Foundations of Schema Mapping Management

Marcelo Arenas
PUC Chile
marenas@ing.puc.cl

Jorge Pérez
PUC Chile
jperez@ing.puc.cl

Juan Reutter
University of Edinburgh
juan.reutter@ed.ac.uk

Cristian Riveros
Oxford University
cristian.riveros@comlab.ox.ac.uk

ABSTRACT

In the last few years, a lot of attention has been paid to the specification and subsequent manipulation of schema mappings, a problem which is of fundamental importance in metadata management. There have been many achievements in this area, and semantics have been defined for operators on schema mappings such as composition and inverse. However, little research has been pursued towards providing formal tools to compare schema mappings, in terms of their ability to transfer data and avoid storing redundant information, which has hampered the development of foundations for more complex operators as many of them involve these notions.

In this paper, we address the problem of providing foundations for metadata management by developing an order to compare the amount of information transferred by schema mappings. From this order we derive several other criteria to compare mappings, we provide tools to deal with these criteria, and we show their usefulness in defining and studying schema mapping operators. More precisely, we show how the machinery developed can be used to study the *extract* and *merge* operators, that have been identified as fundamental for the development of a metadata management framework. We also use our machinery to provide simpler proofs for some fundamental results regarding the inverse operator, and we give an effective characterization for the decidability of the well-known schema evolution problem.

Categories and Subject Descriptors

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

General Terms

Algorithms, Theory

Keywords

Metadata management, model management, schema mapping, data exchange, data integration

PODS'10, June 6–11, 2010, Indianapolis, Indiana, USA.

1. INTRODUCTION

A schema mapping is a specification that describes a relationship between data structured under two independent schemas. In the last few years, a lot of attention has been paid to the specification and subsequent manipulation of schema mappings, a problem which is of fundamental importance in metadata management [5, 22, 19, 6]. In the metadata management context, schema mappings are first class citizens, and high-level algebraic operators are used to manipulate them.

Metadata management and, in particular, schema mapping management, is an area of active research, where there had been many achievements in the recent years. However, there is not yet consensus on the definitive semantics even for the most fundamental metadata management operators (a notable exception is the composition operator [14, 22]). Consider, for example, the cases of the *merge* operator [7, 25, 22, 23, 26] and the *inverse* operator [11, 15, 3, 16, 2, 1], for which several different semantics have been proposed. In fact, we are currently in a situation in which different proposals for the same set of operators are being studied, but little research is being pursued towards understanding the fundamental notions that all these proposals seem to share. In particular, abstract notions of *information*, *redundancy* and *minimality* are part of every proposal for the semantics of schema mapping operators [5, 25, 22, 11, 3, 26]. The formalization of these notions in a general setting would play an essential role in providing a unifying framework for metadata management. The work by Melnik [22] could be considered a first such effort towards the development of a general framework for the area. In this paper, we go a step further in this direction.

We address the problem of providing foundations for metadata management by focusing on the abstract notions of information and redundancy. We develop theoretical tools to compare schema mappings in terms of these two aspects, providing characterizations to deal with these criteria, and showing their usefulness in defining and studying complex metadata management operators.

As an example to motivate our notions, consider the following two mappings given by source-to-target tuple-generating dependencies (st-tgds):

$$\begin{array}{lrcl} \mathcal{M}_1: & A(x,y,z) & \rightarrow & \exists u\, P(x,u) \\ \mathcal{M}_2: & A(x,y,z) & \rightarrow & R(x) \wedge S(x,y) \end{array}$$

Intuitively, $\mathcal{M}_2$ *transfers more information* than $\mathcal{M}_1$ since the first and second components of tuples in A are being transferred to the target under $\mathcal{M}_2$, while only the first component is being transferred under $\mathcal{M}_1$. In fact, notice that the information transferred by $\mathcal{M}_1$ can be obtained from the target of $\mathcal{M}_2$ by means of the mapping $R(x) \rightarrow \exists u\, P(x,u)$. However, the opposite is not true; we cannot obtain the information transferred by $\mathcal{M}_2$ from the tar-

get of $\mathcal{M}_1$. Consider now the mapping $\mathcal{M}_3$ given by:

$$\mathcal{M}_3: \quad A(x,y,z) \quad \rightarrow \quad T(x,y)$$

Intuitively, mapping $\mathcal{M}_3$ transfers the same amount of information as $\mathcal{M}_2$. Nevertheless, $\mathcal{M}_3$ is more *efficient* in the way that it structures the target data, as $\mathcal{M}_2$ stores *redundant information* in table R. In this paper, we formalize the previous notions, that is, we develop notions to compare mappings in terms of their ability to transfer source data and avoid storing redundant information in its target schema, as well as the symmetric notions of *covering* target data, and storing redundant information in the source schema. In fact, we prove the usefulness of the proposed notions by showing that they can play a central role in the study of complex metadata management operators.

More precisely, we start our investigation by defining a set of natural conditions that an order on the amount of information transferred by a schema mapping should satisfy. We then propose the order $\preceq_{\text{S}}$, that is provably the strictest relation satisfying these conditions. Under our definition, the mappings in the above example satisfy that $\mathcal{M}_1 \preceq_{\text{S}} \mathcal{M}_2$ but $\mathcal{M}_2 \not\preceq_{\text{S}} \mathcal{M}_1$. We study some fundamental properties of $\preceq_{\text{S}}$ and, in particular, we prove its decidability for the class of mappings specified by CQ$^{\neq}$-TO-CQ dependencies, which includes the widely used class of st-tgds. We also contrast our proposal with previous work on comparing mappings [16]. Fagin et al. propose in [16] a notion of *information loss* for schema mappings specified by st-tgds, which gives rise to an order on this type of mappings. In this paper, we show that our notion coincides with the proposal of Fagin et al. for the class of mappings defined by st-tgds. Moreover, we also prove that beyond st-tgds, Fagin et al.'s notion does not satisfy the natural conditions that we impose over an order to compare the amount of information transferred by mappings.

As shown in the previous example, there may exist multiple ways to transfer the same information from a source schema. Thus, one also needs a way to distinguish between different alternatives. In particular, if schemas are designed together with mappings, it is desirable to use schemas that are *optimal* in the way they handle data. To deal with this requirement, we introduce the notion of *target redundancy*, and show that it captures the intuition of using the *exact amount of resources* needed to transfer information using a schema mapping. In fact, our notion formally captures the intuition in the previous example, as $\mathcal{M}_2$ is *target redundant* while $\mathcal{M}_3$ is not.

Furthermore, to complement our information framework, we devise two additional concepts that allow us to compare mappings that share the same target schema. Symmetrically to the definition of $\preceq_{\text{S}}$, we introduce the order $\preceq_{\text{T}}$, that intuitively measures the amount of information *covered* by a mapping, as well as a notion of *source redundancy*. We provide characterizations and tools for all the proposed notions, and show that together they can be used as a powerful framework to study metadata management operators.

As a proof of concept, we show how the machinery developed can be used to study some metadata management problems in the context of data exchange. In particular, we provide simpler proofs for some fundamental results regarding the inverse operator proposed by Fagin [11], and we also give an effective characterization for the decidability of the well-known schema evolution problem [22, 19].

Finally, we use all this machinery to study more complex metadata management operators. More precisely, we revisit the semantics of the *extract* operator [22, 23], that intuitively captures the idea of *upgrading* a legacy schema. We formalize this operator in terms of the general notions developed in this paper, and we provide an algorithm for computing it for a class of mappings that includes the mappings specified by st-tgds. Moreover, we also study the *merge* operator, that as well as the extract operator, has been identified as fundamental for the development of a metadata management framework.

It should be stressed that all the notions proposed in this paper are general and, thus, applicable in a wide context in metadata management.

Organization of the paper. Notation is introduced in Section 2, while the order $\preceq_{\text{S}}$ is defined in Section 3. The fundamental properties of $\preceq_{\text{S}}$ are studied in Section 4, while two of its applications in the data exchange context are shown in Section 5. The order $\preceq_{\text{T}}$ and the notions of target and source redundancy are introduced in Section 6. The extract operator is studied in Section 7, and the merge operator is studied in Section 8. Concluding remarks are given in Section 9.

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 $n_i \geq 0$. 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* $\text{dom}(I)$ of instance I is the set of all elements that occur in any of the relations R_i^I. $\text{Inst}(\mathbf{R})$ is defined to be the set of all instances of $\mathbf{R}$. As is customary, we consider instances with two types of values: constants and nulls [12]. Let $\mathbf{C}$ and $\mathbf{N}$ be infinite and disjoint sets of constants and nulls, respectively, and assume that $\mathbf{D} = \mathbf{C} \cup \mathbf{N}$. Then the instances of a schema are constructed by using elements from both $\mathbf{C}$ and $\mathbf{N}$. And if we refer to a schema $\mathbf{R}$ as *ground*, then we assume that $\text{Inst}(\mathbf{R})$ contains instances that are constructed by using only elements from $\mathbf{C}$.

Mappings: 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 $\text{Inst}(\mathbf{R}_1) \times \text{Inst}(\mathbf{R}_2)$. If $\mathcal{M}$ is a mapping and $(I, J) \in \mathcal{M}$, then we say that J is a *solution for I under $\mathcal{M}$*. The set of solutions for I under $\mathcal{M}$ is denoted by $\text{Sol}_{\mathcal{M}}(I)$. The domain of $\mathcal{M}$, denoted by $\text{dom}(\mathcal{M})$, is defined as the set of instances I such that $\text{Sol}_{\mathcal{M}}(I) \neq \emptyset$, and the range of $\mathcal{M}$, denoted by $\text{range}(\mathcal{M})$ is defined as the set of instances J that are a solution for some instance $I \in \text{dom}(\mathcal{M})$. Given 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 for binary relations, that is, $\mathcal{M}_{12} \circ \mathcal{M}_{23} = \{(I_1, I_3) \mid \exists I_2 : (I_1, I_2) \in \mathcal{M}_{12} \text{ and } (I_2, I_3) \in \mathcal{M}_{23}\}$ [14, 22]. Furthermore, given a mapping $\mathcal{M}$, we denote by $\mathcal{M}^{-1}$ the mapping $\{(J, I) \mid (I, J) \in \mathcal{M}\}$.

Queries and certain answers: A k-ary query Q over a schema $\mathbf{R}$, with $k \geq 0$, is a function that maps every instance $I \in \text{Inst}(\mathbf{R})$ into a k-relation $Q(I) \subseteq \text{dom}(I)^k$. 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 or inequalities, then we use superscripts $=$ and $\neq$, respectively. Thus, for example, UCQ$^{\neq}$ is the class of union of conjunctive queries with inequalities. 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 $\mathbf{C}(\cdot)$.

Let $\mathcal{M}$ be a mapping from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$, I an instance of $\mathbf{R}_1$ and Q a query over $\mathbf{R}_2$. Then $\text{certain}_{\mathcal{M}}(Q, I)$ denotes the set of *certain answers* of Q over I under $\mathcal{M}$, that is, $\text{certain}_{\mathcal{M}}(Q, I) = \bigcap_{J \in \text{Sol}_{\mathcal{M}}(I)} Q(J)$.

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 tu-

ple of free variables in both $\varphi(\bar{x})$ and $\psi(\bar{x})$; (2) $\varphi(\bar{x})$ is an $\mathcal{L}_1$-formula over $\mathbf{R}_1$ (plus $\mathbf{C}(\cdot)$ if $\mathbf{C}(\cdot)$ is allowed in $\mathcal{L}_1$); and (3) $\psi(\bar{x})$ is an $\mathcal{L}_2$-formula over $\mathbf{R}_2$ (plus $\mathbf{C}(\cdot)$ if $\mathbf{C}(\cdot)$ is allowed in $\mathcal{L}_2$). We call $\varphi(\bar{x})$ the *premise* of Φ, and $\psi(\bar{x})$ the *conclusion* of Φ. 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 $\forall \bar{x}\ (\varphi(\bar{x}) \rightarrow \psi(\bar{x}))$. An FO-TO-CQ dependency is *full* if its conclusion does not include existential quantifiers. Notice that the class of source-to-target tuple-generating dependencies (st-tgds) corresponds to the class of CQ-TO-CQ dependencies. The semantics of dependencies is inherited from the semantics of FO. Moreover, we assume that for every $\mathcal{L}_1$-TO-$\mathcal{L}_2$ dependency $\forall \bar{x}\ (\varphi(\bar{x}) \rightarrow \psi(\bar{x}))$, both $\varphi(\bar{x})$ and $\psi(\bar{x})$ are domain-independent (see [10] for a formal definition of domain independence).

