Composition and Inversion of Schema Mappings[*]

Marcelo Arenas
PUC Chile
marenas@ing.puc.cl

Jorge Pérez
PUC Chile
jperez@ing.puc.cl

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

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

1 Introduction

A schema mapping is a specification that describes how data from a source schema is to be mapped to a target schema. Schema mappings have proved to be essential for data-interoperability tasks such as data exchange and data integration. The research on this area has mainly focused on performing these tasks. However, as Bernstein pointed out [7], many information-system problems involve not only the design and integration of complex application artifacts, but also their subsequent manipulation. Driven by this consideration, Bernstein proposed in [7] a general framework for managing schema mappings. In this framework, mappings are usually specified in a logical language, and high-level algebraic operators are used to manipulate them [7, 16, 33, 12, 8].

Two of the most fundamental operators in this framework are the *composition* and *inversion* of schema mappings. Intuitively, the composition can be described as follows. Given a mapping $\mathcal{M}_1$ from a schema $\mathbf{A}$ to a schema $\mathbf{B}$, and a mapping $\mathcal{M}_2$ from $\mathbf{B}$ to a schema $\mathbf{E}$, *the composition* of $\mathcal{M}_1$ and $\mathcal{M}_2$ is a new mapping that describes the relationship between schemas $\mathbf{A}$ and $\mathbf{E}$. This new mapping must be *semantically consistent* with the relationships previously established by $\mathcal{M}_1$ and $\mathcal{M}_2$. On the other hand, *an inverse* of $\mathcal{M}_1$ is a new mapping that describes the *reverse* relationship from $\mathbf{B}$ to $\mathbf{A}$, and is semantically consistent with $\mathcal{M}_1$.

In practical scenarios, the composition and inversion of schema mappings can have several applications. In a data exchange context [13], if a mapping $\mathcal{M}$ is used to exchange data from a source to a target schema, an inverse of $\mathcal{M}$ can be used to exchange the data back to the source, thus *reversing* the application of $\mathcal{M}$. As a second application, consider a peer-data management system (PDMS) [10, 24]. In a PDMS, a peer can act as a data source, a mediator, or both, and the system relates peers by establishing *directional* mappings between the peers schemas. Given a query formulated on a particular peer, the PDMS must proceed to retrieve the answers by reformulating the query using its complex net of semantic mappings. Performing this reformulation at query time may be quite expensive. The composition operator can be used to essentially combine sequences of mappings into a single mapping that can be precomputed and optimized for query answering purposes. Another application is schema evolution, where the inverse together with the composition play a crucial role [8]. Consider a mapping $\mathcal{M}$ between schemas $\mathbf{A}$ and $\mathbf{B}$, and assume that schema $\mathbf{A}$ evolves into a schema $\mathbf{A}'$. This evolution can be expressed as a mapping $\mathcal{M}'$ between $\mathbf{A}$ and $\mathbf{A}'$. Thus, the relationship between the new schema $\mathbf{A}'$ and schema $\mathbf{B}$ can be obtained by inverting mapping $\mathcal{M}'$ and then composing the result with mapping $\mathcal{M}$.

In the recent years, a lot of attention has been paid to the development of solid foundations for the composition [32, 16, 36] and inversion [12, 19, 4, 3] of schema mappings. In this paper, we review the proposals for the semantics of these crucial operators. For each of these proposals, we concentrate on the three following problems: the definition of the semantics of the operator, the language needed to express the operator, and the algorithmic issues associated to the problem of computing the operator. It should be pointed out that we primarily consider the formalization of schema mappings introduced in the work on data exchange [13]. In particular, when studying the problem of computing the composition and inverse of a schema mapping, we will be mostly interested in computing these operators for mappings specified by *source-to-target tuple-generating dependencies* [13]. Although there has been an important amount of work about different *flavors* of composition and inversion motivated by practical applications [9, 34, 38], we focus on the most theoretically-oriented results [32, 16, 12, 19, 4, 3].

Organization of the paper. We begin in Section 2 with the terminology that will be used in the paper. We then continue in Section 3 reviewing the main results for the

[*] **Database Principles Column.** Column editor: Leonid Libkin, School of Informatics, University of Edinburgh, Edinburgh, EH8 9AB, UK. E-mail: libkin@inf.ed.ac.uk.

composition operator proposed in [16]. Section 4 contains a detailed study of the inverse operators proposed in [12, 19, 4]. In Section 5, we review a relaxed approach to define the semantics for the inverse and composition operators that parameterizes these notions by a query-language [32, 3]. Finally, some future work is pointed out in Section 6. Due to the lack of space, the proofs of the new results presented in this survey are given in the extended version of this paper, which can be downloaded from `http://arxiv.org/`.

2 Basic notation

In this paper, we assume that data is represented in the relational model. A *relational schema* $\mathbf{R}$, or just *schema*, is a finite set $\{R_1, \ldots, R_n\}$ of relation symbols, with each R_i having a fixed arity n_i. 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. The *domain* of an instance I, denoted by $\mathrm{dom}(I)$, is the set of all elements that occur in any of the relations R_i^I. In addition, $\mathrm{Inst}(\mathbf{R})$ is defined to be the set of all instances of $\mathbf{R}$.

