

Reverse Data Exchange: Coping with Nulls

Ronald Fagin
IBM Almaden

Phokion G. Kolaitis
UC Santa Cruz & IBM Almaden

Lucian Popa
IBM Almaden

Wang-Chiew Tan
UC Santa Cruz

ABSTRACT

An inverse of a schema mapping \mathcal{M} is intended to “undo” what \mathcal{M} does, thus providing a way to perform “reverse” data exchange. In recent years, three different formalizations of this concept have been introduced and studied, namely, the notions of an inverse of a schema mapping, a quasi-inverse of a schema mapping, and a maximum recovery of a schema mapping. The study of these notions has been carried out in the context in which source instances are restricted to consist entirely of constants, while target instances may contain both constants and labeled nulls. This restriction on source instances is crucial for obtaining some of the main technical results about these three notions, but, at the same time, limits their usefulness, since reverse data exchange naturally leads to source instances that may contain both constants and labeled nulls.

We develop a new framework for reverse data exchange that supports source instances that may contain nulls, thus overcoming the semantic mismatch between source and target instances of the previous formalizations. The development of this new framework requires a careful reformulation of all the important notions, including the notions of the identity schema mapping, inverse, and maximum recovery. To this effect, we introduce the notions of extended identity schema mapping, extended inverse, and maximum extended recovery, by making systematic use of the homomorphism relation on instances. We give results concerning the existence of extended inverses and of maximum extended recoveries, and results concerning their applications to reverse data exchange and query answering. Moreover, we show that maximum extended recoveries can be used to capture in a quantitative way the amount of *information loss* embodied in a schema mapping specified by source-to-target tuple-generating dependencies.

Categories and Subject Descriptors

H.2.5 [Database Management]: Heterogeneous Databases – Data translation; H.2.4 [Database Management]: Systems – Relational Databases

General Terms

Algorithms, Theory

Keywords

Schema mapping, data exchange, data integration, model management, inverse, quasi-inverse, maximum recovery, chase

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PODS'09, June 29–July 2, 2009, Providence, Rhode Island, USA.
Copyright 2009 ACM 978-1-60558-553-6 /09/06 ...\$5.00.

1. Introduction

Background and Motivation. Schema mappings are high-level specifications of how data from a source schema is to be transformed to data in a different schema, called the target schema. More formally, a schema mapping is a triple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where \mathbf{S} is a source schema, \mathbf{T} is a target schema, and Σ is a set of database dependencies that specify the relationship between \mathbf{S} and \mathbf{T} . In recent years, an extensive investigation of schema mappings and of their uses in data exchange and data integration has been carried out. One particular direction of this investigation has focused on the study of operators on schema mappings. Among all such operators originally introduced in [4], the composition operator and the inverse operator have been recognized as two fundamental ones. The intuition behind these two operators is as follows. Given two schema mappings \mathcal{M}_{12} and \mathcal{M}_{23} such that the target schema of \mathcal{M}_{12} is the same as the source schema of \mathcal{M}_{23} , the composition operator yields a schema mapping \mathcal{M}_{13} that is “equivalent” to the successive application of \mathcal{M}_{12} and \mathcal{M}_{23} , thus providing a way to perform data exchange directly between the source schema of \mathcal{M}_{12} and the target schema of \mathcal{M}_{23} . Given a schema mapping \mathcal{M} , the inverse operator yields a schema mapping \mathcal{M}' that “undoes” what \mathcal{M} did, thus providing a way to do “reverse” data exchange. Clearly, these two operators are of interest in their own right; furthermore, when combined together, they attain even greater power since, in combination, they can be used to analyze schema evolution.

By now, the composition operator has been investigated in depth, and consensus has been achieved on what the definitive semantics for composition is [5, 9, 12, 14]. The state of affairs concerning the inverse operator, however, is more complicated and by far less definitive. In [7], rigorous semantics for the inverse operator was given for the first time. The notion of inverse introduced in [7] turned out to be rather restrictive, as most schema mappings specified by source-to-target tuple generating dependencies (s-t tgds) are not invertible. For this reason, the notion of a *quasi-inverse* of a schema mapping was introduced and studied in [10]. After this, a competing notion of a *maximum recovery* of a schema mapping was introduced and studied in [2]. For invertible schema mappings, the notions of inverse, quasi-inverse, and maximum recovery coincide. In contrast, for non-invertible schema mappings, the notions of quasi-inverse and maximum recovery differ in general. Moreover, every schema mapping specified by a set of s-t tgds has a maximum recovery, whereas there are such schema mappings that are not quasi-invertible.

Their differences notwithstanding, all previous studies of “inverse” operators (including inverse, quasi-inverse, and maximum recovery) have the following basic assumption in common: the source instances are *ground*, i.e., they consist entirely of constants, while, on the contrary, target instances may contain both constants

and labeled nulls (variables). In particular, some of the key technical results in [2, 7, 10] very much depend on the assumption that source instances do not contain labeled nulls. However, applications of “inverse” operators naturally lead to source instances that may contain labeled nulls, in addition to constants. The following example illustrates this scenario.