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 $\mathbf{R}_1 \cup \mathbf{R}_2 \cup \{\mathbf{C}(\cdot)\}$. Then 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) satisfies Σ. Moreover, if we refer to $\mathcal{M} = (\mathbf{R}_1, \mathbf{R}_2, \Sigma)$ as an *st-mapping*, then $\mathbf{R}_1$ is a ground schema and $\mathbf{R}_2$ is a non-ground schema (instances of $\mathbf{R}_1$ are constructed by using only elements from $\mathbf{C}$, while instances of $\mathbf{R}_2$ are constructed by using elements from $\mathbf{C}$ and $\mathbf{N}$).

Proviso: In this paper, all sets of dependencies are assumed to be finite.

3. TRANSFERRING SOURCE INFORMATION

In a data exchange scenario, a schema mapping is used to transfer information from a source schema to a target schema. Thus, it is natural to ask how much source information a mapping transfers and, in particular, if two mappings are used to transfer information from the same source schema, it is natural to ask whether one of them transfers *more* source information than the other. The latter question is the main motivation for the first part of our investigation.

The problem of measuring the amount of source information transferred by a mapping has been studied in the data exchange scenario [16, 2, 3]. In fact, the issue of developing an order for comparing the amount of source information transferred by two mappings has been explicitly considered in [16]. However, we follow here a different approach to develop such an order, as we first identity five natural conditions that such an order should satisfy, and then we consider the strictest order according to these conditions.

From now on, we use symbol $\preceq$ to denote an order between mappings that transfer information from the same source schema. That is, if $\mathcal{M}_1 \preceq \mathcal{M}_2$, then we assume that there exists a schema $\mathbf{R}$ such that both $\mathrm{dom}(\mathcal{M}_1)$ and $\mathrm{dom}(\mathcal{M}_2)$ are contained in $\mathrm{Inst}(\mathbf{R})$. The following is the first condition that we impose on $\preceq$.

(C1) $\mathcal{M}_1 \preceq \mathcal{M}_2$ implies $\mathrm{dom}(\mathcal{M}_1) \subseteq \mathrm{dom}(\mathcal{M}_2)$.

If I is an instance in the domain of $\mathcal{M}_1$, then $\mathcal{M}_1$ provides some information about I as it gives a collection of target instances that are considered to be valid translations of I. Thus, if $\mathcal{M}_2$ gives as much source information as $\mathcal{M}_1$, then I should also be in the domain of $\mathcal{M}_2$, as stated by condition (C1).

As usual for any notion of preference, $\preceq$ is also asked to be reflexive and transitive:

(C2) $\mathcal{M} \preceq \mathcal{M}$,

(C3) $\mathcal{M}_1 \preceq \mathcal{M}_2$ and $\mathcal{M}_2 \preceq \mathcal{M}_3$ implies $\mathcal{M}_1 \preceq \mathcal{M}_3$.

Notice that we do not ask relation $\preceq$ to be antisymmetric, as it is usually the case that the same information can be transferred in different ways. Thus, strictly speaking, $\preceq$ is not an order but a preorder.

Furthermore, let $\mathrm{Id}_{\mathbf{R}}$ be the identity schema mapping for a schema $\mathbf{R}$, that is, $\mathrm{Id}_{\mathbf{R}} = \{(I, I) \mid I \in \mathrm{Inst}(\mathbf{R})\}$. This mapping transfers exactly the information that is contained in the instances of $\mathbf{R}$ and, thus, any other mapping that transfers information from $\mathbf{R}$ could not be more informative than $\mathrm{Id}_{\mathbf{R}}$. This gives rise to the fourth condition for the desired order:

(C4) if $\mathcal{M}$ is a mapping from a schema $\mathbf{R}$ to a schema $\mathbf{R}_1$, then $\mathcal{M} \preceq \mathrm{Id}_{\mathbf{R}}$.

Finally, our last condition accounts for the information that is transferred through a composition of schema mappings. Assume that a mapping $\mathcal{M}$ transfers information from a schema $\mathbf{R}$ to a schema $\mathbf{R}_1$ and that $\mathcal{M}_1$, $\mathcal{M}_2$ are mappings that transfer information from $\mathbf{R}_1$. If $\mathcal{M}_2$ maps as much source information as $\mathcal{M}_1$, then given that $\mathcal{M}$ transfers information to schema $\mathbf{R}_1$, one would expect that $\mathcal{M} \circ \mathcal{M}_2$ transfers as much source information as $\mathcal{M} \circ \mathcal{M}_1$. This is stated in our last condition:

(C5) let $\mathcal{M}$ be a mapping from a schema $\mathbf{R}$ to a schema $\mathbf{R}_1$, and $\mathcal{M}_1$, $\mathcal{M}_2$ mappings from $\mathbf{R}_1$ to schemas $\mathbf{R}_2$ and $\mathbf{R}_3$, respectively. If $\mathcal{M}_1 \preceq \mathcal{M}_2$, then $\mathcal{M} \circ \mathcal{M}_1 \preceq \mathcal{M} \circ \mathcal{M}_2$.

Now that we have identified five conditions that the desired order should satisfy, the first question to answer is whether there exists any order that meet them. In the following paragraphs, we give a positive answer to this question by introducing a relation $\preceq_{\mathrm{s}}$. Moreover, we also show that $\preceq_{\mathrm{s}}$ is the strictest order that satisfy the above conditions.

Definition 3.1 (Order $\preceq_{\mathrm{s}}$) *Let $\mathbf{R}$, $\mathbf{R}_1$, $\mathbf{R}_2$ be schemas, and $\mathcal{M}_1$, $\mathcal{M}_2$ mappings from $\mathbf{R}$ to $\mathbf{R}_1$ and $\mathbf{R}$ to $\mathbf{R}_2$, respectively. Then $\mathcal{M}_1 \preceq_{\mathrm{s}} \mathcal{M}_2$ if there exists a mapping $\mathcal{N}$ from $\mathbf{R}_2$ to $\mathbf{R}_1$ such that $\mathcal{M}_1 = \mathcal{M}_2 \circ \mathcal{N}$.*

Intuitively, the preceding definition says that $\mathcal{M}_1 \preceq_{\mathrm{s}} \mathcal{M}_2$ if $\mathcal{M}_2$ transfers enough information from $\mathbf{R}$ to be able to reconstruct the information transferred by $\mathcal{M}_1$.

Example 3.2. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be mappings specified by dependencies $S(x, y) \rightarrow T(x)$ and $S(x, y) \rightarrow U(y, x)$, respectively. Intuitively, $\mathcal{M}_2$ maps more information than $\mathcal{M}_1$ as all the source information is stored in the target according to mapping $\mathcal{M}_2$. In fact, in this case we have that $\mathcal{M}_1 \preceq_{\mathrm{s}} \mathcal{M}_2$ since $\mathcal{M}_1 = \mathcal{M}_2 \circ \mathcal{N}$, where $\mathcal{N}$ is a mapping specified by dependency $U(x, y) \rightarrow T(y)$. In this case, it is also possible to prove that $\mathcal{M}_2 \not\preceq_{\mathrm{s}} \mathcal{M}_1$.

On the other hand, if $\mathcal{M}_3$ is a mapping specified by dependency $S(x, y) \rightarrow V(y)$, then one would expect $\mathcal{M}_1$ and $\mathcal{M}_3$ to be incomparable, as these mappings extract information from different columns of table S. In fact, in this case it is possible to prove that $\mathcal{M}_1 \not\preceq_{\mathrm{s}} \mathcal{M}_3$ and $\mathcal{M}_3 \not\preceq_{\mathrm{s}} \mathcal{M}_1$. □

As a corollary of the fact that the composition is associative, we obtain that $\preceq_{\mathrm{s}}$ satisfies the above conditions:

Proposition 3.3 *The order $\preceq_{\mathrm{s}}$ satisfies (C1), (C2), (C3), (C4) and (C5).*

The following proposition shows the somewhat surprising result that such a simple relation is the strictest order that satisfies the above conditions.

Proposition 3.4 *Assume that an order $\preceq$ satisfies (C1), (C2), (C3), (C4) and (C5). Then for every pair of mappings $\mathcal{M}_1$ and $\mathcal{M}_2$, $\mathcal{M}_1 \preceq_{\text{S}} \mathcal{M}_2$ implies that $\mathcal{M}_1 \preceq \mathcal{M}_2$.*

It remains as an open problem whether the order $\preceq_{\text{S}}$ indeed characterizes the above axioms, in the sense that if an order $\preceq$ satisfies (C1), (C2), (C3), (C4) and (C5), then $\preceq$ is equivalent to $\preceq_{\text{S}}$ (for every pair of mapping $\mathcal{M}_1$, $\mathcal{M}_2$, it holds that $\mathcal{M}_1 \preceq \mathcal{M}_2$ if and only if $\mathcal{M}_1 \preceq_{\text{S}} \mathcal{M}_2$).

In our investigation, we use $\preceq_{\text{S}}$ to compare the amount of information transferred by two mappings from the same source schema. In particular, if $\mathcal{M}_1 \preceq_{\text{S}} \mathcal{M}_2$ and $\mathcal{M}_2 \preceq_{\text{S}} \mathcal{M}_1$, we say that $\mathcal{M}_1$ and $\mathcal{M}_2$ transfer the same amount of source information, which is denoted by $\mathcal{M}_1 \equiv_{\text{S}} \mathcal{M}_2$.

3.1 Comparison with other notions of order

In [16], Fagin et al. propose to use some notions of inversion of schema mappings [11, 3, 16] to measure the *information loss* of a mapping. Loosely speaking, the more invertible a mapping is, the less information the mapping loses [16]. In this section, we contrast and compare Fagin et al's approach with the order $\preceq_{\text{S}}$.

In order to give some intuition behind the definitions presented in [16], we first introduce an order $\preceq_{\text{R}}$ that is based on the notion of maximum recovery [3]. Let $\mathcal{M}$ be a mapping from a schema $\mathbf{R}$ to a schema $\mathbf{R}_1$. A mapping $\mathcal{M}^\star$ is a maximum recovery of $\mathcal{M}$ if $\text{Id}_{\mathbf{R}} \subseteq \mathcal{M} \circ \mathcal{M}^\star$ and for every other mapping $\mathcal{M}'$ such that $\text{Id}_{\mathbf{R}} \subseteq \mathcal{M} \circ \mathcal{M}'$, it holds that $\mathcal{M} \circ \mathcal{M}^\star \subseteq \mathcal{M} \circ \mathcal{M}'$. It is easy to see that if $\mathcal{M}_1^\star$ and $\mathcal{M}_2^\star$ are both maximum recoveries of $\mathcal{M}$, then $\mathcal{M} \circ \mathcal{M}_1^\star = \mathcal{M} \circ \mathcal{M}_2^\star$. Thus, the composition of a mapping $\mathcal{M}$ with any of its maximum recoveries depends only on $\mathcal{M}$. In fact, it was shown in [3] that if $\mathcal{M}^\star$ is a maximum recovery of $\mathcal{M}$, then the composition $\mathcal{M} \circ \mathcal{M}^\star$ is equal to the set $\{(I, K) \mid \text{Sol}_{\mathcal{M}}(K) \subseteq \text{Sol}_{\mathcal{M}}(I)\}$. The definition of order $\preceq_{\text{R}}$ is based on this property. More precisely, a mapping $\mathcal{M}_2$ is said to be *less lossy than* a mapping $\mathcal{M}_1$ if for every pair of instances I, K, it holds that $\text{Sol}_{\mathcal{M}_1}(I) \subseteq \text{Sol}_{\mathcal{M}_1}(K)$ whenever $\text{Sol}_{\mathcal{M}_2}(I) \subseteq \text{Sol}_{\mathcal{M}_2}(K)$ holds. Let $\preceq_{\text{R}}$ denote the order induced by this notion, that is, $\mathcal{M}_1 \preceq_{\text{R}} \mathcal{M}_2$ if and only if $\mathcal{M}_2$ is less lossy than $\mathcal{M}_1$.

