The Recovery of a Schema Mapping: Bringing Exchanged Data Back

Marcelo Arenas
PUC Chile
marenas@ing.puc.cl

Jorge Pérez
PUC Chile
jperez@ing.puc.cl

Cristian Riveros
PUC Chile
crj@ing.puc.cl

ABSTRACT

A schema mapping is a specification that describes how data from a source schema is to be mapped to a target schema. Once the data has been transferred from the source to the target, a natural question is whether one can undo the process and recover the initial data, or at least part of it. In fact, it would be desirable to find a *reverse* schema mapping from target to source that specifies how to bring the exchanged data back.

In this paper, we introduce the notion of a recovery of a schema mapping: it is a reverse mapping $\mathcal{M}'$ for a mapping $\mathcal{M}$ that recovers sound data with respect to $\mathcal{M}$. We further introduce an order relation on recoveries. This allows us to choose mappings that recover the maximum amount of sound information. We call such mappings maximum recoveries. We study maximum recoveries in detail, providing a necessary and sufficient condition for their existence. In particular, we prove that maximum recoveries exist for the class of mappings specified by FO-TO-CQ source-to-target dependencies. This class subsumes the class of source-to-target tuple-generating dependencies used in previous work on data exchange. For the class of mappings specified by FO-TO-CQ dependencies, we provide an exponential-time algorithm for computing maximum recoveries, and a simplified version for full dependencies that works in quadratic time. We also characterize the language needed to express maximum recoveries, and we include a detailed comparison with the notion of inverse (and quasi-inverse) mapping previously proposed in the data exchange literature. In particular, we show that maximum recoveries strictly generalize inverses. We study the complexity of some decision problems related to the notions of recovery and maximum recovery. Finally, we report our initial results about a relaxed notion of maximal recovery, showing that it strictly generalizes the notion of maximum recovery.

Categories and Subject Descriptors
H.2.5 [**Heterogeneous Databases**]: Data translation

General Terms
Algorithms, Theory

Keywords
Metadata management, schema mapping, data exchange, data integration, inverse, recovery, maximum recovery

PODS'08, June 9–12, 2008, Vancouver, BC, Canada.

1. INTRODUCTION

A schema mapping is a specification that describes how data from a source schema is to be mapped to a target schema. In the last years, a lot of attention has been paid to the development of solid foundations for the problem of exchanging data using schema mappings [8, 14, 11]. These developments are a first step towards providing a general framework for exchanging information, but they are definitely not the last one. As pointed out by Bernstein [3], many information system problems involve not only the design and integration of complex application artifacts, but also their subsequent manipulation. This has motivated the need for the development of a general infrastructure for managing schema mappings.

A framework for managing schema mappings, called model management, was proposed by Bernstein in [3]. In this framework, operators like match, merge and compose are used to manipulate mappings [3, 15, 16]. Another important operator that naturally arises in this context is the inverse, which plays an important role in schema evolution [4]. Once the data has been transferred from the source to the target, the goal of the inverse is to recover the initial source data.

If a mapping $\mathcal{M}'$ is an inverse of a mapping $\mathcal{M}$, then $\mathcal{M}'$ is an ideal mapping to bring the data exchanged through $\mathcal{M}$ back to the source. Unfortunately, the process of inverting schema mappings turned out to be a nontrivial task [7, 10]. In [7], Fagin proposes a first formal definition for what it means for a schema mapping $\mathcal{M}'$ to be an inverse of a schema mapping $\mathcal{M}$. Roughly speaking, Fagin's definition is based on the idea that a mapping composed with its inverse should be equal to the *identity schema mapping*. More formally, Fagin introduces in [7] an identity schema mapping $\overline{\mathrm{Id}}$, suitably adapted for the case of mappings specified by source-to-target tuple-generating dependencies (st-tgds). Then he says that $\mathcal{M}'$ is an inverse of $\mathcal{M}$ if $\mathcal{M} \circ \mathcal{M}' = \overline{\mathrm{Id}}$. This notion turns out to be rather restrictive, as it is rare that a schema mapping possesses an inverse. In view of this limitation, in a subsequent work [10], Fagin et al. introduce the notion of a quasi-inverse of a schema mapping. The idea of the quasi-inverse is to relax the notion of inverse by not differentiating between source instances that are equivalent for data exchange purposes. Although numerous non-invertible schema mappings possess natural and useful quasi-inverses [10], there are still simple mappings specified by st-tgds that have no quasi-inverse. Moreover, the notions of inverse and quasi-inverse are defined by considering identity mapping $\overline{\mathrm{Id}}$, that is only appropriate for mappings that are closed down on the left [7] and, in particular, for mappings specified by st-tgds. This leaves out numerous mappings of practical interest.

In this paper, we revisit the problem of inverting schema mappings. Although our motivation is similar to that of previous work, we follow a different approach. In fact, our main goal is not to define a notion of inverse mapping, but instead to give a formal definition for what it means for a schema mapping $\mathcal{M}'$ to recover *sound* informa-

tion with respect to a schema mapping $\mathcal{M}$. We call such an $\mathcal{M}'$ a *recovery* of $\mathcal{M}$. In this paper, we use a general definition of schema mapping, where mappings are simply defined as binary relations with pairs (I, J), where I is a source instance and J is a target instance. Our notion of recovery is applicable to this general definition of mapping. Given that, in general, there may exist many possible recoveries for a mapping, we introduce an order relation on recoveries. This naturally gives rise to the notion of maximum recovery, which is a mapping that brings back the maximum amount of sound information.

As a motivating example, consider a database with relations *Emp*(*name, works_in, lives_in*) and *DrivesWork*(*name*), the former to store names of employees and the places where they work and live, and the latter to store the names of employees that drive to work. Assume that the information about employees has to be transferred to an independent database that contains relation *Shuttle*(*name*), that stores the names of employees that take a shuttle bus to go to work. A schema mapping $\mathcal{M}_{E\text{-}S}$ between these two databases is defined by the following dependency:

$$\textit{Emp}(x, y, z) \wedge y \neq z \wedge \neg \textit{DrivesWork}(x) \;\rightarrow\; \textit{Shuttle}(x). \qquad (1)$$

An example of a reverse mapping $\mathcal{M}_1$ that recovers sound information w.r.t. $\mathcal{M}_{E\text{-}S}$ is $\textit{Shuttle}(x) \rightarrow \exists u \exists v\, \textit{Emp}(x, u, v)$; it is correct to bring back to relation *Emp* every employee in relation *Shuttle*, but since *Shuttle* does not store information about the places where employees work and live, variables u and v are existentially quantified. Furthermore, it is also correct to assume that if an employee name has been brought back from relation *Shuttle*, then the places where this employee works and lives are different. Thus, mapping $\mathcal{M}_2$ defined by $\textit{Shuttle}(x) \rightarrow \exists u \exists v\, (\textit{Emp}(x, u, v) \wedge u \neq v)$ is also a correct way of recovering information w.r.t. $\mathcal{M}_{E\text{-}S}$. On the other hand, it is clear that mapping $\mathcal{M}_3$ defined by $\textit{Shuttle}(x) \rightarrow \exists u\, \textit{Emp}(x, u, u)$, is not a correct way of recovering information w.r.t. $\mathcal{M}_{E\text{-}S}$, since $\mathcal{M}_3$ assumes that in every recovered instance, every employee in relation *Shuttle* works and lives in the same place.

Formally, if $\mathcal{M}$ is a mapping from a source schema to a target schema and $\mathcal{M}'$ is a reverse mapping from target to source, then we say that $\mathcal{M}'$ is a recovery of $\mathcal{M}$ if for every source instance I, the space of solutions for I under the composition of mappings $\mathcal{M}$ and $\mathcal{M}'$ contains I itself. That is, I must be a possible solution for itself under mapping $\mathcal{M} \circ \mathcal{M}'$. Under this definition, mappings $\mathcal{M}_1$ and $\mathcal{M}_2$ above are recoveries of $\mathcal{M}_{E\text{-}S}$, while mapping $\mathcal{M}_3$ is not a recovery of $\mathcal{M}_{E\text{-}S}$.

Being a recovery is a sound but mild requirement. Then it would be desirable to have some criteria to compare alternative recoveries. In our motivating example, if one has to choose between $\mathcal{M}_1$ and $\mathcal{M}_2$ as a recovery of $\mathcal{M}$, then one would probably choose $\mathcal{M}_2$, since this mapping says not only that every employee that takes a shuttle bus works and lives in some place, but also that those places must be different. Intuitively, $\mathcal{M}_2$ is *more informative than* $\mathcal{M}_1$ w.r.t. $\mathcal{M}$. Furthermore, if $\mathcal{M}_4$ is a mapping defined by dependency:

$$\textit{Shuttle}(x) \;\rightarrow\; \exists u \exists v\, (\textit{Emp}(x, u, v) \wedge u \neq v \wedge \neg \textit{DrivesWork}(x)),$$

then $\mathcal{M}_4$ is a recovery of $\mathcal{M}_{E\text{-}S}$ that is more informative than $\mathcal{M}_2$; $\mathcal{M}_4$ additionally states that if an employee is brought back from relation *Shuttle*, then it is known that she/he does not drive to work. In general, if $\mathcal{M}'$ is a recovery of $\mathcal{M}$, then the smaller the space of solutions generated by the composition $\mathcal{M} \circ \mathcal{M}'$, the more informative $\mathcal{M}'$ is about the initial source instances. We formalize this notion by saying that $\mathcal{M}'$ is at least as informative as $\mathcal{M}''$ w.r.t. $\mathcal{M}$, if for every source instance I, the space of solutions for I under $\mathcal{M} \circ \mathcal{M}'$ is contained in the space of solutions for I under $\mathcal{M} \circ \mathcal{M}''$. This *order* on recoveries gives rise to a notion of maximum recovery. Going back to our example, it can be shown that mapping $\mathcal{M}_4$ is a maximum recovery of $\mathcal{M}_{E\text{-}S}$.

In this paper, we study the notions of recovery and maximum recovery. The following are our main technical contributions:

- For the general notion of schema mapping considered in this paper, we provide a necessary and sufficient condition for the existence of a maximum recovery. We use this condition to show that maximum recoveries are guaranteed to exist for a large class of schema mappings, namely for mappings specified by FO-TO-CQ source-to-target dependencies. An FO-TO-CQ dependency is a formula of the form $\forall \bar{x}(\varphi_{\mathbf{S}}(\bar{x}) \rightarrow \exists \bar{y}\, \psi_{\mathbf{T}}(\bar{x}, \bar{y}))$, where $\varphi_{\mathbf{S}}(\bar{x})$ is a first-order formula over the source schema and $\psi_{\mathbf{T}}(\bar{x}, \bar{y})$ is a conjunction of relational atoms over the target schema. Notice that every st-tgd is an FO-TO-CQ dependency. We further show that maximum recoveries exist even if we enrich the class of FO-TO-CQ dependencies with arbitrary source dependencies, equality-generating target dependencies and weakly acyclic sets of tuple-generating target dependencies.
- We provide a detailed comparison between the notions of inverse, quasi-inverse, and maximum recovery. Most notably, we show that for the class of mappings considered in [7, 10], if a mapping $\mathcal{M}$ is invertible, then $\mathcal{M}'$ is an inverse of $\mathcal{M}$ if and only if $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$. For this class of mappings, we also show that, if a mapping $\mathcal{M}$ is quasi-invertible, then $\mathcal{M}$ has a maximum recovery, and, furthermore, every maximum recovery of $\mathcal{M}$ is a quasi-inverse of $\mathcal{M}$.
- In the above example, a maximum recovery for mapping $\mathcal{M}_{E\text{-}S}$ is obtained just by "reversing the arrow" of dependency (1). However, in general the process of computing maximum recoveries is more involved. For mappings specified by FO-TO-CQ dependencies, we provide an exponential-time algorithm for computing maximum recoveries. For the case of full FO-TO-CQ dependencies, that is, dependencies that do not use existential quantifiers in their consequents, we provide a quadratic-time algorithm for computing maximum recoveries. It is worth mentioning that these algorithms can also be used for computing inverses and quasi-inverses. We also investigate the language needed to express maximum recoveries for mappings specified by FO-TO-CQ dependencies, providing justification for the dependency language used in the output of these algorithms.
- We study the complexity of some problems related to the notions of recovery and maximum recovery. We show that even for the case of st-tgds, testing whether a mapping $\mathcal{M}'$ is a recovery of a mapping $\mathcal{M}$ is undecidable. As a corollary, we obtain the same undecidability result for the notions of inverse, quasi-inverse, and maximum recovery. When restricted to full st-tgds, we prove lower complexity bounds for this problem.
- We also consider a relaxed notion of maximal recovery, and we present our initial results about this notion. In particular, we provide a necessary and sufficient condition for the existence of a maximal recovery, and we use this condition to prove that maximal recoveries are guaranteed to exist for the class of mappings specified by FO-TO-UCQ$^{\neq}$ dependencies (the extension of FO-TO-CQ with disjunctions and inequalities in the consequents).