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

Schema mappings and solutions. Schema mappings are used to define a semantic relationship between two schemas. In this paper, we use a general representation of mappings; given two schemas $\mathbf{R}_1$ and $\mathbf{R}_2$, a mapping $\mathcal{M}$ from $\mathbf{R}_1$ to $\mathbf{R}_2$ is a set of pairs (I, J), where I is an instance of $\mathbf{R}_1$, and J is an instance of $\mathbf{R}_2$. Further, we say that J is a *solution for* I *under* $\mathcal{M}$ if $(I, J) \in \mathcal{M}$. The set of solutions for I under $\mathcal{M}$ is denoted by $\mathrm{Sol}_{\mathcal{M}}(I)$. The domain of $\mathcal{M}$, denoted by $\mathrm{dom}(\mathcal{M})$, is defined as the set of instances I such that $\mathrm{Sol}_{\mathcal{M}}(I) \neq \emptyset$.

Dependencies. As usual, we use a class of dependencies to specify schema mappings [13]. 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 $\mathbf{R}_1 \cup \mathbf{R}_2$ is an $\mathcal{L}_1$-TO-$\mathcal{L}_2$ *dependency from* $\mathbf{R}_1$ *to* $\mathbf{R}_2$ if Φ is of the form $\forall \bar{x}\, (\varphi(\bar{x}) \rightarrow \psi(\bar{x}))$, where (1) $\bar{x}$ is the tuple of free variables in both $\varphi(\bar{x})$ and $\psi(\bar{x})$; (2) $\varphi(\bar{x})$ is an $\mathcal{L}_1$-formula over $\mathbf{R}_1$; and (3) $\psi(\bar{x})$ is an $\mathcal{L}_2$-formula over $\mathbf{R}_2$. Furthermore, we usually omit the outermost universal quantifiers from $\mathcal{L}_1$-TO-$\mathcal{L}_2$ dependencies and, thus, we write $\varphi(\bar{x}) \rightarrow \psi(\bar{x})$ instead of $\forall \bar{x}\, (\varphi(\bar{x}) \rightarrow \psi(\bar{x}))$. Finally, the semantics of an $\mathcal{L}_1$-TO-$\mathcal{L}_2$ dependency is defined as usual (e.g., see [13, 4]).

If $\mathbf{S}$ is a source schema and $\mathbf{T}$ is a target schema, an $\mathcal{L}_1$-TO-$\mathcal{L}_2$ dependency from $\mathbf{S}$ to $\mathbf{T}$ is called an $\mathcal{L}_1$-TO-$\mathcal{L}_2$ *source-to-target dependency* ($\mathcal{L}_1$-TO-$\mathcal{L}_2$ st-dependency), and an $\mathcal{L}_1$-TO-$\mathcal{L}_2$ dependency from $\mathbf{T}$ to $\mathbf{S}$ is called an $\mathcal{L}_1$-TO-$\mathcal{L}_2$ *target-to-source dependency* ($\mathcal{L}_1$-TO-$\mathcal{L}_2$ ts-dependency). Notice that the fundamental class of source-to-target tuple-generating dependencies (st-tgds) [13] corresponds to the class of CQ-TO-CQ st-dependencies.

When considering a mapping specified by a set of dependencies, we use the usual semantics given by logical satisfaction. That is, if $\mathcal{M}$ is a mapping from $\mathbf{R}_1$ to $\mathbf{R}_2$ specified by a set Σ of $\mathcal{L}_1$-TO-$\mathcal{L}_2$ dependencies, we have that $(I, J) \in \mathcal{M}$ if and only if $I \in \mathrm{Inst}(\mathbf{R}_1)$, $J \in \mathrm{Inst}(\mathbf{R}_2)$, and (I, J) satisfies Σ.

Query Answering. In this paper, we use CQ to denote the class of conjunctive queries and UCQ to denote the class of unions of conjunctive queries. Given a query Q and a database instance I, we denote by $Q(I)$ the evaluation of Q over I. Moreover, we use predicate $\mathbf{C}(\cdot)$ to differentiate between constants and nulls, that is, $\mathbf{C}(a)$ holds if and only if a is a constant value. We use $=$, $\neq$, and $\mathbf{C}$ as superscripts to denote a class of queries enriched with equalities, inequalities, and predicate $\mathbf{C}(\cdot)$, respectively. Thus, for example, UCQ$^{=,\mathbf{C}}$ is the class of unions of conjunctive queries with equalities and predicate $\mathbf{C}(\cdot)$.

As usual, the semantics of queries in the presence of schema mappings is defined in terms of the notion of *certain answer*. Assume that $\mathcal{M}$ is a mapping from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$. Then given an instance I of $\mathbf{R}_1$ and a query Q over $\mathbf{R}_2$, the *certain answers of* Q *for* I *under* $\mathcal{M}$, denoted by $\underline{\mathrm{certain}}_{\mathcal{M}}(Q, I)$, is the set of tuples that belong to the evaluation of Q over every possible solution for I under $\mathcal{M}$, that is, $\bigcap\{Q(J) \mid J$ is a solution for I under $\mathcal{M}\}$.