EXAMPLE 1.1. Let \mathcal{M} be the schema mapping given by the tuple-generating dependency

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

which describes a decomposition of a source relation P into two target relations Q and R . It was shown in [10] that \mathcal{M} is not invertible but is quasi-invertible. Furthermore, a natural “inverse” of \mathcal{M} , which is both a quasi-inverse of \mathcal{M} and a maximum recovery for \mathcal{M} , is the schema mapping \mathcal{M}' given by the following set of “reverse” tgds:

$$\Sigma' = \{ Q(x, y) \rightarrow \exists z P(x, y, z), R(y, z) \rightarrow \exists x P(x, y, z) \}.$$

Consider the ground source instance $I = \{P(a, b, c)\}$, where a, b, c are constants. The result of chasing I with \mathcal{M} (i.e., the result of performing data exchange with \mathcal{M}) is the instance $U = \{Q(a, b), R(b, c)\}$. If we now chase U with \mathcal{M}' (i.e., perform the “reverse” data exchange with \mathcal{M}'), we obtain the source instance $V = \{P(a, b, Z), P(X, b, c)\}$, where Z and X are nulls. Note that V , which is the canonical result of “reverse” data exchange, is no longer a ground instance and, thus, it is ruled out from the semantics. \square

Another limitation of restricting source instances to be ground is that data exchange cannot be performed on source instances that are the result of a prior data exchange with s-t tgds, since, in general, labeled nulls may be generated by chasing source instances with tgds to produce universal solutions [8]. Thus, the preceding considerations suggest that previous studies of inverse, quasi-inverse, and maximum recovery suffer from a *semantic mismatch* that stems from the assumption that only ground instances are allowed.

Summary of Contributions. Our goal in this paper is to develop a new framework for “reverse” data exchange that overcomes this semantic mismatch. This new framework supports source instances with nulls, and makes it possible to recover source instances using reverse data exchange and to permit target instances that result from one data exchange to be used as source instances of another data exchange (thus, in the long run, this framework will enable the analysis of schema evolution using composition and inverse).

The development of this new framework requires a careful reformulation of all the important notions, including the notions of the identity schema mapping (which was used in the definition of inverse [7]), inverse, and maximum recovery. This is so because these earlier notions, which were studied under the assumption that only ground source instances are allowed, lose their desirable properties when schema mappings are used to perform data exchange between source and target instances that may both contain nulls. For example, using the concept of maximum recovery in [2], it can be shown (see Proposition 4.2) that the schema mapping specified by the tgd $P(x, y) \rightarrow \exists z(Q(x, z) \wedge Q(z, y))$ has a maximum recovery when only ground source instances are allowed, but has no maximum recovery when source instances may contain nulls.

The key to finding the “right” notions in the new framework is to replace the *containment* relation $I_1 \subseteq I_2$ between instances by the *homomorphism* relation $I_1 \rightarrow I_2$, which holds if there is a homomorphism from I_1 to I_2 that maps constants to themselves (note that if I_1 is a ground instance, then $I_1 \rightarrow I_2$ if and only

if $I_1 \subseteq I_2$). We use the notation \rightarrow to denote the binary relation $\{(I_1, I_2) : I_1 \rightarrow I_2\}$ between instances that may contain nulls. The next step is to replace the identity schema mapping $\text{Id} = \{(I_1, I_2) : I_1, I_2 \text{ are ground instances such that } I_1 \subseteq I_2\}$ with the *extended identity* schema mapping $e(\text{Id}) = \rightarrow$. Thus,

$$e(\text{Id}) = \{(I_1, I_2) : I_1, I_2 \text{ are instances such that } I_1 \rightarrow I_2\}.$$

In fact, even the notion of a *solution* needs to be reformulated in the new framework. Specifically, we say that a target instance J is an *extended solution* for a source instance I w.r.t. a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ if there are a source instance I' and a target instance J' such that $I \rightarrow I'$, $(I', J') \models \Sigma$, and $J' \rightarrow J$. In effect, an extended solution J for I w.r.t. \mathcal{M} is a solution for I w.r.t. the schema mapping $e(\mathcal{M}) = \rightarrow \circ \mathcal{M} \circ \rightarrow$, which we call the *homomorphic extension* of \mathcal{M} . The intuition behind extended solutions is that, since labeled nulls represent unknown information, it is “legal” to homomorphically map them to other values before or after taking the standard notion of solution. In a sense, the notion of an extended solution is a relaxation of the notion of solution where the exact values of the labeled nulls (in both the source instance and the target instance) do not matter. Further, our definition of extended solutions relaxes the notion of solution even more by allowing the possibility that additional facts may be present (in the spirit of the “open-world assumption”).¹ With these new notions at hand, we can then define the notions of *extended inverse*, *extended recovery*, and *maximum extended recovery* by systematically replacing the identity schema mapping Id by the extended identity schema mapping $e(\text{Id})$, and the composition $\mathcal{M} \circ \mathcal{M}'$ by the composition $e(\mathcal{M}) \circ e(\mathcal{M}')$ of the homomorphic extensions of \mathcal{M} and \mathcal{M}' . Specifically, we say that \mathcal{M}' is an *extended inverse* of \mathcal{M} if $e(\mathcal{M}) \circ e(\mathcal{M}') = e(\text{Id})$. We say that \mathcal{M}' is a *recovery* of \mathcal{M} if for every source instance I , we have that $(I, I) \in e(\mathcal{M}) \circ e(\mathcal{M}')$; finally, we say that \mathcal{M}' is a *maximum extended recovery* of \mathcal{M} if \mathcal{M}' is an extended recovery of \mathcal{M} and for every extended recovery \mathcal{M}'' of \mathcal{M} we have that $e(\mathcal{M}) \circ e(\mathcal{M}') \subseteq e(\mathcal{M}) \circ e(\mathcal{M}'')$.