Organization of the paper. In Section 2, we introduce the terminology used in the paper. In Section 3, we define the notions of recovery and maximum recovery. In Section 4, we study the problem of the existence of maximum recoveries. In Section 5, we compare the notions of maximum recovery, inverse and quasi-inverse. In Section 6, we provide algorithms for computing maximum recoveries. In Section 7, we study the language needed to express maximum recoveries.

In Section 8, we study the complexity of some decision problems related to the notions of recovery and maximum recovery. In Section 9, we study the notion of maximal recovery. Concluding remarks are in Section 10.

2. PRELIMINARIES

A *schema* $\mathbf{R}$ is a finite set $\{R_1, \ldots, R_k\}$ of relation symbols, with each R_i having a fixed arity n_i. Let $\mathbf{D}$ be a countably infinite domain. An instance I of $\mathbf{R}$ assigns to each relation symbol R_i of $\mathbf{R}$ a finite n_i-ary relation $R_i^I \subseteq \mathbf{D}^{n_i}$. The *domain* $\mathrm{dom}(I)$ of instance I is the set of all elements that occur in any of the relations R_i^I. $\mathrm{Inst}(\mathbf{R})$ is defined to be the set of all instances of $\mathbf{R}$. Given instances $I, J \in \mathrm{Inst}(\mathbf{R})$, we write $I \subseteq J$ to denote that, for every relation symbol R_i of $\mathbf{R}$, it holds that $R_i^I \subseteq R_i^J$.

As is customary in the data exchange literature, we consider instances with two types of values: constants and nulls [8, 7, 10]. More precisely, let $\mathbf{C}$ and $\mathbf{N}$ be infinite and disjoint sets of constants and nulls, respectively, and assume that $\mathbf{D} = \mathbf{C} \cup \mathbf{N}$. If we refer to a schema $\mathbf{S}$ as a *source* schema, then $\mathrm{Inst}(\mathbf{S})$ is defined to be the set of all instances of $\mathbf{S}$ that are constructed by using only elements from $\mathbf{C}$, and if we refer to a schema $\mathbf{T}$ as a *target* schema, then $\mathrm{Inst}(\mathbf{T})$ is defined as usual (instances of $\mathbf{T}$ are constructed by using elements from both $\mathbf{C}$ and $\mathbf{N}$). In this paper, we use $\mathbf{S}$ to refer to a source schema and $\mathbf{T}$ to refer to a target schema.

Given schemas $\mathbf{R}_1$ and $\mathbf{R}_2$, a *schema mapping* (or just *mapping*) from $\mathbf{R}_1$ to $\mathbf{R}_2$ is a nonempty subset of $\mathrm{Inst}(\mathbf{R}_1) \times \mathrm{Inst}(\mathbf{R}_2)$. As is customary in the data exchange literature, if $\mathbf{S}$ is a source schema and $\mathbf{T}$ is a target schema, a mapping from $\mathbf{S}$ to $\mathbf{T}$ is called *source-to-target mapping* (st-mapping), and a mapping from $\mathbf{T}$ to $\mathbf{S}$ is called *target-to-source mapping* (ts-mapping) [10].

If $\mathcal{M}$ is a schema mapping from $\mathbf{R}_1$ to $\mathbf{R}_2$ and I is an instance of $\mathbf{R}_1$, then we say that an instance J of $\mathbf{R}_2$ is a *solution for I under $\mathcal{M}$*, if $(I, J) \in \mathcal{M}$. The set of solutions for I under $\mathcal{M}$ is denoted by $\mathrm{Sol}_{\mathcal{M}}(I)$. The domain of $\mathcal{M}$, denoted by $\mathrm{dom}(\mathcal{M})$, is defined as the set of instances I such that $\mathrm{Sol}_{\mathcal{M}}(I) \neq \emptyset$. Furthermore, given schema mappings $\mathcal{M}_{12}$ from $\mathbf{R}_1$ to $\mathbf{R}_2$ and $\mathcal{M}_{23}$ from $\mathbf{R}_2$ to $\mathbf{R}_3$, the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is defined as the usual composition of binary relations, that is, $\mathcal{M}_{12} \circ \mathcal{M}_{23} = \{(I_1, I_3) \mid \exists I_2 : (I_1, I_2) \in \mathcal{M}_{12} \text{ and } (I_2, I_3) \in \mathcal{M}_{23}\}$. If $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is nonempty, then there exists a unique mapping $\mathcal{M}_{13}$ from $\mathbf{R}_1$ to $\mathbf{R}_3$ such that $\mathcal{M}_{13} = \mathcal{M}_{12} \circ \mathcal{M}_{23}$.

2.1 Dependencies and definability of mappings

In this paper, CQ is the class of conjunctive queries and UCQ is the class of unions of conjunctive queries. If we extend these classes by allowing equalities, inequalities or negation (of atoms), then we use superscripts $=$, $\neq$ and $\neg$, respectively. Thus, for example, $\mathrm{CQ}^{=}$ is the class of conjunctive queries with equalities and $\mathrm{UCQ}^{\neg}$ is the class of unions of conjunctive queries with negation. FO is the class of all first-order formulas with equality. Slightly abusing notation, we use $\mathbf{C}(\cdot)$ to denote a built-in unary predicate such that $\mathbf{C}(a)$ holds if and only if a is a constant, that is, $a \in \mathbf{C}$. If $\mathcal{L}$ is any of the previous query languages, then $\mathcal{L}^{\mathbf{C}}$ is the extension of $\mathcal{L}$ allowing predicate $\mathbf{C}(\cdot)$. For example, $\mathrm{CQ}^{\neq,\mathbf{C}}$ is the class of conjunctive queries with inequalities and predicate $\mathbf{C}(\cdot)$.

Dependencies: Let $\mathcal{L}_1$, $\mathcal{L}_2$ be query languages and $\mathbf{R}_1$, $\mathbf{R}_2$ be schemas with no relation symbols in common. A sentence Φ over $\mathbf{R}_1 \cup \mathbf{R}_2 \cup \{\mathbf{C}(\cdot)\}$ is an $\mathcal{L}_1$-TO-$\mathcal{L}_2$ *dependency from $\mathbf{R}_1$ to $\mathbf{R}_2$* if Φ is of the form $\forall \bar{x}\,(\varphi(\bar{x}) \rightarrow \psi(\bar{x}))$, where (1) $\bar{x}$ is the tuple of free variables in both $\varphi(\bar{x})$ and $\psi(\bar{x})$; (2) $\varphi(\bar{x})$ is an $\mathcal{L}_1$-formula over $\mathbf{R}_1 \cup \{\mathbf{C}(\cdot)\}$ if $\mathbf{C}(\cdot)$ is allowed in $\mathcal{L}_1$, and over $\mathbf{R}_1$ otherwise; and (3) $\psi(\bar{x})$ is an $\mathcal{L}_2$-formula over $\mathbf{R}_2 \cup \{\mathbf{C}(\cdot)\}$ if $\mathbf{C}(\cdot)$ is allowed in $\mathcal{L}_2$, and over $\mathbf{R}_2$ otherwise. We call $\varphi(\bar{x})$ the *premise* of Φ, and $\psi(\bar{x})$ the *consequent* of Φ. If $\mathbf{S}$ is a source schema and $\mathbf{T}$ is a target schema, an $\mathcal{L}_1$-TO-$\mathcal{L}_2$ dependency from $\mathbf{S}$ to $\mathbf{T}$ is called an $\mathcal{L}_1$-TO-$\mathcal{L}_2$ *source-to-target dependency* ($\mathcal{L}_1$-TO-$\mathcal{L}_2$ st-dependency), and an $\mathcal{L}_1$-TO-$\mathcal{L}_2$ dependency from $\mathbf{T}$ to $\mathbf{S}$ is called an $\mathcal{L}_1$-TO-$\mathcal{L}_2$ *target-to-source dependency* ($\mathcal{L}_1$-TO-$\mathcal{L}_2$ ts-dependency).

Three fundamental classes of dependencies for data exchange, and in particular for inverting schema mappings, are source-to-target tuple-generating dependencies (st-tgds), full source-to-target tuple-generating dependencies (full st-tgds) and target-to-source disjunctive tuple-generating dependencies with inequalities and predicate $\mathbf{C}(\cdot)$ [8, 10]. The former corresponds to the class of CQ-TO-CQ st-dependencies, and the latter is an extension of the class of $\mathrm{CQ}^{\neq,\mathbf{C}}$-TO-UCQ ts-dependencies. An FO-TO-CQ dependency is full if its consequent does not include existential quantifiers and, thus, the class of full st-tgds corresponds to the class of full CQ-TO-CQ st-dependencies.

Semantics of dependencies, safeness: Assume that I is an instance of a schema $\mathbf{R} = \{R_1, \ldots, R_m\}$. Instance I can be represented as an $(\mathbf{R} \cup \{\mathbf{C}(\cdot)\})$-structure $\mathfrak{A}_I = \langle A, R_1^A, \ldots, R_m^A, \mathbf{C}^A \rangle$, where $A = \mathrm{dom}(I)$ is the universe of $\mathfrak{A}_I$, $R_i^A = R_i^I$ for $i \in [1, m]$ and $\mathbf{C}^A = A \cap \mathbf{C}$. This representation is used to define the semantics of FO over source and target instances (here we assume familiarity with some basic notions of first-order logic).

Let $\mathbf{R}_1 = \{S_1, \ldots, S_m\}$ be a schema and I an instance of $\mathbf{R}_1$. If $\varphi(\bar{x})$ is an FO-formula over $\mathbf{R}_1 \cup \{\mathbf{C}(\cdot)\}$ and $\bar{a}$ is a tuple of elements from $\mathrm{dom}(I)$, then we say that I satisfies $\varphi(\bar{a})$, denoted by $I \models \varphi(\bar{a})$, if and only if $\mathfrak{A}_I \models \varphi(\bar{a})$. Furthermore, let $\mathbf{R}_2 = \{T_1, \ldots, T_n\}$ be a schema with no relation symbols in common with $\mathbf{R}_1$, and J an instance of $\mathbf{R}_2$. Then $K = (I, J)$ is an instance of $\mathbf{R}_1 \cup \mathbf{R}_2$ defined as $S_i^K = S_i^I$ and $T_j^K = T_j^J$, for $i \in [1, m]$ and $j \in [1, n]$. Notice that $\mathrm{dom}(K) = \mathrm{dom}(I) \cup \mathrm{dom}(J)$. If $\varphi(\bar{x})$ is an FO-formula over $\mathbf{R}_1 \cup \mathbf{R}_2 \cup \{\mathbf{C}(\cdot)\}$ and $\bar{a}$ is a tuple of elements from $\mathrm{dom}(I) \cup \mathrm{dom}(J)$, then we say that (I, J) satisfies $\varphi(\bar{a})$, denoted by $(I, J) \models \varphi(\bar{a})$, if and only if $\mathfrak{A}_K \models \varphi(\bar{a})$. As usual, we say that an instance satisfies a set Σ of dependencies if the instance satisfies each dependency in Σ.