3 Composition of Schema Mappings

The composition operator has been identified as one of the fundamental operators for the development of a framework for managing schema mappings [7, 33, 35]. The goal of this operator is to generate a mapping $\mathcal{M}_{13}$ that has the same effect as applying successively two given mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, provided that the target schema of $\mathcal{M}_{12}$ is the same as the source schema of $\mathcal{M}_{23}$. In [16], Fagin et al. study the composition for the widely used class of st-tgds. In particular, they provide solutions to the three fundamental problems for mapping operators considered in this paper, that is, they provide a formal semantics for the composition operator, they identify a mapping language that is appropriate for expressing this operator, and they study the complexity of composing schema mappings. In this section, we present these solutions.

In [16, 33], the authors propose a semantics for the composition operator that is based on the semantics of this operator for binary relations:

Definition 3.1 ([16, 33]) *Let $\mathcal{M}_{12}$ be a mapping from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$, and $\mathcal{M}_{23}$ a mapping from $\mathbf{R}_2$ to a schema $\mathbf{R}_3$. Then the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is defined as $\mathcal{M}_{12} \circ \mathcal{M}_{23} = \{(I_1, I_3) \mid \exists I_2 : (I_1, I_2) \in \mathcal{M}_{12}$ and $(I_2, I_3) \in \mathcal{M}_{23}\}$.*

Then Fagin et al. consider in [16] the natural question of whether the composition of two mappings specified by st-tgds can also be specified by a set of these dependencies. Unfortunately, they prove in [16] that this is not the case, as shown in the following example.

Example 3.2. (from [16]) Consider a schema $\mathbf{R}_1$ consisting of one binary relation Takes, that associates a student name with a course she/he is taking, a schema $\mathbf{R}_2$ consisting of a relation Takes$_1$, that is intended to be a copy of Takes, and of an additional relation symbol Student, that associates a student with a student id; and a schema $\mathbf{R}_3$ consisting of a binary relation symbol Enrollment, that associates a student id with the courses this student is taking. Consider now mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ specified by the following sets of st-tgds:

$$\begin{aligned}\Sigma_{12} &= \{\texttt{Takes}(n,c) \to \texttt{Takes}_1(n,c),\\ &\qquad \texttt{Takes}(n,c) \to \exists s\,\texttt{Student}(n,s)\},\\ \Sigma_{23} &= \{\texttt{Student}(n,s) \wedge \texttt{Takes}_1(n,c) \ \to\\ &\qquad\qquad \texttt{Enrollment}(s,c)\}.\end{aligned}$$

Mapping $\mathcal{M}_{12}$ requires that a copy of every tuple in Takes must exist in Takes$_1$ and, moreover, that each student name n must be associated with some student id s in the relation Student. Mapping $\mathcal{M}_{23}$ requires that if a student with name n and id s takes a course c, then (s, c) is a tuple in the relation Enrollment. Intuitively, in the composition mapping one would like to replace the name n of a student by a student id i_n, and then for each course c that is taken by n, one would like to include the tuple (i_n, c) in the table Enrollment. Unfortunately, as shown in [16], it is not possible to express this relationship by using a set of st-tgds. In particular, a st-tgd of the form:

$$\texttt{Takes}(n,c) \quad\to\quad \exists y\,\texttt{Enrollment}(y,c) \qquad (1)$$

does not express the desired relationship, as it may associate a distinct student id y for each tuple (n, c) in Takes and, thus, it may create several identifiers for the same student name. □

The previous example shows that in order to express the composition of mappings specified by st-tgds, one has to use a language more expressive than st-tgds. However, the example gives little information about what the right language for composition is. In fact, the composition of mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ in this example can be defined in first-order logic (FO):

$$\forall n \exists y \forall c\,(\texttt{Takes}(n,c) \to \texttt{Enrollment}(y,c)),$$

which may lead to the conclusion that FO is a good alternative to define the composition of mappings specified by st-tgds. However, a complexity argument shows that this conclusion is wrong. More specifically, given mappings $\mathcal{M}_{12} = (\mathbf{R}_1, \mathbf{R}_2, \Sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{R}_2, \mathbf{R}_3, \Sigma_{23})$, where Σ_{12} and Σ_{23} are sets of st-tgds, define the *composition problem for $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$*, denoted by COMPOSITION$(\mathcal{M}_{12}, \mathcal{M}_{23})$, as the problem of verifying, given $I_1 \in \text{Inst}(\mathbf{R}_1)$ and $I_3 \in \text{Inst}(\mathbf{R}_3)$, whether $(I_1, I_3) \in \mathcal{M}_{12} \circ \mathcal{M}_{23}$. If the composition of $\mathcal{M}_{12}$ with $\mathcal{M}_{23}$ is defined by a set Σ of formulas in some logic, then COMPOSITION$(\mathcal{M}_{12}, \mathcal{M}_{23})$ is reduced to the problem of verifying whether a pair of instances (I_1, I_3) satisfies Σ. In particular, if Σ is a set of FO formulas, then the complexity of COMPOSITION$(\mathcal{M}_{12}, \mathcal{M}_{23})$ is in LOGSPACE, as the complexity of verifying whether a fixed set of FO formulas is satisfied by an instance is in LOGSPACE [39]. Thus, if for some mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, the complexity of the composition problem is higher than LOGSPACE, one can conclude that FO is not capable of expressing the composition. In fact, this higher complexity is proved in [16].