The first group of our technical results concerns the properties of the extended inverse and its relation to the inverse. We give a necessary and sufficient condition for an arbitrary schema mapping to have an extended inverse and then focus on schema mappings specified by s-t tgds. We show that if \mathcal{M} is specified by s-t tgds and is extended invertible, then it is invertible, but not vice-versa. An extended inverse, however, need not be an inverse (and, of course, an inverse need not be an extended inverse, since invertibility does not imply extended invertibility). We also show that if \mathcal{M} and \mathcal{M}' are schema mappings such that both are specified by tgds, then \mathcal{M}' is an extended inverse of \mathcal{M} if and only if \mathcal{M}' is a *chase-inverse* of \mathcal{M} , that is, if every source instance I is homomorphically equivalent to $\text{chase}_{\mathcal{M}'}(\text{chase}_{\mathcal{M}}(I))$, which is the source instance obtained from I by first chasing with \mathcal{M} and then with \mathcal{M}' . This result reveals that extended inverses specified by tgds are ideally suited for “reverse” data exchange, since if the original source instance I is no longer available, we can recover a homomorphically equivalent one by chasing a universal solution for I with an extended inverse specified by tgds. It should be noted that this result does not hold for (non-extended) inverses.

After this, we focus on maximum extended recoveries and obtain the main technical results of this paper. We show that every schema mapping \mathcal{M} specified by s-t tgds has a maximum extended recovery \mathcal{M}' . Note that if both \mathcal{M}' and \mathcal{M}'' are maximum extended

¹This is automatic in the case of schema mappings specified by dependencies, such as s-t tgds.

recoveries of \mathcal{M} , then

$$e(\mathcal{M}) \circ e(\mathcal{M}') = e(\mathcal{M}) \circ e(\mathcal{M}'').$$

For schema mappings \mathcal{M} specified by s-t tgds, we characterize the quantity $e(\mathcal{M}) \circ e(\mathcal{M}')$ in terms of \mathcal{M} alone by showing that $e(\mathcal{M}) \circ e(\mathcal{M}') = \rightarrow_{\mathcal{M}}$, where

$$\rightarrow_{\mathcal{M}} = \{(I_1, I_2) : \text{chase}_{\mathcal{M}}(I_1) \rightarrow \text{chase}_{\mathcal{M}}(I_2)\}.$$

This result makes it possible to capture in a quantitative way the *information loss* embodied in a schema mapping. Specifically, we can define the information loss of a schema mapping \mathcal{M} specified by s-t tgds as the set-theoretic difference $\rightarrow_{\mathcal{M}} \setminus e(\text{Id})$; note also that $e(\text{Id}) \subseteq \rightarrow_{\mathcal{M}}$. In effect, the information loss of a schema mapping \mathcal{M} specified by s-t tgds measures the amount by which \mathcal{M} deviates from being extended invertible, since we show that \mathcal{M} is extended invertible if and only if $\rightarrow_{\mathcal{M}} = e(\text{Id})$. The concept of information loss can also be used to *compare* two schema mappings in a quantitative way and determine which of the two is “more invertible”.

We also embark on an investigation of the language needed to express maximum extended recoveries of schema mappings specified by s-t tgds. To this effect, we show that if \mathcal{M} is a schema mapping specified by full s-t tgds, then it has a maximum extended recovery specified by disjunctive tgds with inequalities; this turns out to be an optimal result, since we show the necessity of both disjunctions and inequalities. We leave it as an open problem, however, to pinpoint the exact language needed to express maximum extended recoveries of schema mappings specified by arbitrary s-t tgds.

Finally, we show that maximum extended recoveries specified by disjunctive tgds can be used to perform “reverse” data exchange and “reverse” query answering, where the latter means answering queries over the *source* schema when the source instance is no longer available. A key notion that is central to all these applications (reverse data exchange, reverse query answering, and also comparing schema mappings) is the notion of a *universal-faithful* schema mapping, which provides a procedural counterpart to the notion of maximum extended recovery of a schema mapping.

2. Background, Basic Notions, and Notation

A *schema* \mathbf{R} is a finite sequence $\langle R_1, \dots, R_k \rangle$ of relation symbols, each of a fixed arity. An *instance* I over \mathbf{R} is a sequence $\langle R_1^I, \dots, R_k^I \rangle$, 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.

We assume that we have a fixed infinite set Const of *constants* and an infinite set Var of *labeled nulls* that is disjoint from Const. A *ground* instance over some schema is an instance such that all values occurring in its relations are constants. We will also assume that \mathbf{S} is a fixed *source* schema and \mathbf{T} is a fixed *target* schema. Starting with [8], earlier work on data exchange was carried out under the assumption that instances over the source schema \mathbf{S} are ground, while instances over the target schema \mathbf{T} may contain both constants and labeled nulls. This models the situation in which we perform data exchange from \mathbf{S} to \mathbf{T} under the assumption that the individual values of source instances are known, while the under-specification of a data exchange setting may give rise to null values in the target instances. In this paper, we will drop the assumption that source instances are ground and, instead, we will assume that instances over both the source and the target schema may have individual values from Const \cup Var. Allowing source instances to contain both constants and labeled nulls models the situation where

source instances may also contain incomplete information. In particular, under this assumption, the target instance of one data exchange can be used as source instances of another data exchange.