We impose the following safety condition on $\mathcal{L}_1$-TO-$\mathcal{L}_2$ dependencies. Recall that an FO-formula $\varphi(\bar{x})$ is *domain-independent* if its answer depends only on the database instance but not on the underlying domain (see [6] for a formal definition). Let $\mathbf{R}_1$ and $\mathbf{R}_2$ be schemas with no relation symbols in common and $\Phi = \forall \bar{x}\,(\varphi(\bar{x}) \rightarrow \psi(\bar{x}))$ an $\mathcal{L}_1$-TO-$\mathcal{L}_2$ dependency from $\mathbf{R}_1$ to $\mathbf{R}_2$. Then we say that Φ is *domain-independent* if both $\varphi(\bar{x})$ and $\psi(\bar{x})$ are domain-independent. The following strategy can be used to evaluate Φ: Given instances I, J of $\mathbf{R}_1$ and $\mathbf{R}_2$, respectively, we have that $(I, J) \models \Phi$ if and only if for every tuple $\bar{a}$ of elements from $\mathrm{dom}(I)$, if $I \models \varphi(\bar{a})$, then every component of $\bar{a}$ is in $\mathrm{dom}(J)$ and $J \models \psi(\bar{a})$. We note that this strategy cannot be used for non domain-independent $\mathcal{L}_1$-TO-$\mathcal{L}_2$ dependencies.

Definability of mappings: Let $\mathbf{R}_1$ and $\mathbf{R}_2$ be schemas with no relation symbols in common and Σ a set of FO-sentences over $\mathbf{R}_1 \cup \mathbf{R}_2 \cup \{\mathbf{C}(\cdot)\}$. We say that a mapping $\mathcal{M}$ from $\mathbf{R}_1$ to $\mathbf{R}_2$ is *specified* by Σ, denoted by $\mathcal{M} = (\mathbf{R}_1, \mathbf{R}_2, \Sigma)$, if for every $(I, J) \in \mathrm{Inst}(\mathbf{R}_1) \times \mathrm{Inst}(\mathbf{R}_2)$, we have that $(I, J) \in \mathcal{M}$ if and only if $(I, J) \models \Sigma$.

Proviso: In this paper, every set Σ of dependencies is finite, and if Σ is a set of $\mathcal{L}_1$-TO-$\mathcal{L}_2$ dependencies, then we assume that every dependency in Σ is domain-independent (as defined above). Furthermore, we omit the outermost universal quantifiers from $\mathcal{L}_1$-TO-$\mathcal{L}_2$ dependencies and, thus, we write $\varphi(\bar{x}) \rightarrow \psi(\bar{x})$ instead of $\forall \bar{x}\,(\varphi(\bar{x}) \rightarrow \psi(\bar{x}))$. Finally, for the sake of readability, we write $\varphi(\bar{x}, \bar{y}) \rightarrow \psi(\bar{x})$ instead of $(\exists \bar{y}\,\varphi(\bar{x}, \bar{y})) \rightarrow \psi(\bar{x})$ in some examples, as these two formulas are equivalent.

3. RECOVERIES AND THEIR MAXIMA

Let $\mathcal{M}$ be a mapping from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$, and Id the *identity schema mapping* over $\mathbf{R}_1$, that is, $\mathrm{Id} = \{(I, I) \mid I \in \mathrm{Inst}(\mathbf{R}_1)\}$. When trying to invert $\mathcal{M}$, the ideal would be to find a mapping $\mathcal{M}'$ from $\mathbf{R}_2$ to $\mathbf{R}_1$ such that, $\mathcal{M} \circ \mathcal{M}' = \mathrm{Id}$. If such a mapping exists, we know that if we use $\mathcal{M}$ to exchange data, the application of $\mathcal{M}'$ gives as result exactly the initial source instance. Unfortunately, in most cases this ideal is impossible to reach. For example, it is impossible to obtain such an inverse if $\mathcal{M}$ is specified by a set of st-tgds [7]. The main problem with such an ideal definition of inverse is that, in general, no matter what $\mathcal{M}'$ we choose, we will have not one but many solutions for a source instance under $\mathcal{M} \circ \mathcal{M}'$.

If for a mapping $\mathcal{M}$, there is no mapping $\mathcal{M}_1$ such that $\mathcal{M} \circ \mathcal{M}_1 = \mathrm{Id}$, at least we would like to find a schema mapping $\mathcal{M}_2$ that *does not forbid* the possibility of recovering the initial source data. That is, we would like that for every instance $I \in \mathrm{dom}(\mathcal{M})$, the space of solutions for I under $\mathcal{M} \circ \mathcal{M}_2$ contains I itself. Such a schema mapping $\mathcal{M}_2$ is called a *recovery* of $\mathcal{M}$.

DEFINITION 3.1. *Let $\mathbf{R}_1$ and $\mathbf{R}_2$ be two schemas, $\mathcal{M}$ a mapping from $\mathbf{R}_1$ to $\mathbf{R}_2$ and $\mathcal{M}'$ a mapping from $\mathbf{R}_2$ to $\mathbf{R}_1$. Then $\mathcal{M}'$ is a* recovery *of $\mathcal{M}$ iff $(I, I) \in \mathcal{M} \circ \mathcal{M}'$ for every instance $I \in \mathrm{dom}(\mathcal{M})$.*

Being a recovery is a sound but mild requirement. Indeed, a schema mapping $\mathcal{M}$ from $\mathbf{R}_1$ to $\mathbf{R}_2$ always has as recoveries, for example, mappings $\mathcal{M}_1 = \mathrm{Inst}(\mathbf{R}_2) \times \mathrm{Inst}(\mathbf{R}_1)$ and $\mathcal{M}_2 = \mathcal{M}^{-1} = \{(J, I) \mid (I, J) \in \mathcal{M}\}$. If one has to choose between $\mathcal{M}_1$ and $\mathcal{M}_2$ as a recovery of $\mathcal{M}$, then one would probably choose $\mathcal{M}_2$ since the space of possible solutions for a source instance I under $\mathcal{M} \circ \mathcal{M}_2$ is smaller than under $\mathcal{M} \circ \mathcal{M}_1$. In fact, if there exists a mapping $\mathcal{M}_3$ such that $\mathcal{M} \circ \mathcal{M}_3 = \mathrm{Id}$, then one would definitely prefer $\mathcal{M}_3$ over $\mathcal{M}_1$ and $\mathcal{M}_2$. In general, if $\mathcal{M}'$ is a recovery of $\mathcal{M}$, then the smaller the space of solutions generated by $\mathcal{M} \circ \mathcal{M}'$, the more informative $\mathcal{M}'$ is about the initial source instances. This notion induces an *order* among recoveries:

DEFINITION 3.2. *Let $\mathcal{M}$ be a mapping and $\mathcal{M}'$, $\mathcal{M}''$ recoveries of $\mathcal{M}$. We say that $\mathcal{M}'$ is* at least as informative as *$\mathcal{M}''$ for $\mathcal{M}$, and write $\mathcal{M}'' \preceq_{\mathcal{M}} \mathcal{M}'$, iff $\mathcal{M} \circ \mathcal{M}' \subseteq \mathcal{M} \circ \mathcal{M}''$.*

Moreover, we say that $\mathcal{M}'$ and $\mathcal{M}''$ are *equally informative for $\mathcal{M}$*, denoted by $\mathcal{M}' \equiv_{\mathcal{M}} \mathcal{M}''$, if $\mathcal{M}'' \preceq_{\mathcal{M}} \mathcal{M}'$ and $\mathcal{M}' \preceq_{\mathcal{M}} \mathcal{M}''$.

If for a mapping $\mathcal{M}$, there exists a recovery $\mathcal{M}'$ that is at least as informative as any other recovery of $\mathcal{M}$, then $\mathcal{M}'$ is the best alternative to bring exchanged data back, among all the recoveries. Intuitively, such a mapping $\mathcal{M}'$ recovers the maximum amount of sound information. Such a mapping $\mathcal{M}'$ is called a maximum recovery of $\mathcal{M}$.

DEFINITION 3.3. *Let $\mathcal{M}'$ be a recovery of a mapping $\mathcal{M}$. We say that $\mathcal{M}'$ is a* maximum recovery *of $\mathcal{M}$ if for every recovery $\mathcal{M}''$ of $\mathcal{M}$, it is the case that $\mathcal{M}'' \preceq_{\mathcal{M}} \mathcal{M}'$.*

Notice that, if $\mathcal{M}_1$ and $\mathcal{M}_2$ are maximum recoveries of a mapping $\mathcal{M}$, then they are equally informative for $\mathcal{M}$, that is, $\mathcal{M}_1 \equiv_{\mathcal{M}} \mathcal{M}_2$.

A first important issue about the notion of recovery is whether for every mapping $\mathcal{M}$, there always exists a maximum recovery. To answer this question, we introduce the notion of *witness*, and use it to provide a necessary and sufficient condition for the existence of a maximum recovery for a mapping $\mathcal{M}$.

DEFINITION 3.4. *Let $\mathcal{M}$ be a mapping from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$ and $I \in \mathrm{Inst}(\mathbf{R}_1)$. Then $J \in \mathrm{Inst}(\mathbf{R}_2)$ is a* witness for I under $\mathcal{M}$ *if for every $I' \in \mathrm{Inst}(\mathbf{R}_1)$, if $J \in \mathrm{Sol}_{\mathcal{M}}(I')$, then $\mathrm{Sol}_{\mathcal{M}}(I) \subseteq \mathrm{Sol}_{\mathcal{M}}(I')$.*

Notice that a witness for an instance I is not necessarily a solution for I under $\mathcal{M}$. We say that J is a *witness solution* for I if J is both a witness and a solution for I. A witness solution can be considered as an *identifier* for a space of solutions; if J is a witness solution for instances I_1 and I_2, then $\mathrm{Sol}_{\mathcal{M}}(I_1) = \mathrm{Sol}_{\mathcal{M}}(I_2)$. Other identifiers for spaces of solutions have been proposed in the data exchange literature. For example, for the specific case of st-tgds, we prove in Section 4.1 that the notion of *universal solution* introduced in [8] is stronger than the notion of witness solution, in the sense that every universal solution is a witness solution but the opposite does not hold. For other classes of st-dependencies, the notions of universal and witness solution are incomparable (see Section 4.2 for some examples).

THEOREM 3.5. *A mapping $\mathcal{M}$ has a maximum recovery iff for every $I \in \mathrm{dom}(\mathcal{M})$, there exists a witness solution for I under $\mathcal{M}$.*

The previous theorem shows that the notion of witness can be used to provide a necessary and sufficient condition for the existence of a maximum recovery for a mapping $\mathcal{M}$. This notion can also be used to characterize when a mapping $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$. In fact, the following theorem shows that witness instances are the building blocks of maximum recoveries.

THEOREM 3.6. *$\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$ if and only if, $\mathcal{M}'$ is a recovery of $\mathcal{M}$ and for every $(I_1, J) \in \mathcal{M}$ and $(J, I_2) \in \mathcal{M}'$, it holds that $I_2 \in \mathrm{dom}(\mathcal{M})$ and J is a witness for I_2 under $\mathcal{M}$.*

4. ON THE EXISTENCE OF MAXIMUM RECOVERIES

In this section, we focus on source-to-target mappings, that is, mappings from a source schema $\mathbf{S}$ to a target schema $\mathbf{T}$. Recall that instances of $\mathbf{S}$ are constructed by using only elements from $\mathbf{C}$ (constants), while instances of $\mathbf{T}$ are constructed by using elements from both $\mathbf{C}$ and $\mathbf{N}$ (constants and nulls). This is the most common class of mappings in the data exchange literature [8, 2, 9, 1], and specifically in the literature on inverting schema mappings [7, 10]. We note that the recovery of an st-mapping is a target-to-source mapping.

On the positive side, we prove our main results regarding classes of st-mappings that admit maximum recoveries. Namely, we show that if $\mathcal{M}$ is an st-mapping specified by a set of FO-TO-CQ dependencies, then $\mathcal{M}$ has a maximum recovery. Furthermore, we also show that the extension of this class with source dependencies, equality-generating target dependencies and weakly acyclic sets of tuple-generating target dependencies [5, 8] also admits maximum recoveries (these classes of dependencies are defined in Section 4.1). These results are in sharp contrast with the results of [7, 10] , where it was shown that even for full st-tgds, inverses and quasi-inverses are not guaranteed to exist.

On the negative side, we show that if we enrich the consequent of FO-TO-CQ dependencies by adding inequalities, or disjunction, or negation, the existence of maximum recoveries is not guaranteed.