Theorem 3.3 ([16]) *For every pair of mappings $\mathcal{M}_{12}$, $\mathcal{M}_{23}$ specified by st-tgds,* COMPOSITION$(\mathcal{M}_{12}, \mathcal{M}_{23})$ *is in NP. Moreover, there exist mappings $\mathcal{M}^\star_{12}$ and $\mathcal{M}^\star_{23}$ specified by st-tgds such that* COMPOSITION$(\mathcal{M}^\star_{12}, \mathcal{M}^\star_{23})$ *is NP-complete.*

Theorem 3.3 not only shows that FO is not the right language to express the composition of mappings given by st-tgds, but also gives a good insight on what needs to be added to st-tgds to obtain a language closed under composition. Given that COMPOSITION$(\mathcal{M}_{12}, \mathcal{M}_{23})$ is in NP, we know by Fagin's Theorem that the composition can be defined by an existential second-order logic formula [26]. In fact, Fagin et al. use this property in [16] to obtain the right language for composition. More specifically, Fagin et al. extend st-tgds with existential second-order quantification, which gives rise to the class of SO-tgds [16]. Formally, given schemas $\mathbf{R}_1$ and $\mathbf{R}_2$ with no relation symbols in common, a *second-order tuple-generating dependency from* $\mathbf{R}_1$ *to* $\mathbf{R}_2$ (SO-tgd) is a formula of the form $\exists \bar{f}\,(\forall \bar{x}_1(\varphi_1 \rightarrow \psi_1) \wedge \cdots \wedge \forall \bar{x}_n(\varphi_n \rightarrow \psi_n))$, where (1) each member of $\bar{f}$ is a function symbol, (2) each formula φ_i ($1 \leq i \leq n$) is a conjunction of relational atoms of the form $S(y_1, \ldots, y_k)$ and equality atoms of the form $t = t'$, where S is a k-ary relation symbol of $\mathbf{R}_1$ and y_1, …, y_k are (not necessarily distinct) variables in $\bar{x}_i$, and t, t' are terms built from $\bar{x}_i$ and $\bar{f}$, (3) each formula ψ_i ($1 \leq i \leq n$) is a conjunction of relational atomic formulas over $\mathbf{R}_2$ mentioning terms built from $\bar{x}_i$ and $\bar{f}$, and (4) each variable in $\bar{x}_i$ ($1 \leq i \leq n$) appears in some relational atom of φ_i.

In [16], Fagin et al. show that SO-tgds are the right dependencies for expressing the composition of mappings given by st-tgds. First, it is not difficult to see that every set of st-tgds can be transformed into an SO-tgd. For example, set Σ_{12} from Example 3.2 is equivalent to the following SO-tgd:

$$\exists f\Big(\forall n \forall c\,(\texttt{Takes}(n,c) \rightarrow \texttt{Takes}_1(n,c)) \wedge \\ \forall n \forall c\,(\texttt{Takes}(n,c) \rightarrow \texttt{Student}(n, f(n,c)))\Big).$$

Second, Fagin et al. show that SO-tgds are closed under composition.

Theorem 3.4 ([16]) *Let $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ be mappings specified by SO-tgds. Then the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ can also be specified by an SO-tgd.*

It should be noticed that the previous theorem can also be applied to mappings that are specified by finite sets of SO-tgds, as these dependencies are closed under conjunction. Moreover, it is important to notice that Theorem 3.4 implies that the composition of a finite number of mappings specified by st-tgds can be defined by an SO-tgd, as every set of st-tgds can be expressed as an SO-tgd.

Theorem 3.5 ([16]) *The composition of a finite number of mappings, each defined by a finite set of st-tgds, is defined by an SO-tgd.*

Example 3.6. Let $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ be the mappings defined in Example 3.2. The following SO-tgd correctly specifies the composition of these two mappings:

$$\exists g\Big(\forall n \forall c\,(\texttt{Takes}(n,c) \rightarrow \texttt{Enrollment}(g(n), c))\Big).$$

□

Third, Fagin et al. prove in [16] that the converse of Theorem 3.5 also holds, thus showing that SO-tgds are exactly the right language for representing the composition of mappings given by st-tgds.

Theorem 3.7 ([16]) *Every SO-tgd defines the composition of a finite number of mappings, each defined by a finite set of st-tgds.*

Finally, Fagin et al. in [16] also study the complexity of composing schema mappings. More specifically, they provide an exponential-time algorithm that given two mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, each specified by an SO-tgd, returns a mapping $\mathcal{M}_{13}$ specified by an SO-tgd and equivalent to the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$. Furthermore, they show that exponentiality is unavoidable in such an algorithm, as there exist mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, each specified by a finite set of st-tgds, such that every SO-tgd that defines the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is of size exponential in the size of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$.

In [36], Nash et al. also study the composition problem and extend the results of [16]. In particular, they study the composition of mappings given by dependencies that need not be source-to-target, and for all the classes of mappings considered in that paper, they provide an algorithm that attempts to compute the composition and give sufficient conditions that guarantee that the algorithm will succeed.

