Quasi-inverses of Schema Mappings

Ronald Fagin Phokion G. Kolaitis* Lucian Popa
IBM Almaden Research Center
{fagin,kolaitis,lucian}@almaden.ibm.com

Wang-Chiew Tan†
UC Santa Cruz
wctan@cs.ucsc.edu

ABSTRACT

Schema mappings are high-level specifications that describe the relationship between two database schemas. Two operators on schema mappings, namely the composition operator and the inverse operator, are regarded as especially important. Progress on the study of the inverse operator was not made until very recently, as even finding the exact semantics of this operator turned out to be a fairly delicate task. Furthermore, this notion is rather restrictive, since it is rare that a schema mapping possesses an inverse.

In this paper, we introduce and study the notion of a quasi-inverse of a schema mapping. This notion is a principled relaxation of the notion of an inverse of a schema mapping; intuitively, it is obtained from the notion of an inverse by not differentiating between instances that are equivalent for data-exchange purposes. For schema mappings specified by source-to-target tuple-generating dependencies (s-t tgds), we give a necessary and sufficient combinatorial condition for the existence of a quasi-inverse, and then use this condition to obtain both positive and negative results about the existence of quasi-inverses. In particular, we show that every LAV (local-as-view) schema mapping has a quasi-inverse, but that there are schema mappings specified by full s-t tgds that have no quasi-inverse. After this, we study the language needed to express quasi-inverses of schema mappings specified by s-t tgds, and we obtain a complete characterization. We also characterize the language needed to express inverses of schema mappings, and thereby solve a problem left open in the earlier study of the inverse operator. Finally, we show that quasi-inverses can be used in many cases to recover the data that was exported by the original schema mapping when performing data exchange.

Categories and Subject Descriptors

H.2.5 [**Heterogeneous Databases**]: Data translation; H.2.4 [**Systems**]: Relational Databases

General Terms

Algorithms, Theory

*On leave from UC Santa Cruz

†Supported in part by NSF CAREER Award IIS-0347065 and NSF grant IIS-0430994. Work partially done while at IBM Almaden.

PODS'07, June 11–13, 2007, Beijing, China.

Keywords

Schema mapping, data exchange, data integration, metadata model management, inverse, quasi-inverse, dependencies, chase

1. INTRODUCTION

Schema mappings are high-level specifications that describe the relationship between two database schemas. More precisely, a schema mapping is a triple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ consisting of a source schema $\mathbf{S}$, a target schema $\mathbf{T}$, and a set Σ of database dependencies that specify the relationship between the source schema and the target schema. Since schema mappings form the essential building blocks of such crucial data inter-operability tasks as data integration and data exchange (see the surveys [6, 7]), several different operators on schema mappings have been singled out as deserving study in their own right [1]. The composition operator and the inverse operator have emerged as two of the most fundamental operators on schema mappings.

By now, the composition operator has been investigated in depth [5, 8, 9, 10]; however, progress on the study of the inverse operator was not made until very recently, as even finding the exact semantics of this operator turned out to be a delicate task. In [3], the concept of an inverse of a schema mapping was rigorously defined and its basic properties were studied. The definition of an inverse was given by first defining the concept of the *identity* schema mapping Id and then stipulating that a schema mapping $\mathcal{M}'$ is an *inverse* of a schema mapping $\mathcal{M}$ if the composition of $\mathcal{M}$ with $\mathcal{M}'$ yields the identity schema mapping Id, in symbols $\mathcal{M} \circ \mathcal{M}' = \mathrm{Id}$.

Unfortunately, the notion of an inverse of a schema mapping turned out to be rather restrictive, since it is rare that a schema mapping possesses an inverse. Indeed, as shown in [3], if a schema mapping $\mathcal{M}$ is invertible, then $\mathcal{M}$ satisfies the *unique-solutions property*, which asserts that different source instances must have different *spaces of solutions* (that is, different sets of target instances satisfying the specifications of $\mathcal{M}$). This necessary condition for invertibility can be used as a simple, yet powerful, sufficient condition for non-invertibility. In particular, none of the following natural schema mappings possesses an inverse, because it is easy to see that none of them has the unique-solutions property:

Projection: This is the schema mapping specified by the dependency $P(x, y) \rightarrow Q(x)$.

Union: This is the schema mapping specified by the dependencies $P(x) \rightarrow S(x)$ and $Q(x) \rightarrow S(x)$.

Decomposition: This is the schema mapping specified by the dependency $P(x, y, z) \rightarrow Q(x, y) \wedge R(y, z)$.

Moreover, the invertibility of a schema mapping is not robust, as it is affected by changes to the source schema, even when the dependencies remain intact. Specifically, assume that $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is an invertible schema mapping. If the source schema $\mathbf{S}$ is aug-

mented with a new relation symbol R, then the new schema mapping $\mathcal{M}^* = (\mathbf{S} \cup \{R\}, \mathbf{T}, \Sigma)$ is no longer invertible.

In view of these limitations of the notion of an inverse of a schema mapping, it is natural to ask: is there a good alternative notion of an inverse that is not as restrictive as the original notion in [3], but is still useful in data exchange? In what follows, we address this question by formulating the notion of a *quasi-inverse* of a schema mapping, by exploring its properties in depth, and by making a case for its usefulness.

Conceptual contributions We introduce the notion of a *quasi-inverse* of a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ as a principled relaxation of the notion of an inverse mapping of $\mathcal{M}$. Intuitively, the notion of a quasi-inverse is obtained from the notion of an inverse by not differentiating between *ground* instances (i.e., null-free source instances) that are equivalent for data-exchange purposes. In more formal terms, we first consider the equivalence relation $\sim_{\mathcal{M}}$ between ground instances such that $I_1 \sim_{\mathcal{M}} I_2$ holds if I_1 and I_2 have the same *space of solutions*, that is, for every target instance J, we have that $(I_1, J) \models \Sigma$ if and only if $(I_2, J) \models \Sigma$. We then say that a schema mapping $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \mathcal{M}')$ is a *quasi-inverse* of $\mathcal{M}$ if, in a precise technical sense, $\mathcal{M} \circ \mathcal{M}' = \mathrm{Id}$ holds modulo the equivalence relation $\sim_{\mathcal{M}}$.

We show that the concept of a quasi-inverse of a schema mapping is actually part of a unifying framework in which different relaxations of the notion of an inverse of a schema mapping can be obtained by using different equivalence relations that are *refinements* of the equivalence relation $\sim_{\mathcal{M}}$ (i.e., they are contained in $\sim_{\mathcal{M}}$). This framework captures, as special cases, both inverses and quasi-inverses. In fact, the notion of an inverse is the strictest one, while the notion of a quasi-inverse is the most relaxed one; all other relaxations of the notion of an inverse lie in between.

Numerous non-invertible schema mappings possess natural and useful quasi-inverses. Indeed, let us revisit the preceding examples of non-invertible schema mappings.

Projection: The schema mapping specified by $P(x, y) \rightarrow Q(x)$ has a quasi-inverse specified by $Q(x) \rightarrow \exists y P(x, y)$. Intuitively, this quasi-inverse describes the "best" you can do to recover ground instances.

Union: The schema mapping specified by the dependencies $P(x) \rightarrow S(x)$ and $Q(x) \rightarrow S(x)$ has a quasi-inverse specified by $S(x) \rightarrow P(x) \vee Q(x)$. Quasi-inverses need not be unique up to logical equivalence (the same holds true for inverses as well). Indeed, the schema mapping specified by $S(x) \rightarrow P(x)$ is also a quasi-inverse, as are the schema mapping specified by $S(x) \rightarrow Q(x)$ and the schema mapping specified by $S(x) \rightarrow P(x) \wedge Q(x)$.

Decomposition: The schema mapping specified by the dependency $P(x, y, z) \rightarrow Q(x, y) \wedge R(y, z)$ has a quasi-inverse specified by $Q(x, y) \wedge R(y, z) \rightarrow P(x, y, z)$. Another quasi-inverse of this schema mapping is specified by $Q(x, y) \rightarrow \exists z P(x, y, z)$ and $R(y, z) \rightarrow \exists x P(x, y, z)$.

Finally, if $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is an invertible schema mapping and we augment $\mathbf{S}$ with a new relation symbol R, then *every* inverse of $\mathcal{M}$ is a quasi-inverse of the resulting non-invertible schema mapping $\mathcal{M}^* = (\mathbf{S} \cup \{R\}, \mathbf{T}, \Sigma)$. Moreover, if a schema mapping $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$ is a quasi-inverse of a non-invertible schema mapping $\mathcal{M}$, then the schema mapping $\mathcal{M}'' = (\mathbf{T}, \mathbf{S} \cup \{R\}, \Sigma')$ is a quasi-inverse of $\mathcal{M}^*$. Thus, unlike the notion of an inverse, the notion of a quasi-inverse is robust when relation symbols are added to the source schema.

Technical Contributions Our results span three different directions: an exact criterion for the existence of quasi-inverses, complete characterizations of the languages needed to express quasi-inverses and inverses, and results on the use of quasi-inverses in data exchange.

Existence of quasi-inverses For schema mappings specified by source-to-target tuple-generating dependencies (s-t tgds), we give a necessary and sufficient combinatorial condition, called the *subset property*, for the existence of a quasi-inverse. We then apply this condition to obtain both positive and negative results about the existence of quasi-inverses. On the positive side, we use the subset property as a sufficient condition for quasi-invertibility to show that every LAV (local-as-view) schema mapping has a quasi-inverse; this result provides a unifying explanation for the quasi-invertibility of the projection, union, and decomposition schema mappings. On the negative side, we use the subset property as a necessary condition for quasi-invertibility to show that there are simple schema mappings specified by full s-t tgds that have no quasi-inverse; a fortiori, such schema mappings have no inverse.

The language of inverses and quasi-inverses We investigate the language needed to express quasi-inverses of schema mappings specified by s-t tgds, and we obtain a complete characterization. Specifically, we show that if a schema mapping specified by a finite set of s-t tgds is quasi-invertible, then it has a quasi-inverse specified by a finite set of target-to-source disjunctive tgds with constants and inequalities (in fact, inequalities among constants suffice). Moreover, we give an exponential-time algorithm for constructing such a quasi-inverse. The left-hand side of a target-to-source disjunctive tgd with constants and inequalities is a conjunction of target atoms, formulas of the form *Constant*(x) that evaluate to true only when x is instantiated to a constant (non-null) value, and inequalities $x_i \neq x_j$; the right-hand side is a disjunction of conjunctive queries over the source. We show that our expressibility result is optimal by proving that *no* proper fragment of the language of disjunctive tgds with constants and inequalities suffices to express quasi-inverses; that is, both constants and inequalities in the left-hand side of dependencies are needed, as are both disjunctions and existential quantifiers in the right-hand side of dependencies. For schema mappings specified by a finite set of full s-t tgds, we show that if such a schema mapping is quasi-invertible, then it has a quasi-inverse specified by a finite set of target-to-source disjunctive tgds with inequalities; in other words, the predicate *Constant* is not needed in this case. We also show that every LAV schema mapping has a quasi-inverse specified by a finite set of target-to-source tgds with inequalities and constants; thus, in this case, there is no need for disjunctions in the right-hand side of dependencies.