4.1 Positive results

In [8], the class of *universal solutions* for st-mappings was identified as a class of solutions that has good properties for data exchange. These solutions play an important role in this section. To formally introduce this concept, we review the necessary terminology from [8].

Let J_1 and J_2 be instances of the same schema $\mathbf{R}$. A *homomorphism* h from J_1 to J_2 is a function $h : \mathrm{dom}(J_1) \to \mathrm{dom}(J_2)$ such that, for every $R \in \mathbf{R}$ and every tuple $(a_1, \ldots, a_k) \in R^{J_1}$, it holds $(h(a_1), \ldots, h(a_k)) \in R^{J_2}$. Given a set $A \subseteq \mathbf{D}$, we say that a homomorphism h from J_1 to J_2 is the identity on A, if $h(a) = a$ for every $a \in A \cap \mathrm{dom}(J_1)$. Let $\mathcal{M}$ be an st-mapping, I a source

instance and J a solution for I under $\mathcal{M}$. Then J is a *universal solution* for I under $\mathcal{M}$, if for every solution J' for I under $\mathcal{M}$, there exists a homomorphism from J to J' that is the identity on $\mathbf{C}$. The next lemma shows an important relationship between universal and witness solutions.

LEMMA 4.1.

(1) Let $\mathcal{M}$ be an st-mapping specified by a set of FO-TO-CQ *dependencies and I a source instance. Then every universal solution for I under $\mathcal{M}$ is a witness solution for I under $\mathcal{M}$.*

(2) There exists an st-mapping $\mathcal{M}$ specified by a set of st-tgds and a source instance I such that, I has a witness solution under $\mathcal{M}$ that is not a universal solution for I under $\mathcal{M}$.

It is known that, for st-mappings specified by FO-TO-CQ dependencies, universal solutions exist for every source instance [8, 2]. Then from Theorem 3.5 and Lemma 4.1, we obtain the following theorem.

THEOREM 4.2. *If $\mathcal{M}$ is an st-mapping specified by a set of* FO-TO-CQ *st-dependencies, then $\mathcal{M}$ has a maximum recovery.*

EXAMPLE 4.3. In [10], it was shown that the schema mapping $\mathcal{M}$ specified by full st-tgd $E(x,z) \wedge E(z,y) \rightarrow F(x,y) \wedge M(z)$ has neither a quasi-inverse nor an inverse. It is possible to show that the schema mapping $\mathcal{M}'$ specified by:

$$\begin{aligned} F(x,y) &\rightarrow \exists u(E(x,u) \wedge E(u,y)), \\ M(z) &\rightarrow \exists v \exists w(E(v,z) \wedge E(z,w)), \end{aligned}$$

is a maximum recovery of $\mathcal{M}$. □

Source and target dependencies. Fix source and target schemas $\mathbf{S}$ and $\mathbf{T}$. If α is an FO-sentence over $\mathbf{S}$, then we say that α is a source FO-dependency, and if β is an FO-sentence over $\mathbf{T} \cup \{\mathbf{C}(\cdot)\}$, then we say that β is a target FO-dependency. We assume that both source and target FO-dependencies are domain independent.

Let $\Sigma_{\mathbf{st}}$, $\Gamma_{\mathbf{s}}$, $\Gamma_{\mathbf{t}}$ be sets of source-to-target, source, and target FO-dependencies, respectively. We say that an st-mapping $\mathcal{M}$ is specified by $\Sigma_{\mathbf{st}}$, $\Gamma_{\mathbf{s}}$, and $\Gamma_{\mathbf{t}}$, and we write $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathbf{st}}, \Gamma_{\mathbf{s}}, \Gamma_{\mathbf{t}})$, if $\mathcal{M}$ is specified by $\Sigma_{\mathbf{st}} \cup \Gamma_{\mathbf{s}} \cup \Gamma_{\mathbf{t}}$. Given that both $\Gamma_{\mathbf{s}}$ and $\Gamma_{\mathbf{t}}$ are sets of domain-independent sentences, we have that $(I,J) \models \Sigma_{\mathbf{st}} \cup \Gamma_{\mathbf{s}} \cup \Gamma_{\mathbf{t}}$ if and only if $(I,J) \models \Sigma_{\mathbf{st}}$, $I \models \Gamma_{\mathbf{s}}$ and $J \models \Gamma_{\mathbf{t}}$. Thus, source constraints affect the domain of an st-mapping, while target constraints affect its set of possible solutions. Notice that these roles switch when considering ts-mappings. Our next results show that maximum recoveries have good properties regarding source constraints.

LEMMA 4.4. *Let $\mathcal{M}_1$ be an st-mapping and $\mathcal{M}_1^\star$ a maximum recovery of $\mathcal{M}_1$. If $\Gamma_{\mathbf{s}}$ is a set of source* FO*-dependencies and $\mathcal{M}_2 = \{(I,J) \in \mathcal{M}_1 \mid I \models \Gamma_{\mathbf{s}}\}$, then $\mathcal{M}_2^\star = \{(J,I) \in \mathcal{M}_1^\star \mid I \models \Gamma_{\mathbf{s}}\}$ is a maximum recovery of $\mathcal{M}_2$.*

Notice that $\mathcal{M}_1$ in the above lemma is an arbitrary st-mapping. Thus, we obtain the following corollary from Theorem 4.2.

PROPOSITION 4.5. *If $\mathcal{M}$ is an st-mapping specified by a set of* FO-TO-CQ *st-dependencies together with a set of source* FO*-dependencies, then $\mathcal{M}$ has a maximum recovery.*

EXAMPLE 4.6. Let $\mathcal{M}_2 = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathbf{st}}, \Gamma_{\mathbf{s}})$ be an st-mapping, where $\mathbf{S} = \{A(\cdot,\cdot,\cdot)\}$, $\mathbf{T} = \{B(\cdot,\cdot),\ C(\cdot,\cdot)\}$ and

$$\begin{aligned} \Sigma_{\mathbf{st}} &= \{A(x,y,z) \rightarrow B(x,y) \wedge C(y,z)\}, \\ \Gamma_{\mathbf{s}} &= \{A(x,y,z) \wedge A(x',y,z') \rightarrow z = z'\}. \end{aligned}$$

Notice that $\Gamma_{\mathbf{s}}$ is a set of functional dependencies. Consider st-mapping $\mathcal{M}_1 = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathbf{st}})$. Then ts-mapping specified by $\Sigma_{\mathbf{ts}} = \{B(x,y) \wedge C(y,z) \rightarrow \exists u\, A(x,y,u) \wedge \exists w\, A(w,y,z)\}$ is a maximum recovery of $\mathcal{M}_1$. Thus, we have by Lemma 4.4 that ts-mapping $\mathcal{M}_2^\star$ specified by $\Sigma_{\mathbf{ts}}$ and $\Gamma_{\mathbf{s}}$ is a maximum recovery of $\mathcal{M}_2$. We observe that $\Sigma_{\mathbf{ts}} \cup \Gamma_{\mathbf{s}}$ is logically equivalent to:

$$B(x,y) \wedge C(y,z) \rightarrow A(x,y,z), \quad (2)$$

$$A(x,y,z) \wedge A(x',y,z') \rightarrow z = z'. \quad (3)$$

In this case, we obtained what was expected; since $\Sigma_{\mathbf{st}}$ is a lossless decomposition of relation A according to $\Gamma_{\mathbf{s}}$, dependency (2) joins relations B and C to reconstruct the source instances. □

We show in Section 4.2 that, if the full power of FO is allowed in target dependencies, then maximum recoveries are not guaranteed to exist. For this reason, we focus here on equality- and tuple-generating dependencies. Let $\mathbf{R}$ be a schema. An *equality-generating dependency* (egd) over $\mathbf{R}$ is an FO-sentence $\forall \bar{x}(\varphi(\bar{x}) \rightarrow (x_i = x_j))$, where $\varphi(\bar{x})$ is a conjunctive query over $\mathbf{R}$, and x_i, x_j are among the variables in $\bar{x}$. A *tuple-generating dependency* (tgd) over $\mathbf{R}$ is an FO-sentence $\forall \bar{x}(\varphi(\bar{x}) \rightarrow \psi(\bar{x}))$, where both $\varphi(\bar{x})$ and $\psi(\bar{x})$ are conjunctive queries over $\mathbf{R}$. In the following theorem, we show that maximum recoveries are guaranteed to exist in the general setting where target egds and weakly acyclic sets of target tgds are allowed. Due to the lack of space, we omit the formal definition of weak acyclicity (see [8] for a formal definition). We just mention that over the past few years, weak acyclicity has shown to be indispensable for the tractability of some important data exchange problems [8, 13, 12] and, thus, it is a common assumption in the area.

THEOREM 4.7. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathbf{st}}, \Gamma_{\mathbf{s}}, \Gamma_{\mathbf{t}})$ be an st-mapping, where $\Sigma_{\mathbf{st}}$ is a set of* FO-TO-CQ *st-dependencies, $\Gamma_{\mathbf{s}}$ is a set of source* FO*-dependencies and $\Gamma_{\mathbf{t}}$ is the union of a set of target egds and a weakly acyclic set of target tgds. Then $\mathcal{M}$ has a maximum recovery.*

Notice that the positive results of this section do not say anything about the language needed to express maximum recoveries. In Sections 6 and 7, we study this problem.

4.2 Negative results

In Section 4.1, we prove that FO-TO-CQ st-mappings have maximum recoveries using the relationship between universal and witness solutions shown in Lemma 4.1. If we go beyond CQ in the consequents of dependencies, these notions become incomparable. For example, consider an st-mapping $\mathcal{M}_1$ specified by CQ-TO-CQ$^{\neq}$ dependencies $P(x) \rightarrow \exists y\, R(x,y)$ and $S(x) \rightarrow \exists y\, (R(x,y) \wedge x \neq y)$, and let I be a source instance such that $P^I = \{a\}$. Target instance J_1 such that $R^{J_1} = \{(a,n)\}$, with $n \in \mathbf{N}$, is a universal solution but not a witness for I, while target instance J_2 such that $R^{J_2} = \{(a,a)\}$ is a witness but not a universal solution for I. In this example, every source instance has a witness solution, and, thus, $\mathcal{M}_1$ has a maximum recovery. In fact, dependencies $R(x,y) \rightarrow P(x) \vee S(x)$ and $R(x,y) \wedge x \neq y \rightarrow S(x)$ specify a maximum recovery of $\mathcal{M}_1$. As a second example, consider st-mapping $\mathcal{M}_2$ specified by CQ-TO-UCQ dependency $P(x) \rightarrow R(x) \vee S(x)$. In this case, every source instance has a witness solution, and only the empty source instance has a universal solution. In fact, dependencies $R(x) \rightarrow P(x)$ and $S(x) \rightarrow P(x)$ specify a maximum recovery of $\mathcal{M}_2$.

We have shown examples of mappings that have maximum recoveries and are specified by dependencies with inequalities and disjunctions in the consequents. However, the following proposition shows that this is not a general phenomenon. If we slightly enrich the language used in the consequents of FO-TO-CQ dependencies, then the

existence of maximum recoveries is not guaranteed, even if premises are restricted to be conjunctive queries.

PROPOSITION 4.8. *There exist st-mappings specified by (1)* CQ-TO-CQ$^{\neq}$*, (2)* CQ-TO-UCQ*, and (3)* CQ-TO-CQ$^{\neg}$ *dependencies, that have no maximum recoveries.*

We conclude this section by showing that, if the full power of FO is allowed in target dependencies, then maximum recoveries are not guaranteed to exist.

PROPOSITION 4.9. *There exists an st-mapping specified by a set of st-tgds plus a set of target* FO*-dependencies that has no maximum recovery.*

5. COMPARISON WITH THE NOTIONS OF INVERSE AND QUASI-INVERSE

In this section, we study the relationship between the notion of maximum recovery and the notions of inverse and quasi-inverse [7, 10].