3.1 Composition under closed world semantics

In [27], Libkin proposes an alternative semantics for schema mappings and, in particular, for data exchange. Roughly speaking, the main idea in [27] is that when exchanging data with a set Σ of st-tgds and a source instance I, one generates a target instance J such that every tuple in J is *justified* by a formula in Σ and a set of tuples from I. A target instance J that satisfies the above property is called a *closed-world solution* for I under Σ [27]. In [28], Libkin and Sirangelo propose the language of CQ-SkSTDs, that slightly extends the syntax of SO-tgds, and study the composition problem under the closed-world semantics for mappings given by sets of CQ-SkSTDs. Due to the lack of space, we do not give here the formal definition of the closed-world semantics, but instead we give an example that shows the intuition behind it (see [28] for a formal definition of the semantics and of CQ-SkSTDs).

Example 3.8. Let σ be the SO-tgd of Example 3.6. Formula σ is also a CQ-SkSTD [28]. Consider now a source instance I such that $\texttt{Takes}^I = \{(\text{Chris}, \text{logic})\}$, and the instances J_1 and J_2 such that:

$$\begin{aligned} \texttt{Enrollment}^{J_1} &= \{(075, \text{logic})\} \\ \texttt{Enrollment}^{J_2} &= \{(075, \text{logic}), (084, \text{algebra})\} \end{aligned}$$

Notice that both (I, J_1) and (I, J_2) satisfy σ (considering an interpretation for function g such that $g(\text{Chris}) = 075$). Thus, under the semantics based on logical satisfaction [16], both J_1 and J_2 are solutions for I. The crucial difference between J_1 and J_2 is that J_2 has an *unjustified* tuple [27]; tuple $(075, \text{logic})$ is *justified* by tuple $(\text{Chris}, \text{logic})$, while $(084, \text{algebra})$ *has no justification*. In fact, J_1 is a closed-world solution for I under σ, but J_2 is not [27, 28]. □

Given a set Σ of CQ-SkSTDs from $\mathbf{R}_1$ to $\mathbf{R}_2$, we say that $\mathcal{M}$ is *specified by* Σ *under the closed-world semantics*, denoted by $\mathcal{M} = \text{cws}(\Sigma, \mathbf{R}_1, \mathbf{R}_2)$, if $\mathcal{M} = \{(I, J) \mid I \in \text{Inst}(\mathbf{R}_1), J \in \text{Inst}(\mathbf{R}_2)$ and J is a closed-world solution for I under $\Sigma\}$. Notice that, as Example 3.8 shows, the mapping specified by a formula (or a set of formulas) under the closed-world semantics is different from the mapping specified by the same formula but under the semantics of [16]. Thus, it is not immediately clear whether a closure property like the one in Theorem 3.4 can be directly translated to the closed-world semantics. In this respect, Libkin and Sirangelo [28] show that the language of CQ-SkSTDs is closed under composition.

Theorem 3.9 ([28]) *Let* $\mathcal{M}_{12} = \text{cws}(\Sigma_{12}, \mathbf{R}_1, \mathbf{R}_2)$ *and* $\mathcal{M}_{23} = \text{cws}(\Sigma_{23}, \mathbf{R}_2, \mathbf{R}_3)$*, where* Σ_{12} *and* Σ_{23} *are sets of* CQ-SkSTDs*. Then there exists a set* Σ_{13} *of* CQ-SkSTDs *such that* $\mathcal{M}_{12} \circ \mathcal{M}_{23} = \text{cws}(\Sigma_{13}, \mathbf{R}_1, \mathbf{R}_3)$.

4 Inversion of Schema Mappings

In the recent years, the problem of inverting schema mappings has received a lot of attention. In particular, the issue of providing a *good* semantics for this operator turned out to be a difficult problem. Three main proposals for inverting mappings have been considered so far in the literature: *Fagin-inverse* [12], *quasi-inverse* [19] and *maximum recovery* [5]. In this section, we present and compare these approaches.

Some of the notions mentioned above are only appropriate for certain classes of mappings. In particular, the following two classes of mappings are used in this section when defining and comparing inverses. A mapping $\mathcal{M}$ from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$ is said to be *total* if $\text{dom}(\mathcal{M}) = \text{Inst}(\mathbf{R}_1)$, and is said to be *closed-down on the left* if whenever $(I, J) \in \mathcal{M}$ and $I' \subseteq I$, it holds that $(I', J) \in \mathcal{M}$.

Furthermore, whenever a mapping is specified by a set of formulas, we consider source instances as just containing constants values, and target instances as containing constants and null values. This is a natural assumption in a data exchange context, since target instances generated as a result of exchanging data may be *incomplete*, thus, null values are used as place-holders for unknown information. In Section 4.3, we consider inverses for alternative semantics of mappings and, in particular, inverses for the *extended semantics* that was proposed in [17] to deal with incomplete information in source instances.

4.1 Fagin-inverse and quasi-inverse

We start by considering the notion of inverse proposed by Fagin in [12], and that we call Fagin-inverse in this paper[1]. Roughly speaking, Fagin's definition is based on the idea that a mapping composed with its inverse should be equal to the identity schema mapping. Thus, given a schema $\mathbf{R}$, Fagin first defines an *identity mapping* $\overline{\text{Id}}$ as $\{(I_1, I_2) \mid I_1, I_2$ are instances of $\mathbf{R}$ and $I_1 \subseteq I_2\}$. Then a mapping $\mathcal{M}'$ is said to be a *Fagin-inverse* of a mapping $\mathcal{M}$ if $\mathcal{M} \circ \mathcal{M}' = \overline{\text{Id}}$. Notice that $\overline{\text{Id}}$ is not the usual identity relation over $\mathbf{R}$. As explained in [12], $\overline{\text{Id}}$ is appropriate as an identity for mappings that are total and closed-down on the left and, in particular, for the class of mappings specified by st-tgds.