Schema mappings A *schema mapping* is a triple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a set of constraints (typically, formulas in some logic) that describe the relationship between \mathbf{S} and \mathbf{T} . We say that Σ *specifies* \mathcal{M} . This is the syntactic view of schema mappings. For all practical purposes, however, a schema mapping can be identified with the binary relation

$$\{(I, J) : I \text{ is a } \mathbf{S}\text{-instance, } J \text{ is a } \mathbf{T}\text{-instance, } (I, J) \models \Sigma\}.$$

This is the semantic view of schema mappings. In what follows, we will switch freely between the syntactic view and the semantic view of schema mappings. In particular, we will use the notation $(I, J) \in \mathcal{M}$ to denote that the ordered pair (I, J) satisfies the constraints of \mathcal{M} ; furthermore, we will sometimes define schema mappings by simply defining the set of ordered pairs (I, J) that constitute \mathcal{M} (instead of giving a set of constraints that specify \mathcal{M}). We will also say that J is a *solution* for I w.r.t. \mathcal{M} if $(I, J) \in \mathcal{M}$. We use $\text{sol}_{\mathcal{M}}(I)$ to denote the set of all solutions of I w.r.t. \mathcal{M} .

Dependencies and schema mappings A *source-to-target tuple-generating dependency*, in short, an *s-t tgd*, is a first-order formula of the form $\forall \mathbf{x}(\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x}, \mathbf{y}))$, where $\varphi(\mathbf{x})$ is a conjunction of atomic formulas over \mathbf{S} , $\psi(\mathbf{x}, \mathbf{y})$ is a conjunction of atomic formulas over \mathbf{T} , and every variable in \mathbf{x} occurs in an atomic formula in $\varphi(\mathbf{x})$. A *full* s-t tgd is a tgd with no existential quantifiers $\exists \mathbf{y}$. In this paper, we will focus on *schema mappings* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ with Σ a set of s-t tgds. Our goal is to study *extended inverses* and *maximum extended recoveries* of such schema mappings. In particular, we will also investigate the language needed to express maximum extended recoveries. This will require bringing into the picture a richer class of dependencies that was first considered in the study of quasi-inverses of schema mappings [10] and then in the study of maximum recoveries [2].

Let *Constant* be a relation symbol that is different from all relation symbols in \mathbf{S} and \mathbf{T} . A *disjunctive tgd with constants and inequalities from \mathbf{T} to \mathbf{S}* is a first-order formula of the form

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

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

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

In this paper, we also make use of *disjunctive tgds with inequalities*, which are obtained by not allowing condition (2) in the above definition. Moreover, if neither condition (2) nor (3) is allowed, then we refer to these formulas simply as *disjunctive tgds*. We shall also make use of *tgds with constants*, which are obtained by disallowing both disjunction and condition (3).

Composing and inverting schema mappings Next, we recall the concept of the *composition* of two schema mappings, introduced in [9, 13], and the concept of an *inverse* of a schema mapping, introduced in [7].

If $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$ are two schema mappings, their *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 a \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 $\mathcal{M}_{12} \circ \mathcal{M}_{23}$.

The study of inverses of schema mappings in [7] assumes that source instances are ground. Let $\hat{\mathbf{S}}$ be a replica of the source schema \mathbf{S} , i.e., 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 . Thus, every source instance I has a replica instance \hat{I} over $\hat{\mathbf{S}}$.

The *identity schema mapping* is $\text{Id} = (\mathbf{S}, \hat{\mathbf{S}}, \Sigma_{\text{Id}})$, 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, from the semantic point of view, Id is 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 $\hat{I}_1 \subseteq I_2$.

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 $\mathcal{M} \circ \mathcal{M}' = \text{Id}$ (as sets of pairs of instances). 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. Extended Inverses

In this section, we introduce and study the notion of an extended inverse of a schema mapping. For this, we first introduce the notions of an extended solution and of the extended identity mapping.

We begin by giving the standard definitions of homomorphisms and homomorphic equivalence.

DEFINITION 3.1. Let I_1 and I_2 be instances over a schema \mathbf{R} , with values in $\text{Const} \cup \text{Var}$. A function h from $\text{Const} \cup \text{Var}$ to $\text{Const} \cup \text{Var}$ is a *homomorphism* from I_1 to I_2 if for every c in Const , we have that $h(c) = c$, and for every relation symbol R in \mathbf{R} and every tuple $(a_1, \dots, a_n) \in R^{I_1}$, we have that $(h(a_1), \dots, h(a_n)) \in R^{I_2}$.

We use the notation $I_1 \rightarrow I_2$ to denote that there is a homomorphism from I_1 to I_2 . We say that I_1 is *homomorphically equivalent* to I_2 if $I_1 \rightarrow I_2$ and $I_2 \rightarrow I_1$. We shall also use the notation \rightarrow for the binary relation $\{(I_1, I_2) \mid I_1 \rightarrow I_2\}$. \square

Note that \rightarrow is itself a schema mapping in which the source schema is the same as the target schema. An important property of \rightarrow that we shall often use is that $\rightarrow \circ \rightarrow = \rightarrow$.