We start by recalling the definition of inverse proposed in [7]. A mapping $\mathcal{M}$ is *closed-down on the left* if whenever $(I, J) \in \mathcal{M}$ and $I' \subseteq I$, it holds that $(I', J) \in \mathcal{M}$. In [7], Fagin defines a notion of inverse focusing on mappings that satisfy this condition. More precisely, let $\mathbf{S}$ be a source schema. Fagin first defines an identity mapping $\overline{\mathrm{Id}}$ as $\{(I_1, I_2) \mid I_1, I_2 \in \mathrm{Inst}(\mathbf{S}) \text{ and } I_1 \subseteq I_2\}$, which is appropriate for closed-down on the left mappings [7]. Then he says that a ts-mapping $\mathcal{M}'$ is an *inverse* of an st-mapping $\mathcal{M}$ if $\mathcal{M} \circ \mathcal{M}' = \overline{\mathrm{Id}}$.

Since it is rare that a schema mapping possesses an inverse [10], Fagin et al. introduce the notion of a quasi-inverse of a schema mapping in [10]. The idea behind quasi-inverses is to relax the notion of inverse of a mapping by not differentiating between source instances that are data-exchange equivalent. Let $\mathcal{M}$ be a mapping from a source schema $\mathbf{S}$ to a target schema $\mathbf{T}$. Instances I_1 and I_2 of $\mathbf{S}$ are *data-exchange equivalent* w.r.t. $\mathcal{M}$, denoted by $I_1 \sim_{\mathcal{M}} I_2$, if $\mathrm{Sol}_{\mathcal{M}}(I_1) = \mathrm{Sol}_{\mathcal{M}}(I_2)$. Furthermore, given a mapping $\mathcal{M}_1$ from $\mathbf{S}$ to $\mathbf{S}$, mapping $\mathcal{M}_1[\sim_{\mathcal{M}}, \sim_{\mathcal{M}}]$ is defined as $\{(I_1, I_2) \in \mathrm{Inst}(\mathbf{S}) \times \mathrm{Inst}(\mathbf{S}) \mid \exists (I_1', I_2') : I_1 \sim_{\mathcal{M}} I_1', I_2 \sim_{\mathcal{M}} I_2' \text{ and } (I_1', I_2') \in \mathcal{M}_1\}$. Then a ts-mapping $\mathcal{M}'$ is a *quasi-inverse* of an st-mapping $\mathcal{M}$ if $(\mathcal{M} \circ \mathcal{M}')[\sim_{\mathcal{M}}, \sim_{\mathcal{M}}] = \overline{\mathrm{Id}}[\sim_{\mathcal{M}}, \sim_{\mathcal{M}}]$.

The definitions of inverse and quasi-inverse are appropriate for closed-down on the left mappings. In fact, some counterintuitive results are obtained if one removes this restriction. For example, let $\mathbf{S} = \{P(\cdot)\}$, $\mathbf{T} = \{R(\cdot)\}$ and $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$ specified by dependency $\forall x\,(P(x) \leftrightarrow R(x))$. In this case, mapping $\mathcal{M}'$ specified by $\forall x\,(R(x) \leftrightarrow P(x))$ is an *ideal* inverse of $\mathcal{M}$ since $\mathcal{M} \circ \mathcal{M}' = \mathrm{Id} = \{(I, I) \mid I \in \mathrm{Inst}(\mathbf{S})\}$. However, $\mathcal{M}'$ is neither an inverse nor a quasi-inverse of $\mathcal{M}$ (although it is a maximum recovery of $\mathcal{M}$). Moreover, the definitions of inverse and quasi-inverse are only appropriate for total mappings, that is, mappings $\mathcal{M}$ such that $\mathrm{dom}(\mathcal{M})$ is the set of all source instances. According to the definitions of inverse and quasi-inverse in [10], if an st-mapping $\mathcal{M}$ is not total, then $\mathcal{M}$ is neither invertible nor quasi-invertible.

From the discussion in the previous paragraph, to compare the notions of maximum recovery, inverse and quasi-inverse, we need to focus on the class of total st-mappings that are closed-down on the left. This class includes, for example, total st-mappings specified by UCQ$^{\neq}$-TO-FO st-dependencies. Our first result is a corollary of Propositions 3.9 and 3.12 in [10] and Theorem 4.2.

PROPOSITION 5.1. *There exists an st-mapping $\mathcal{M}$ specified by a set of full st-tgds that is neither invertible nor quasi-invertible, but has a maximum recovery.*

This result combined with the following theorem, shows that the notion of maximum recovery strictly generalizes the notion of inverse.

THEOREM 5.2. *Let $\mathcal{M}$ be a total st-mapping that is closed-down on the left, and assume that $\mathcal{M}$ is invertible. Then $\mathcal{M}'$ is an inverse of $\mathcal{M}$ iff $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$.*

The exact relationship between the notions of quasi-inverse and maximum recovery is shown in the following theorem. It is worth emphasizing that if an st-mapping $\mathcal{M}$ is quasi-invertible, then it admits a maximum recovery and, furthermore, every maximum recovery of $\mathcal{M}$ is also a quasi-inverse of $\mathcal{M}$.

THEOREM 5.3.

(1) Let $\mathcal{M}$ be a total st-mapping that is closed-down on the left, and assume that $\mathcal{M}$ is quasi-invertible. Then $\mathcal{M}$ has a maximum recovery and, furthermore, $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$ iff $\mathcal{M}'$ is a quasi-inverse and a recovery of $\mathcal{M}$.

(2) There exists an st-mapping $\mathcal{M}$ specified by a set of st-tgds and a ts-mapping $\mathcal{M}'$ specified by a set of ts-tgds such that, $\mathcal{M}'$ is a quasi-inverse of $\mathcal{M}$ but not a maximum recovery of $\mathcal{M}$.

On necessary and sufficient conditions for the existence of inverses and quasi-inverses. In Section 3, we identify a necessary and sufficient condition for the existence of maximum recoveries. For the case of the inverse (quasi-inverse), a condition called *subset property* ($(\sim_{\mathcal{M}}, \sim_{\mathcal{M}})$*-subset property*) was identified in [10] as necessary and sufficient for testing invertibility (quasi-invertibility), for the case of st-mappings specified by st-tgds. In this section, we first show that the subset property ($(\sim_{\mathcal{M}}, \sim_{\mathcal{M}})$-subset property) is not a sufficient condition for testing invertibility (quasi-invertibility) if one goes beyond st-tgds. Then we show that these conditions can be extended to the class of total and closed-down on the left st-mappings, by combining them with any necessary and sufficient condition for the existence of maximum recoveries.

An st-mapping has the subset property if for every pair of instances I_1, I_2 such that $\mathrm{Sol}_{\mathcal{M}}(I_2) \subseteq \mathrm{Sol}_{\mathcal{M}}(I_1)$, it holds that $I_1 \subseteq I_2$. An st-mapping $\mathcal{M}$ has the $(\sim_{\mathcal{M}}, \sim_{\mathcal{M}})$*-subset property* if for every pair of instances I_1, I_2 such that $\mathrm{Sol}_{\mathcal{M}}(I_2) \subseteq \mathrm{Sol}_{\mathcal{M}}(I_1)$, there exist instances I_1' and I_2' such that $I_1 \sim_{\mathcal{M}} I_1'$, $I_2 \sim_{\mathcal{M}} I_2'$ and $I_1' \subseteq I_2'$.

PROPOSITION 5.4. *There exist total and closed-down on the left st-mappings specified by (1)* CQ-TO-CQ$^{\neq}$*, and (2)* CQ-TO-UCQ *dependencies, that satisfy both the subset and $(\sim_{\mathcal{M}}, \sim_{\mathcal{M}})$-subset property and are neither invertible nor quasi-invertible.*

It turns out that by using the machinery developed for maximum recoveries, it is possible to provide necessary and sufficient conditions for the existence of inverses and quasi-inverses.

PROPOSITION 5.5. *Let $\mathcal{M}$ be a total st-mapping that is closed-down on the left.*

(1) $\mathcal{M}$ is invertible iff $\mathcal{M}$ has a maximum recovery and satisfies the subset property.

(2) $\mathcal{M}$ is quasi-invertible iff $\mathcal{M}$ has a maximum recovery and satisfies the $(\sim_{\mathcal{M}}, \sim_{\mathcal{M}})$-subset property.

As a corollary of Proposition 5.5, we obtain that an extension of the notion of witness solution can be used to provide a necessary and sufficient condition for invertibility. Given an st-mapping $\mathcal{M}$, we say that a target instance J is a *strong witness* for a source instance I under $\mathcal{M}$, if for every source instance I' such that $J \in \mathrm{Sol}_{\mathcal{M}}(I')$, it holds that $I' \subseteq I$. Notice that if a mapping $\mathcal{M}$ is closed-down on the left and J is a strong witness for I, then J is a witness for I.

COROLLARY 5.6. *A total and closed-down on the left st-mapping $\mathcal{M}$ is invertible iff every source instance has a strong witness solution under $\mathcal{M}$.*

6. COMPUTING MAXIMUM RECOVERIES

In Section 4.1, we show that every st-mapping specified by a set of FO-TO-CQ dependencies has a maximum recovery. In this section, we provide an exponential-time algorithm that computes maximum recoveries for st-mappings specified by FO-TO-CQ dependencies, and a quadratic-time algorithm for the case of full FO-TO-CQ dependencies. Moreover, we show how to extend these algorithms to handle st-mappings with arbitrary source constraints.

It is known that the simple process of "reversing the arrows" of source-to-target dependencies does not necessarily produce inverses as consequents of different dependencies may be related [7]; a consequent of a dependency may be implied by the consequents of other dependencies. In this section, we present an algorithm that searches for relations among consequents of dependencies, suitably *composes* the premises of related dependencies, and then "reverses the arrows" to obtain a maximum recovery.

We start by focusing on full FO-TO-CQ dependencies. Recall that a full FO-TO-CQ dependency does not include any existential quantifiers in its consequent. We illustrate the intuition behind our algorithm with some examples. Consider an st-mapping $\mathcal{M}_1$ specified by dependencies $\varphi_1(x) \rightarrow P(x)$ and $\varphi_2(y) \rightarrow R(y)$. The consequents of these dependencies are not related since they mention different relation names. In this case, the algorithm simply reverses the arrows to obtain dependencies $P(x) \rightarrow \varphi_1(x)$ and $R(y) \rightarrow \varphi_2(y)$, which specify a maximum recovery of $\mathcal{M}_1$. As a more involved example, consider an st-mapping $\mathcal{M}_2$ specified by (1) $\varphi_1(x, y) \rightarrow P(x, y)$ and (2) $\varphi_2(z) \rightarrow P(z, z)$. Clearly the consequents of these dependencies are related; the consequent of (1) implies the consequent of (2) when x is equal to y. In fact, this set of dependencies is logically equivalent to (3) $\varphi_1(u_1, u_2) \vee (\varphi_2(u_1) \wedge u_1 = u_2) \rightarrow P(u_1, u_2)$. The algorithm replaces the original set of dependencies by a slight variation of (3), and then simply reverses the arrow to obtain a maximum recovery for $\mathcal{M}_2$.

Next we give a detailed description of the algorithm for full FO-TO-CQ dependencies. In this description, given tuples $\bar{x} = (x_1, \ldots, x_k)$ and $\bar{u} = (u_1, \ldots, u_k)$, we use formula $\bar{x} = \bar{u}$ as a shorthand for $x_1 = u_1 \wedge \cdots \wedge x_k = u_k$.

Algorithm MAXIMUMRECOVERYFULL($\mathcal{M}$)
Input: An st-mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ where Σ is a set of full FO-TO-CQ dependencies, and every dependency has a single atom in its consequent.
Output: A ts-mapping $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$ where Σ' is a set of CQ-TO-FO dependencies, and $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$.

Step 1: (*Create the premises of* Σ') Without loss of generality, assume that the sets of variables of the dependencies of Σ are pairwise disjoint. Let $\mathcal{P}$ be the set of atoms $R(\bar{x})$ such that, $R(\bar{x})$ is the consequent of a dependency in Σ.