Concerning the language needed to express inverses of schema mappings specified by s-t tgds, the paper [3] focussed only on inverses specified by target-to-source tgds, and left open the problem of characterizing the language needed to express inverses of schema mappings. Here, we settle this problem by showing that if a schema mapping specified by a finite set of s-t tgds is invertible, then it has an inverse specified by a finite set of target-to-source tgds with constants and inequalities. This turns out to be an optimal result as well.

Although we have completely characterized the language needed to express quasi-inverses and inverses of schema mappings specified by a finite set of s-t tgds, the complexity of deciding the existence a quasi-inverse and of an inverse of such schema mappings remain open problems. In fact, even the decidability of these problems is open.

Using quasi-inverses in data exchange Since it is rare that a schema mapping $\mathcal{M}$ has an inverse, we cannot hope to always obtain an exact copy of the original ground instance from target instances. The notion of a quasi-inverse is motivated from the idea that "similarity up to the space of solutions" is often good enough for data-exchange applications; this is why the definition of a quasi-inverse of a schema mapping $\mathcal{M}$ relaxes exact equality between ground instances to $\sim_{\mathcal{M}}$-equivalence. We show that, even though it is not possible to recover an exact copy of the original source instance, in many cases quasi-inverses allow us to recover a source instance that has "equivalent" properties from the data-exchange point of view.

More formally, assume that $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is a schema mapping specified by a finite set of s-t tgds and $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$ is a quasi-

inverse of $\mathcal{M}$ specified by a finite set of target-to-source disjunctive tgds with constants and inequalities among constants. We show that $\mathcal{M}'$ is *sound*, which means that $\mathcal{M}'$ has the following property. Let I be an arbitrary ground instance and let U be the result of chasing I with Σ. Suppose we chase U back from target to source with the disjunctive dependencies in Σ' to obtain a set $\mathcal{V}$ of source instances and then we chase every member of $\mathcal{V}$ with the dependencies in Σ to obtain a set $\mathcal{U}'$ of target instances. Then $\mathcal{U}'$ contains a target instance U' that can be mapped homomorphically into U; thus, the target instance U' contains (up to a homomorphism) only facts of U, although not necessarily all of them. Furthermore, if $\mathcal{M}'$ is obtained by applying our algorithm for constructing a quasi-inverse of $\mathcal{M}$, then $\mathcal{M}'$ is *faithful*, which means that the set $\mathcal{U}'$ contains a target instance U' that is homomorphically equivalent to U. In other words, there is a source instance V in $\mathcal{V}$ whose chase with Σ is homomorphically equivalent to U. This instance V is thus "data-exchange equivalent" to the original source instance I.

2. PRELIMINARIES

A *schema* $\mathbf{R}$ is a finite sequence $(R_1, \ldots, R_k)$ of relation symbols, each of a fixed arity. An *instance* I over $\mathbf{R}$ is a sequence $(R_1^I, \ldots, R_k^I)$, where each R_i^I is a finite relation of the same arity as R_i. We shall often use R_i to denote both the relation symbol and the relation R_i^I that interprets it. An *atom over* $\mathbf{R}$ is a formula $P(v_1, \ldots, v_m)$, where P is a relation symbol in $\mathbf{R}$ and $v_1, \ldots, v_m$ are variables.

In what follows, we assume that $\mathbf{S}$ is a fixed *source* schema and $\mathbf{T}$ is a fixed *target* schema. We also assume that we have a fixed infinite set Const of constants and an infinite set Var of nulls that is disjoint from Const. For source instances, our main focus is on instances with individual values from Const only; we call such source instances *ground* instances. In contrast, target instances typically have individual values from Const $\cup$ Var. Intuitively, this models the situation in which we perform data exchange from $\mathbf{S}$ to $\mathbf{T}$: the individual values of source instances are known, while incomplete information in the specification of data exchange may give rise to null values in the target instances.

Dependencies, schema mappings, and data exchange notions Recall that a schema mapping is a triple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ consisting of a source schema $\mathbf{S}$, a target schema $\mathbf{T}$, and a set Σ of database dependencies that specify the relationship between the source schema and the target schema. We say that $\mathcal{M}$ is *specified by* Σ.

We review several notions from [4] that will be needed in this paper. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping. If I a is ground instance, then a *solution for* I under $\mathcal{M}$ is a target instance J such that $(I, J) \models \Sigma$. The set of all solutions for I under $\mathcal{M}$ is denoted by $\mathrm{Sol}(\mathcal{M}, I)$.

Let J, J' be two target instances. A function h from Const$\cup$Var to Const$\cup$Var is a *homomorphism* from J to J' if for every c in Const, we have that $h(c) = c$, and for every relation symbol R in $\mathbf{T}$ and every tuple $(a_1, \ldots, a_n) \in R^J$, we have that $(h(a_1), \ldots, h(a_n)) \in R^{J'}$. The instances J and J' are said to be *homomorphically equivalent* if there are homomorphisms from J to J' and from J' to J.

Given a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and a ground instance I, a *universal solution* for I under $\mathcal{M}$ is a solution J for I under $\mathcal{M}$ such that for every solution J' for I under $\mathcal{M}$, there is a homomorphism $h : J \to J'$. Intuitively, universal solutions are the "most general" solutions among the space of all solutions for I.

A *source-to-target tuple-generating dependency*, in short, an *s-t tgd*, is a first-order formula of the form $\forall \mathbf{x}(\varphi(\mathbf{x}) \to \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y}))$, where $\varphi(\mathbf{x})$ is a conjunction of atoms over $\mathbf{S}$, $\psi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms over $\mathbf{T}$, and every variable in $\mathbf{x}$ occurs in an atom in $\varphi(\mathbf{x})$.

If $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is a schema mapping specified by a finite set Σ of s-t tgds, then *chasing* I with Σ produces a target instance U such that U is a universal solution for I under $\mathcal{M}$. We shall often write that $U = \mathit{chase}_\Sigma(I)$ and say that U is *the* result of the chase. (In general, there may be several such instances U but they are all homomorphically equivalent.)

Our goal in this paper is to investigate inverses and quasi-inverses of schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a finite set of s-t tgds. In particular, we will identify the languages needed for expressing such inverses and quasi-inverses, and will show that these languages must be richer than the language of target-to-source tgds. The following definition introduces the richer classes of dependencies needed.

DEFINITION 2.1. *Let Constant be a relation symbol that is different from all relation symbols in* $\mathbf{S}$ *and* $\mathbf{T}$.

1. *A* disjunctive tgd with constants and inequalities from $\mathbf{T}$ to $\mathbf{S}$ *is a first-order formula of the form*

$$\forall \mathbf{x}(\varphi(\mathbf{x}) \to \bigvee_{i=1}^{n} \exists \mathbf{y}_i \psi_i(\mathbf{x}, \mathbf{y}_i)),$$

where:

- *the formula $\varphi(\mathbf{x})$ is a conjunction of*
  (1) atoms over $\mathbf{T}$, such that every variable in $\mathbf{x}$ occurs in one of them;
  (2) formulas of the form Constant(x), where x is a variable in $\mathbf{x}$;
  (3) inequalities $x \neq x'$, where x and x' are variables in $\mathbf{x}$.
- *Each formula $\psi_i(\mathbf{x}, \mathbf{y}_i)$ is a conjunction of atoms over $\mathbf{S}$.*

Naturally, a formula Constant(x) evaluates to true if and only if x is interpreted by a value in Const.

2. *A* disjunctive tgd with constants and inequalities among constants *is a disjunctive tgd with inequalities and constants where the formulas Constant(x) and Constant(x') occur as conjuncts of $\varphi(\mathbf{x})$ whenever the inequality $x \neq x'$ is a conjunct of $\varphi(\mathbf{x})$.*

Clearly, disjunctive tgds with constants and inequalities extend the language of tgds with three features: (1) formulas of the form *Constant*(x) in the left-hand side; (2) inequalities in the left-hand side; and (3) disjunctions in the right-hand side. If the right-hand side consists of a single disjunct, then we talk about *tgds with constants and inequalities*. The concepts of *disjunctive tgds with inequalities*, *tgds with inequalities*, and other such special cases of Definition 2.1 are defined in an analogous way. For example,

$$P(x, y, z) \wedge \mathit{Constant}(x) \wedge x \neq y \to \exists w Q(x, w) \vee Q(x, y)$$

is a disjunctive tgd with constants and inequalities,

$$P(x, y, z) \wedge x \neq y \to \exists w Q(x, w) \vee Q(x, y)$$

is a disjunctive tgd with inequalities, and

$$P(x, y, z) \wedge x \neq y \to Q(x, y)$$

is a tgd with inequalities. Note that, for convenience, we have dropped the universal quantifiers in the front.

Composing and inverting schema mappings Next, we recall the concept of the *composition* of two schema mappings, introduced in [5, 9], and the concept of an *inverse* of a schema mapping, introduced in [3].

Let $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$ be schema mappings. The *composition* $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is a schema mapping $(\mathbf{S}_1, \mathbf{S}_3, \Sigma_{13})$ such that for every $\mathbf{S}_1$-instance I and every $\mathbf{S}_3$-instance K, we have that $(I, K) \models \Sigma_{13}$ if and only if there is an $\mathbf{S}_2$-instance J such that $(I, J) \models \Sigma_{12}$ and $(J, K) \models \Sigma_{23}$. When the schemas involved are understood from the context, we will often write $\Sigma_{12} \circ \Sigma_{23}$ to denote the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$.

Let $\hat{\mathbf{S}}$ be a replica of the source schema $\mathbf{S}$, that is, for every relation symbol R of $\mathbf{S}$, the schema $\hat{\mathbf{S}}$ contains a relation symbol $\hat{R}$ that is not in $\mathbf{S}$ and has the same arity as R. Clearly, every source instance I has a replica instance $\hat{I}$ over $\hat{\mathbf{S}}$.