Example 4.1. Let $\mathcal{M}$ be a mapping specified by st-tgds $S(x) \rightarrow U(x)$ and $S(x) \rightarrow V(x)$. Intuitively, $\mathcal{M}$ is Fagin-invertible since all the information in the source relation S is transferred to both relations U and V in the target. In fact, the mapping $\mathcal{M}'$ specified by ts-tgd $U(x) \rightarrow S(x)$ is a Fagin-inverse of $\mathcal{M}$ since $\mathcal{M} \circ \mathcal{M}' = \overline{\text{Id}}$. Moreover, the mapping $\mathcal{M}''$ specified by ts-tgd $V(x) \rightarrow S(x)$ is also a Fagin-inverse of $\mathcal{M}$, which shows that there need not be a unique Fagin-inverse. □

A first fundamental question about any notion of inverse is for which class of mappings is guaranteed to exist. The following example from [12] shows that Fagin-inverses are not guaranteed to exist for mappings specified by st-tgds.

Example 4.2. Let $\mathcal{M}$ be a mapping specified by st-tgd $S(x, y) \rightarrow T(x)$. Intuitively, $\mathcal{M}$ has no Fagin-inverse since $\mathcal{M}$ only transfers the information about the first component of S. In fact, it is formally proved in [12] that this mapping is not Fagin-invertible. □

[1] Fagin [12] named his notion just as *inverse* of a schema mapping. Since we are comparing different semantics for the *inverse* operator, we reserve the term *inverse* to refer to this operator in general, and use the name *Fagin-inverse* for the notion proposed in [12].

As pointed out in [19], the notion of Fagin-inverse is rather restrictive as it is rare that a schema mapping possesses a Fagin-inverse. Thus, there is a need for weaker notions of inversion, which is the main motivation for the introduction of the notion of quasi-inverse of a schema mapping in [19].

The idea behind quasi-inverses is to relax the notion of Fagin-inverse by not differentiating between source instances that have the same space of solutions. More precisely, let $\mathcal{M}$ be a mapping from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$. Instances I_1 and I_2 of $\mathbf{R}_1$ are *data-exchange equivalent* w.r.t. $\mathcal{M}$, denoted by $I_1 \sim_{\mathcal{M}} I_2$, if $\mathrm{Sol}_{\mathcal{M}}(I_1) = \mathrm{Sol}_{\mathcal{M}}(I_2)$. For example, for the mapping $\mathcal{M}$ in Example 4.2, we have that $I_1 \sim_{\mathcal{M}} I_2$, with $I_1 = \{S(1,2)\}$ and $I_2 = \{S(1,3)\}$. Then $\mathcal{M}'$ is said to be a quasi-inverse of $\mathcal{M}$ if the property $\mathcal{M} \circ \mathcal{M}' = \overline{\mathrm{Id}}$ holds *modulo* the equivalence relation $\sim_{\mathcal{M}}$. Formally, given a mapping $\mathcal{N}$ from $\mathbf{R}$ to $\mathbf{R}$, mapping $\mathcal{N}[\sim_{\mathcal{M}}, \sim_{\mathcal{M}}]$ is defined as

$$\{(I_1, I_2) \in \mathrm{Inst}(\mathbf{R}) \times \mathrm{Inst}(\mathbf{R}) \mid \text{ exist } I_1', I_2' \text{ with } I_1 \sim_{\mathcal{M}} I_1',\ I_2 \sim_{\mathcal{M}} I_2' \text{ and } (I_1', I_2') \in \mathcal{N}\}$$

Then a mapping $\mathcal{M}'$ is said to be a *quasi-inverse* of a mapping $\mathcal{M}$ if $(\mathcal{M} \circ \mathcal{M}')[\sim_{\mathcal{M}}, \sim_{\mathcal{M}}] = \overline{\mathrm{Id}}[\sim_{\mathcal{M}}, \sim_{\mathcal{M}}]$.

Example 4.3. Let $\mathcal{M}$ be a mapping specified by st-tgd $S(x,y) \rightarrow T(x)$. It was shown in Example 4.2 that $\mathcal{M}$ does not have a Fagin-inverse. However, mapping $\mathcal{M}'$ specified by ts-tgd $T(x) \rightarrow \exists y\, S(x,y)$ is a quasi-inverse of $\mathcal{M}$ [19]. Notice that for the source instance $I_1 = \{S(1,2)\}$, we have that I_1 and $I_2 = \{S(1,3)\}$ are both solutions for I_1 under the composition $\mathcal{M} \circ \mathcal{M}'$. In fact, for every I such that $I \sim_{\mathcal{M}} I_1$, we have that I is a solution for I_1 under $\mathcal{M} \circ \mathcal{M}'$. □

In [19], the authors show that if a mapping $\mathcal{M}$ is Fagin-invertible, then a mapping $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$ if and only if $\mathcal{M}'$ is a quasi-inverse of $\mathcal{M}$. Example 4.3 shows that the opposite direction does not hold. Thus, the notion of quasi-inverse is a strict generalization of the notion of Fagin-inverse. Furthermore, the author provides in [19] a necessary and sufficient condition for the existence of quasi-inverses for mappings specified by st-tgds, and use this condition to show the following result:

Proposition 4.4 ([19]) *There is a mapping $\mathcal{M}$ specified by a single st-tgd that has no quasi-inverse.*

Thus, although numerous non-Fagin-invertible schema mappings possess natural and useful quasi-inverses [19], there are still simple mappings specified by st-tgds that have no quasi-inverse. This leaves as an open problem the issue of finding an appropriate notion of inversion for st-tgds, and it is the main motivation for the introduction of the notion of inversion discussed in the following section.