DEFINITION 3.2. Let \mathcal{M} be a schema mapping. We say that J is an *extended solution* of I w.r.t. \mathcal{M} if there exist instances I' and J' such $I \rightarrow I'$, $(I', J') \in \mathcal{M}$, and $J' \rightarrow J$. This is the same as saying that $(I, J) \in \rightarrow \circ \mathcal{M} \circ \rightarrow$. We use $\text{eSol}_{\mathcal{M}}(I)$ to denote the set of all extended solutions of I w.r.t. \mathcal{M} . \square

EXAMPLE 3.3. Recall the earlier Example 1.1. The target instance U is not a solution for the source instance V w.r.t. schema mapping \mathcal{M} because every solution for V w.r.t. \mathcal{M} must contain $R(b, Z)$ and $Q(X, b)$. However, U is an extended solution for V w.r.t. \mathcal{M} . To see this, consider the target instance

$$U' = \{Q(a, b), Q(X, b), R(b, c), R(b, Z)\},$$

which is a solution for V w.r.t. \mathcal{M} . Furthermore, there is a homomorphism from U' to U (where X is mapped to a , and Z is mapped to c). Thus, $(V, U') \in \mathcal{M}$, and $U' \rightarrow U$. Hence, U is an extended solution for V w.r.t. \mathcal{M} .

Another way to see that U is an extended solution for V w.r.t. \mathcal{M} is to observe that if I is as in Example 1.1, then $V \rightarrow I$, and U itself is a solution for I w.r.t. \mathcal{M} .

As this example illustrates, the main idea behind extended solutions is that, since nulls represent unknown information, it is “legal” to homomorphically map them to other values either before or after taking the standard notion of solution. \square

The following proposition shows that extended solutions coincide with solutions in an important special case.

PROPOSITION 3.4. *If I is a ground source instance and \mathcal{M} is a schema mapping specified by s-t tgds, then $\text{eSol}_{\mathcal{M}}(I) = \text{Sol}_{\mathcal{M}}(I)$.*

Based on extended solutions, we define extended universal solutions by mimicking the definition of universal solutions in [8].

DEFINITION 3.5. Let \mathcal{M} be a schema mapping. We say that J is an *extended universal solution* for I w.r.t. \mathcal{M} if $J \in \text{eSol}_{\mathcal{M}}(I)$ and, for every $J' \in \text{eSol}_{\mathcal{M}}(I)$, we have that $J \rightarrow J'$. \square

We now define the notion of a homomorphic extension of a schema mapping. This plays a central role in what follows.

DEFINITION 3.6. Let \mathcal{M} be a schema mapping. The *homomorphic extension* of \mathcal{M} , denoted by $e(\mathcal{M})$, is the schema mapping $\rightarrow \circ \mathcal{M} \circ \rightarrow$. \square

Note that for every source instance I , the extended solutions for I w.r.t. \mathcal{M} are exactly the (standard) solutions of I w.r.t. $e(\mathcal{M})$.

In the same spirit as the extended notion of solution, we now consider an extended notion of the identity schema mapping. This is obtained by applying the homomorphic extension operator e on the standard identity schema mapping Id .

DEFINITION 3.7. The *extended identity schema mapping* is the schema mapping $e(\text{Id})$. \square

Note that, by definition, $e(\text{Id}) = \rightarrow \circ \text{Id} \circ \rightarrow$. It is easy to see that $\rightarrow \circ \text{Id} \circ \rightarrow$ is the same as \rightarrow and, therefore, $e(\text{Id}) = \rightarrow$. Thus, the key difference from the standard notion of the identity schema mapping is that $e(\text{Id})$ considers pairs (I_1, I_2) of instances such that I_1 is not necessarily a subset of I_2 but, instead, I_1 can be homomorphically mapped into I_2 . Intuitively, I_1 is a subset of I_2 up to homomorphic mapping of nulls. Note that when I_1 and I_2 are ground, we have that $I_1 \rightarrow I_2$ if and only if $I_1 \subseteq I_2$. Thus, for ground instances, $e(\text{Id})$ coincides with Id .

We are now ready to give the definition of an extended inverse of a schema mapping.

DEFINITION 3.8. Let \mathcal{M} be a schema mapping.

- A schema mapping \mathcal{M}' is an *extended inverse* of \mathcal{M} if

$$e(\mathcal{M}) \circ e(\mathcal{M}') = e(\text{Id}).$$

Since $\rightarrow \circ \rightarrow = \rightarrow$, the above equation is the same as

$$\rightarrow \circ \mathcal{M} \circ \rightarrow \circ \mathcal{M}' \rightarrow = \rightarrow.$$

- \mathcal{M} is *extended-invertible* if it has an extended inverse. \square

We next introduce the notion of a *capturing function*, which will be used to characterize extended invertibility.

DEFINITION 3.9. Let \mathcal{M} be a schema mapping.