- If $\mathcal{M}^* = (\mathbf{S}, \hat{\mathbf{S}}, \Sigma^*)$ is a schema mapping, we write $\mathrm{Inst}(\mathcal{M}^*)$ for the set of all pairs (I_1, I_2) such that I_1 is a ground $\mathbf{S}$-instance, I_2 is a ground $\hat{\mathbf{S}}$-instance, and $(I_1, I_2) \models \Sigma^*$.
- The *identity schema mapping* is, by definition, the schema mapping $\mathrm{Id} = (\mathbf{S}, \hat{\mathbf{S}}, \Sigma_{\mathrm{Id}})$, where Σ_{Id} consists of the dependencies $R(\mathbf{x}) \rightarrow \hat{R}(\mathbf{x})$ as R ranges over the relation symbols in $\mathbf{S}$. Thus, $\mathrm{Inst}(\mathrm{Id})$ consists of all pairs (I_1, I_2) of a ground $\mathbf{S}$-instance I_1 and a ground $\hat{\mathbf{S}}$-instance I_2 such that $\hat{I}_1 \subseteq I_2$.
- Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping. We say that a schema mapping $\mathcal{M}' = (\mathbf{T}, \hat{\mathbf{S}}, \Sigma')$ is an *inverse* of $\mathcal{M}$ if $\mathrm{Inst}(\mathrm{Id}) = \mathrm{Inst}(\mathcal{M} \circ \mathcal{M}')$. This means that, for every pair (I_1, I_2) of a ground $\mathbf{S}$-instance I_1 and a ground $\hat{\mathbf{S}}$-instance I_2, we have that $\hat{I}_1 \subseteq I_2$ if and only if there is a target instance J such that $(I_1, J) \models \Sigma$ and $(J, I_2) \models \Sigma'$.

From now on and for notational simplicity, we will write $\mathbf{S}$ to also denote its replica $\hat{\mathbf{S}}$; it will be clear from the context if we refer to $\mathbf{S}$ or to its replica. Moreover, we will use the same symbol to denote both a ground $\mathbf{S}$-instance I and its replica $\hat{\mathbf{S}}$-instance $\hat{I}$.

3. A UNIFYING FRAMEWORK

In this section, we develop a unifying framework for defining and studying a spectrum of notions that relax the notion of an inverse of a schema mapping in a principled manner. The key idea is not to differentiate between ground instances that are *equivalent* for data-exchange purposes.

DEFINITION 3.1. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping and let I, I' be two ground instances.*

We write $I \sim_{\mathcal{M}} I'$ to denote that $\mathrm{Sol}(\mathcal{M}, I) = \mathrm{Sol}(\mathcal{M}, I')$. When $\mathcal{M}$ is understood from the context, we may write $\sim$ in place of $\sim_{\mathcal{M}}$.

Clearly, $\sim_{\mathcal{M}}$ is an equivalence relation on ground instances. We will use this equivalence relation to define relaxations of the notion of an inverse. Before doing so, we introduce an auxiliary concept.

DEFINITION 3.2. *Let $\sim_1$ and $\sim_2$ be two equivalence relations on ground instances, and let D be a binary relation between ground instances. We set*

$$D[\sim_1, \sim_2] = \{(I_1, I_2) : \exists I_1' I_2' ((I_1 \sim_1 I_1') \wedge (I_1', I_2') \in D \wedge (I_2 \sim_2 I_2'))\}.$$

It is easy to see that $D[\sim_1, \sim_2] = \sim_1 \circ D \circ \sim_2$, where $\circ$ denotes the composition of two binary relations.

In what follows, we will use the notation $\sim_{(1,2)}$ to denote the *product* equivalence relation of two equivalence relations $\sim_1$ and $\sim_2$. This means that $(I_1, I_2) \sim_{(1,2)} (I_1', I_2')$ if and only if $I_1 \sim_1 I_1'$ and $I_2 \sim_2 I_2'$. Thus, we have that

$$D[\sim_1, \sim_2] = \{(I_1, I_2) : \exists I_1' I_2' ((I_1, I_2) \sim_{(1,2)} (I_1', I_2') \wedge (I_1', I_2') \in D)\}.$$

We are now ready to give the crucial definition of an *inverse with respect to two equivalence relations*, which generalizes the concept of an inverse in [3].

DEFINITION 3.3. *Assume that $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is a schema mapping and $\sim_1$, $\sim_2$ are two equivalence relations on ground instances such that $\sim_1 \subseteq \sim_{\mathcal{M}}$ and $\sim_2 \subseteq \sim_{\mathcal{M}}$. We say that schema mapping $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$ is a $(\sim_1, \sim_2)$-*inverse *of $\mathcal{M}$ if*

$$\mathrm{Inst}(\mathrm{Id})[\sim_1, \sim_2] = \mathrm{Inst}(\mathcal{M} \circ \mathcal{M}')[\sim_1, \sim_2].$$

This means that, for every pair (I_1, I_2) of ground instances, the following statements are equivalent:

1. *There are ground instances I_1' and I_2' such that $(I_1, I_2) \sim_{(1,2)} (I_1', I_2')$ and $I_1' \subseteq I_2'$.*
2. *There are ground instances I_1'' and I_2'' and a target instance J such that $(I_1, I_2) \sim_{(1,2)} (I_1'', I_2'')$, $(I_1'', J) \models \Sigma$, and $(J, I_2'') \models \Sigma'$.*

As an immediate consequence of the definitions, we have that $\mathcal{M}'$ is an inverse of $\mathcal{M}$ if and only if $\mathcal{M}'$ is an $(=, =)$-inverse of Σ. Thus, informally, a $(\sim_1, \sim_2)$-inverse of $\mathcal{M}$ is an inverse of $\mathcal{M}$ modulo the equivalence relations $\sim_1$, $\sim_2$.

Next, we introduce a combinatorial property that will be used to characterize the existence of $(\sim_1, \sim_2)$-inverses.

DEFINITION 3.4. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping. We say that $\mathcal{M}$ has the $(\sim_1, \sim_2)$-*subset property *if for every pair (I_1, I_2) of ground instances such that $\mathrm{Sol}(\mathcal{M}, I_2) \subseteq \mathrm{Sol}(\mathcal{M}, I_1)$, there is a pair (I_1', I_2') of ground instances such that $(I_1, I_2) \sim_{(1,2)} (I_1', I_2')$ and $I_1' \subseteq I_2'$.*

Before stating any technical results, let us give some insight to the $(\sim_1, \sim_2)$-subset property. It is easy to see that if Σ is a set of s-t tgds and $I_1 \subseteq I_2$, then $\mathrm{Sol}(\mathcal{M}, I_2) \subseteq \mathrm{Sol}(\mathcal{M}, I_1)$. Note also that the $(=, =)$-subset property asserts that if $\mathrm{Sol}(\mathcal{M}, I_2) \subseteq \mathrm{Sol}(\mathcal{M}, I_1)$, then $I_1 \subseteq I_2$; hence, it is the converse of the preceding fact. Thus, the $(\sim_1, \sim_2)$-subset property can be construed as a weak converse of the fact that if $I_1 \subseteq I_2$, then $\mathrm{Sol}(\mathcal{M}, I_2) \subseteq \mathrm{Sol}(\mathcal{M}, I_1)$; informally, it is a converse modulo the equivalence relations $\sim_1$ and $\sim_2$.

Our first theorem asserts that the $(\sim_1, \sim_2)$-subset property is a necessary and sufficient condition for the existence of a $(\sim_1, \sim_2)$-inverse of a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ in which Σ is a set of s-t tgds.

THEOREM 3.5. *Assume that $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is a schema mapping in which Σ is a finite set of s-t tgds and let $\sim_1$, $\sim_2$ be two equivalence relations on ground instances such that $\sim_1 \subseteq \sim_{\mathcal{M}}$ and $\sim_2 \subseteq \sim_{\mathcal{M}}$. Then the following are equivalent:*

1. *$\mathcal{M}$ has a $(\sim_1, \sim_2)$-inverse.*
2. *$\mathcal{M}$ has the $(\sim_1, \sim_2)$-subset property.*

PROOF. *(Sketch)* Assume that $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$ is a $(\sim_1, \sim_2)$-inverse of Σ. Let (I_1, I_2) be a pair of ground instances such that $\mathrm{Sol}(\mathcal{M}, I_2) \subseteq \mathrm{Sol}(\mathcal{M}, I_1)$. Since $(I_2, I_2) \in \mathrm{Inst}(\mathrm{Id})$, it follows that $(I_2, I_2) \in \mathrm{Inst}(\mathcal{M} \circ \mathcal{M}')[\sim_1, \sim_2]$. Therefore, there is a pair (I_3, I_4) of ground instances and a target instance J such that $(I_2, I_2) \sim_{(1,2)} (I_3, I_4)$, and such that $(I_3, J) \models \Sigma$ and $(J, I_4) \models \Sigma'$. Since $I_2 \sim_1 I_3$ and $\sim_1 \subseteq \sim_{\mathcal{M}}$, we obtain that $I_2 \sim_{\mathcal{M}} I_3$. Hence, since $(I_3, J) \models \Sigma$, it follows that $(I_2, J) \models \Sigma$. Since $\mathrm{Sol}(\Sigma, I_2) \subseteq \mathrm{Sol}(\Sigma, I_1)$, we have that $(I_1, J) \models \Sigma$. Consequently, $(I_1, I_4) \in \mathrm{Inst}(\mathcal{M} \circ \mathcal{M}')$, which implies that $(I_1, I_4) \in \mathrm{Inst}(\mathrm{Id})[\sim_1, \sim_2]$. In turn, this implies that there is a pair (I_1', I_4') of ground instances such that $(I_1, I_4) \sim_{(1,2)} (I_1', I_4')$ and $I_1' \subseteq I_4'$. Since $I_2 \sim_2 I_4$ and $I_4 \sim_2 I_4'$, we have that $I_2 \sim_2 I_4'$. Therefore, $(I_1, I_2) \sim_{(1,2)} (I_1', I_4')$ where $I_1' \subseteq I_4'$. This establishes the implication $(1) \Longrightarrow (2)$.

For the implication $(2) \Longrightarrow (1)$, assume that $\mathcal{M}$ has the $(\sim_1, \sim_2)$-subset property. Put

$$D = \{(J, I) : \ J \text{ is universal for } I \text{ under } \mathcal{M}\}.$$

Let $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$ be the schema mapping such that $(J, I) \in D$ if and only if $(J, I) \models \Sigma'$. It can be shown that $\mathcal{M}'$ is a $(\sim_1, \sim_2)$-inverse of Σ; details appear in the full paper. □

Theorem 3.5 yields the following necessary and sufficient condition for the existence of an inverse.