It is important to notice that if mappings $\mathcal{M}_1$ and $\mathcal{M}_2$ have maximum recoveries, say $\mathcal{M}_1^\star$ and $\mathcal{M}_2^\star$, respectively, then $\mathcal{M}_1 \preceq_{\text{R}} \mathcal{M}_2$ if and only if $\text{Id}_{\mathbf{R}} \subseteq \mathcal{M}_2 \circ \mathcal{M}_2^\star \subseteq \mathcal{M}_1 \circ \mathcal{M}_1^\star$. Thus, $\mathcal{M}_1 \preceq_{\text{R}} \mathcal{M}_2$ if the composition of $\mathcal{M}_2$ with its maximum recovery is more similar to the identity mapping than the composition of $\mathcal{M}_1$ with its maximum recovery.

As a first result, we prove that $\preceq_{\text{R}}$ does not satisfy all the conditions identified in this section for a natural order, thus showing that $\preceq_{\text{S}}$ can be considered as a better alternative than $\preceq_{\text{R}}$ to compare the information transferred by schema mappings. In particular, $\preceq_{\text{R}}$ does not satisfy (C5).

Proposition 3.5 *There exist mappings $\mathcal{M}$, $\mathcal{M}_1$ and $\mathcal{M}_2$ such that $\mathcal{M}_1 \preceq_{\text{R}} \mathcal{M}_2$ and $\mathcal{M} \circ \mathcal{M}_1 \not\preceq_{\text{R}} \mathcal{M} \circ \mathcal{M}_2$.*

In [16], Fagin et al. were interested in studying mappings with null values in source and target instances. In particular, for a mapping $\mathcal{M}$ of this type, Fagin et al. define a mapping $e(\mathcal{M})$ that extends $\mathcal{M}$ by giving a semantics to the nulls that distinguish them from the constants (see [16] for the precise definition of $e(\mathcal{M})$). In [16], Fagin et al. introduce a notion of information loss of a schema mapping $\mathcal{M}$ by considering the extension $e(\mathcal{M})$ of $\mathcal{M}$. More precisely, if $\mathcal{M}_1$ and $\mathcal{M}_2$ are two mappings containing null values in source and target instances, then $\mathcal{M}_2$ is said to be less lossy than $\mathcal{M}_1$ if for every pair of instances I, K, it holds that $\text{Sol}_{e(\mathcal{M}_1)}(I) \subseteq \text{Sol}_{e(\mathcal{M}_1)}(K)$ whenever $\text{Sol}_{e(\mathcal{M}_2)}(I) \subseteq \text{Sol}_{e(\mathcal{M}_2)}(K)$ holds [16]. Let $\preceq_{\text{E}}$ be the order induced by this notion. Notice that $\preceq_{\text{E}}$ is tightly connected with $\preceq_{\text{R}}$; in fact, it holds that $\mathcal{M}_1 \preceq_{\text{E}} \mathcal{M}_2$ if and only if $e(\mathcal{M}_1) \preceq_{\text{R}} e(\mathcal{M}_2)$. The following proposition shows that as for the case of $\preceq_{\text{R}}$, the order $\preceq_{\text{E}}$ does not satisfy (C5).

Proposition 3.6 *There exist mappings $\mathcal{M}$, $\mathcal{M}_1$ and $\mathcal{M}_2$ such that $\mathcal{M}$, $\mathcal{M}_1$ and $\mathcal{M}_2$ contain null values in source and target instances, $\mathcal{M}_1 \preceq_{\text{E}} \mathcal{M}_2$ and $\mathcal{M} \circ \mathcal{M}_1 \not\preceq_{\text{E}} \mathcal{M} \circ \mathcal{M}_2$.*

No restrictions on mappings were imposed when defining the order $\preceq_{\text{S}}$. In particular, $\preceq_{\text{S}}$ can be used to compare mappings containing null values in source and target instances. Thus, Proposition 3.6 gives evidence that $\preceq_{\text{S}}$ is a better alternative than $\preceq_{\text{E}}$ to compare the information transferred by schema mappings.

It should be pointed out that in [16], the authors introduce the order $\preceq_{\text{E}}$ but do not study its fundamental properties. Interestingly, the following result shows that $\preceq_{\text{S}}$, $\preceq_{\text{R}}$ and $\preceq_{\text{E}}$ coincide for the class of st-mappings (mappings not containing null values in source instances) that are specified by st-tgds. Thus, the machinery developed in this paper for $\preceq_{\text{S}}$ can also be used for $\preceq_{\text{E}}$ and $\preceq_{\text{R}}$ in this case.

Proposition 3.7 *Let $\mathcal{M}_1 = (\mathbf{S}, \mathbf{T}_1, \Sigma_1)$ and $\mathcal{M}_2 = (\mathbf{S}, \mathbf{T}_2, \Sigma_2)$ be st-mappings, where Σ_1, Σ_2 are sets of st-tgds. Then the following statements are equivalent:*

(1) $\mathcal{M}_1 \preceq_{\text{S}} \mathcal{M}_2$.

(2) $\mathcal{M}_1 \preceq_{\text{R}} \mathcal{M}_2$.

(3) $\mathcal{M}_1 \preceq_{\text{E}} \mathcal{M}_2$.

4. FUNDAMENTAL PROPERTIES OF THE ORDER $\preceq_{\text{S}}$

In this section, we provide a characterization of the order $\preceq_{\text{S}}$, that gives evidence of why it is appropriate to compare the amount of source information being transferred by two mappings. Furthermore, we use this characterization to deal with some algorithmic problems related to this notion. In particular, we show in Section 4.2 that the order $\preceq_{\text{S}}$ is decidable for the class of mappings specified by $\text{CQ}^{\neq}$-TO-CQ dependencies, that includes the widely used class of st-tgds.

4.1 Characterizing the order $\preceq_{\text{S}}$

In this section, we present a characterization of the order $\preceq_{\text{S}}$ for mappings given by FO-TO-CQ dependencies, that is based on query rewriting. More specifically, given a mapping $\mathcal{M}$ from a schema $\mathbf{S}$ to a schema $\mathbf{T}$ and a query Q over $\mathbf{S}$, we say that Q is *target rewritable* under $\mathcal{M}$ if there exists a query Q' over $\mathbf{T}$ such that for every instance I of $\mathbf{S}$, it holds that $Q(I) = \text{certain}_{\mathcal{M}}(Q', I)$. That is, Q is target rewritable under a mapping $\mathcal{M}$ if $\mathcal{M}$ transfers enough source information to be able to answer Q by using the target data. Thus, the amount of source information transferred by two mappings can be compared in terms of the queries that are target rewritable under them, which gives rise to the following characterization:

Theorem 4.1 *Let $\mathcal{M}_1 = (\mathbf{S}, \mathbf{T}_1, \Sigma_1)$ and $\mathcal{M}_2 = (\mathbf{S}, \mathbf{T}_2, \Sigma_2)$ be st-mappings, where Σ_1, Σ_2 are sets of* FO-TO-CQ *dependencies. Then the following statements are equivalent:*

(1) $\mathcal{M}_1 \preceq_{\text{S}} \mathcal{M}_2$.

(2) For every query Q over $\mathbf{S}$, if Q is target rewritable in $\mathcal{M}_1$, then Q is target rewritable in $\mathcal{M}_2$.

It is important to notice that the preceding theorem considers the class of all queries, as defined in Section 2. Thus, Theorem 4.1 gives strong evidence in favor of the order $\preceq_{\mathrm{s}}$.

4.2 Fundamental algorithmic issues for the order $\preceq_{\mathrm{s}}$

Some algorithmic issues related to the order $\preceq_{\mathrm{s}}$ play a key role in the development of algorithms for some metadata management operators. In this section, we study two such fundamental issues. More precisely, we start by answering the question of whether relation $\preceq_{\mathrm{s}}$ is decidable. Not surprisingly, we obtain a negative answer for the class of FO-TO-CQ dependencies.

Theorem 4.2 *The problem of verifying, given st-mappings $\mathcal{M}_1$ and $\mathcal{M}_2$ specified by sets of* FO-TO-CQ *dependencies, whether $\mathcal{M}_1 \preceq_{\mathrm{s}} \mathcal{M}_2$ is undecidable.*

Interestingly, the characterization of $\preceq_{\mathrm{s}}$ can be used to prove that the order is decidable for mappings given by $\mathrm{CQ}^{\neq}$-TO-CQ dependencies, which includes the fundamental class of st-tgds.

Theorem 4.3 *The problem of verifying, given st-mappings $\mathcal{M}_1$ and $\mathcal{M}_2$ specified by sets of $\mathrm{CQ}^{\neq}$-TO-CQ dependencies, whether $\mathcal{M}_1 \preceq_{\mathrm{s}} \mathcal{M}_2$ is in* coNEXPTIME.

A second fundamental algorithmic issue is the problem of constructing a mapping $\mathcal{N}$ such that $\mathcal{M}_1 = \mathcal{M}_2 \circ \mathcal{N}$, whenever $\mathcal{M}_1 \preceq_{\mathrm{s}} \mathcal{M}_2$. Next, we present an algorithm that solves this problem for $\mathrm{CQ}^{\neq}$-TO-CQ dependencies, which uses the following terminology.

Let $\mathcal{M}$ be a mapping and Q be a query that is target rewritable under $\mathcal{M}$, and let Q' be a query such that $Q(I) = \mathrm{certain}_{\mathcal{M}}(Q', I)$ holds for every instance I. Then, we say that Q' is a *target rewriting* for Q under $\mathcal{M}$. Correspondingly, we also say that Q is a *source rewriting* of Q'.

It can be proved that for mappings specified by FO-TO-CQ dependencies, it is always possible to compute the *source rewriting* of a conjunctive query Q', that is, there exists a procedure SOURCEREWRITING that, given such a mapping $\mathcal{M}$ and a query Q' in CQ, computes a query Q in FO that is a source rewriting of Q'. In particular, if the input mapping is specified by $\mathrm{CQ}^{\neq}$-TO-CQ dependencies, then the output of the procedure is a query Q in $\mathrm{UCQ}^{=,\neq}$ [3, 9].

For the class of mappings specified by $\mathrm{CQ}^{\neq}$-TO-CQ dependencies, target rewritings can also be computed. More precisely, it follows from the proof of Theorem 4.1 that there exists a procedure TARGETREWRITING that, given a mapping $\mathcal{M}$ specified by a set of $\mathrm{CQ}^{\neq}$-TO-CQ dependencies and a target rewritable query Q in $\mathrm{UCQ}^{=,\neq}$, computes a query in $\mathrm{UCQ}^{=,\neq,\mathbf{C}}$ that is a target rewriting of Q. The following algorithm uses this procedure and the algorithm mentioned above as black boxes.

Algorithm COMPUTEORDER($\mathcal{M}_1, \mathcal{M}_2$)

Input: st-mappings $\mathcal{M}_1 = (\mathbf{S}, \mathbf{T}_1, \Sigma_1)$ and $\mathcal{M}_2 = (\mathbf{S}, \mathbf{T}_2, \Sigma_2)$, where Σ_1, Σ_2 are sets of $\mathrm{CQ}^{\neq}$-TO-CQ dependencies and $\mathcal{M}_1 \preceq_{\mathrm{s}} \mathcal{M}_2$.

Output: A mapping $\mathcal{N}$ such that $\mathcal{M}_1 = \mathcal{M}_2 \circ \mathcal{N}$.