- We say that a target instance J *captures* a source instance I (for \mathcal{M}) if the following two conditions hold: (a) $J \in \text{eSol}_{\mathcal{M}}(I)$; and (b) if K is a source instance such that $J \in \text{eSol}_{\mathcal{M}}(K)$, then $K \rightarrow I$.
- A *capturing function* for \mathcal{M} is a (total) function F from source instances to target instances such that for every source instance I , we have that $F(I)$ captures I . \square

The definitions imply that if J captures both I_1 and I_2 , then I_1 and I_2 are homomorphically equivalent. Thus, the source instance that is captured by J is unique up to homomorphic equivalence. In general, for a given I there may not exist a J that captures I . If such J exists for every I , then a capturing function for \mathcal{M} exists.

We note that the notion of a target instance capturing a source instance is an extended version of the notion of a strong witness solution in [2]. The next theorem shows that extended invertibility is equivalent to the existence of a capturing function.

THEOREM 3.10. *Let \mathcal{M} be a schema mapping. The following statements are equivalent:*

- (1) \mathcal{M} is extended-invertible.
- (2) There exists a capturing function for \mathcal{M} .

Furthermore, if \mathcal{M} is extended-invertible and F is a capturing function for \mathcal{M} , then $\mathcal{M}' = \{(J, I) \mid J = F(I)\}$ is an extended inverse of \mathcal{M} .

3.1 Extended Inverses of Mappings Specified by s-t tgds

We now address the important case in which the schema mapping \mathcal{M} is specified by a finite set of s-t tgds. First we relate extended-invertibility of such schema mappings to the existence of a special capturing function given by the chase, and also to a homomorphism-based property, which we define shortly. We consider here the standard chase with tgds as introduced in [3] (see also [1]) but applied to data exchange [8]. In particular, given a source instance I , we write $\text{chase}_{\mathcal{M}}(I)$ to denote a target instance J such that (I, J) is the result of chasing (I, \emptyset) with the dependencies in \mathcal{M} . Note here that (I, \emptyset) and (I, J) are instances over the combined source and target schema.

The following proposition is analogous to (and follows easily from) the result in [8] that $\text{chase}_{\mathcal{M}}(I)$ is a universal solution for I .

PROPOSITION 3.11. *Let \mathcal{M} be a schema mapping specified by a finite set of s-t tgds. If J is a universal solution for I w.r.t. \mathcal{M} , then J is an extended universal solution for I w.r.t. \mathcal{M} . In particular, $\text{chase}_{\mathcal{M}}(I)$ is an extended universal solution for I .*

DEFINITION 3.12. Assume that \mathcal{M} is a schema mapping specified by a finite set of s-t tgds. We say that \mathcal{M} has the *homomorphism property* if for all source instances I_1 and I_2 , the following holds: if $\text{chase}_{\mathcal{M}}(I_1) \rightarrow \text{chase}_{\mathcal{M}}(I_2)$, then $I_1 \rightarrow I_2$. \square

THEOREM 3.13. *Let \mathcal{M} be a schema mapping specified by a finite set of s-t tgds. The following statements are equivalent:*

- (1) \mathcal{M} is extended-invertible.
- (2) There exists a capturing function for \mathcal{M} .
- (3) The function F with $F(I) = \text{chase}_{\mathcal{M}}(I)$ is a capturing function for \mathcal{M} .
- (4) \mathcal{M} has the homomorphism property.

Moreover, if \mathcal{M} is extended-invertible, then the schema mapping $\mathcal{M}^* = \{(J, I) \mid J = \text{chase}_{\mathcal{M}}(I)\}$ is an extended inverse of \mathcal{M} .

Theorems 3.10 and 3.13 provide useful tools for verifying whether a schema mapping is extended invertible or not. Next, we give an example that illustrates how Theorem 3.13 can be applied to show that a schema mapping given by s-t tgds is *not* extended invertible.

EXAMPLE 3.14. Consider the “union” schema mapping \mathcal{M} specified by the s-t tgds $P(x) \rightarrow R(x)$ and $Q(x) \rightarrow R(x)$. We

now prove that \mathcal{M} is not extended-invertible by showing that \mathcal{M} does not have the homomorphism property. Let $I_1 = \{P(0)\}$ and $I_2 = \{Q(0)\}$. It is obvious that $\text{chase}_{\mathcal{M}}(I_1) \rightarrow \text{chase}_{\mathcal{M}}(I_2)$; however, it is not true that $I_1 \rightarrow I_2$. \square

We next relate the notions of extended invertibility and extended inverses to the notions of invertibility and inverses.

THEOREM 3.15. *The following hold:*

- (1) If \mathcal{M} is a schema mapping specified by a finite set of s-t tgds and \mathcal{M} is extended invertible, then \mathcal{M} is invertible.
- (2) There is a schema mapping \mathcal{M} specified by a finite set of s-t tgds that is invertible but not extended-invertible.
- (3) There is a schema mapping \mathcal{M} specified by a finite set of s-t tgds that is extended-invertible and such that:
 - (a) \mathcal{M} has an extended inverse \mathcal{M}' that is not an inverse of \mathcal{M} .
 - (b) \mathcal{M} has an inverse \mathcal{M}'' that is not an extended inverse of \mathcal{M} .

PROOF. Part (1) follows from the fact that the homomorphism property (which is equivalent to extended invertibility) implies the “subset property” [10] (which characterizes invertibility).