COROLLARY 3.6. *Assume that $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is a schema mapping where Σ is a finite set of s-t tgds. Then the following statements are equivalent:*

1. *$\mathcal{M}$ has an inverse.*
2. *$\mathcal{M}$ has the $(=, =)$-subset property, that is to say, if I_1 and I_2 are two ground instances such that $\mathrm{Sol}(\mathcal{M}, I_2) \subseteq \mathrm{Sol}(\mathcal{M}, I_1)$, then $I_1 \subseteq I_2$.*

As mentioned in the Introduction, the *unique-solutions property* was identified in [3] as a necessary condition for a schema mapping $\mathcal{M}$ to have an inverse. By definition, this property says that if I_1 and I_2 are ground instances such that $I_1 \neq I_2$, then we have that $\mathrm{Sol}(\mathcal{M}, I_1) \neq \mathrm{Sol}(\mathcal{M}, I_2)$. Clearly, the $(=,=)$-subset property implies the unique-solutions property. Indeed, if $\mathrm{Sol}(\mathcal{M}, I_1) = \mathrm{Sol}(\mathcal{M}, I_2)$, then by applying the $(=,=)$-subset property twice, we have that $I_1 \subseteq I_2$ and $I_2 \subseteq I_1$, and so $I_1 = I_2$. In the full paper we show that there is a schema mapping $\mathcal{M}$ that is specified by a finite set of s-t tgds and has the unique-solutions property, but does not have the $(=,=)$-property. Hence, the unique-solutions property is not a sufficient condition for the existence of an inverse.

By varying the equivalence relations $\sim_1$ and $\sim_2$, we can obtain a variety of $(\sim_1, \sim_2)$-inverses. The next proposition provides a basic tool for comparing them.

PROPOSITION 3.7. *Let $\mathcal{M}$ be a schema mapping and let $\sim_1$, $\sim_2$, $\sim_3$, $\sim_4$ be four equivalence relations on ground instances such that $\sim_1 \subseteq \sim_3 \subseteq \sim_{\mathcal{M}}$ and $\sim_2 \subseteq \sim_4 \subseteq \sim_{\mathcal{M}}$. Every $(\sim_1, \sim_2)$-inverse of $\mathcal{M}$ is also a $(\sim_3, \sim_4)$-inverse of $\mathcal{M}$.*

Proposition 3.7 implies that the spectrum of $(\sim_1, \sim_2)$-inverses has both "strongest" and "weakest" elements. Indeed, when $\mathcal{M}'$ is an $(=,=)$-inverse of $\mathcal{M}$ (i.e., $\mathcal{M}'$ is an inverse of $\mathcal{M}$), then for every two equivalence relations $\sim_1$ and $\sim_2$ contained in $\sim_{\mathcal{M}}$, we have that $\mathcal{M}'$ is a $(\sim_1, \sim_2)$-inverse of $\mathcal{M}$. At the other end of the spectrum, if $\mathcal{M}'$ is a $(\sim_1, \sim_2)$-inverse of $\mathcal{M}$, then $\mathcal{M}'$ is also a $(\sim_{\mathcal{M}}, \sim_{\mathcal{M}})$-inverse of $\mathcal{M}$. In what follows, we will focus on $(=,=)$-inverses (i.e., on inverses) and on $(\sim_{\mathcal{M}}, \sim_{\mathcal{M}})$-inverses, which we will refer to from now on as *quasi-inverses*.

DEFINITION 3.8. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping. We say that a schema mapping $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$ is a* quasi-inverse *of $\mathcal{M}$ if $\mathcal{M}'$ is a $(\sim_{\mathcal{M}}, \sim_{\mathcal{M}})$-inverse of $\mathcal{M}$, that is,*

$$\mathrm{Inst}(\mathrm{Id})[\sim_{\mathcal{M}}, \sim_{\mathcal{M}}] = \mathrm{Inst}(\mathcal{M} \circ \mathcal{M}')[\sim_{\mathcal{M}}, \sim_{\mathcal{M}}].$$

We say that $\mathcal{M}$ is quasi-invertible *if it has a quasi-inverse, and* invertible *if it has an inverse.*

PROPOSITION 3.9. *Every quasi-inverse of an invertible schema mapping $\mathcal{M}$ is an inverse of $\mathcal{M}$.*

This proposition holds because if $\mathcal{M}$ is invertible, then the unique-solutions property implies that the equivalence relation $\sim_{\mathcal{M}}$ coincides with the equality relation $=$ on ground instances. Thus, for invertible schema mappings, there is no distinction between inverses and quasi-inverses. In contrast, there are schema mappings that are not invertible, but have natural quasi-inverses. As a matter of fact, the three examples of schema mappings given in the Introduction (*Projection*, *Union*, and *Decomposition*) have this property. We revisit one of them.

EXAMPLE 3.10. Let $\mathcal{M}$ be the *Decomposition* schema mapping specified by the tgd

$$P(x, y, z) \rightarrow Q(x, y) \wedge R(y, z).$$

First, $\mathcal{M}$ does not have an inverse, since it does not have the unique-solutions property. For example, if I_1 and I_2 are source instances, where P^{I_1} has exactly the tuples $\{(0,0,0), (0,0,1), (1,0,0)\}$, and P^{I_2} has these tuples along with $(1,0,1)$, then I_1 and I_2 have exactly the same solutions. We claim, however, that $\mathcal{M}$ has the $(\sim_{\mathcal{M}}, \sim_{\mathcal{M}})$-subset property. To see this, let I_1 and I_2 be two ground instances such that $\mathrm{Sol}(\mathcal{M}, I_2) \subseteq \mathrm{Sol}(\mathcal{M}, I_1)$. Let J be the solution for I_2 obtained by taking $Q^J = \pi_{12}(P^{I_2})$ and $R^J = \pi_{23}(P^{I_2})$. Since $\mathrm{Sol}(\mathcal{M}, I_2) \subseteq \mathrm{Sol}(\mathcal{M}, I_1)$, we have that J is also in $\mathrm{Sol}(\mathcal{M}, I_1)$, so $\pi_{12}(P^{I_1}) \subseteq \pi_{12}(P^{I_2})$ and $\pi_{23}(P^{I_1}) \subseteq \pi_{23}(P^{I_2})$. Let $I_2' = I_1 \cup I_2$. From the two inclusions we have just established, it follows that $I_2' \sim_{\mathcal{M}} I_2$; moreover, we have that $I_1 \subseteq I_2'$. This shows that $\mathcal{M}$ has the $(\sim_{\mathcal{M}}, \sim_{\mathcal{M}})$-subset property (actually, this shows that $\mathcal{M}$ has the stronger $(=, \sim_{\mathcal{M}})$-subset property).

Theorem 3.5 implies that $\mathcal{M}$ has a quasi-inverse. As a matter of fact, one can show directly that the schema mapping $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$ with Σ' consisting of the tgd

$$Q(x, y) \wedge R(y, z) \rightarrow P(x, y, z)$$

is a quasi-inverse of $\mathcal{M}$. One can also show directly that another quasi-inverse of $\mathcal{M}$ is the schema mapping $\mathcal{M}'' = (\mathbf{T}, \mathbf{S}, \Sigma'')$, where Σ'' consists of the tgds

$$\begin{aligned} Q(x, y) &\rightarrow \exists z P(x, y, z) \\ R(y, z) &\rightarrow \exists x P(x, y, z). \end{aligned}$$

This also shows that the notion of a quasi-inverse of a schema mapping need not be unique up to logical equivalence. The same holds true for the notion of an inverse [3]. □

The *Projection*, *Union*, and *Decomposition* schema mappings are LAV *(local-as-view)* schema mappings, that is, the left-hand side of each dependency is a single atom. The next result shows that *every* LAV schema mapping has a quasi-inverse. The proof generalizes the argument in Example 3.10.

PROPOSITION 3.11. *If $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is a LAV schema mapping, then $\mathcal{M}$ has the $(\sim_{\mathcal{M}}, \sim_{\mathcal{M}})$-subset property. Consequently, every LAV schema mapping has a quasi-inverse.*

PROOF. *(Hint)* Assume that I_1 and I_2 are two ground instances such that $\mathrm{Sol}(\mathcal{M}, I_2) \subseteq \mathrm{Sol}(\mathcal{M}, I_1)$. Let J_1 be a universal solution for I_1, and let J_2 be a universal solution for I_2. Let $I_2' = I_1 \cup I_2$. Clearly, $I_1 \subseteq I_2'$. In the full paper, we show that $I_2 \sim_{\mathcal{M}} I_2'$, which implies that $\mathcal{M}$ has the $(\sim_{\mathcal{M}}, \sim_{\mathcal{M}})$-subset property (and in fact the stronger $(=, \sim_{\mathcal{M}})$-subset property). □

Our next result asserts that, in contrast to LAV schema mappings, there are schema mappings specified by *full* s-t tgds that have no quasi-inverses. Recall that an s-t tgd is *full* if its right-hand side has no existential quantifiers; this means that it is of the form $\forall \mathbf{x}(\varphi(\mathbf{x}) \rightarrow \psi(\mathbf{x}))$, where $\varphi(\mathbf{x})$ is a conjunction of source atoms and $\psi(\mathbf{x})$ is a conjunction of target atoms.

PROPOSITION 3.12. *There is a schema mapping $\mathcal{M}$ that is specified by a single full s-t tgd and has no quasi-inverse.*

PROOF. *(Hint)* Let $\mathcal{M}$ be the schema mapping specified by the following full s-t tgd:

$$E(x, z) \wedge E(z, y) \rightarrow F(x, y) \wedge M(z).$$

It can be shown that $\mathcal{M}$ does not have the $(\sim_{\mathcal{M}}, \sim_{\mathcal{M}})$-subset property, which, by Theorem 3.5, implies that $\mathcal{M}$ has no quasi-inverse. The details can be found in the full paper. □

Note that the $(\sim_{\mathcal{M}}, \sim_{\mathcal{M}})$-subset property is used "positively" in the proof of Proposition 3.11 and "negatively" in the proof of Proposition 3.12. More precisely, the $(\sim_{\mathcal{M}}, \sim_{\mathcal{M}})$-subset property is used as a sufficient condition for the existence of quasi-inverses in Proposition 3.11 and as a necessary condition in Proposition 3.12.

4. THE LANGUAGE OF QUASI-INVERSES

One of our main results is the following theorem, characterizing the language for quasi-inverses of schema mappings specified by tgds.