1. Construct a set Σ' of dependencies as follows. For every dependency $\varphi(\bar{x}) \rightarrow \psi(\bar{x}) \in \Sigma_1$ repeat the following:
   (a) Let $\alpha(\bar{x})$ be the query in $\mathrm{UCQ}^{=,\neq}$ that is the output of SOURCEREWRITING($\mathcal{M}_1, \psi(\bar{x})$).
   (b) Let $\beta(\bar{x})$ be the query in $\mathrm{UCQ}^{=,\neq,\mathbf{C}}$ that is the output of TARGETREWRITING($\mathcal{M}_2, \alpha(\bar{x})$).
   (c) For every disjunct $\gamma(\bar{x})$ of $\beta(\bar{x})$, add formula $\gamma(\bar{x}) \wedge \mathbf{C}(\bar{x}) \rightarrow \psi(\bar{x})$ to Σ'.
2. Let Σ be the set obtained from Σ' by eliminating the equalities by using variable replacements.
3. Return the mapping $\mathcal{N} = (\mathbf{T}_2, \mathbf{T}_1, \Sigma)$

□

Proposition 4.4 COMPUTEORDER($\mathcal{M}_1, \mathcal{M}_2$) *returns a mapping $\mathcal{N}$ specified by a set of $\mathrm{CQ}^{\neq,\mathbf{C}}$-TO-CQ dependencies such that $\mathcal{M}_1 = \mathcal{M}_2 \circ \mathcal{N}$.*

5. TWO APPLICATIONS OF $\preceq_{\mathrm{s}}$ IN DATA EXCHANGE

The issue of providing foundations for metadata management has appeared in different contexts. In particular, in the data exchange context, the schema evolution problem has been a driving force for the development of the composition and inverse operators [14, 19, 11]. Here, we show that the machinery developed in the previous sections can be used as a uniform framework to study the inverse operator and the schema evolution problem.

5.1 Inverting schema mappings

In this section, we focus on the definition of the inverse operator given by Fagin in [11], and we show that it can be defined in terms of the order $\preceq_{\mathrm{s}}$. Interestingly, this characterization can be used to extend, and provide simpler proofs of, some of the fundamental results about this operator.

We start by recalling the definition of inverse given by Fagin [11]. For a ground schema $\mathbf{R}$, let $\widehat{\mathbf{R}}$ be the schema $\{\widehat{R} \mid R \in \mathbf{R}\}$, and $\widehat{\mathrm{Id}}_{\mathbf{R}} = (\mathbf{R}, \widehat{\mathbf{R}}, \Sigma)$ be an *identity mapping*, where Σ contains a dependency of the form $R(x_1, \ldots, x_k) \rightarrow \widehat{R}(x_1, \ldots, x_k)$, for every k-ary predicate $R \in \mathbf{R}$. Then given $\mathcal{M}$ from $\mathbf{R}$ to $\mathbf{R}_1$ and $\mathcal{M}'$ from $\mathbf{R}_1$ to $\widehat{\mathbf{R}}$, mapping $\mathcal{M}'$ is said to be an inverse of mapping $\mathcal{M}$ if $\mathcal{M} \circ \mathcal{M}' = \widehat{\mathrm{Id}}_{\mathbf{R}}$ [11].

The following theorem shows that the notion of inverse introduced by Fagin can be defined in terms of the order $\preceq_{\mathrm{s}}$. It should be noticed that this theorem focuses on a particular class of mappings, as $\widehat{\mathrm{Id}}_{\mathbf{R}}$ is appropriate as an identity only for mappings in this class. Specifically, given a mapping $\mathcal{M}$ from $\mathbf{R}$ to $\mathbf{R}_1$, $\mathcal{M}$ is said to be *total* if $\mathrm{dom}(\mathcal{M}) = \mathrm{Inst}(\mathbf{R})$, and $\mathcal{M}$ is said to be *closed-down on the left* if whenever $(I, J) \in \mathcal{M}$ and $I' \subseteq I$, it holds that $(I', J) \in \mathcal{M}$.

Theorem 5.1 *Let $\mathcal{M}$ be a mapping from a ground schema $\mathbf{R}$ to a schema $\mathbf{R}_1$ that is total and closed-down on the left. Then the following statements are equivalent.*

(1) $\mathcal{M}$ is invertible.

(2) $\mathcal{M}$ is $\preceq_{\mathrm{s}}$-maximal in the class of total and closed-down on the left mappings.

(3) $\widehat{\mathrm{Id}}_{\mathbf{R}} \preceq_{\mathrm{s}} \mathcal{M}$.

The preceding theorem can be used to extend some of the fundamental results that have been obtained for the inverse operator. It is important to notice that these results were proved in a series of papers by using different techniques [11, 15, 17]. Interestingly, our approach provides a unifying framework for these results. More specifically, a first fundamental question about the notion of inverse

is whether it is decidable. In [17], it is shown that the problem of verifying, given a mapping $\mathcal{M}$ specified by a set of st-tgds, whether $\mathcal{M}$ is invertible is decidable. From Theorem 5.1, we know that if $\mathcal{M}$ is a mapping from a ground schema $\mathbf{R}$ to a schema $\mathbf{R}_1$ specified by a set of st-tgds, then the previous problem can be reduced to the problem of verifying whether $\widehat{\mathrm{Id}}_{\mathbf{R}} \preceq_{\mathrm{S}} \mathcal{M}$, which is known to be decidable by Theorem 4.3. Moreover, we also known from Theorems 5.1 and 4.3 that the same holds for the class of $\mathrm{CQ}^{\neq}$-TO-CQ dependencies, which gives us the following extension of the decidability result in [17]:

Corollary 5.2 *The problem of verifying, given an st-mapping $\mathcal{M}$ specified by a set of* $\mathrm{CQ}^{\neq}$-TO-CQ *dependencies, whether $\mathcal{M}$ is invertible is decidable.*

It should be noticed that in [17], the authors prove that testing invertibility of st-tgds is coNP-complete, which cannot be obtained from our results. A second fundamental question about any notion of inverse is how to compute it. In [15], there is given an algorithm for computing the inverse of a mapping specified by st-tgds. From Theorem 5.1, we know that algorithm COMPUTEORDER can also be used for this task, and not only for the case of st-tgds but also for the larger class of $\mathrm{CQ}^{\neq}$-TO-CQ dependencies.

Proposition 5.3 *Let $\mathcal{M} = (\mathbf{R}, \mathbf{R}_1, \Sigma)$ be an invertible st-mapping specified by a set Σ of* $\mathrm{CQ}^{\neq}$-TO-CQ *dependencies. Then the output of* COMPUTEORDER$(\widehat{\mathrm{Id}}_{\mathbf{R}}, \mathcal{M})$ *is an inverse of $\mathcal{M}$.*

The use of query rewriting in COMPUTEORDER makes the above approach for computing inverses more suitable than the approach proposed in [15] for optimization. In fact, one can reuse the large number of techniques developed for query rewriting [20, 8, 18, 27] when implementing procedure COMPUTEORDER.

As a corollary of Proposition 5.3, we obtain a third fundamental result for the notion of inverse proposed by Fagin [11].

Corollary 5.4 *For every invertible st-mapping $\mathcal{M}$ specified by a set of* $\mathrm{CQ}^{\neq}$-TO-CQ *dependencies, there exists an inverse of $\mathcal{M}$ that is specified by a set of* $\mathrm{CQ}^{\neq,\mathbf{C}}$-TO-CQ *dependencies.*

It is important to notice that Fagin et al. showed in [15] that if a mapping $\mathcal{M}$ specified by a set of st-tgds is invertible, then it has an inverse that is given by a set of $\mathrm{CQ}^{\neq,\mathbf{C}}$-TO-CQ dependencies. The above corollary extends this results for the class of invertible mappings specified by $\mathrm{CQ}^{\neq}$-TO-CQ dependencies.

As we mentioned in Section 3.1, the notion of information loss presented in [16] is tightly connected with invertibility of mappings. In fact, Fagin et al. claim in [16] that if $\mathcal{M}_1$ and $\mathcal{M}_2$ are mappings specified by st-tgds, then $\mathcal{M}_2$ is less lossy than $\mathcal{M}_1$ when, intuitively, "$\mathcal{M}_2$ is *more invertible* than $\mathcal{M}_1$". We conclude this subsection by showing that if one goes beyond st-tgds (which is the class of mappings considered in [16]), the orders $\preceq_{\mathrm{R}}$ and $\preceq_{\mathrm{E}}$ fail to capture the idea of *being more invertible*. Notice that Theorem 5.1 shows that our order $\preceq_{\mathrm{S}}$ captures exactly the intuition mentioned in [16] for a large class of mappings, which gives evidence of the usefulness of $\preceq_{\mathrm{S}}$ to compare schema mappings. We begin by showing that $\preceq_{\mathrm{R}}$ does not capture invertibility beyond st-tgds.

Proposition 5.5 *There exists an st-mapping $\mathcal{M}$ specified by a set of* CQ-TO-UCQ *dependencies such that $\mathcal{M}$ is $\preceq_{\mathrm{R}}$-maximal on the class of total and closed-down on the left mappings and $\mathcal{M}$ is not invertible.*

In [16], the authors introduce an alternative notion of inversion for mappings with null values in source and target instances. They call this notion *extended invertibility*, and show that the order $\preceq_{\mathrm{E}}$ is tightly connected with this extended notion of inversion (see [16] for details about extended inverses). The following result shows that beyond st-tgds, $\preceq_{\mathrm{E}}$ captures neither the notion of invertibility nor the notion of extended invertibility.

Proposition 5.6 *There exists a mapping $\mathcal{M}$ such that (1) $\mathcal{M}$ is specified by a set of* CQ-TO-UCQ *dependencies, (2) $\mathcal{M}$ has nulls in source and target instances, (3) $\mathcal{M}$ is $\preceq_{\mathrm{E}}$-maximal in the class of all mappings, and (4) $\mathcal{M}$ is neither invertible nor extended invertible.*

We conclude this subsection by pointing out that we have mainly focused here on the notion of inverse proposed by Fagin in [11]. However, in the last few years other alternative semantics for the inverse operator have been proposed [15, 3, 16, 2]. Thus, it is natural to ask whether these notions can also be characterized in term of the order $\preceq_{\mathrm{S}}$, and whether the machinery proposed in this paper can be used to improve our understanding of these notions. Although we have made a bit of progress in this direction, these questions remain unanswered.

5.2 Schema evolution

The schema evolution problem has been one of the driving forces behind the study of the composition and inverse operators [14, 19, 11]. Two main scenarios have been identified for this problem. In the first scenario, one is given a mapping $\mathcal{M}$ from a schema $\mathbf{S}$ to a schema $\mathbf{T}$, and a mapping $\mathcal{M}'$ that specifies how $\mathbf{T}$ evolves into a new schema $\mathbf{T}'$. The schema evolution problem is then to provide a mapping from $\mathbf{S}$ to $\mathbf{T}'$, that captures the metadata provided by $\mathcal{M}$ and $\mathcal{M}'$. In this scenario, it is always possible to find a solution for this problem by using the composition operator [14, 19], as mapping $\mathcal{M} \circ \mathcal{M}'$ correctly represents the relationship between $\mathbf{S}$ and $\mathbf{T}'$. In the second scenario, one is also given a mapping $\mathcal{M}$ from a schema $\mathbf{S}$ to a schema $\mathbf{T}$, but in this case $\mathbf{S}$ evolves into a new schema $\mathbf{S}'$, and the relationship between $\mathbf{S}$ and $\mathbf{S}'$ is given by a mapping $\mathcal{M}'$. Then again the question is how to construct a mapping from $\mathbf{S}'$ to $\mathbf{T}$ that captures the metadata provided by $\mathcal{M}$ and $\mathcal{M}'$. In this section, we use the machinery developed in the previous sections to formally study this problem. It is important to notice that we focus on the second scenario, as the first one has been completely solved by using the composition operator [14].

Let $\mathcal{M}_1$ be a mapping from a schema $\mathbf{R}$ to a schema $\mathbf{R}_1$, and $\mathcal{M}_2$ a mapping from $\mathbf{R}$ to a schema $\mathbf{R}_2$. Then a mapping $\mathcal{N}$ is *a solution for the schema evolution problem for* $(\mathcal{M}_1, \mathcal{M}_2)$ if $\mathcal{M}_1 = \mathcal{M}_2 \circ \mathcal{N}$. The following result shows that the schema evolution problem can be characterized in terms of the order $\preceq_{\mathrm{S}}$, as it is just a reformulation of the definition of $\preceq_{\mathrm{S}}$.