For part (2), consider the schema mapping \mathcal{M} that is specified by the following s-t tgds:

$$P(x) \rightarrow \exists y R(x, y) \quad Q(y) \rightarrow \exists x R(x, y)$$

We now show that \mathcal{M} is invertible but not extended-invertible. First, it can be easily verified that the schema mapping \mathcal{M}' given by the following s-t tgds with constants is an inverse of \mathcal{M} :

$$\begin{aligned} R(x, y) \wedge \text{Constant}(x) &\rightarrow P(x) \\ R(x, y) \wedge \text{Constant}(y) &\rightarrow Q(y) \end{aligned}$$

Thus, \mathcal{M} is invertible. We now show that \mathcal{M} is not extended-invertible by showing it fails to satisfy the homomorphism property. Consider the source instances $I_1 = \{P(n_1)\}$ and $I_2 = \{Q(n_2)\}$ where n_1 and n_2 are nulls. Then $\text{chase}_{\mathcal{M}}(I_1)$ and $\text{chase}_{\mathcal{M}}(I_2)$ are homomorphically equivalent. In particular, $\text{chase}_{\mathcal{M}}(I_1) \rightarrow \text{chase}_{\mathcal{M}}(I_2)$. However, it is not the case that $I_1 \rightarrow I_2$.

For part (3), consider the schema mapping \mathcal{M} given by the following s-t tgd:

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

Moreover, consider the following “reverse” schema mappings \mathcal{M}' and \mathcal{M}'' given, respectively, by the following dependencies:

$$\begin{aligned} \mathcal{M}' : \quad & Q(x, z) \wedge Q(z, y) \rightarrow P(x, y) \\ \mathcal{M}'' : \quad & Q(x, z) \wedge Q(z, y) \wedge \text{Constant}(x) \wedge \text{Constant}(y) \\ & \rightarrow P(x, y) \end{aligned}$$

We shall show in the next subsection (Example 3.18) that \mathcal{M}' is an extended inverse of \mathcal{M} . At the same time, it was shown in [10] that there is no inverse of \mathcal{M} that is specified by s-t tgds without the *Constant* predicate. Hence, \mathcal{M}' cannot be an inverse of \mathcal{M} . Thus, \mathcal{M}' is an extended inverse of \mathcal{M} that is not an inverse of \mathcal{M} . As for condition (b), it was shown in [10] that \mathcal{M}'' is an inverse of \mathcal{M} . We show in the next subsection (Example 3.19) that \mathcal{M}'' is not an extended inverse of \mathcal{M} . \square

Theorem 3.15 tells us that the notion of extended invertibility is stronger than invertibility. Intuitively, it is harder for a schema mapping to be extended invertible, since there are more instances (i.e., non-ground source instances) to consider. Furthermore, extended inverses and inverses do not necessarily coincide, even for schema mappings that are extended-invertible (hence, also invertible).

3.2 The Goodness of Extended Inverses

In this section, we show that extended inverses that are given by s-t tgds have desirable properties that make them ideally suited for “reverse” data exchange. The data exchange problem associated with a schema mapping \mathcal{M} is to materialize a “good” solution J from a source instance I , based on \mathcal{M} . The canonical procedure for data exchange [8] uses the chase of I with \mathcal{M} to produce a (canonical) universal solution. The *reverse data exchange problem* is then to materialize a source instance I' from a target instance J according to a reverse schema mapping \mathcal{M}' (from the target schema to the source schema). Reverse data exchange is typically performed after an initial data exchange with \mathcal{M} was performed; in such a case, the goal of reverse data exchange is to recover a source instance that is as “close as possible” to the *original* source instance I .

DEFINITION 3.16. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping specified by a finite set of s-t tgds. Let $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$ be a schema mapping specified by a finite set of s-t tgds. We say that \mathcal{M}' is a *chase-inverse* of \mathcal{M} if I and $\text{chase}_{\mathcal{M}'}(\text{chase}_{\mathcal{M}}(I))$ are homomorphically equivalent, for every source instance I . \square

Note that a chase inverse makes it possible to recover the original source instance up to homomorphic equivalence.

The next theorem shows that extended inverses that are given by s-t tgds have an equivalent characterization as chase-inverses. This characterization is a precise measure of the goodness of extended inverses for reverse data exchange.

THEOREM 3.17. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping specified by a finite set of s-t tgds, and let $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$ be a “reverse” schema mapping specified by a finite set of s-t tgds. The following statements are equivalent:

- (1) \mathcal{M}' is an extended inverse of \mathcal{M} .
- (2) \mathcal{M}' is a chase-inverse of \mathcal{M} .

Theorem 3.17 easily extends to the case when \mathcal{M}' is given by tgds with constants, and we also allow these as chase-inverses. Also, Theorem 3.17 gives us another tool to verify whether a schema mapping \mathcal{M}' is an extended inverse of a schema mapping \mathcal{M} .

EXAMPLE 3.18. As in the proof of part (3) of Theorem 3.15, consider the schema mapping \mathcal{M} specified by the s-t tgd

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

Let \mathcal{M}' be the schema mapping specified by the target-to-source tgd $Q(x, z) \wedge Q(z, y) \rightarrow P(x, y)$. We can show that \mathcal{M}' is an extended inverse of \mathcal{M} by showing that \mathcal{M}' is a chase-inverse of \mathcal{M} . Indeed, let I be a source instance, let $U = \text{chase}_{\mathcal{M}}(I)$ and let $V = \text{chase}_{\mathcal{M}'}(U)$. We shall show that $I \subseteq V$ and that $V \rightarrow I$, which imply that V and I are homomorphically equivalent.