4.2 Maximum recovery

We consider now the notion of maximum recovery introduced by Arenas et al. in [4]. In that paper, the authors follow a different approach to define a notion of inversion. In fact, the main goal of [4] is not to define a notion of inverse mapping, but instead to give a formal definition for what it means for a mapping $\mathcal{M}'$ to recover *sound information* with respect to a mapping $\mathcal{M}$. Such a mapping $\mathcal{M}'$ is called a recovery of $\mathcal{M}$ in [4]. Given that, in general, there may exist many possible recoveries for a given mapping, Arenas et al. introduce an order relation on recoveries in [4], and show that this naturally gives rise to the notion of maximum recovery, which is a mapping that brings back the maximum amount of sound information.

Let $\mathcal{M}$ be a mapping from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$, and Id the identity schema mapping over $\mathbf{R}_1$, that is, $\mathrm{Id} = \{(I,I) \mid I \in \mathrm{Inst}(\mathbf{R}_1)\}$. When trying to invert $\mathcal{M}$, the ideal would be to find a mapping $\mathcal{M}'$ from $\mathbf{R}_2$ to $\mathbf{R}_1$ such that $\mathcal{M} \circ \mathcal{M}' = \mathrm{Id}$. Unfortunately, in most cases this ideal is impossible to reach (for example, for the case of mappings specified by st-tgds [12]). If for a mapping $\mathcal{M}$, there is no mapping $\mathcal{M}_1$ such that $\mathcal{M} \circ \mathcal{M}_1 = \mathrm{Id}$, at least one would like to find a schema mapping $\mathcal{M}_2$ that does not forbid the possibility of recovering the initial source data. This gives rise to the notion of recovery proposed in [4]. Formally, given a mapping $\mathcal{M}$ from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$, a mapping $\mathcal{M}'$ from $\mathbf{R}_2$ to $\mathbf{R}_1$ is a *recovery* of $\mathcal{M}$ if $(I,I) \in \mathcal{M} \circ \mathcal{M}'$ for every instance $I \in \mathrm{dom}(\mathcal{M})$ [4].

In general, if $\mathcal{M}'$ is a recovery of $\mathcal{M}$, then the smaller the space of solutions generated by $\mathcal{M} \circ \mathcal{M}'$, the more informative $\mathcal{M}'$ is about the initial source instances. This naturally gives rise to the notion of maximum recovery; given a mapping $\mathcal{M}$ and a recovery $\mathcal{M}'$ of it, $\mathcal{M}'$ is said to be a *maximum recovery* of $\mathcal{M}$ if for every recovery $\mathcal{M}''$ of $\mathcal{M}$, it holds that $\mathcal{M} \circ \mathcal{M}' \subseteq \mathcal{M} \circ \mathcal{M}''$ [4].

Example 4.5. In [19], it was shown that the schema mapping $\mathcal{M}$ specified by st-tgd

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

has neither a Fagin-inverse nor a quasi-inverse. However, it is possible to show that the schema mapping $\mathcal{M}'$ specified by ts-tgds:

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

is a maximum recovery of $\mathcal{M}$. Notice that, intuitively, the mapping $\mathcal{M}'$ is making the *best effort* to recover the initial data transferred by $\mathcal{M}$. □

In [4], Arenas et al. study the relationship between the notions of Fagin-inverse, quasi-inverse and maximum recovery. It should be noticed that the first two notions are only appropriate for total and closed-down on the left mappings [12, 4]. Thus, the comparison in [4] focus on these mappings. More precisely, it is shown in [4] that for every mapping $\mathcal{M}$ that is total and closed-down on the left, if $\mathcal{M}$ is Fagin-invertible, then $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$ if and only if $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$. Thus, from Example 4.5, one can conclude that the notion of maximum recovery strictly generalizes the notion of Fagin-inverse. The exact relationship between the notions of quasi-inverse and maximum recovery is a bit more involved. For every mapping $\mathcal{M}$ that is total and closed-down on the left, it is shown in [4] that if $\mathcal{M}$ is quasi-invertible, then $\mathcal{M}$ has a maximum recovery and, furthermore, every maximum recovery of $\mathcal{M}$ is also a quasi-inverse of $\mathcal{M}$.

In [4], the authors provide a necessary and sufficient condition for the existence of a maximum recovery. It is important to notice that this is general condition as it can be applied to any mapping, as long as it is defined as a set of pairs of instances. This condition is used in [4] to prove that every mapping specified by a set of st-tgds has a maximum recovery.