Proposition 5.7 *There exists a solution for the schema evolution problem for $(\mathcal{M}_1, \mathcal{M}_2)$ iff $\mathcal{M}_1 \preceq_{\mathrm{S}} \mathcal{M}_2$.*

Thus, as a corollary of Theorem 4.3 and Proposition 4.4, we obtain a solution for the schema evolution problem for the class of mappings specified by $\mathrm{CQ}^{\neq}$-TO-CQ dependencies:

Corollary 5.8 *The schema evolution problem is decidable for the class of st-mappings specified by* $\mathrm{CQ}^{\neq}$-TO-CQ *dependencies. Moreover, for every $\mathcal{M}_1$, $\mathcal{M}_2$ in this class, if there exists a solution for the schema evolution problem for $(\mathcal{M}_1, \mathcal{M}_2)$, then* COMPUTEORDER$(\mathcal{M}_1, \mathcal{M}_2)$ *returns a solution for this problem specified by a set of* $\mathrm{CQ}^{\neq,\mathbf{C}}$-TO-CQ *dependencies.*

The previous approach is specific to the class of $CQ^{\neq}$-TO-CQ dependencies and, thus, it is natural to ask how one can deal with the schema evolution problem in a more general setting. It has been argued that the combination of the inverse and composition operators can be used for this purpose, and the mapping $\mathcal{M}_2^\star \circ \mathcal{M}_1$ has been proposed as a solution for the schema evolution problem for $(\mathcal{M}_1, \mathcal{M}_2)$ [11], where $\mathcal{M}_2^\star$ represents an inverse of mapping $\mathcal{M}_2$. Next we show that this approach is appropriate to solve the schema evolution problem, if one considers the notion of maximum recovery [3] which generalize the notion of inverse proposed in [11] (see Section 3.1 for the definition of maximum recovery).

Proposition 5.9 *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be st-mappings. If $\mathcal{M}_1 \preceq_{\text{S}} \mathcal{M}_2$ and $\mathcal{M}_2^\star$ is a maximum recovery of $\mathcal{M}_2$, then $\mathcal{M}_2^\star \circ \mathcal{M}_1$ is a solution for the schema evolution problem for $(\mathcal{M}_1, \mathcal{M}_2)$.*

6. TARGET INFORMATION AND REDUNDANCY

In this section, we use the order $\preceq_{\text{S}}$ to define three additional concepts which, together with $\preceq_{\text{S}}$, provide a theoretical framework to study complex metadata management operators such as extract and merge [22]. More precisely, we introduce in Section 6.1 an order to compare mappings that possess the same target schema. This order, denoted by $\preceq_{\text{T}}$, intuitively measures the amount of target information *covered* by a mapping. As there may exist multiple ways to transfer the same information from a source schema, or to cover the same information of a target schema, one also needs a way of distinguishing between different alternatives. To deal with this requirement, in Sections 6.2 and 6.3, we use the orders $\preceq_{\text{S}}$ and $\preceq_{\text{T}}$ to introduce the notions of target redundancy and source redundancy, and show that they capture the intuition of using the *exact amount of resources* needed to transfer information between schemas.

6.1 Target information covered by a mapping

In some metadata management scenarios, it is important to measure the amount of target information *covered* by a mapping. When $\preceq_{\text{S}}$ was introduced, we said that $\mathcal{M}_1 \preceq_{\text{S}} \mathcal{M}_2$ if $\mathcal{M}_2$ transfers enough source information to be able to reconstruct the information transferred by $\mathcal{M}_1$. Similarly, we say that $\mathcal{M}_2$ *covers as much target information as* $\mathcal{M}_1$, denoted by $\mathcal{M}_1 \preceq_{\text{T}} \mathcal{M}_2$, if $\mathcal{M}_2$ covers enough target information to be able to reconstruct the information that is covered by $\mathcal{M}_1$. More precisely,

Definition 6.1 (Order $\preceq_{\text{T}}$) *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be mappings that share the target schema. Then $\mathcal{M}_1 \preceq_{\text{T}} \mathcal{M}_2$ if there exists a mapping $\mathcal{N}$ such that $\mathcal{M}_1 = \mathcal{N} \circ \mathcal{M}_2$.*

Moreover, we say that $\mathcal{M}_1$ and $\mathcal{M}_2$ *cover the same target information*, and write $\mathcal{M}_1 \equiv_{\text{T}} \mathcal{M}_2$, if $\mathcal{M}_1 \preceq_{\text{T}} \mathcal{M}_2$ and $\mathcal{M}_2 \preceq_{\text{T}} \mathcal{M}_1$. As pointed out above, $\preceq_{\text{T}}$ can be defined in terms of the order $\preceq_{\text{S}}$.

Proposition 6.2 $\mathcal{M}_1 \preceq_{\text{T}} \mathcal{M}_2$ *iff* $(\mathcal{M}_1)^{-1} \preceq_{\text{S}} (\mathcal{M}_2)^{-1}$.

Characterizing the order $\preceq_{\text{T}}$. We provide here a characterization of the order $\preceq_{\text{T}}$ for mappings given by FO-TO-CQ dependencies. But before we need to recall some definitions. Let K_1 and K_2 be instances of a schema $\mathbf{R}$. A *homomorphism* h from K_1 to K_2 is a function $h : \text{dom}(K_1) \to \text{dom}(K_2)$ such that: (1) $h(c) = c$ for every $c \in \mathbf{C} \cap \text{dom}(K_1)$, and (2) for every $R \in \mathbf{R}$ and tuple $(a_1, \ldots, a_k) \in R^{K_1}$, it holds that $(h(a_1), \ldots, h(a_k)) \in R^{K_2}$ [12]. Moreover, an instance J is said to be a universal solution for an instance I under a mapping $\mathcal{M}$ if $J \in \text{Sol}_{\mathcal{M}}(I)$ and for every $J' \in \text{Sol}_{\mathcal{M}}(I)$, there exists a homomorphism from J to J' [12].

Theorem 6.3 *Let $\mathcal{M}_1 = (\mathbf{S}_1, \mathbf{T}, \Sigma_1)$ and $\mathcal{M}_2 = (\mathbf{S}_2, \mathbf{T}, \Sigma_2)$ be st-mappings, where Σ_1, Σ_2 are sets of* FO-TO-CQ *dependencies. Then the following statements are equivalent:*

(1) $\mathcal{M}_1 \preceq_{\text{T}} \mathcal{M}_2$.

(2) For every instance J of $\mathbf{T}$, if J is a universal solution for some instance under $\mathcal{M}_1$, then J is a universal solution for some instance under $\mathcal{M}_2$.

The above characterization supports our claim that $\preceq_{\text{T}}$ measures the amount of information covered by a mapping. In fact, universal solutions have been identified as a fundamental class of solutions in data exchange, as they represent (in a precise sense) the entire space of solutions [12, 13]. Our characterization shows that if $\mathcal{M}_1 \preceq_{\text{T}} \mathcal{M}_2$, then the space of possible universal solutions for $\mathcal{M}_1$ is contained in that of $\mathcal{M}_2$.

We conclude this subsection by pointing out that although for the case of st-mappings given by FO-TO-CQ dependencies, we have provided a characterization of the order $\preceq_{\text{T}}$ in term of the well-studied notion of universal solution, it remains as an open problem whether the order $\preceq_{\text{T}}$ is decidable in this case. That is, it is open whether there exists an algorithm that, given st-mappings $\mathcal{M}_1$ and $\mathcal{M}_2$ specified by sets of FO-TO-CQ dependencies, verifies if $\mathcal{M}_1 \preceq_{\text{T}} \mathcal{M}_2$.

6.2 Target redundancy in schema mappings

There may exist many different ways to transfer the same information and, hence, metadata management systems should handle some criteria that help them in identifying the best alternatives, in terms of the resources they use. In this section, we introduce one such criteria, the notion of target redundancy. We use the following example to motivate our definition.

Example 6.4. Let $\mathcal{M}_1 = (\mathbf{R}, \mathbf{R}_1, \Sigma_1)$ and $\mathcal{M}_2 = (\mathbf{R}, \mathbf{R}_2, \Sigma_2)$ be mappings specified by dependencies $A(x) \to R(x)$ and $A(x) \to P(x, x)$, respectively, and where $\mathbf{R}$, $\mathbf{R}_1$ and $\mathbf{R}_2$ are ground schemas. It is easy to see that $\mathcal{M}_1 \equiv_{\text{S}} \mathcal{M}_2$. However, $\mathcal{M}_1$ can be considered *better* than $\mathcal{M}_2$ in the sense that it does not *waste* resources when transferring information from $\mathbf{R}$. In fact, every instance in $\text{range}(\mathcal{M}_1)$ is *essential* for $\mathcal{M}_1$, as it is a universal solution for an instance of $\mathbf{R}$ under $\mathcal{M}_1$. On the other hand, every universal solution of $\mathcal{M}_2$ can only contain tuples of the form $P(a, a)$, which implies that several instances in $\text{range}(\mathcal{M}_2)$ are not essential for this mapping. □

As shown in Example 6.4, it would be advisable to design mappings for which every target instance is *essential* in transferring source information. In the following definition, we use the order $\preceq_{\text{S}}$ to formalize this notion.

Definition 6.5 (Target redundancy) *A mapping $\mathcal{M}$ is* target redundant *if there exists an instance $J^\star \in \text{range}(\mathcal{M})$ such that $\mathcal{M}^\star = \{(I, J) \in \mathcal{M} \mid J \neq J^\star\}$ satisfies that $\mathcal{M}^\star \equiv_{\text{S}} \mathcal{M}$.*

Thus, intuitively, we say that a mapping $\mathcal{M}$ is *target non-redundant* if we cannot remove any target instance from $\mathcal{M}$, and still be able to transfer the same amount of information.

Example 6.6. (Example 6.4 continued) $\mathcal{M}_1$ is target non-redundant, but $\mathcal{M}_2$ is target redundant as $\mathcal{M}_2 \equiv_{\text{S}} \mathcal{M}^\star$, where $\mathcal{M}^\star$ is generated from $\mathcal{M}_2$ by removing from $\text{range}(\mathcal{M}_2)$ an arbitrary instance that contains a tuple $P(a, b)$ with $a \neq b$. Notice that if we add $P(x, y) \to x = y$ as a target constraint to $\mathcal{M}_2$, the resulting mapping is target non-redundant. □

Characterizing target redundancy. We provide here a characterization of the notion of target redundancy for mappings specified

by FO-TO-CQ dependencies. But first, we shed light on the issue of how the use of null values generate redundant information.

Null values are used in data exchange to deal with incomplete information. For example, assume that one needs to transfer data from a schema $\text{Emp}_1(\cdot)$ storing a list of employee names, to a schema $\text{Emp}_2(\cdot,\cdot)$ storing a list of employee names and the departments where they work. Given that the source schema does not contain any information about departments, one has to use a dependency of the form $\text{Emp}_1(x) \rightarrow \exists y\, \text{Emp}_2(x,y)$. Thus, when exchanging data, a null value n is included in a tuple $\text{Emp}_2(a,n)$ if one does not know the department where employee a works. Null values introduce redundant information, as they allow one to represent the same data in many different ways. For example, a target instance $\text{Emp}_2(a,n)$ contains exactly the same information as a target instance $\text{Emp}_2(a,n')$ if n and n' are null values. Thus, instance $\text{Emp}_2(a,n')$ is really not needed when transferring source data. In fact, the next result shows that every st-mapping specified by FO-TO-CQ dependencies that allows null values in the target schema is target redundant (recall that we use the term st-mapping for mappings that only have constants in their source instances, and constants and nulls in their target instances).

Proposition 6.7 *Let $\mathcal{M}$ be an st-mapping specified by a set of* FO-TO-CQ *dependencies. Then $\mathcal{M}$ is target redundant.*

It is important to notice that target redundancy does not mean that a mapping is poorly designed, as in some cases the redundancy, and in particular the use of null values, is unavoidable (like in the above mapping $\text{Emp}_1(x) \rightarrow \exists y\, \text{Emp}_2(x,y)$). Nevertheless, when mappings are specified by using dependencies without existential quantifiers in their conclusions, that is, full dependencies, there is no need to use null values as one does not need to deal with incomplete information. We provide in the following theorem a characterization of the notion of target redundancy for mappings specified by full dependencies that allow only constant values in source and target schemas.