First, we observe that every fact $P(a, b)$ in I generates (via the chase with \mathcal{M}) two facts in U , namely $Q(a, Z_{ab})$ and $Q(Z_{ab}, b)$, where Z_{ab} is a fresh new null, distinct for every choice of a and b . (Moreover, these are the only types of facts that are generated in U .) Then V must contain every fact $P(a, b)$ (in order to satisfy \mathcal{M}' for $Q(a, Z_{ab})$ and $Q(Z_{ab}, b)$). Thus, $I \subseteq V$.

We now observe that every extra fact of V (i.e., not in I) can only be of the form $P(Z_{ab}, Z_{bc})$, arising via \mathcal{M}' from two facts of U of the form $Q(Z_{ab}, b)$ and $Q(b, Z_{bc})$. But then U must contain two additional facts, $Q(a, Z_{ab})$ and $Q(Z_{bc}, c)$. At the same time, the only way to have $Q(a, Z_{ab})$ and $Q(Z_{bc}, c)$ in U is to have the fact $P(a, b)$ in I . Consider now the mapping h where $h(x) = x$ for members x of I , and where $h(Z_{ab}) = a$. Then h is a homomorphism from V to I . In particular, for every extra fact $P(Z_{ab}, Z_{bc})$ in V , we have that $P(h(Z_{ab}), h(Z_{bc}))$ is $P(a, b)$, which we have shown must exist in I . \square

We have already noted that the schema mapping \mathcal{M}' in the above example cannot be an inverse of \mathcal{M} (since, as shown in [10], such an inverse would have to make use of the *Constant* predicate, for this particular \mathcal{M}). Thus, this example shows in particular that the characterization via chase-inverses does not hold for inverses (since \mathcal{M}' is a chase-inverse of \mathcal{M} but not an inverse of \mathcal{M}).

The next example uses Theorem 3.17 to prove that a schema mapping \mathcal{M}'' is *not* an extended inverse of a schema mapping \mathcal{M} .

EXAMPLE 3.19. As in the proof of part (3) of Theorem 3.15 and Example 3.18, consider the schema mapping \mathcal{M} specified by

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

Moreover, let \mathcal{M}'' be the schema mapping specified by

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

It was shown in [10] that \mathcal{M}'' is an inverse of \mathcal{M} . We now show that \mathcal{M}'' is not an extended inverse of \mathcal{M} by showing that \mathcal{M}'' fails to be a chase-inverse of \mathcal{M} . (Here we allow chase-inverses to be specified by tgds with constants, as discussed after Theorem 3.17.)

To show that \mathcal{M}'' is not a chase-inverse of \mathcal{M} , consider the source instance $I = \{P(W, Z)\}$, where W and Z are nulls. Chasing I with \mathcal{M} gives the target instance $U = \{Q(W, Y), Q(Y, Z)\}$ where Y is a null. Chasing U with \mathcal{M}'' gives the empty instance, since there are no constants in U . Hence, $\text{chase}_{\mathcal{M}''}(\text{chase}_{\mathcal{M}}(I))$ and I are not homomorphically equivalent. \square

The above two examples point out problems with the notion of inverse, which do not arise for the new notion of extended inverse. Example 3.18 shows a natural “inverse” (the chase-inverse) that is not captured by the notion of inverse. Conversely, Example 3.19 shows an inverse that fails to be a chase-inverse.

Most schema mappings occurring in practice do not possess extended inverses, since they do not even possess inverses. Invertibility and extended invertibility are strong notions that, intuitively, cover the case of “no information loss” in a schema mapping. To address schema mappings with information loss, which is the frequent case, we will go beyond extended invertibility and study extended recoveries in Section 4.

4. Extended Recoveries

Arenas, Pérez, and Riveros [2] introduced the notions of recovery and maximum recovery.

DEFINITION 4.1. ([2]) Let \mathcal{M} be a schema mapping from a source schema \mathbf{S} to a target schema \mathbf{T} . A schema mapping \mathcal{M}' from \mathbf{T} to \mathbf{S} is a *recovery* of \mathcal{M} if for every source instance I , the pair (I, I) is in $\mathcal{M} \circ \mathcal{M}'$. The schema mapping \mathcal{M}' is a *maximum recovery* of \mathcal{M} if (1) \mathcal{M}' is a recovery for \mathcal{M} , and (2) for every recovery \mathcal{M}'' of \mathcal{M} , we have that $\mathcal{M} \circ \mathcal{M}' \subseteq \mathcal{M} \circ \mathcal{M}''$. \square

Thus, a maximum recovery is a recovery that is optimal among all recoveries in the sense that the composition of \mathcal{M} with \mathcal{M}' is the smallest among the compositions of \mathcal{M} with every other recovery. Similar to the framework in [7, 10], the study of recoveries in [2] is carried out in the context in which source instances are ground. While some of the results in [2] continue to hold even when source instances are not restricted to be ground, certain other results do not. In particular, one of the important results in [2] is that every schema mapping specified by s-t tgds has a maximum recovery. However, we show in the next proposition that there is a schema mapping specified by s-t tgds with no maximum recovery when source instances can be non-ground.