THEOREM 4.1. *Let $\mathcal{M}$ be a schema mapping specified by a finite set of s-t tgds. If $\mathcal{M}$ has a quasi-inverse then the following hold.*

1. *$\mathcal{M}$ has a quasi-inverse $\mathcal{M}'$ specified by a finite set of disjunctive tgds with constants and inequalities.*
2. *There is an exponential-time algorithm for producing $\mathcal{M}'$.*
3. *Statement (1) is not necessarily true if we disallow either constants or inequalities in the left-hand side, or disallow disjunctions or existential quantifiers in the right-hand side.*

In fact, the quasi-inverse $\mathcal{M}'$ that the algorithm produces has inequalities only among constants.

We illustrate the intuition behind the construction of $\mathcal{M}'$, with two examples. We begin with the union example, where Σ consists of the s-t tgds $P(x) \rightarrow S(x)$ and $Q(x) \rightarrow S(x)$. There are two possible "generators" of $S(x)$, namely $P(x)$ and $Q(x)$. These possibilities are reflected by the disjunctive tgd $S(x) \rightarrow P(x) \vee Q(x)$ (we shall put a variation of this disjunctive tgd into Σ'). As another example, let Σ consist of the s-t tgds $S(x,y) \rightarrow P(x,y)$ and $T(x,y) \rightarrow P(x,x)$. There is only one possible generator of $P(x,y)$ if x and y are different, namely $S(x,y)$, and this is reflected by the tgd with inequalities $P(x,y) \wedge (x \neq y) \rightarrow S(x,y)$. However, there are two possible generators of $P(x,x)$, namely $S(x,x)$ and $T(x,y)$, and this is reflected by the disjunctive tgd $P(x,x) \rightarrow S(x,x) \vee \exists y T(x,y)$. The algorithm for producing quasi-inverses systematically considers all such generators.

We now discuss the machinery behind the algorithm to produce $\mathcal{M}'$, including a formal definition of "generator". If α is a conjunction of atoms (or an instantiation of atoms), define I_α to be an instance whose facts are the conjuncts of α. Note that I_α may not be an instance in the usual sense, because the active domain may include variables, in addition to constants or nulls. Thus, I_α is a type of canonical instance. Let $\mathbf{x}$ be a vector of distinct variables. A *complete description* $\delta(\mathbf{x})$ is a conjunction of equalities $x_i = x_j$ and inequalities $x_k \neq x_\ell$ among the variables in $\mathbf{x}$ in a consistent manner, that completely describes which variables are equal and which are unequal.

Let Σ be a finite set of s-t tgds. We now define a set Σ^* that includes Σ and that is logically equivalent to Σ. For each member σ of Σ, and for each complete description δ of the variables that each appear in both the left-hand side and the right-hand side of σ, select a unique representative of each equivalence class determined by δ, and let $f(\sigma, \delta)$ be obtained from σ by replacing every variable in σ by the representative of its equivalence class. Let Σ^* consist of Σ and all such formulas $f(\sigma, \delta)$ (for all choices of σ in Σ and all complete descriptions δ of the variables that each appear in both the left-hand side and the right-hand side of σ). For example, if σ is $R(x_1,x_2,x_3,x_4) \rightarrow \exists y(Q(x_1,y) \wedge S(y,x_2,x_3))$, and if δ is $(x_1 = x_3) \wedge (x_1 \neq x_2) \wedge (x_2 \neq x_3)$, then $\{x_1, x_3\}$ forms one equivalence class and $\{x_2\}$ is the other equivalence class, and $f(\sigma, \delta)$ is $R(x_1,x_2,x_1,x_4) \rightarrow \exists y(Q(x_1,y) \wedge S(y,x_2,x_1))$.

DEFINITION 4.2. Let $\beta(\mathbf{x}, \mathbf{z})$ be a conjunction of source atoms, and let $\psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$ be a conjunction of target atoms, where the members of $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are all distinct, and the members of $\mathbf{x}$ are exactly the variables that appear in both $\beta(\mathbf{x}, \mathbf{z})$ and $\psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$. Let Σ be a finite set of s-t tgds. We say that $\beta(\mathbf{x}, \mathbf{z})$ is a *generator of* $\exists \mathbf{y} \psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$ *(with respect to* Σ*)* if the s-t tgd $\beta(\mathbf{x}, \mathbf{z}) \rightarrow \exists \mathbf{y} \psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$ is a logical consequence of Σ.

When Σ is understood, we shall often drop the words "with respect to Σ". It follows easily from the standard theory of the chase that $\beta(\mathbf{x}, \mathbf{z})$ is a generator of $\exists \mathbf{y} \psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$ with respect to Σ if and only if the chase of $I_{\beta(\mathbf{x},\mathbf{z})}$ with Σ gives at least $I_{\psi_{\mathbf{T}}(\mathbf{x},\mathbf{y}')}$ for a substitution where some $\mathbf{y}'$ substitutes for $\mathbf{y}$.

DEFINITION 4.3. The source formula $\beta(\mathbf{x}, \mathbf{z})$ is a *minimal generator of* $\exists \mathbf{y} \psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$ if $\beta(\mathbf{x}, \mathbf{z})$ is a generator of $\exists \mathbf{y} \psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$ and there is no $\beta'(\mathbf{x}, \mathbf{z})$ that is a conjunction of a strict subset of the conjuncts of $\beta(\mathbf{x}, \mathbf{z})$ such that $\beta'(\mathbf{x}, \mathbf{z})$ is a generator of $\exists \mathbf{y} \psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$.

We shall make use of the following simple lemma.

LEMMA 4.4. *Let Σ be a finite set of s-t tgds, each with at most s_1 conjuncts in its left-hand side. Let $\psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$ be a conjunction of s_2 target atoms. Then every minimal generator of $\exists \mathbf{y} \psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$ with respect to Σ has at most $s_1 s_2$ conjuncts.*

From Lemma 4.4, we see that there is a simple exhaustive-search algorithm for finding minimal generators:

Algorithm MinGen($\mathcal{M}, \exists \mathbf{y} \psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$)
Input: A schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a finite set of s-t tgds, and a formula $\exists \mathbf{y} \psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$, where $\psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$ is a conjunction of target atoms, and where the variables in $\mathbf{x}, \mathbf{y}$ are all distinct, and all appear in $\psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$.
Output: A finite set of the minimal generators of $\exists \mathbf{y} \psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$ with respect to Σ.

1. (*Initialization.*)
   Initialize the set G of minimal generators of $\exists \mathbf{y} \psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$ to the empty set.
2. (*Exhaustive search.*)
   Let s_1 and s_2 be as in Lemma 4.4. Systematically check every conjunction $\beta(\mathbf{x}, \mathbf{z})$ (up to renaming of variables in $\mathbf{z}$) of at most $s_1 s_2$ atoms where the variables in $\mathbf{z}$ are distinct and distinct from members of $\mathbf{x}, \mathbf{y}$, to see if the chase of $I_{\beta(\mathbf{x},\mathbf{z})}$ with Σ gives at least $I_{\psi_{\mathbf{T}}(\mathbf{x},\mathbf{y}')}$ for a substitution where some $\mathbf{y}'$ substitutes for $\mathbf{y}$. If so, add $\beta(\mathbf{x}, \mathbf{z})$ to G.
3. (*Minimize.*)
   For each member $\beta(\mathbf{x}, \mathbf{z})$ of G, check to see if there is some other $\beta'(\mathbf{x}, \mathbf{z}')$ in G whose conjuncts are a subset of the conjuncts of $\beta(\mathbf{x}, \mathbf{z})$ (up to renaming of variables in $\mathbf{z}, \mathbf{z}'$). If so, remove $\beta(\mathbf{x}, \mathbf{z})$ from G. Continue the process until there is no more change in G.
   Return G. □

The next algorithm produces a finite set of disjunctive tgds with constants and inequalities that defines a quasi-inverse if one exists.

Algorithm QuasiInverse($\mathcal{M}$)
Input: A schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a finite set of s-t tgds.
Output: A schema mapping $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$, where Σ' is a finite set of disjunctive tgds with constants and inequalities, that is a quasi-inverse of $\mathcal{M}$ if $\mathcal{M}$ has a quasi-inverse.

1. (*Create* Σ^*.)
   Create Σ^* from Σ as defined earlier.
2. (*Create the formulas* σ'.)
   For each member σ of Σ^*, create σ' as follows. Assume that σ is is $\phi_{\mathbf{S}}(\mathbf{x}, \mathbf{u}) \rightarrow \exists \mathbf{y} \psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$, where the variables in $\mathbf{x}$ are distinct, and consist exactly of the variables that appear in both $\phi_{\mathbf{S}}(\mathbf{x}, \mathbf{u})$ and $\psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$. The left-hand side of σ' is the conjunction of $\psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$, along with each of the formulas *Constant*(x) for members x of $\mathbf{x}$, along with the formulas $x_i \neq x_j$ for each pair x_i, x_j of distinct variables in $\mathbf{x}$. For each formula $\beta(\mathbf{x}, \mathbf{z})$ in the output of MinGen($\mathcal{M}, \exists \mathbf{y} \psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$), let $\exists \mathbf{z} \beta(\mathbf{x}, \mathbf{z})$ be a disjunct in the right-hand side of σ'.
3. (*Construct* Σ'.)
   Let Σ' consist of each of these formulas σ'.
   Return $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$. □

We prove in the full paper that this algorithm defines a quasi-inverse of $\mathcal{M}$ if one exists. Note that the disjunction in the right-hand side that is created in Step (2) of the algorithm is nonempty, since $\phi_{\mathbf{S}}(\mathbf{x}, \mathbf{u})$, the left-hand side of σ, is a generator of $\exists \mathbf{y} \psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$, and so some subset of the conjunctions of $\phi_{\mathbf{S}}(\mathbf{x}, \mathbf{u})$ forms a minimal generator.

EXAMPLE 4.5. Let Σ consist of the tgds:

$$P(x_1, x_2, x_3) \rightarrow \exists y(S(x_1, x_2, y) \wedge Q(y, y))$$
$$U(x_1) \rightarrow \exists y(S(x_1, x_1, y) \wedge Q(y, y) \wedge Q(x_1, y))$$
$$T(x_3, x_4) \rightarrow S(x_4, x_4, x_3)$$
$$R(x_1, x_2, x_4) \rightarrow Q(x_1, x_2).$$

Let σ_1 be the first tgd in Σ. Let σ_2 be

$$P(x_1, x_1, x_3) \rightarrow \exists y(S(x_1, x_1, y) \wedge Q(y, y)),$$

the result of replacing each occurrence of x_2 in σ_1 by x_1. Then σ_1 and σ_2 are both in Σ^*. To show Step (2) of the algorithm QuasiInverse, in this example we shall produce σ_1' from σ_1, and we shall produce σ_2' from σ_2. Thus, the algorithm puts σ_1' and σ_2' into Σ'.