Theorem 6.8 *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S}$ and $\mathbf{T}$ are ground schemas and Σ is a set of full* FO-TO-CQ *dependencies. Then the following properties are equivalent:*

(1) $\mathcal{M}$ is target non-redundant.

(2) Every instance in $\text{range}(\mathcal{M})$ is a universal solution for some instance in $\text{dom}(\mathcal{M})$.

Thus, our characterization shows that a mapping $\mathcal{M}$ defined by a set of full FO-TO-CQ dependencies is target non-redundant if and only if every instance in $\text{range}(\mathcal{M})$ is essential for $\mathcal{M}$, as it is a universal solution for some instance in $\text{dom}(\mathcal{M})$.

As for the case of the order $\preceq_{\text{T}}$, we have provided a characterization of the notion of target non-redundancy in terms of the well-studied notion of universal solution. However, it remains as an open problem whether there exists an algorithm that, given a mapping $\mathcal{M}$ specified by a set of full FO-TO-CQ dependencies, verifies whether $\mathcal{M}$ is target non-redundant.

6.3 Source redundancy

Just as there exists a symmetric definition for the order $\preceq_{\text{s}}$, so is the case for the notion of target redundancy. In fact, we use the order $\preceq_{\text{T}}$ in the following definition to introduce the notion of source redundancy, which also plays a fundamental role in providing foundations for metadata management.

Definition 6.9 (Source redundancy) *A mapping $\mathcal{M}$ is* source redundant *if there exists an instance $I^\star \in \text{dom}(\mathcal{M})$ such that $\mathcal{M}^\star = \{(I,J) \in \mathcal{M} \mid I \neq I^\star\}$ satisfies $\mathcal{M}^\star \equiv_{\text{T}} \mathcal{M}$.*

That is, a mapping $\mathcal{M}$ is source redundant if one can eliminate an instance from $\text{dom}(\mathcal{M})$ and still cover the same amount of target information. Not surprisingly, there is a tight relation between target and source redundancy.

Proposition 6.10 *$\mathcal{M}$ is source redundant if and only if $\mathcal{M}^{-1}$ is target redundant.*

Characterizing source redundancy. We conclude this section by providing a characterization of source redundancy for the class of mappings specified by FO-TO-CQ dependencies. From the point of view of covering target information, a non-redundant mapping should not assign the same space of solutions to two different source instances, as this means that one of them is not necessary. The following theorem shows that the notion of source redundancy captures this intuition.

Theorem 6.11 *Let $\mathcal{M}$ be an st-mapping specified by a set of* FO-TO-CQ *dependencies. Then the following statements are equivalent:*

(1) $\mathcal{M}$ is source non-redundant.

(2) For every pair of source instances I_1, I_2, if $I_1 \neq I_2$ then $\text{Sol}_{\mathcal{M}}(I_1) \neq \text{Sol}_{\mathcal{M}}(I_2)$.

Property (2) above is called *unique-solutions* property in [11], where it is shown to be a necessary condition for invertibility.

As a final remark, it should be noticed that it is open whether there exists an algorithm that, given an st-mapping $\mathcal{M}$ specified by a set of FO-TO-CQ dependencies, verifies if $\mathcal{M}$ is source non-redundant.

7. THE EXTRACT OPERATOR

Consider a mapping $\mathcal{M}$ between schemas $\mathbf{S}$ and $\mathbf{T}$, and assume that $\mathbf{S}$ is the schema of a database that is only being used to map data through $\mathcal{M}$. In general, not all the information of $\mathbf{S}$ *participates* in the mapping and, thus, it is natural to ask whether one can *upgrade* $\mathbf{S}$ into a new schema that stores only the information being mapped by $\mathcal{M}$, that is, whether one can *extract* from $\mathbf{S}$ the portion of the schema that is actually participating in $\mathcal{M}$. This is the intended meaning of the *extract operator* [22, 23], as shown in the following example.

Example 7.1. Let $\mathbf{S} = \{P(\cdot,\cdot), R(\cdot,\cdot), S(\cdot,\cdot)\}$ and $\mathbf{T} = \{T(\cdot,\cdot), U(\cdot,\cdot), V(\cdot,\cdot,\cdot)\}$, and assume that $\mathbf{S}$ is a ground schema. Consider a mapping $\mathcal{M}$ from $\mathbf{S}$ to $\mathbf{T}$ given by the following dependencies:

$$P(x,y) \rightarrow \exists u\, T(x,u) \wedge U(x,x) \quad (1)$$

$$P(x,y) \wedge R(y,z) \rightarrow \exists v\, V(x,y,v) \quad (2)$$

The first column of P is being transferred from the source by dependency (1), while all the tuples in P that can be joined with some tuples in R are being transferred by dependency (2). Moreover, notice that relation S is not participating at all in the mapping.

A natural way to upgrade $\mathbf{S}$, and store only the data that is transferred by $\mathcal{M}$, is to have a new ground schema $\mathbf{S}' = \{P_1(\cdot), P_2(\cdot,\cdot)\}$, where relation $P_1(\cdot)$ is used to store the first component of P, and relation $P_2(\cdot,\cdot)$ is used to store the tuples in P that can be joined with some tuples in R. But we can do even better. Notice that by the intended meaning of relations P_1 and P_2, one knows

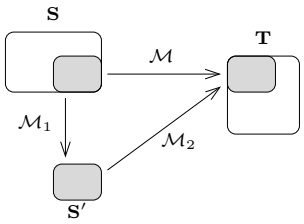

Figure 1: $(\mathcal{M}_1, \mathcal{M}_2)$ is an EXTRACT of $\mathcal{M}$.

that they must satisfy the inclusion dependency $P_2(x, y) \rightarrow P_1(x)$. Thus, schema $\mathbf{S}'$ plus this dependency still have enough capacity to store all the source information being transferred by $\mathcal{M}$. □

Given a mapping $\mathcal{M}$ from a schema $\mathbf{S}$ to a schema $\mathbf{T}$, the idea of the *extract* operator is to create a new source schema $\mathbf{S}'$ that captures *exactly* the information that is participating in $\mathcal{M}$ and *no other information* [22, 23]. As shown in Figure 1, a solution for the extract operator has two components, a mapping $\mathcal{M}_1$ from $\mathbf{S}$ to $\mathbf{S}'$ that drives the *migration* from the old to the new source schema, and a mapping $\mathcal{M}_2$ from $\mathbf{S}'$ to $\mathbf{T}$ that states how data should be mapped from the new source schema to the target schema. But what are the conditions that have to be imposed on mappings $\mathcal{M}_1$ and $\mathcal{M}_2$ (and schema $\mathbf{S}'$) to capture the intuition behind the extract operator? A set of such conditions was proposed by Melnik et al. in [22, 23]. In what follows, we show that the machinery developed in the previous sections can be used to provide a natural semantics for the extract operator. We compare our proposal with that of Melnik et al. [22, 23] in Section 7.2.

Assume that $\mathcal{M}$, $\mathcal{M}_1$ and $\mathcal{M}_2$ are the mappings shown in Figure 1. The first condition that we impose on $\mathcal{M}_1$ and $\mathcal{M}_2$, to consider them a valid extract of $\mathcal{M}$, is that the composition of $\mathcal{M}_1$ and $\mathcal{M}_2$ is equal to $\mathcal{M}$:

(E1) $\mathcal{M}_1 \circ \mathcal{M}_2 = \mathcal{M}$.

In this way, one ensures that for every instance I of $\mathbf{S}$, if one first *migrates* I from $\mathbf{S}$ to $\mathbf{S}'$, and then maps the result to $\mathbf{T}$, then one obtains exactly the same space of possible solutions as if I is being mapped by using the initial mapping $\mathcal{M}$. Notice that (E1) does not impose any restrictions over $\mathcal{M}_1$ and $\mathcal{M}_2$ alone. We do that with the next conditions.

The intended meaning of the extract operator is to store in a new schema exactly the information that is being transferred by the initial mapping. Thus, we require that $\mathcal{M}_1$ transfers from $\mathbf{S}$ to $\mathbf{S}'$ the same amount of source information as $\mathcal{M}$. Similarly, since $\mathcal{M}_2$ is used as the new way of mapping the information from $\mathbf{S}'$, we require that $\mathcal{M}_2$ covers exactly the same target information as $\mathcal{M}$. Thus, we impose the following condition on $\mathcal{M}_1$ and $\mathcal{M}_2$:

(E2) $\mathcal{M}_1 \equiv_{\mathrm{s}} \mathcal{M}$ and $\mathcal{M}_2 \equiv_{\mathrm{T}} \mathcal{M}$.

To complete the description of the extract operator, we only need a condition that captures the *optimality* of the new source schema. To do this, we do not impose an explicit condition on this schema, but instead we impose conditions over the range of $\mathcal{M}_1$ and the domain of $\mathcal{M}_2$. Notice that, although we require $\mathcal{M}_1$ to transfer exactly the same source information as $\mathcal{M}$, this mapping can be *redundant* and store the data in $\mathbf{S}'$ in a suboptimal way. Thus, we require $\mathcal{M}_1$ to be target non-redundant, as well as $\mathcal{M}_2$ to be source non-redundant. In that way, we force $\mathrm{range}(\mathcal{M}_1)$ and $\mathrm{dom}(\mathcal{M}_2)$ to be *minimal*, since one cannot lose an instance from $\mathrm{range}(\mathcal{M}_1)$ or $\mathrm{dom}(\mathcal{M}_2)$, and still obtain mappings that fulfill conditions (E1) and (E2). Thus, our last condition is:

(E3) $\mathcal{M}_1$ is target non-redundant and $\mathcal{M}_2$ is source non-redundant.

Notice that it is not difficult to show that under the above conditions, it holds that $\mathrm{range}(\mathcal{M}_1) = \mathrm{dom}(\mathcal{M}_2)$.

We finally have all the necessary ingredients to define the semantics of the extract operator.

Definition 7.2 (Extract operator) $(\mathcal{M}_1, \mathcal{M}_2)$ *is an* extract *of* $\mathcal{M}$ *if* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *satisfy conditions (E1), (E2), and (E3).*

Example 7.3. Consider schemas $\mathbf{S}$, $\mathbf{S}'$, $\mathbf{T}$, and mapping $\mathcal{M}$ from Example 7.1. Let Σ_1 be the set that consists of dependencies:

$$\begin{aligned} P(x, y) &\rightarrow P_1(x), \\ P(x, y) \wedge R(y, z) &\rightarrow P_2(x, y), \end{aligned}$$

Σ_2 the set that consists of:

$$\begin{aligned} P_1(x) &\rightarrow \exists u\, T(x, u) \wedge U(x, x), \\ P_2(x, y) &\rightarrow \exists v\, V(x, y, v), \end{aligned}$$

and $\Gamma_{\mathbf{S}'}$ the set containing the inclusion dependency over $\mathbf{S}'$:

$$P_2(x, y) \rightarrow P_1(x).$$

Consider now the mappings $\mathcal{M}_1 = (\mathbf{S}, \mathbf{S}', \Sigma_1 \cup \Gamma_{\mathbf{S}'})$, and $\mathcal{M}_2 = (\mathbf{S}', \mathbf{T}, \Sigma_2 \cup \Gamma_{\mathbf{S}'})$. Then it can be shown that $(\mathcal{M}_1, \mathcal{M}_2)$ is an extract of $\mathcal{M}$. □

7.1 Computing the extract operator

Two fundamental questions about any metadata management operator are for which classes of mappings is the operator defined, and how can it be computed. In this section, we provide answers to both questions for the class of mappings specified by FO-TO-CQ dependencies, as we provide an algorithm that, given a mapping $\mathcal{M}$ specified by a set of FO-TO-CQ dependencies, computes an extract $(\mathcal{M}_1, \mathcal{M}_2)$ of $\mathcal{M}$.