Step 2: (*Create the consequents of* Σ') For every formula $R(\bar{x})$ in $\mathcal{P}$, create a set $\mathcal{C}_{R(\bar{x})}$ as follows. Initially $\mathcal{C}_{R(\bar{x})} = \emptyset$. Then for every dependency $\varphi(\bar{u}) \rightarrow R(\bar{u}) \in \Sigma$ do the following. If $\bar{u}$ and $\bar{x}$ are equal, then add $\varphi(\bar{u})$ to $\mathcal{C}_{R(\bar{x})}$. Otherwise, $\bar{x}$ and $\bar{u}$ are disjoint, and formula $\exists \bar{u} \, (\varphi(\bar{u}) \wedge \bar{u} = \bar{x})$ is added to $\mathcal{C}_{R(\bar{x})}$.

Step 3: (*Construct* Σ') For every formula $R(\bar{x})$ in $\mathcal{P}$, add to Σ' dependency $R(\bar{x}) \rightarrow \alpha(\bar{x})$, where $\alpha(\bar{x})$ is the disjunction of all the formulas in $\mathcal{C}_{R(\bar{x})}$. Return $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$. □

From Theorems 5.2 and 5.3, we know that the above algorithm computes an inverse (quasi-inverse) if Σ is an invertible (quasi-invertible) set of full st-tgds. In [10], algorithms for computing inverses and quasi-inverses are proposed. Let $\|\Sigma\|$ denote the size of Σ. The algorithm in [10] for computing an inverse of a set Σ of full st-tgds returns a set Σ' of CQ$^{\neq}$-TO-CQ dependencies of exponential size in $\|\Sigma\|$. The algorithm in [10] for computing a quasi-inverse of a set Σ of full st-tgds returns a set Σ' of CQ$^{\neq}$-TO-UCQ dependencies which is also of exponential size in $\|\Sigma\|$. In both cases, our algorithm works in quadratic time and returns a set Σ' of CQ-TO-UCQ$^{=}$ dependencies which is of quadratic size in $\|\Sigma\|$.

THEOREM 6.1. *Let $\mathcal{M}$ be an st-mapping specified by a set Σ of full* FO-TO-CQ *st-dependencies, each dependency with a single atom in its consequent. Then* MAXIMUMRECOVERYFULL($\mathcal{M}$) *computes a maximum recovery of $\mathcal{M}$ in time $O(\|\Sigma\|^2)$, which is specified by a set of* CQ-TO-FO *dependencies.*

The algorithm is much more involved for the case of non-full FO-TO-CQ dependencies. As an example, consider a mapping $\mathcal{M}_3$ specified by: (1) $\varphi_1(x_1, x_2) \rightarrow \exists v(P(x_1, v) \wedge R(v, x_2))$, (2) $\varphi_2(y_1, y_2) \rightarrow P(y_1, y_2)$, and (3) $\varphi_3(z_1, z_2) \rightarrow R(z_1, z_2)$. In this case, the conjunction of the consequents of (2) and (3) implies the consequent of (1) when y_2 is equal to z_1 and both are existentially quantified. The algorithm replaces (1) by a slight variation of (4) $\beta(u_1, u_2) \rightarrow \exists v(P(u_1, v) \wedge R(v, u_2))$, where $\beta(u_1, u_2)$ is equal to $\varphi_1(u_1, u_2) \vee \exists y_2 \exists z_1(\varphi_2(u_1, y_2) \wedge \varphi_3(z_1, u_2) \wedge y_2 = z_1)$, and then it reverses the arrows of (2), (3) and (4) to generate a maximum recovery of $\mathcal{M}_3$. Given that (1) is a non-full dependency, in this case we also need to impose an additional constraint. When reversing (2), the algorithm needs to force variable y_2 in $P(y_1, y_2)$ to take values only from the set $\mathbf{C}$, that is, the algorithm uses dependency $P(y_1, y_2) \wedge \mathbf{C}(y_2) \rightarrow \varphi_2(y_1, y_2)$ instead of $P(y_1, y_2) \rightarrow \varphi_2(y_1, y_2)$. This is because, given a source instance I such that $I \models \varphi_1(a, b)$, dependency (1) could be satisfied by including a tuple of the form $P(a, n)$ in a target instance, where $n \in \mathbf{N}$, and value n should not be passed to a source instance by a recovery (see Proposition 7.1 for a formal justification for the use of predicate $\mathbf{C}(\cdot)$).

We now introduce the terminology used in the algorithm for the case of non-full FO-TO-CQ dependencies. The basic notion used in the algorithm is that of *existential replacement*. In an existential replacement of a formula β, we are allowed to existentially quantify some of the positions of the free variables of β. For example, if $\beta(x_1, x_2, x_3) = P(x_1, x_2) \wedge R(x_2, x_3)$, then two existential replacements of $\beta(x_1, x_2, x_3)$ are $\gamma_1(x_2) = \exists u \exists v \, (P(u, x_2) \wedge R(x_2, v))$ and $\gamma_2(x_1, x_2, x_3) = \exists z \, (P(x_1, z) \wedge R(x_2, x_3))$. We note that both γ_1 and γ_2 are implied by β. In an existential replacement, we are also allowed to use the same quantifier for different positions. For example, $\gamma_3(x_2) = \exists w \, (P(w, x_2) \wedge R(x_2, w))$ is also an existential replacement of β. We note that γ_3 is implied by β if x_1 and x_3 have the same value, that is, $\beta(x_1, x_2, x_3) \wedge x_1 = x_3$ implies γ_3. In an existential replacement, these equalities are also included. Formally, given a formula $\beta(\bar{x})$ where $\bar{x} = (x_1, \ldots, x_k)$ is a tuple of distinct variables, an *existential replacement* of $\beta(\bar{x})$ is a pair of formulas $(\exists \bar{z} \, \gamma(\bar{x}', \bar{z}), \theta(\bar{x}''))$, where: (1) $\exists \bar{z} \, \gamma(\bar{x}', \bar{z})$ is obtained from $\beta(\bar{x})$ by existentially quantifying some of the position of the free variables of $\beta(\bar{x})$, and $\bar{z}$ is the tuple of fresh variables used in these quantifications, (2) $\theta(\bar{x}'')$ is a conjunction of equalities such that, $x_i = x_j$ is in θ ($1 \leq i, j \leq k$ and $i \neq j$) if we replace a position with variable x_i and a position with variable x_j by the same variable z from $\bar{z}$, and (3) $\bar{x}'$ and $\bar{x}''$ are the tuples of free variables of $\exists \bar{z} \, \gamma(\bar{x}', \bar{z})$ and θ, respectively. Notice that $\exists \bar{z} \, \gamma(\bar{x}', \bar{z})$ is a logical consequence of $\beta(\bar{x}) \wedge \theta(\bar{x}'')$. For example, the followings are existential replacements of the formula $\beta(x_1, x_2, x_3) = \exists y_1 \, (R(x_1, x_2, y_1) \wedge T(y_1, x_3, x_2))$:

$$\big(\exists z_1 \exists z_2 \exists y_1 \, (R(z_1, x_2, y_1) \wedge T(y_1, x_3, z_2)), \ \underline{\text{true}}\big),$$
$$\big(\exists z_1 \exists z_2 \exists y_1 \, (R(z_1, z_1, y_1) \wedge T(y_1, z_2, z_2)), \ x_1 = x_2 \wedge x_3 = x_2\big).$$

In the first existential replacement above, we include sentence $\underline{\text{true}}$ since no distinct variables are replaced by the same variable from

(z_1, z_2). In the algorithm, we use the following terminology for tuples of variables: $\bar{x} \subseteq \bar{y}$ indicates that every variable in $\bar{x}$ is also mentioned in $\bar{y}$, $(\bar{x}, \bar{y})$ is a tuple of variables obtained by placing the variables of $\bar{x}$ followed by the variables of $\bar{y}$, $f : \bar{x} \to \bar{y}$ is a substitution that replaces every variable of $\bar{x}$ by a variable of $\bar{y}$ (f is not necessarily a 1-1 function), and $f(\bar{x})$ is a tuple of variables obtained by replacing every variable x in $\bar{x}$ by $f(x)$. Furthermore, if $\bar{x} = (x_1, \ldots, x_k)$, then $\mathbf{C}(\bar{x})$ is a shorthand for $\mathbf{C}(x_1) \wedge \cdots \wedge \mathbf{C}(x_k)$.

Algorithm MAXIMUMRECOVERY($\mathcal{M}$)
Input: An st-mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ where Σ is a set of FO-TO-CQ dependencies.
Output: A ts-mapping $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$ where Σ' is a set of $\text{CQ}^{\mathbf{C}}$-TO-FO dependencies and $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$.

Step 1: (*Create the premises of* Σ') Let $\mathcal{P} = \emptyset$. Then for every dependency $\varphi(\bar{x}) \to \exists \bar{y}\, \psi(\bar{x}, \bar{y})$ in Σ, add $\exists \bar{y}\, \psi(\bar{x}, \bar{y})$ to $\mathcal{P}$.

Step 2: (*Create the consequents of* Σ') For every formula $\exists \bar{y}\, \psi(\bar{x}, \bar{y})$ in $\mathcal{P}$, create a set $\mathcal{C}_{\exists \bar{y}\, \psi(\bar{x}, \bar{y})}$ as follows. Let $\mathcal{C}_{\exists \bar{y}\, \psi(\bar{x}, \bar{y})} = \emptyset$ and m be the number of atoms in $\psi(\bar{x}, \bar{y})$. Then for every $p \in \{1, \ldots, m\}$ and tuple $(\sigma_1, \ldots, \sigma_p) \in \Sigma^p$ do the following.

Let $(\xi_1, \ldots, \xi_p)$ be a tuple obtained from $(\sigma_1, \ldots, \sigma_p)$ by renaming the variables of the formulas σ_1, ..., σ_p in such a way that the sets of variables of the formulas ξ_1, ..., ξ_p are pairwise disjoint. Assume that ξ_i is equal to $\varphi_i(\bar{u}_i) \to \exists \bar{v}_i\, \psi_i(\bar{u}_i, \bar{v}_i)$, where $\bar{u}_i$ and $\bar{v}_i$ are tuples of distinct variables. Then do the following for every tuple $(\chi_1(\bar{w}_1, \bar{z}_1), \ldots, \chi_p(\bar{w}_p, \bar{z}_p))$, where $\chi_i(\bar{w}_i, \bar{z}_i)$ is a nonempty conjunction of atoms from $\psi_i(\bar{u}_i, \bar{v}_i)$, $\bar{w}_i \subseteq \bar{u}_i$, $\bar{z}_i \subseteq \bar{v}_i$ and both $\bar{w}_i$ and $\bar{z}_i$ are tuples of distinct variables. Let $\exists \bar{z}\, \chi(\bar{w}, \bar{z})$ be the formula $\exists \bar{z}_1 \cdots \exists \bar{z}_p (\chi_1(\bar{w}_1, \bar{z}_1) \wedge \cdots \wedge \chi_p(\bar{w}_p, \bar{z}_p))$ with $\bar{w} = (\bar{w}_1, \ldots, \bar{w}_p)$ and $\bar{z} = (\bar{z}_1, \ldots, \bar{z}_p)$. Then for every existential replacement $(\exists \bar{s} \exists \bar{z}\, \gamma(\bar{w}', \bar{z}, \bar{s}), \theta(\bar{w}''))$ of $\exists \bar{z}\, \chi(\bar{w}, \bar{z})$ (up to renaming of variables in $\bar{s}$), and for every pair of variable substitutions $f : \bar{x} \to \bar{x}$ and $g : \bar{w}' \to \bar{x}$, check whether there exists a variable substitution $h : \bar{y} \to (\bar{z}, \bar{s})$ such that $\psi(f(\bar{x}), h(\bar{y}))$ and $\gamma(g(\bar{w}'), \bar{z}, \bar{s})$ are syntactically equal (up to reordering of atoms). If this is the case, then add to $\mathcal{C}_{\exists \bar{y}\, \psi(\bar{x}, \bar{y})}$ the following formula:

$$\exists \bar{u}_1 \cdots \exists \bar{u}_p \, (\bigwedge_{i=1}^{p} \varphi_i(\bar{u}_i) \wedge \theta(\bar{w}'') \wedge \bar{x} = f(\bar{x}) \wedge \bar{w}' = g(\bar{w}')) \quad (5)$$

Step 3: (*Construct* Σ') For every formula $\exists \bar{y}\, \psi(\bar{x}, \bar{y})$ in $\mathcal{P}$, add to Σ' dependency $\exists \bar{y}\, \psi(\bar{x}, \bar{y}) \wedge \mathbf{C}(\bar{x}) \to \alpha(\bar{x})$, where $\alpha(\bar{x})$ is the disjunction of all the formulas in $\mathcal{C}_{\exists \bar{y}\, \psi(\bar{x}, \bar{y})}$. Return $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$. □

Notice that $\bar{x}$ is the set of free variables of formula (5) since both $\bar{w}'$ and $\bar{w}''$ are subsets of $(\bar{u}_1, \ldots, \bar{u}_p)$. Also notice that since $\psi(f(\bar{x}), h(\bar{y}))$ and $\gamma(g(\bar{w}'), \bar{z}, \bar{s})$ are identical, $f : \bar{x} \to \bar{x}$, $g : \bar{w}' \to \bar{x}$ and $h : \bar{y} \to (\bar{z}, \bar{s})$, we can infer that every variable x in $\bar{x}$ is equal to some variable u in $(\bar{u}_1, \ldots, \bar{u}_p)$ from the subformula $\bar{x} = f(\bar{x}) \wedge \bar{w}' = g(\bar{w}')$. This implies that formula (5) is domain independent since each formula $\varphi_i(\bar{u}_i)$ is domain independent.