The only generator of $\exists y(S(x_1, x_2, y) \wedge Q(y, y))$, the right-hand side of σ_1, is $P(x_1, x_2, x_3)$, so σ_1' is

$$S(x_1, x_2, y) \wedge Q(y, y) \wedge \mathit{Constant}(x_1) \wedge \mathit{Constant}(x_2) \wedge (x_1 \neq x_2) \rightarrow \exists x_3 P(x_1, x_2, x_3)$$

There are four minimal generators of $\exists y(S(x_1, x_1, y) \wedge Q(y, y))$, the right-hand side of σ_2. The first is $P(x_1, x_1, x_3)$, the left-hand side of σ_2. The second is $U(x_1)$, since its chase yields $S(x_1, x_1, y), Q(y, y), Q(x_1, y)$, which includes the conjuncts in the right-hand side of σ_2. The third is $T(x_1, x_1) \wedge R(x_1, x_1, x_4)$, since chasing the two facts in this conjunct yields $S(x_1, x_1, x_1), Q(x_1, x_1)$, where the role of y in the right-hand side of σ_2 is played by the variable x_1. The fourth is $T(x_3, x_1) \wedge R(x_3, x_3, x_4)$, since the chase of the two facts in this conjunct yields $S(x_1, x_1, x_3), Q(x_3, x_3)$, where the role of y in the right-hand side of σ_2 is played by the variable x_3. Then σ_2' is:

$$S(x_1, x_1, y) \wedge Q(y, y) \wedge \mathit{Constant}(x_1) \rightarrow \exists x_3 P(x_1, x_1, x_3) \vee U(x_1) \vee \exists x_4(T(x_1, x_1) \wedge R(x_1, x_1, x_4)) \vee \exists x_3 \exists x_4(T(x_3, x_1) \wedge R(x_3, x_3, x_4))$$

Note that the fourth disjunct is implied by the third disjunct (by letting the role of x_3 be played by x_1). So the third disjunct could be removed, since we need only keep the more general disjunct. □

The next theorem asserts that the language of quasi-inverses is slightly simplified in the case of full s-t tgds.

THEOREM 4.6. *Let $\mathcal{M}$ be a schema mapping specified by a finite set of full s-t tgds. If $\mathcal{M}$ has a quasi-inverse, then $\mathcal{M}$ has a quasi-inverse specified by a finite set of disjunctive tgds with inequalities. Thus, constants are not needed.*

Proposition 3.11 tells us that every LAV schema mapping has a quasi-inverse. The next theorem asserts that disjunctions are not needed in the language of quasi-inverses of LAV schema mappings.

THEOREM 4.7. *Every LAV schema mapping has a quasi-inverse specified by a finite set of tgds with constants and inequalities. Thus, disjunctions are not needed.*

4.1 Necessity of the Language

In this section, we exhibit the schema mappings used to prove Part (3) of Theorem 4.1, which says that constants, inequalities, disjunctions, and existential quantifiers are needed in general to express a quasi-inverse. We shall take advantage of Proposition 3.9 to turn results about inverses into results about quasi-inverses. We shall also show the optimality of Theorems 4.6 and 4.7.

THEOREM 4.8. *(Necessity of constants.) There is a LAV schema mapping that has an inverse, but no inverse specified by a set of disjunctive tgds with inequalities.*

PROOF. *(Hint)* Let $\mathbf{S}$ consist of a binary relation symbol P, and let $\mathbf{T}$ consist of a binary relation symbol Q. Let Σ consist of the tgd $P(x, y) \rightarrow \exists z(Q(x, z) \wedge Q(z, y))$. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$. Let Σ' consist of the following tgd with constants:

$$Q(x, z) \wedge Q(z, y) \wedge \mathit{Constant}(x) \wedge \mathit{Constant}(y) \rightarrow P(x, y).$$

Let $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$. It is shown in the full paper that $\mathcal{M}'$ is an inverse of $\mathcal{M}$, but $\mathcal{M}$ has no inverse specified by a set of disjunctive tgds with inequalities (but no constants). □

THEOREM 4.9. *(Necessity of inequalities.) There is a LAV schema mapping specified by full s-t tgds that has an inverse, but no inverse specified by a set of disjunctive tgds with constants.*

PROOF. *(Hint)* Let $\mathbf{S}$ consist of the binary relation symbol P and the unary relation symbol T. Let $\mathbf{T}$ consist of the binary relation symbol P' and the unary relation symbols Q and T'. Let Σ consist of the tgds $P(x, y) \rightarrow P'(x, y)$, $P(x, x) \rightarrow Q(x)$. $T(x) \rightarrow T'(x)$, $T(x) \rightarrow P'(x, x)$. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$. It is shown in the full paper that $\mathcal{M}$ has an inverse, but no inverse specified by a set of disjunctive tgds with constants. □

THEOREM 4.10. *(Necessity of disjunctions.) There is a schema mapping specified by a finite set of full s-t tgds that has a quasi-inverse, but has no quasi-inverse specified by a set of tgds with constants and inequalities.*

PROOF. *(Hint)* Let $\mathbf{S}$ consist of four unary relation symbols P_1, P_2, P_3, P_4. and let $\mathbf{T}$ consist of six unary relation symbols S_1, S_2, R_{13}, R_{14}, R_{23}, R_{24}. Let Σ consist of the tgds $P_1(x) \rightarrow S_1(x)$, $P_2(x) \rightarrow S_1(x)$, $P_3(x) \rightarrow S_2(x)$, $P_4(x) \rightarrow S_2(x)$, along with the four tgds $P_i(x) \wedge P_j(x) \rightarrow R_{ij}(x)$, for $i \in \{1, 2\}$ and $j \in \{3, 4\}$. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$. It is shown in the full paper that $\mathcal{M}$ has a quasi-inverse, but no quasi-inverse specified by a set of tgds with constants and inequalities. □

THEOREM 4.11. *(Necessity of existential quantifiers.) There is a LAV schema mapping specified by full s-t tgds that has a quasi-inverse, but no quasi-inverse specified by a set of full disjunctive tgds with constants and inequalities.*

PROOF. *(Hint)* Let $\mathbf{S}$ consist of a single binary relation symbol P, and let $\mathbf{T}$ consist of two unary relation symbols R and S. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ where Σ consists of the tgds $P(x, y) \rightarrow R(x), P(x, x) \rightarrow S(x)$. Since $\mathcal{M}$ is a LAV mapping, it has a quasi-inverse by Proposition 3.11. It is shown in the full paper that $\mathcal{M}$ has no quasi-inverse that is specified by a set of full disjunctive tgds with constants and inequalities. □

Part (3) of Theorem 4.1 follows from Theorems 4.8, 4.9, 4.10, and 4.11, along with Proposition 3.9. Note that Theorems 4.9, 4.10, and 4.11 (along with Proposition 3.9) tell us that the result of Theorem 4.6 is optimal, in that inequalities, disjunctions, and existential quantifiers are needed in general to specify a quasi-inverse of a schema mapping specified by a finite set of full s-t tgds. Similarly, Theorems 4.8, 4.9, and 4.11 (along with Proposition 3.9) tell us that the result of Theorem 4.7 is optimal, in that constants, inequalities, and existential quantifiers are needed in general to specify a quasi-inverse of a LAV schema mapping.

5. THE LANGUAGE OF INVERSES

The focus in [3] is on inverses that are specified by a finite set of tgds. For example, given a schema mapping $\mathcal{M}$ specified by a finite set of s-t tgds, [3] gives an algorithm for constructing a schema mapping specified by finite set of tgds that is an inverse of $\mathcal{M}$ if and only if there

is an inverse of $\mathcal{M}$ that is specified by a finite set of tgds. If there is an inverse $\mathcal{M}'$ but there is no inverse specified by a finite set of tgds, then the algorithm in [3] will not find $\mathcal{M}'$. The "language of inverses" is left as an open problem in [3]. This is the question as to what language is needed to specify the inverse of $\mathcal{M}$, when $\mathcal{M}$ is specified by a finite set of s-t tgds. The next theorem resolves this open problem.

THEOREM 5.1. *Let $\mathcal{M}$ be a schema mapping specified by a finite set of s-t tgds. If $\mathcal{M}$ has an inverse then the following hold.*

1. *$\mathcal{M}$ has an inverse $\mathcal{M}'$ specified by a finite set of full tgds with constants and inequalities.*
2. *There is an exponential-time algorithm for producing $\mathcal{M}'$.*
3. *Statement (1) is not necessarily true if we disallow either constants or inequalities in the left-hand side, even if we allow existential quantifiers in the right-hand side (and so allow non-full dependencies to specify $\mathcal{M}'$).*

In fact, the inverse $\mathcal{M}'$ that the algorithm produces has inequalities only among constants.

Part (3) of Theorem 5.1 follows from Theorems 4.8 and 4.9. When $\mathcal{M}$ is a schema mapping specified by a finite set of *full* s-t tgds, we show in the full paper that constants are no longer needed, although Theorem 4.9 tells us that inequalities are still needed.

We now discuss the machinery used to prove Theorem 5.1.

DEFINITION 5.2. A schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a finite set of s-t tgds, satisfies the *constant-propagation property* if for every ground instance I, every member of the active domain of I is in the active domain of $chase_\Sigma(I)$.

It is straightforward to see that $\mathcal{M}$ satisfies the constant-propagation property precisely if, for each relation symbol R in $\mathbf{S}$, the chase of $R(x_1, \ldots, x_m)$ with Σ includes each of the m distinct variables x_1, $\ldots$, x_m, where m is the arity of R.

We shall use the following proposition from [3].

PROPOSITION 5.3. *[3] Every invertible schema mapping that is specified by a finite set of s-t tgds satisfies the constant-propagation property.*

Define a *prime atom* to be one that contains precisely the variables $x_1, x_2, \ldots, x_k$ for some k, and where the initial appearance of x_i precedes the initial appearance of x_j if $i < j$. For example, $P(x_1, x_2, x_1, x_3, x_2)$ is a prime atom, but $Q(x_2, x_1)$ and $R(x_2, x_3)$ are not. Note that for every atom, there is a unique renaming of variables to obtain a prime atom. Define a *prime instance* to be an instance whose only fact is a single prime atom. As with our definition of I_α, a prime instance is not an instance in the usual sense, but is a type of canonical instance. We now give an algorithm that produces an inverse if one exists.