To present our algorithm, we need to introduce some terminology. In what follows, we use a procedure COMPOSE that given pairwise disjoint schemas $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$, a set Σ_1 of dependencies from $\mathbf{S}_1$ to $\mathbf{S}_2$ and a set Σ_2 of dependencies from $\mathbf{S}_2$ to $\mathbf{S}_3$, computes a set Σ of dependencies from $\mathbf{S}_1$ to $\mathbf{S}_3$ such that $(I, J) \models \Sigma$ if and only if there exists K such that $(I, K) \models \Sigma_1$ and $(K, J) \models \Sigma_2$. That is, COMPOSE(Σ_1, Σ_2) returns a set of dependencies Σ specifying a mapping that represents the composition of the mappings specified by Σ_1 and Σ_2. As pointed out in [24, 23], there exists a straightforward implementation of COMPOSE when Σ_1 and Σ_2 are sets of FO sentences; if $\Sigma_1 = \{\sigma_1, \ldots, \sigma_n\}$, $\Sigma_2 = \{\gamma_1, \ldots, \gamma_m\}$ are set of FO-sentences, and $\mathbf{S}_2 = \{S_1, \ldots, S_k\}$, then a set Σ consisting of second-order dependency $\exists S_1 \cdots \exists S_k\, (\sigma_1 \wedge \cdots \wedge \sigma_n \wedge \gamma_1 \wedge \cdots \wedge \gamma_m)$ satisfies the above condition.

It should be noticed that second-order quantification is unavoidable to express the composition of mappings specified by FO dependencies, even for the case of st-tgds [14]. In what follows, we use COMPOSE as a black box, which could have been implemented by considering the idea shown above and the techniques presented in [14, 24, 23]. In particular, we use COMPOSE in Step 2 of the following algorithm to create constraints that eliminate the redundancy of mappings

Algorithm EXTRACT($\mathcal{M}$)

Input:	st-mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a set of FO-TO-CQ dependencies.
Output:	An extract $(\mathcal{M}_1, \mathcal{M}_2)$ of $\mathcal{M}$.

1. Construct sets Σ_1, Σ_2 of dependencies, and a ground schema $\mathbf{R}$ as follows. For every $\varphi(\bar{x}) \rightarrow \psi(\bar{x}) \in \Sigma$, where $\bar{x}$ is an n-ary tuple of variables, do the following.
   (a) Include a fresh n-ary relational symbol R into $\mathbf{R}$.
   (b) Let $\alpha(\bar{x})$ be a formula in FO that is the output of SOURCEREWRITING$(\mathcal{M}, \psi(\bar{x}))$.
   (c) Include dependency $\alpha(\bar{x}) \rightarrow R(\bar{x})$ into Σ_1 and dependency $R(\bar{x}) \rightarrow \psi(\bar{x})$ into Σ_2.
2. Construct a set of formulas $\Gamma_{\mathbf{R}}$ over $\mathbf{R}$ as follows.
   (a) Let $\widehat{\mathbf{R}} = \{\widehat{R} \mid R \in \mathbf{R}\}$ and Σ_1^- be the set of dependencies $\{\widehat{R}(\bar{x}) \rightarrow \beta(\bar{x}) \mid \beta(\bar{x}) \rightarrow R(\bar{x}) \in \Sigma_1\}$.
   (b) Let Σ' be an SO-formula over $\mathbf{R} \cup \widehat{\mathbf{R}}$ that is the output of COMPOSE(Σ_1^-, Σ_1).
   (c) Let $\Gamma_{\mathbf{R}}$ be the set of formulas over $\mathbf{R}$ obtained from Σ' by replacing every symbol $\widehat{R} \in \widehat{\mathbf{R}}$ by R.
3. Let $\mathcal{M}_1 = (\mathbf{S}, \mathbf{R}, \Sigma_1 \cup \Gamma_{\mathbf{R}})$ and $\mathcal{M}_2 = (\mathbf{R}, \mathbf{T}, \Sigma_2 \cup \Gamma_{\mathbf{R}})$. Return $(\mathcal{M}_1, \mathcal{M}_2)$.

□

Theorem 7.4 EXTRACT$(\mathcal{M})$ *returns an extract of* $\mathcal{M}$.

We conclude this section by investigating what is the language needed to express the extract operator. First, it should be noticed that algorithm EXTRACT uses target dependencies in its output. As our first result, we show that the use of these dependencies is unavoidable.

Proposition 7.5 *There exists an st-mapping* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$*, where* Σ *is a set of* CQ-TO-CQ *dependencies, that does not admit an extract* $(\mathcal{M}_1, \mathcal{M}_2)$ *in which* $\mathcal{M}_1 = (\mathbf{S}, \mathbf{R}, \Sigma_1)$ *and* Σ_1 *is a set of* FO-TO-CQ *dependencies from* $\mathbf{S}$ *to* $\mathbf{R}$*.*

Second, it should also be noticed that algorithm EXTRACT uses SO dependencies (Step 2) to eliminate redundancy. Thus, it is natural to ask whether SO is unavoidable in this algorithm and, in particular, whether FO would not be enough. Although this remains as an open problem, we conjecture that FO is not expressive enough to be used to eliminate redundancy in this algorithm, as there is some evidence in this direction. In particular, the following result shows that if one tries to eliminate target redundancy from a mapping by only forcing some target constraints, then one needs to use a dependency language more expressive than FO (notice that this can be done with SO dependencies, as shown in algorithm EXTRACT).

Proposition 7.6 *There exists a mapping* $\mathcal{M} = (\mathbf{S}, \mathbf{R}, \Sigma)$*, where* $\mathbf{S}$*,* $\mathbf{R}$ *are ground schemas and* Σ *is a set of full* CQ-TO-CQ *dependencies, for which there is no* $\mathcal{M}' = (\mathbf{S}, \mathbf{R}, \Sigma \cup \Gamma)$ *such that* Γ *is a set of* FO*-sentences over* $\mathbf{R}$*,* $\mathcal{M}'$ *is target non-redundant and* $\mathcal{M}' \equiv_s \mathcal{M}$*.*

7.2 On the semantics of the extract operator

The extract operator was considered by Melnik et al. in [21, 22, 23]. Given a mapping $\mathcal{M}$ from a schema $\mathbf{S}$ to a schema $\mathbf{T}$, the output of this operator according to Melnik et al. is a mapping $\mathcal{M}_1$ from $\mathbf{S}$ to a schema $\mathbf{S}'$ together with the schema constraints $\Gamma_{\mathbf{S}'}$ that $\mathbf{S}'$ should satisfy. Moreover, the following two conditions should be satisfied by $\mathcal{M}_1$ according to [22, 23]: (1) $\mathcal{M}_1 \circ (\mathcal{M}_1)^{-1} \circ \mathcal{M}$ is equal to $\mathcal{M}$, and (2) $\mathrm{range}(\mathcal{M}_1)$ is the set of instances of $\mathbf{S}'$ that satisfy $\Gamma_{\mathbf{S}'}$.

Although our semantics for the extract operator was inspired by the work of Melnik et al. [22, 23], there are two features of Melnik et al.'s definition that limit its applicability, in particular if more expressive languages are used to specify mappings. First, if mapping $\mathcal{M}_1$ above is specified by a set of st-tgds (or, in general, by a set of FO-TO-CQ dependencies), then $\mathcal{M}_1 \circ (\mathcal{M}_1)^{-1}$ is a trivial mapping that contains all the pairs of instances from $\mathbf{S}$. Thus, $\mathcal{M}_1 \circ (\mathcal{M}_1)^{-1} \circ \mathcal{M}$ is also a trivial mapping in this case and, therefore, $\mathcal{M}_1 \circ (\mathcal{M}_1)^{-1} \circ \mathcal{M} = \mathcal{M}$ does not hold in general. This rules out the possibility of having natural solutions for the extract operator specified by st-tgds, as the one shown in Example 7.3. Second, in the semantics proposed in [23], no minimality restriction is imposed on the generated schema $\mathbf{S}'$, thus allowing redundant information. Moreover, in the semantics proposed in [22], a minimality criterion based on *counting* the number of instances of a schema is imposed, which is only meaningful when instances are generated from a finite domain, and thus, not applicable in our context.

In view of the aforementioned limitations of the semantics of the extract operator proposed in [22, 23], we have imposed some new conditions on this operator that try to capture the intuition behind it [22, 23]. In particular, we have used the notions of redundancy proposed in Section 6 to impose a minimality condition over the generated schemas. Moreover, we have imposed some conditions on the information transferred by the generated mappings, that ensure that condition (1) above holds when $(\cdot)^{-1}$ is replaced by the notion of maximum recovery, which has proved to be a more natural notion of inverse when mappings are given by FO-TO-CQ dependencies [3, 2, 1, 16] (see Section 3.1 for the definition of maximum recovery).

Proposition 7.7 *Let* $\mathcal{M}$ *be an st-mapping specified by a set of* FO-TO-CQ *dependencies,* $(\mathcal{M}_1, \mathcal{M}_2)$ *the output of* EXTRACT$(\mathcal{M})$ *and* $\mathcal{M}_1^\star$ *a maximum recovery of* $\mathcal{M}_1$*. Then it holds that* $\mathcal{M}_1 \circ \mathcal{M}_1^\star \circ \mathcal{M} = \mathcal{M}$*.*

8. THE MERGE OPERATOR

Consider two independent database schemas $\mathbf{S}_1$ and $\mathbf{S}_2$ and a mapping $\mathcal{M}$ between them, and assume that both schemas have materialized data that is being queried by several applications. Mapping $\mathcal{M}$ describes how data in these schemas is related, and, thus, the relationship stated by $\mathcal{M}$ leads to some *redundancy of storage*: there are *corresponding pieces of data* stored twice in these schemas. Thus, it is natural to ask whether one can have a single *global schema* $\mathbf{S}$ that simultaneously stores the data of $\mathbf{S}_1$ and $\mathbf{S}_2$, but that is not redundant in the storage of the shared information. This is the intuition behind the *merge operator* [4, 5, 22], and, hence, we say that $\mathbf{S}$ is the result of *merging* $\mathbf{S}_1$ and $\mathbf{S}_2$ with respect to the relationship established by $\mathcal{M}$. A complete solution for the merge operator should also include mappings $\mathcal{M}_1$ and $\mathcal{M}_2$ from $\mathbf{S}$ to $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively, that describe the relationship between the global schema and the initial schemas [22, 23]. These mappings ensure that an application that has used the initial schemas independently, would also be able to obtain the required data from the global schema. A diagram of the complete process is shown in Figure 2.

Example 8.1. Let $\mathbf{S}_1 = \{A(\cdot, \cdot)\}$ and $\mathbf{S}_2 = \{B(\cdot, \cdot)\}$ be ground schemas, and consider a mapping $\mathcal{M}$ given by dependency:

$$A(x, y) \quad \rightarrow \quad B(x, y).$$

This simple mapping states that all the tuples of relation A in $\mathbf{S}_1$ should also be part of relation B in $\mathbf{S}_2$. A natural way to store the information of both $\mathbf{S}_1$ and $\mathbf{S}_2$ in a non-redundant way is to consider a schema $\mathbf{S}$ with one relation $A'(\cdot, \cdot)$ storing all the information in A, and a new relation $D(\cdot, \cdot)$ storing the *difference* between B and A. By the intended meaning of relations A' and D,

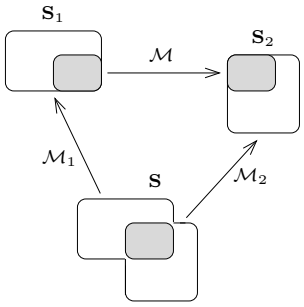

Figure 2: $(\mathcal{M}_1, \mathcal{M}_2)$ is a merge of $\mathcal{M}$.

we know that they should satisfy the denial constraint:

$$\forall x \forall y \neg(A'(x,y) \wedge D(x,y)). \quad (3)$$

In fact, schema $\mathbf{S}$ plus this constraint have enough capacity to store the information of both $\mathbf{S}_1$ and $\mathbf{S}_2$. Moreover, let $\mathcal{M}_1 = (\mathbf{S}, \mathbf{S}_1, \Sigma_1 \cup \Gamma_\mathbf{S})$ and $\mathcal{M}_2 = (\mathbf{S}, \mathbf{S}_2, \Sigma_2 \cup \Gamma_\mathbf{S})$, where Σ_1 consists of dependency:

$$A'(x,y) \quad \rightarrow \quad A(x,y),$$

Σ_2 consists of dependencies:

$$\begin{aligned} A'(x,y) &\rightarrow B(x,y), \\ D(x,y) &\rightarrow B(x,y), \end{aligned}$$

and $\Gamma_\mathbf{S}$ is the set that consists of denial constraint (3). Then $\mathcal{M}_1$ and $\mathcal{M}_2$ can be used to relate the new schema $\mathbf{S}$ with schemas $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively. □

In what follows, we propose a semantics for the merge operator using the machinery developed in the previous sections. As for the case of the extract operator, we formalize the merge considering only the mappings $\mathcal{M}$, $\mathcal{M}_1$ and $\mathcal{M}_2$, as the schemas will be implicit in the mappings.