EXAMPLE 6.2. Assume that Σ contains dependencies:

$$\varphi_1(x_1, x_2, x_3) \to \exists y_1 \, (R(x_1, x_2, y_1) \wedge R(y_1, x_3, x_3)), \quad (6)$$
$$\varphi_2(x_1, x_2) \to R(x_1, x_1, x_2). \quad (7)$$

In Step 1 of the algorithm, $\exists y_1 (R(x_1, x_2, y_1) \wedge R(y_1, x_3, x_3))$ and $R(x_1, x_1, x_2)$ are added to the set $\mathcal{P}$. Let $\psi(x_1, x_2, x_3, y_1) = R(x_1, x_2, y_1) \wedge R(y_1, x_3, x_3)$. Given that $\psi(x_1, x_2, x_3, y_1)$ has two atoms, the algorithm considers the two tuples in Σ^1 and the four tuples in $\Sigma^2 = \Sigma \times \Sigma$ to construct the set $\mathcal{C}_{\exists y_1 \psi(x_1, x_2, x_3, y_1)}$. We show here how tuple $(\sigma_1, \sigma_2) \in \Sigma^2$ is processed, where both σ_1 and σ_2 are equal to formula (7).

First, the algorithm generates a tuple (ξ_1, ξ_2) from (σ_1, σ_2) by renaming the variables of σ_1 and σ_2. Assume that ξ_1 is equal to $\varphi_2(u_1, u_2) \to R(u_1, u_1, u_2)$ and ξ_2 is equal to $\varphi_2(u_3, u_4) \to R(u_3, u_3, u_4)$. The algorithm continues by considering all the tuples $(\chi_1(u_1, u_2), \chi_2(u_3, u_4))$ such that $\chi_1(u_1, u_2)$ and $\chi_2(u_3, u_4)$ are nonempty conjunctions of atoms from the consequents of ξ_1 and ξ_2, respectively. In this case, the algorithm only needs to consider tuple $(R(u_1, u_1, u_2), R(u_3, u_3, u_4))$. The algorithm uses this tuple to construct formula $\chi(u_1, u_2, u_3, u_4) = R(u_1, u_1, u_2) \wedge R(u_3, u_3, u_4)$, and then looks for all the existential replacements of $\chi(u_1, u_2, u_3, u_4)$ that can be made identical to $\exists y_1\, \psi(x_1, x_2, x_3, y_1)$ by substituting some variables. For instance, $(\exists s_1 \, (R(u_1, u_1, s_1) \wedge R(s_1, u_3, u_4)), u_2 = u_3)$ is one of these existential replacements: $R(g(u_1), g(u_1), s_1) \wedge R(s_1, g(u_3), g(u_4))$ is syntactically equal to $\psi(f(x_1), f(x_2), f(x_3), h(y_1))$, where $f(x_1) = f(x_2) = x_1$, $f(x_3) = x_3$, $g(u_1) = x_1$, $g(u_3) = g(u_4) = x_3$ and $h(y_1) = s_1$. The algorithm uses functions f, g and condition $u_2 = u_3$ from the existential replacement to generate the following formula $\beta(x_1, x_2, x_3)$ (omitting trivial equalities like $x_1 = x_1$):

$$\exists u_1 \exists u_2 \exists u_3 \exists u_4 \, (\varphi_2(u_1, u_2) \wedge \varphi_2(u_3, u_4) \wedge u_2 = u_3 \wedge x_2 = x_1 \wedge u_1 = x_1 \wedge u_3 = x_3 \wedge u_4 = x_3).$$

Formula $\beta(x_1, x_2, x_3)$ is added to $\mathcal{C}_{\exists y_1\, \psi(x_1, x_2, x_3, y_1)}$. It is important to notice that $\beta(x_1, x_2, x_3)$ represents a way to deduce $\exists y_1\, \psi(x_1, x_2, x_3, y_1)$ from φ_2, that is, $\beta(x_1, x_2, x_3) \to \exists y_1\, \psi(x_1, x_2, x_3, y_1)$ is a logical consequence of formula (7). In the last step of the algorithm, a ts-dependency is generated for every formula in $\mathcal{P}$. Formula $\beta(x_1, x_2, x_3)$ is one of the disjuncts in the consequent of the ts-dependency for $\exists y_1\, \psi(x_1, x_2, x_3, y_1)$. □

THEOREM 6.3. *Let $\mathcal{M}$ be an st-mapping specified by a set Σ of* FO-TO-CQ *dependencies. Then* MAXIMUMRECOVERY($\mathcal{M}$) *computes a maximum recovery of $\mathcal{M}$ in exponential time, which is specified by a set of* $\text{CQ}^{\mathbf{C}}$-TO-FO *dependencies.*

As for the case of full st-tgds, from Theorems 5.2 and 5.3, we have that if Σ is an invertible (quasi-invertible) set of st-tgds, then MAXIMUMRECOVERY computes an inverse (quasi-inverse) of Σ. In fact, if Σ is a set of st-tgds, then the algorithm returns a set of $\text{CQ}^{\mathbf{C}}$-TO-$\text{UCQ}^{=}$ dependencies. It is important to notice that our algorithm works not only for st-tgds but also for the larger class of FO-TO-CQ dependencies. For the latter class, it is not clear how to extend the algorithms from [10] to produce inverses and quasi-inverses, as the notion of generator used in these algorithms (Definition 4.2 in [10]) becomes undecidable for FO-TO-CQ dependencies.

Computing maximum recoveries for mappings with source dependencies. By using Lemma 4.4, we can extend algorithm MAXIMUMRECOVERY to handle source constraints. Given an st-mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathbf{st}}, \Gamma_{\mathbf{s}})$, where $\Sigma_{\mathbf{st}}$ is a set of FO-TO-CQ st-dependencies and $\Gamma_{\mathbf{s}}$ is a set of source FO-dependencies, algorithm MAXIMUMRECOVERY can be used to produce a maximum recovery $\mathcal{M}_1^{\star} = (\mathbf{T}, \mathbf{S}, \Sigma_{\mathbf{ts}})$ for st-mapping $\mathcal{M}_1 = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathbf{st}})$, and then $\mathcal{M}^{\star} = (\mathbf{T}, \mathbf{S}, \Sigma_{\mathbf{ts}}, \Gamma_{\mathbf{s}})$ is output as a maximum recovery of $\mathcal{M}$.

7. THE LANGUAGE OF MAXIMUM RECOVERIES

Given a mapping $\mathcal{M}$ specified by a set of FO-TO-CQ dependencies, algorithm MAXIMUMRECOVERY produces a maximum recovery of $\mathcal{M}$ that is specified by a set of $\text{CQ}^{\mathbf{C}}$-TO-FO dependencies. In this section, we study some properties of the language needed to express

maximum recoveries, which provide justification for the language used in the output of algorithm MAXIMUMRECOVERY. Moreover, we also show that the extension of this algorithm to handle target constraints is not immediate, as there exists a mapping specified by a set of FO-TO-CQ dependencies plus a set of target egds that has no maximum recovery specified by a set of FO-sentences, and the same holds for a weakly acyclic set of target tgds.

A first question about the output of MAXIMUMRECOVERY is whether predicate $\mathbf{C}(\cdot)$ is really needed. In [10], it is proved that $\mathbf{C}(\cdot)$ is needed when computing quasi-inverses of st-mappings specified by st-tgds, if quasi-inverses are expressed using st-tgds with inequalities in the premises and disjunction in the consequents. Here we show that $\mathbf{C}(\cdot)$ is needed when computing maximum recoveries for st-mappings specified by st-tgds, even if we allow the full power of FO to express maximum recoveries.

PROPOSITION 7.1. *There exists an st-mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ specified by a set Σ of st-tgds that has no maximum recovery specified by a set of* FO*-sentences over $\mathbf{S} \cup \mathbf{T}$ not using predicate $\mathbf{C}(\cdot)$.*

In Section 4, it is proved that adding disjunction, inequalities or negation to the consequents of FO-TO-CQ dependencies generates st-mappings that do not necessarily have maximum recoveries. Hence, it would be desirable to stay in the class of FO-TO-CQ dependencies when dealing with maximum recoveries. In particular, it would be desirable to have an algorithm that takes as input a set Σ of FO-TO-CQ st-dependencies, and produces a set Σ' of $\text{FO}^{\mathbf{C}}$-TO-CQ ts-dependencies which is a maximum recovery of Σ. Thus, a second important question about the algorithm MAXIMUMRECOVERY is whether it could be modified to produce a set of $\text{FO}^{\mathbf{C}}$-TO-CQ ts-dependencies as output. Unfortunately, the following proposition shows that this could not be the case, even if we allow disjunction in the consequents of the output dependencies.

PROPOSITION 7.2. *There exists an st-mapping specified by a set of* FO-TO-CQ *st-dependencies that has no maximum recovery specified by a set of* $\text{FO}^{\mathbf{C}}$-TO-UCQ *ts-dependencies.*

In fact, from the proof of Proposition 7.2 , we obtain that there exists an st-mapping specified by a set of $\text{CQ}^{\neq}$-TO-CQ dependencies that has no maximum recovery specified by a set of $\text{FO}^{\mathbf{C}}$-TO-UCQ dependencies.

A third question about the output of MAXIMUMRECOVERY is whether the full power of FO is really needed in the consequents of the dependencies returned by the algorithm. For example, could it be the case that $\text{CQ}^{\mathbf{C}}$-TO-$\text{UCQ}^{=,\neg}$ dependencies suffice to specify maximum recoveries for st-mappings specified by FO-TO-CQ dependencies? Theorem 7.3 below shows that this could not be the case. In fact, we show that for $\mathcal{L}$ and $\mathcal{L}'$ fragments of FO (satisfying some regularity conditions), if $\text{CQ}^{\mathbf{C}}$-TO-$\mathcal{L}'$ dependencies suffice to specify maximum recoveries for mappings given by $\mathcal{L}$-TO-CQ dependencies, then $\mathcal{L}'$ must be at least as expressive as $\mathcal{L}$.

In Theorem 7.3, we use the following terminology. We say that a fragment $\mathcal{L}$ of FO is closed under conjunction and existential quantification, if for every pair of formulas φ and ψ in $\mathcal{L}$, there exist formulas α and β in $\mathcal{L}$ such that, α is equivalent to $\varphi \wedge \psi$ and β is equivalent to $\exists x\, \varphi$. Furthermore, we say that $\mathcal{L}$ is closed under free-variable substitution, if for every formula $\varphi(\bar{x})$ in $\mathcal{L}$ and substitution μ for $\bar{x}$, there exists a formula $\alpha(\mu(\bar{x}))$ in $\mathcal{L}$ that is equivalent to $\varphi(\mu(\bar{x}))$. Notice that all the fragments of FO used in this paper are closed under conjunction, existential quantification and free-variable substitution. Finally, we say that an FO-sentence Φ is nontrivial if Φ is neither a contradiction nor a valid sentence.