Algorithm Inverse($\mathcal{M}$)
Input: A schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a finite set of s-t tgds.
Output: A schema mapping $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$, where Σ' is a finite set of full tgds with constants and inequalities, and $\mathcal{M}'$ is an inverse of $\mathcal{M}$ if $\mathcal{M}$ has an inverse. There is no output if $\mathcal{M}$ does not satisfy the constant-propagation property.

1. (*Verify that $\mathcal{M}$ satisfies the constant-propagation property.*)
   Check to see if, for each relation symbol R in $\mathbf{S}$, the chase of $R(x_1, \ldots, x_m)$ with Σ includes each of the m distinct variables $x_1, \ldots, x_m$, where m is the arity of R. If not, halt without output. If so, continue to the next step.
2. (*Generate all prime source atoms in lexicographic order.*)
   For example, if R is a ternary source relation symbol, the atoms for R, in lexicographic order, are $R(x_1, x_1, x_1)$, $R(x_1, x_1, x_2)$, $R(x_1, x_2, x_1)$, $R(x_1, x_2, x_2)$, $R(x_1, x_2, x_3)$.
3. (*Construct a full tgd $\omega(\Sigma, I)$ for each prime instance I.*)
   For each prime source atom α generated in Step (1), let I_α be the prime instance containing only α. Let ψ_α be the conjunction of the facts of $chase_\Sigma(I_\alpha)$. Form a full tgd $\omega(\Sigma, I_\alpha)$ whose left-hand side is the conjunction of ψ_α with the formulas *Constant*(x) for each variable x that appears in α, along with inequalities $x_i \neq x_j$ for each pair x_i, x_j of distinct variables that appear in α, and whose right-hand side is α.
4. (*Construct Σ'.*)
   Let Σ' consist of each of these formulas $\omega(\Sigma, I)$, one for each prime instance I.
   Return $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$. □

Assume that $\mathcal{M}$ satisfies the constant-propagation property. Then the algorithm gives an output. Furthermore, $\omega(\Sigma, I_\alpha)$, as formed in Step (3), is then a well-defined full tgd with constants and inequalities, since every variable in the right-hand side of $\omega(\Sigma, I_\alpha)$ necessarily appears in the left-hand side.

EXAMPLE 5.4. Let $\mathbf{S}$ consist of a binary relation symbol R. Let $\mathbf{T}$ consist of a binary relation symbol Q, ternary relation symbol S, and unary relation symbol U. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ where Σ consists of the tgds:

$$R(x_1, x_2) \wedge R(x_2, x_1) \rightarrow \exists y Q(x_1, y)$$
$$R(x_1, x_2) \rightarrow \exists y S(x_1, x_2, y)$$
$$R(x_1, x_1) \rightarrow U(x_1)$$

Then $\mathcal{M}$ satisfies the constant-propagation property, since the chase of $R(x_1, x_2)$ is $S(x_1, x_2, y)$, which contains both of the variables x_1 and x_2 of $R(x_1, x_2)$. The two prime source atoms are $R(x_1, x_1)$ and $R(x_1, x_2)$. The two prime instances are $I_{R(x_1,x_1)} = \{R(x_1, x_1)\}$ and $I_{R(x_1,x_2)} = \{R(x_1, x_2)\}$. The tgd $\omega(\Sigma, I_{R(x_1,x_1)})$ is

$$Q(x_1, y_1) \wedge S(x_1, x_1, y_2) \wedge U(x_1) \wedge \mathit{Constant}(x_1) \rightarrow R(x_1, x_1) \quad (1)$$

The tgd $\omega(\Sigma, I_{R(x_1,x_2)})$ is

$$S(x_1, x_2, y) \wedge \mathit{Constant}(x_1) \wedge \mathit{Constant}(x_2) \wedge (x_1 \neq x_2) \rightarrow R(x_1, x_2) \quad (2)$$

The output of Inverse($\mathcal{M}$) is $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$, where Σ' consists of (1) and (2). □

We show in the full paper that if $\mathcal{M}$ is invertible, then the output $\mathcal{M}'$ of the algorithm is an inverse of $\mathcal{M}$. Also, we show that $\mathcal{M}'$ is the most general (or "weakest") inverse, in the sense that if $\mathcal{M}'' = (\mathbf{T}, \mathbf{S}, \Sigma'')$ is another inverse of $\mathcal{M}$, then Σ'' logically implies Σ'.

Proposition 3.9 tells us that every quasi-inverse of an invertible schema mapping $\mathcal{M}$ is an inverse of $\mathcal{M}$. The reader might therefore wonder why we need both the algorithms QuasiInverse and Inverse, since the QuasiInverse algorithm will necessarily produce an inverse if the input is an invertible schema mapping. The answer is that in this case, the QuasiInverse algorithm will produce an inverse specified by disjunctive tgds with constants and equalities where disjunctions may actually appear, even though there is an inverse specified by full (and non-disjunctive) tgds with constants and equalities that the Inverse algorithm will find (an example appears in the full paper).

6. QUASI-INVERSES IN DATA EXCHANGE

Next, we shall describe two desirable properties that an "inverse" should possess for data exchange. Here, we use the term "inverse" loosely, to mean a schema mapping $\mathcal{M}'$ that goes in the reverse direction of $\mathcal{M}$. We will then show that quasi-inverses have the two properties.

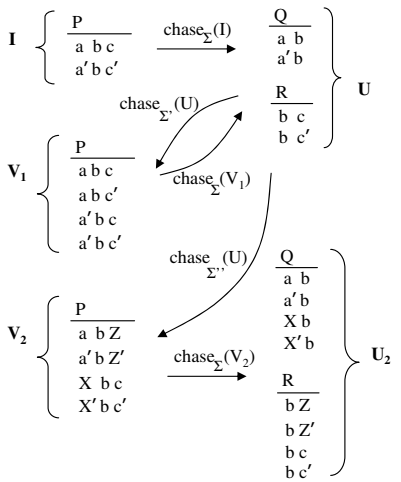

Figure 1: $\mathcal{M}'$ and $\mathcal{M}''$ are faithful with respect to $\mathcal{M}$.

First, it is desirable for an inverse to be *sound*. Specifically, assume that $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is a schema mapping where Σ is a finite set of s-t tgds, and assume that $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$ is an "inverse" schema mapping. For the moment, assume that Σ' is given by a finite set of tgds. Suppose that we perform data exchange with $\mathcal{M}$, by chasing a ground instance I with Σ, to obtain a target instance U, denoted by $U = \mathit{chase}_\Sigma(I)$. We can then perform a reverse data exchange from U with $\mathcal{M}'$ and obtain V (i.e., compute $V = \mathit{chase}_{\Sigma'}(U)$). Then $\mathcal{M}'$ is *sound* with respect to $\mathcal{M}$ if the following holds for every choice of ground instance I: When we redo the original exchange with Σ but this time starting from V, we obtain a *subset* of the facts that are in U (modulo homomorphic images of nulls). Intuitively, the result of the reverse data exchange with $\mathcal{M}'$, followed by a data exchange with $\mathcal{M}$ (i.e., $\mathit{chase}_\Sigma(V)$), does not introduce any new information that cannot be found in U. If, additionally, all the data in U can be embedded homomorphically into $\mathit{chase}_\Sigma(V)$, then no information that is in U has been lost. We then say that $\mathcal{M}'$ is *faithful* with respect to $\mathcal{M}$.

EXAMPLE 6.1. Let us revisit the earlier Decomposition example with a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ where Σ consists of the following s-t tgd:

$$P(x, y, z) \rightarrow Q(x, y) \wedge R(y, z).$$

Let us recall, from Example 3.10, that $\mathcal{M}$ has quasi-inverses $\mathcal{M}'$ and $\mathcal{M}''$ specified by the following sets Σ' and Σ'' of tgds:

$$\begin{aligned} \Sigma' &= \{\ Q(x, y) \wedge R(y, z) \rightarrow P(x, y, z)\ \} \\ \Sigma'' &= \{\ Q(x, y) \rightarrow \exists z P(x, y, z), \\ &\qquad R(y, z) \rightarrow \exists x P(x, y, z)\ \} \end{aligned}$$

Let I be the ground instance shown in Figure 1. The result of chasing I with Σ (i.e., the result of the data exchange with $\mathcal{M}$) is the instance U shown in the figure. If we now chase U with Σ' (i.e., perform the reverse data exchange with $\mathcal{M}'$), we obtain the source instance V_1. Furthermore, if we now redo the original data exchange with $\mathcal{M}$ starting from V_1, the result is identical to U. In fact, it can be shown that, for every ground instance I, the result of redoing the original data exchange on V_1 is identical to U. Hence, $\mathcal{M}'$ is faithful with respect to $\mathcal{M}$.

Consider now $\mathcal{M}''$. Again, let U be the result of the first data exchange on I with $\mathcal{M}$. Let V_2 be obtained, as in the figure, by a reverse data exchange with $\mathcal{M}''$ from U. If we now redo the original data exchange with $\mathcal{M}$ starting from V_2, the result is the instance U_2. The instance U_2 is different from the target instance U because U_2 contains extra tuples with nulls. The two instances U and U_2, however, are homomorphically equivalent. It can be shown that this is true for every ground instance I, and therefore $\mathcal{M}''$ is faithful with respect to $\mathcal{M}$.

It turns out that it is not an accident that $\mathcal{M}$ has faithful quasi-inverses. In this section, we show that if $\mathcal{M}$ is a schema mapping that is specified by a finite set of s-t tgds and has a quasi-inverse, then $\mathcal{M}$ is guaranteed to have a faithful quasi-inverse (and the algorithm QuasiInverse produces one). □

Note that nulls may arise when we chase I with a schema mapping $\mathcal{M}$, and also when we chase the result U with an "inverse" $\mathcal{M}'$. In particular, the result of the reverse data exchange may not necessarily be a ground instance, but rather a source instance with nulls. However, if the inverse is faithful, these nulls do not matter: when we redo the data exchange with $\mathcal{M}$, we obtain a target instance that is homomorphically equivalent to the original result U.

In order to define soundness and faithfulness in the general case, when $\mathcal{M}'$ is expressed by a set of disjunctive tgds with constants and inequalities, we need to consider an extension of the chase for this more general language. The standard notion of the chase can be easily extended to handle the *Constant* predicate and the inequalities in the left-hand side of the tgds in Σ'. However, when the right-hand side of a tgd in Σ' contains disjunction, we need to use the *disjunctive chase*. Chasing with disjunctive dependencies has been considered before in various contexts [2, 4]; we use a similar notion here, which we make precise via the following three definitions. When defining the disjunctive chase, we do not need to assume a separation into a source and a target schema. However, the subsequent definitions and results about soundness and faithfulness will apply the disjunctive chase in the context where such separation exists.