To define the semantics of the merge operator, we need to introduce the notion of $\mathcal{M}$-*confluence*, which is inspired by the notion of confluence proposed in [22, 23]. Let $\mathbf{S}$, $\mathbf{S}_1$, $\mathbf{S}_2$ be schemas, where $\mathbf{S}_1$ and $\mathbf{S}_2$ have no relation symbols in common, $\mathcal{M}_1$, $\mathcal{M}_2$ mappings from $\mathbf{S}$ to $\mathbf{S}_1$ and from $\mathbf{S}$ to $\mathbf{S}_2$, respectively, and $\mathcal{M}$ a mapping from $\mathbf{S}_1$ to $\mathbf{S}_2$. Then the $\mathcal{M}$-*confluence* of $\mathcal{M}_1$ and $\mathcal{M}_2$, denoted by $\mathcal{M}_1 \oplus_\mathcal{M} \mathcal{M}_2$, is defined as the following mapping from $\mathbf{S}$ to $\mathbf{S}_1 \cup \mathbf{S}_2$:

$$\{(I, J \cup K) \mid (J,K) \in \mathcal{M},\ (I,J) \in \mathcal{M}_1,\ (I,K) \in \mathcal{M}_2\},$$

where $J \cup K$ is the union of instances J and K, that is, $R^{J \cup K} = R^J$ for every $R \in \mathbf{S}_1$, and $S^{J \cup K} = S^K$ for every $S \in \mathbf{S}_2$. Intuitively, $\mathcal{M}_1 \oplus_\mathcal{M} \mathcal{M}_2$ describes the *unified* instances $J \cup K$ that are valid pairs according to $\mathcal{M}$ and also simultaneously mapped from $\mathcal{M}_1$ and $\mathcal{M}_2$.

As pointed out before, $\mathbf{S}$ is a valid global schema if every instance I of $\mathbf{S}$ represents in a non-redundant way a valid unified instance of $\mathbf{S}_1$ and $\mathbf{S}_2$ according to $\mathcal{M}$. Then, if the pair $(\mathcal{M}_1, \mathcal{M}_2)$ of mappings from $\mathbf{S}$ to $\mathbf{S}_1$ and $\mathbf{S}$ to $\mathbf{S}_2$, respectively, is given as a solution for the merge operator, one can formalize this intuition by imposing conditions over $\mathcal{M}_1 \oplus_\mathcal{M} \mathcal{M}_2$, and considering $\mathrm{dom}(\mathcal{M}_1 \oplus_\mathcal{M} \mathcal{M}_2)$ as the new global schema. More precisely, we impose the following conditions:

(M1) $\mathrm{range}(\mathcal{M}_1 \oplus_\mathcal{M} \mathcal{M}_2) = \{J \cup K \mid (J,K) \in \mathcal{M}\}$ and $\mathcal{M}_1 \oplus_\mathcal{M} \mathcal{M}_2$ is target non-redundant.

(M2) $\mathcal{M}_1 \oplus_\mathcal{M} \mathcal{M}_2$ is source non-redundant.

(M3) $\mathrm{dom}(\mathcal{M}_1 \oplus_\mathcal{M} \mathcal{M}_2) = \mathrm{dom}(\mathcal{M}_1) = \mathrm{dom}(\mathcal{M}_2)$.

Condition (M1) indicates that every valid unified instance of $\mathcal{M}$ is covered in an *essential way* by $\mathcal{M}_1 \oplus_\mathcal{M} \mathcal{M}_2$. Condition (M2) specifies that every instance in the global schema is necessary to cover the unified instances of $\mathcal{M}$. Finally, condition (M3) indicates that $\mathcal{M}_1$ and $\mathcal{M}_2$ do not consider instances that are outside the schema defined by $\mathrm{dom}(\mathcal{M}_1 \oplus_\mathcal{M} \mathcal{M}_2)$. We use these conditions to define the merge operator.

Definition 8.2 (Merge operator) $(\mathcal{M}_1, \mathcal{M}_2)$ *is a merge of* $\mathcal{M}$ *if* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *satisfy conditions (M1), (M2) and (M3).*

For instance, $(\mathcal{M}_1, \mathcal{M}_2)$ is a merge of $\mathcal{M}$ in Example 8.1.

The merge operator has been studied in different contexts [7, 25, 22, 23, 26]. Our definition is inspired by the definition of Melnik et al. [22, 23]. However, as for the case of the extract operator, there are some features of their definition that limit its applicability, such as the use of $(\cdot)^{-1}$ (as shown in Section 7.2). This has leaded us to impose some new conditions to define this operator. Moreover, in [25, 26] Pottinger and Bernstein define a semantics for the merge operator that is based on some *preservation of information* and *minimality* conditions. However, whereas this approach is similar to the one taken in this section, the semantics proposed in [26] is specific to a class of mappings that specify the overlap between schemas by means of conjunctive queries.

8.1 Computing the merge operator

In Theorem 4.2.4 in [22], Melnik proposes a straightforward algorithm for the computation of the merge operator, which can also be used in our context to compute this operator for mappings specified by FO-TO-CQ dependencies. However, motivated by Example 8.1, we develop here an algorithm for the case of mappings given by full FO-TO-CQ dependencies, that outputs a merge that makes use of smaller instances in the global schema.

Algorithm MERGE($\mathcal{M}$)

Input: Mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma)$, where $\mathbf{S}_1, \mathbf{S}_2$ are disjoint ground schemas and Σ is a set of full FO-TO-CQ dependencies.

Output: A merge $(\mathcal{M}_1, \mathcal{M}_2)$ of $\mathcal{M}$.

1. Let $\mathbf{S} = \{\widehat{P} \mid P \in \mathbf{S}_1\} \cup \{D_R \mid R \in \mathbf{S}_2\}$ be a ground schema.
2. Construct a set Σ_1 as follows. For every n-ary relation symbol P in $\mathbf{S}_1$, include dependency $\widehat{P}(\bar{x}) \rightarrow P(\bar{x})$ into Σ_1, with $\bar{x}$ an n-ary tuple of distinct variables.
3. Construct sets Σ_2 and $\Gamma_\mathbf{S}$ as follows. For every n-ary relation symbol R in $\mathbf{S}_2$ do the following. Let $\bar{x}$ be an n-ary tuple of distinct variables.
   (a) Include $D_R(\bar{x}) \rightarrow R(\bar{x})$ into Σ_2.
   (b) Let $\alpha(\bar{x})$ the output of SOURCEREWRITING($\mathcal{M}, R(\bar{x})$), and $\widehat{\alpha}(\bar{x})$ the formula obtained from $\alpha(\bar{x})$ by replacing every relation symbol $P \in \mathbf{S}_1$ by $\widehat{P}$.
   (c) Include $\widehat{\alpha}(\bar{x}) \rightarrow R(\bar{x})$ into Σ_2 and $\forall \bar{x} \neg(\widehat{\alpha}(\bar{x}) \wedge D_R(\bar{x}))$ into $\Gamma_\mathbf{S}$.
4. Let $\mathcal{M}_1 = (\mathbf{S}, \mathbf{S}_1, \Sigma_1 \cup \Gamma_\mathbf{S})$ and $\mathcal{M}_2 = (\mathbf{S}, \mathbf{S}_2, \Sigma_2 \cup \Gamma_\mathbf{S})$. Return $(\mathcal{M}_1, \mathcal{M}_2)$.

□

Theorem 8.3 MERGE($\mathcal{M}$) *returns a merge of* $\mathcal{M}$.

9. CONCLUDING REMARKS

We provide foundations for metadata management by developing a theory to compare schema mappings in terms of notions of information and redundancy. We proved the usefulness of our proposal by presenting applications in the schema evolution problem, and the definition and computation of the inverse, extract and merge operators. Although we have focused in the relational case, our theory is bounded neither to a particular data model nor to a specific language for expressing mappings, thus providing general foundations for schema mapping management.

Acknowledgments

We thank the anonymous referees and Ron Fagin for many helpful comments. Arenas was supported by FONDECYT grant 1090565, Pérez by CONICYT Ph.D. Scholarship, Reutter by EPSRC grant G049165 and FET-Open project FoX, and Riveros by EPSRC EP/G004021/1.

10. REFERENCES

[1] M. Arenas, J. Pérez, J. Reutter, and C. Riveros. Composition and Inversion of Schema Mappings. *SIGMOD Record*, 38(3):17-28, 2009.

[2] M. Arenas, J. Pérez, J. Reutter, and C. Riveros. Inverting schema mappings: bridging the gap between theory and practice. In *VLDB*, pages 1018–1029, 2009.

[3] M. Arenas, J. Pérez, and C. Riveros. The recovery of a schema mapping: bringing exchanged data back. In *TODS*, 34(4), 2009.

[4] P. Bernstein, A. Halevy and R. Pottinger. A Vision of Management of Complex Models. *SIGMOD Record*, 29(4):55-63, 2000.

[5] P. Bernstein. Applying Model Management to Classical Meta Data Problems. In *CIDR*, 2003.

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

[7] P. Buneman, S. B. Davidson and A. Kosky. Theoretical aspects of schema merging. In *EDBT*, pages 152–167, 1992.

[8] O. Duschka and M. Genesereth. Answering recursive queries using views. In *PODS*, pages 109–116, 1997.

[9] B. ten Cate, P. Kolaitis. Structural Characterizations of Schema-Mapping Languages. In *ICDT*, pages 63–72, 2009.

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

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

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

[13] R. Fagin, P. G. Kolaitis and L. Popa. Data exchange: getting to the core. *TODS*,. 30(1):174–210, 2005.

[14] 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.

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

[16] R. Fagin, P. Kolaitis, L. Popa, and W.-C. Tan. Reverse data exchange: coping with nulls. In *PODS*, pages 23–32, 2009.

[17] R. Fagin, A. Nash. The structure of inverses in schema mappings. IBM Research Report RJ10425, version 4, April 2008.

[18] A. Y. Halevy. Answering queries using views: A survey. VLDB J., 10(4):270–294, 2001.

[19] P. G. Kolaitis. Schema mappings, data exchange, and metadata management. In *PODS*, pages 61–75, 2005.

[20] A. Y. Levy, A. O. Mendelzon, Y. Sagiv and D. Srivastava. Answering queries using views. In *PODS*, pages 95–104, 1995.

[21] S. Melnik, E. Rahm, and P. Bernstein. Rondo: A Programming Platform for Generic Model Management. In *SIGMOD*, pages 193–204, 2003.

[22] S. Melnik. Generic model management: concepts and algorithms. Volume 2967 of *LNCS*, Springer, 2004.

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

[24] A. Nash, P. Bernstein, S. Melnik. Composition of mappings given by embedded dependencies. In *PODS* pages 172–183, 2005.

[25] R. Pottinger and P. A. Bernstein. Merging models based on given correspondences. In *VLDB*, pages 826–873, 2003.

[26] R. Pottinger and P. A. Bernstein. Schema merging and mapping creation for relational sources. In *EDBT*, pages 73–84, 2008.

[27] R. Pottinger and A. Y. Halevy. MiniCon: A scalable algorithm for answering queries using views. VLDB J., 10(2-3):182–198, 2001.