THEOREM 7.3. *Let $\mathcal{L}$ and $\mathcal{L}'$ be fragments of* FO *that are closed under conjunction, existential quantification and free-variable substitution. If there exists a nontrivial sentence Φ in $\mathcal{L}$ that is not equivalent to any sentence in $\mathcal{L}'$, then there exists an st-mapping specified by a set of $\mathcal{L}$-TO-CQ st-dependencies that has no maximum recovery specified by a set of $\text{CQ}^{\mathbf{C}}$-TO-$\mathcal{L}'$ ts-dependencies.*

In Section 6, we show that algorithm MAXIMUMRECOVERY can be extended to handle arbitrary source constraints. In Theorem 4.7, we show that if an st-mapping $\mathcal{M}$ is specified by a set of FO-TO-CQ dependencies, a set of target egds and a weakly acyclic set of target tgds, then $\mathcal{M}$ has a maximum recovery. Thus, a natural question is whether MAXIMUMRECOVERY can be extended to this class of mappings with target dependencies. Unfortunately, the following proposition shows that the extension of the algorithm to handle target constraints is by no means immediate.

PROPOSITION 7.4.

(1) There exists an st-mapping $\mathcal{M}$ specified by a set of st-tgds plus a set of target egds that has no maximum recovery specified by a set of FO*-sentences.*

(2) There exists an st-mapping $\mathcal{M}$ specified by a set of st-tgds plus a weakly acyclic set of target tgds that has no maximum recovery specified by a set of FO*-sentences.*

8. COMPLEXITY RESULTS

In [7], two problems are identified as important decision problems for the notion of inverse: (1) to check whether a mapping $\mathcal{M}$ is invertible, and (2) to check whether a mapping $\mathcal{M}_2$ is an inverse of a mapping $\mathcal{M}_1$. These questions are considered in the context of st-tgds in [7]. In this context, the problem of verifying whether a mapping $\mathcal{M}$ has a maximum recovery becomes trivial, as every mapping specified by this type of dependencies admits a maximum recovery. In fact, this question is also trivial for the larger class of mappings specified by FO-TO-CQ dependencies. The goal of this section is to show that the problem of verifying, given mappings $\mathcal{M}$ and $\mathcal{M}'$, whether $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$ is undecidable. To this end, we prove a stronger result, namely that undecidability still holds if maximum recovery is replaced by the weaker notion of recovery in the previous problem. We start by considering mappings specified by full st-tgds.

PROPOSITION 8.1. *The problem of verifying, given mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$, where Σ is a set of full st-tgds and Σ' is a set of ts-tgds, whether $\mathcal{M}'$ is a recovery of $\mathcal{M}$ is Π_2^P-complete. Moreover, if Σ' is a set of full ts-tgds, then this problem is coNP-complete.*

Proposition 8.1 is in sharp contrast with the results of [7], where it is shown that the problem of verifying, given schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$, with Σ a set of full st-tgds and Σ' a set of full ts-tgds, whether $\mathcal{M}'$ is an inverse of $\mathcal{M}$ is DP-complete. The lower complexity for the case of the recovery is not surprising as the notion of recovery is much weaker than the notion of inverse. However, the situation is different for non-full st-tgds.

THEOREM 8.2. *The problem of verifying, given mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$, where Σ is a set of st-tgds and Σ' is a set of ts-tgds, whether $\mathcal{M}'$ is a recovery of $\mathcal{M}$ is undecidable.*

As a corollary of Theorem 8.2 and the results in Section 5, we obtain the following undecidability results for maximum recoveries, inverses and quasi-inverses.

COROLLARY 8.3. *The problems of verifying, given mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$, where Σ is a set of st-tgds and Σ' is a set of ts-tgds, whether (1) $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$, (2) $\mathcal{M}'$ is an inverse of $\mathcal{M}$, and (3) $\mathcal{M}'$ is a quasi-inverse of $\mathcal{M}$, are all undecidable.*

9. MAXIMAL RECOVERY

Although maximum recoveries exist for a large class of mappings, there are classes of practical interest for which the existence of maximum recoveries is not guaranteed. To overcome this limitation, one has to look for a weaker notion. A straightforward relaxation is to consider not maximum but maximal recoveries. In this section, we report our initial results about maximal recoveries, providing a necessary and sufficient condition for the existence of maximal recoveries, and showing that the notion of maximal recovery strictly generalizes the notion of maximum recovery. In fact, we show that maximal recoveries exist for the larger class of st-mappings specified by FO-TO-UCQ$^{\neq}$ dependencies. This result shows that the notion of maximal recovery is a promising direction for further research.

Recall that for two recoveries $\mathcal{M}'$ and $\mathcal{M}''$ of a mapping $\mathcal{M}$, we say that $\mathcal{M}'$ is at least as informative as $\mathcal{M}''$ for $\mathcal{M}$, and write $\mathcal{M}'' \preceq_{\mathcal{M}} \mathcal{M}'$, if $\mathcal{M} \circ \mathcal{M}' \subseteq \mathcal{M} \circ \mathcal{M}''$. If $\mathcal{M}'' \preceq_{\mathcal{M}} \mathcal{M}'$ and $\mathcal{M}' \not\preceq_{\mathcal{M}} \mathcal{M}''$, then we say that $\mathcal{M}'$ is *more informative than* $\mathcal{M}''$ for $\mathcal{M}$, and we write $\mathcal{M}'' \prec_{\mathcal{M}} \mathcal{M}'$.

DEFINITION 9.1. *Let $\mathcal{M}'$ be a recovery of a mapping $\mathcal{M}$. We say that $\mathcal{M}'$ is a* maximal recovery *of $\mathcal{M}$, if there is no recovery $\mathcal{M}''$ of $\mathcal{M}$ such that $\mathcal{M}' \prec_{\mathcal{M}} \mathcal{M}''$.*

That is, $\mathcal{M}'$ is a maximal recovery of $\mathcal{M}$ if there is no other recovery that is more informative for $\mathcal{M}$ than $\mathcal{M}'$. In Section 3, we show that the notion of witness can be used to characterize the existence of maximum recoveries. In the following definition, we introduce a relaxation of this notion that can be used to provide a necessary and sufficient condition for the existence of maximal recoveries. In this definition, $\mathcal{M}^{-1}$ denotes mapping $\{(J, I) \mid (I, J) \in \mathcal{M}\}$.

DEFINITION 9.2. *Let $\mathcal{M}$ be a mapping. Instance J is a* partial-witness for I under $\mathcal{M}$ *iff for every $J' \in \mathrm{Sol}_{\mathcal{M}}(I)$, it is not the case that $\mathrm{Sol}_{\mathcal{M}^{-1}}(J') \subsetneq \mathrm{Sol}_{\mathcal{M}^{-1}}(J)$.*

It is not difficult to prove that an instance J is a witness for an instance I under a mapping $\mathcal{M}$ if and only if for every $J' \in \mathrm{Sol}_{\mathcal{M}}(I)$, it is the case that $\mathrm{Sol}_{\mathcal{M}^{-1}}(J) \subseteq \mathrm{Sol}_{\mathcal{M}^{-1}}(J')$. Thus, the notion of partial-witness is a relaxation of the notion of witness. Notice that a partial-witness for an instance I is not necessarily a solution for I. If J is both a partial-witness and a solution for I under $\mathcal{M}$, then we say that J is a *partial-witness solution* for I under $\mathcal{M}$.

THEOREM 9.3. *$\mathcal{M}$ has a maximal recovery iff for every $I \in \mathrm{dom}(\mathcal{M})$, there exists a partial-witness solution for I under $\mathcal{M}$.*

The following theorem identifies an important class of st-mappings for which the existence of maximal recoveries is guaranteed, and also shows that the notion of maximal recovery strictly generalizes the notion of maximum recovery (see Proposition 4.8).

THEOREM 9.4. *If $\mathcal{M}$ is an st-mapping specified by a set of* FO-TO-UCQ$^{\neq}$ *st-dependencies, then $\mathcal{M}$ has a maximal recovery.*

Our last result shows that there exist mappings that do not have maximal recoveries.

PROPOSITION 9.5. *There exists an st-mapping $\mathcal{M}$ specified by an* FO*-sentence that has no maximal recovery.*

In the proof of the above proposition, we use an st-mapping $\mathcal{M}$ specified by FO-sentence $\forall x(P(x) \rightarrow \neg R(x))$. Notice that formula $\forall x(P(x) \rightarrow \neg R(x))$ does not satisfy the safety condition imposed on FO-TO-FO dependencies since $\neg R(x)$ is not domain-independent. In particular, formula $\forall x(P(x) \rightarrow \neg R(x))$ is not a CQ-TO-CQ$^{\neg}$ dependency since $\neg R(x)$ is not a CQ$^{\neg}$-query. It is open whether for every st-mapping $\mathcal{M}$ specified by a set of CQ-TO-CQ$^{\neg}$ st-dependencies, $\mathcal{M}$ has a maximal recovery.

10. CONCLUDING REMARKS

In this paper, we introduce the notion of a recovery of a mapping: a reverse mapping that recovers sound information. We introduce an order relation on recoveries, from which the notion of maximum recovery naturally arises. As our results show, maximum recoveries possess good properties that justify their usage in data exchange and metadata management. Most notably, maximum recoveries exist for the large class of mappings specified by FO-TO-CQ dependencies. For mappings that do not have maximum recoveries, the notion of maximal recovery is a promising direction to further explore.

An important open problem is the decidability of the existence of maximum recoveries for classes of dependencies beyond FO-TO-CQ, for example, the classes of FO-TO-UCQ and FO-TO-CQ$^{\neq}$ dependencies. Regarding maximal recoveries, the development of algorithms for computing them is an interesting area for future work. Although we have concentrated on the relational case, a characteristic of the notions of recovery and maximum recovery is that, they are bounded neither to a specific data model nor to a specific language for expressing schema mappings. As part of our future work, we plan to study these notions for other semantics, e.g. closed world semantics [14], and for other data models, e.g. XML.

Acknowledgments: We are very grateful to Pablo Barceló, Leopoldo Bertossi, Alejandro Cataldo, Ronald Fagin, Phokion Kolaitis, Leonid Libkin, and Alan Nash, for many helpful discussions, and to the anonymous referees for their comments. The authors were supported by: Arenas and Riveros – FONDECYT grant 1070732; Pérez – CONICYT Ph.D. Scholarship; Arenas and Pérez – grant P04-067-F from the Millennium Nucleus Center for Web Research.

11. REFERENCES

[1] F. Afrati, C. Li, and V. Pavlaki. Data exchange with arithmetic comparisons. Technical report, UCI ICS, 2006.

[2] M. Arenas, P. Barceló, R. Fagin, and L. Libkin. Locally consistent transformations and query answering in data exchange. In *PODS*, pages 229–240, 2004.

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

[4] P. Bernstein and S. Melnik. Model management 2.0: manipulating richer mappings. In *SIGMOD*, pages 1–12, 2007.

[5] A. Deutsch and V. Tannen. Reformulation of XML queries and constraints. In *ICDT*, pages 225–241, 2003.

[6] R. Fagin. Horn clauses and database dependencies. *JACM*, 29(4):952–985, 1982.

[7] R. Fagin. Inverting schema mappings. *TODS*, 32(4), 2007.

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

[9] R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. Composing schema mappings: second-order dependencies to the rescue. *TODS*, 30(4):994–1055, 2005.

[10] R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. Quasi-inverses of schema mappings. In *PODS*, pages 123–132, 2007.

[11] G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. On reconciling data exchange, data integration, and peer data management. In *PODS*, pages 133–142, 2007.

[12] G. Gottlob and A. Nash. Data exchange: computing cores in polynomial time. In *PODS*, pages 40–49, 2006.

[13] P. G. Kolaitis, J. Panttaja, and W.-C. Tan. The complexity of data exchange. In *PODS*, pages 30–39, 2006.

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

[15] S. Melnik. *Generic Model Management: Concepts and Algorithms*, volume 2967 of *LNCS*. Springer, 2004.

[16] S. Melnik, P. Bernstein, A. Y. Halevy, and E. Rahm. Supporting executable mappings in model management. In *SIGMOD*, pages 167–178, 2005.