DEFINITION 6.2. *Let $\phi(\mathbf{x})$ be a conjunction of atoms that may include constants and inequalities as in Definition 2.1. Let K be an instance over* Const ∪ Var. *A* homomorphism h from $\phi(\mathbf{x})$ to K *is a mapping from the variables $\mathbf{x}$ to values in* Const ∪ Var *such that: (1) for every atom $T(x_1, \ldots, x_k)$ in ϕ we have that $T(h(x_1), \ldots, h(x_k))$ is a fact in K, (2) for every inequality $x_i \neq x_j$ in ϕ, we have that $h(x_i) \neq h(x_j)$, and, (3) for every formula Constant(x) in ϕ, we have that $h(x)$ is in* Const.

DEFINITION 6.3 (DISJUNCTIVE CHASE STEP). *Let σ be a disjunctive tgd with constants and inequalities of the form:*

$$\forall \mathbf{x}[\phi(\mathbf{x}) \rightarrow (\exists \mathbf{y_1} \psi_1(\mathbf{x_1}, \mathbf{y_1}) \vee \ldots \vee \exists \mathbf{y_p} \psi_p(\mathbf{x_p}, \mathbf{y_p}))].$$

Let σ_i be the tgd with constants and inequalities that is obtained from σ by taking just one disjunct:

$$\forall \mathbf{x}[\phi(\mathbf{x}) \rightarrow (\exists \mathbf{y_i} \psi_i(\mathbf{x_i}, \mathbf{y_i}))]$$

Let K be an instance over Const ∪ Var. *Assume that h is a homomorphism from $\phi(\mathbf{x})$ to K such that for each $i \in \{1, \ldots, p\}$, there is no extension of h to a homomorphism from $\phi(\mathbf{x}) \wedge \psi_i(\mathbf{x_i}, \mathbf{y_i})$ to K. We say that σ* can be applied to K with homomorphism h. *Note that this also means that σ_i can be be applied to K with homomorphism h (this is the non-disjunctive definition of a chase step).*

Let $K_1, \ldots, K_p$ be the instances that result by applying each of $\sigma_1, \ldots, \sigma_p$ to K with homomorphism h. We say that the result of applying σ to K *is the set $\{K_1, \ldots, K_p\}$, and write $K \xrightarrow{\sigma,h} \{K_1, \ldots, K_p\}$.*

DEFINITION 6.4 (DISJUNCTIVE CHASE). *Let Σ be a finite set of disjunctive tgds with constants and inequalities. The* disjunctive chase of an instance K with Σ *is a tree (finite or infinite) that has K as a root and for each node K', if K' has children $K_1, \ldots, K_p$, then it must be the case that $K' \xrightarrow{\sigma,h} \{K_1, \ldots, K_p\}$ for some σ in Σ and some homomorphism h. Moreover, each leaf K_m in the tree has the requirement that there is no σ in Σ and no homomorphism h such that*

σ can be applied to K with h. When the chase tree is finite we say that the result of the disjunctive chase of K with Σ *is the set of leaves in the chase tree.*

In the case when the disjunctive tgds are from a schema $\mathbf{T}$ to a schema $\mathbf{S}$, we can chase instances of the form (J, I) where J is a $\mathbf{T}$-instance and I is an $\mathbf{S}$-instance. Note that any such chase tree will be finite (since there is no recursion). Our case of interest is applying the disjunctive chase to an instance of the form $(U, \emptyset)$ where $U = \mathit{chase}_\Sigma(I)$, for some ground instance I. The result of such chase is a set $\{(U, V_1), \ldots, (U, V_m)\}$ of instances where $V_1, \ldots, V_m$ are $\mathbf{S}$-instances. If $\mathcal{V}$ denotes the set $\{V_1, \ldots, V_m\}$, we shall also say that $\mathcal{V}$ is the result of chasing U with Σ' and write $\mathcal{V} = \mathit{chase}_{\Sigma'}(U)$. Furthermore, let us denote by $\mathcal{U}' = \mathit{chase}_\Sigma(\mathcal{V})$ the set of all instances U' that are obtained by chasing, in the standard way, each member V of $\mathcal{V}$ with Σ.

DEFINITION 6.5. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \mathbf{\Sigma})$ be a schema mapping where Σ is a finite set of s-t tgds, and let $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \mathbf{\Sigma}')$ be a schema mapping where Σ' is a finite set of disjunctive tgds with constants and inequalities.*

(1) We say that $\mathcal{M}'$ is sound *with respect to $\mathcal{M}$ if:*

for every ground instance I over $\mathbf{S}$, if $U = \mathit{chase}_\Sigma(I)$, $\mathcal{V} = \mathit{chase}_{\Sigma'}(U)$ and $\mathcal{U}' = \mathit{chase}_\Sigma(\mathcal{V})$, then there is a homomorphism from some member of $\mathcal{U}'$ into U.

(2) We say that $\mathcal{M}'$ is faithful *with respect to $\mathcal{M}$ if:*

for every ground instance I over $\mathbf{S}$, if $U = \mathit{chase}_\Sigma(I)$, $\mathcal{V} = \mathit{chase}_{\Sigma'}(U)$ and $\mathcal{U}' = \mathit{chase}_\Sigma(\mathcal{V})$, then there is some member of $\mathcal{U}'$ that is homomorphically equivalent to U.

Regarding the above definition, note that in the case when the dependencies in Σ' have no disjunction, the set $\mathcal{V}$ of source instances becomes a singleton set. Thus, if $\mathcal{M}'$ is faithful, chasing with Σ' recovers a single source instance whose chase (with Σ) is homomorphically equivalent to U. In fact, even when the dependencies in Σ' have disjunction, if $\mathcal{M}'$ is faithful, we can still recover a single source instance: we search among the instances in $\mathcal{V}$ to find the source instance whose chase (with Σ) is homomorphically equivalent to U.

The following proposition states an important property of the disjunctive chase in the context of bidirectional data exchange, when the disjunctive tgds (with constants and inequalities among constants) are part of the "reverse" mapping. The proof of this proposition, which is essential in proving the two main theorems of this section, will be given in the full version of this paper.

PROPOSITION 6.6. [Universality of "chase of the chase"] *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping where Σ is a finite set of s-t tgds and let $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$ be a schema mapping where Σ' is a finite set of disjunctive tgds with constants and inequalities among constants. Moreover, let I be a ground instance over $\mathbf{S}$. If $U = \mathit{chase}_\Sigma(I)$ and $\mathcal{V} = \mathit{chase}_{\Sigma'}(U)$ then for every K such that $(I, K) \in \mathrm{Inst}(\mathcal{M} \circ \mathcal{M}')$, there is $V \in \mathcal{V}$ such that there is a homomorphism from V to K.*

The next theorem shows that every quasi-inverse specified by disjunctive tgds with constants and inequalities among constants is sound. We have shown earlier that this language is sufficient to express quasi-inverses of schema mappings that are specified by s-t tgds. The second theorem states that, furthermore, every quasi-inverse obtained by applying the QuasiInverse algorithm is faithful. The proofs of these two results will be given in the full paper.

THEOREM 6.7. *Let $\mathcal{M}$ be a schema mapping specified by a finite set of s-t tgds. If $\mathcal{M}'$ is a quasi-inverse of $\mathcal{M}$ that is specified by a finite set of disjunctive tgds with constants and inequalities among constants, then $\mathcal{M}'$ is sound with respect to $\mathcal{M}$.*

THEOREM 6.8. *Let $\mathcal{M}$ be a schema mapping specified by a finite set of s-t tgds. If $\mathcal{M}$ has a quasi-inverse, then the schema mapping obtained by applying the algorithm* QuasiInverse *on $\mathcal{M}$ is faithful with respect to $\mathcal{M}$.*

7. CONCLUDING REMARKS

The notion of an inverse of a schema mapping is rather restrictive, since it is rare that a schema mapping has an inverse. We therefore introduced and studied a more relaxed notion of a quasi-inverse of a schema mapping. Both inverses and quasi-inverses are special cases of a unifying framework for inverses that we developed. We gave an exact criterion for the existence of quasi-inverses, complete characterizations of the languages needed to express quasi-inverses and inverses, and results regarding the use of quasi-inverses in data exchange.

Some of the important remaining problems are decision and complexity issues. We have shown that for LAV schema mappings, a quasi-inverse always exists. However, the complexity of the decision problem for the existence of a quasi-inverse of a schema mapping specified by a finite set of s-t tgds (even in the full case) remains open. We do not know whether the problem is even decidable. Similarly, the complexity of the decision problem for the existence of an inverse of a schema mapping specified by a finite set of s-t tgds (even in the full case) remains open. Again, we do not know whether the problem is even decidable. Another open problem concerns the optimality of the algorithms QuasiInverse and Inverse. Given a schema mapping specified by a finite set of s-t tgds, these algorithms produce a schema mapping that is exponential in the size of the input schema mapping. We do not know whether the size of a quasi-inverse is necessarily exponential, and similarly for an inverse. If it turns out that there is always a polynomial-size quasi-inverse, this raises the question of finding a polynomial-time algorithm that can produce it. Similarly, the same question arises for inverses.

8. REFERENCES

[1] P. A. Bernstein. Applying Model Management to Classical Meta-Data Problems. In *Conference on Innovative Data Systems Research (CIDR)*, pages 209–220, 2003.

[2] A. Deutsch and V. Tannen. Optimization Properties for Classes of Conjunctive Regular Path Queries. In *International Workshop on Database Programming Languages (DBPL)*, pages 21–39, 2001.

[3] R. Fagin. Inverting Schema Mappings. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 50–59, 2006. To appear, *ACM Trans. on Database Systems (TODS)*.

[4] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. *Theoretical Computer Science (TCS)*, 336(1):89–124, 2005.

[5] R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. Composing Schema Mappings: Second-order Dependencies to the Rescue. *ACM Transactions on Database Systems (TODS)*, 30(4):994–1055, 2005.

[6] P. G. Kolaitis. Schema Mappings, Data Exchange, and Metadata Management. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 61–75, 2005.

[7] M. Lenzerini. Data Integration: A Theoretical Perspective. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 233–246, 2002.

[8] J. Madhavan and A. Y. Halevy. Composing Mappings Among Data Sources. In *International Conference on Very Large Data Bases (VLDB)*, pages 572–583, 2003.

[9] S. Melnik. *Generic Model Management: Concepts and Algorithms*, volume 2967 of *Lecture Notes in Computer Science*. Springer, 2004.

[10] A. Nash, P. A. Bernstein, and S. Melnik. Composition of Mappings Given by Embedded Dependencies. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 172–183, 2005.