Theorem 4.6 ([4]) *Every mapping $\mathcal{M}$ specified by a set of st-tgds has a maximum recovery.*

4.3 Inverses for alternative semantics

When mappings are specified by sets of logical formulas, we have considered the usual semantics of mappings based on logical satisfaction. However, some alternative semantics have been considered in the literature, such as the *closed world semantics* [27], the *universal semantics* [13], and the *extended semantics* [17]. Although some of the notions of inverse discussed in the previous sections can be directly applied to these alternative semantics, the positive and negative results on the existence of inverses need to be reconsidered in these particular cases. In this section, we focus on this problem for the universal and extended semantics of mappings.

4.3.1 Universal solutions semantics

Recall that a homomorphism from an instance J_1 to an instance J_2 is a function $h : \text{dom}(J_1) \rightarrow \text{dom}(J_2)$ such that (1) $h(c) = c$ for every constant $c \in \text{dom}(J_1)$, and (2) for every tuple $R(a_1, \ldots, a_k)$ in J_1, tuple $R(h(a_1), \ldots, h(a_k))$ is in J_2. Given a mapping $\mathcal{M}$ and a source instance I, a target instance $J \in \text{Sol}_{\mathcal{M}}(I)$ is a universal solution for I under $\mathcal{M}$ if for every $J' \in \text{Sol}_{\mathcal{M}}(I)$, there exists a homomorphism from J to J'. It was shown in [13, 14] that universal solutions have several desirable properties for data exchange. In view of this fact, an alternative semantics based on universal solutions was proposed in [14] for schema mappings. Given a mapping $\mathcal{M}$, the mapping $u(\mathcal{M})$ is defined as the set of pairs

$$\{(I, J) \mid J \text{ is a universal solution for } I \text{ under } \mathcal{M}\}.$$

Mapping $u(\mathcal{M})$ was introduced in [14] in order to give a clean semantics for answering target queries after exchanging data with mapping $\mathcal{M}$. By combining the results on universal solutions for mappings given by st-tgds in [13] and the results in [5] on the existence of maximum recoveries, one can easily prove the following:

Proposition 4.7 *Let $\mathcal{M}$ be a mapping specified by a set of st-tgds. Then $u(\mathcal{M})$ has a maximum recovery. Moreover, the mapping $(u(\mathcal{M}))^{-1} = \{(J, I) \mid (I, J) \in u(\mathcal{M})\}$ is a maximum recovery of $u(\mathcal{M})$.*

4.3.2 Extended solutions semantics

A more delicate issue regarding the semantics of mappings was considered in [17]. In this paper, Fagin et al. made the observation that almost all the literature about data exchange and, in particular, the literature about inverses of schema mappings, assume that source instances do not have null values. Since null values in the source may naturally arise when using inverses of mappings to exchange data, the authors relax the restriction on source instances allowing them to contain values in $\mathbf{C} \cup \mathbf{N}$. In fact, the authors go a step further and propose new refined notions for inverting mappings that consider nulls in the source. In particular, they propose the notions of *extended inverse*, and of *extended recovery* and *maximum extended recovery*. In this section, we review the definitions of the latter two notions and compare them with the previously proposed notions of recovery and maximum recovery.

The first observation to make is that since null values are intended to represent *missing* or *unknown* information, they should not be treated naively as constants [25]. In fact, as shown in [17], if one treats nulls in that way, the existence of a maximum recovery for mappings given by st-tgds is no longer guaranteed.

Example 4.8. Consider a source schema $\{S(\cdot, \cdot)\}$ where instances may contain null values, and let $\mathcal{M}$ be a mapping specified by st-tgd $S(x, y) \rightarrow \exists z\, (T(x, z) \wedge T(z, y))$. Then $\mathcal{M}$ has no maximum recovery if one considers a naïve semantics where null elements are used as constants in the source [17]. □

Since nulls should not be treated naively when exchanging data, in [17] the authors proposed a new way to deal with null values. Intuitively, the idea in [17] is to *close* mappings under homomorphisms. This idea is supported by the fact that nulls are intended to represent unknown data, thus, it should be possible to replace them by arbitrary values. Formally, given a mapping $\mathcal{M}$, define $e(\mathcal{M})$, the *homomorphic extension* of $\mathcal{M}$, as the mapping:

$$\{(I, J) \mid \exists (I', J') : (I', J') \in \mathcal{M} \text{ and there exist homomorphisms from } I \text{ to } I' \text{ and from } J' \text{ to } J\}.$$

Thus, for a mapping $\mathcal{M}$ that has nulls in source and target instances, one does not have to consider $\mathcal{M}$ but $e(\mathcal{M})$ as the mapping to deal with for exchanging data and computing mapping operators, since $e(\mathcal{M})$ treats nulls in a meaningful way [17]. The following result shows that with this new semantics one can avoid anomalies as the one shown in Example 4.8.

Theorem 4.9 ([18]) *For every mapping $\mathcal{M}$ specified by a set of st-tgds and with nulls in source and target instances, $e(\mathcal{M})$ has a maximum recovery.*

As mentioned above, Fagin et al. go a step further in [17] by introducing new notions of inverse for mappings that consider nulls in the source. More specifically, a mapping $\mathcal{M}'$ is said to be an *extended recovery* of $\mathcal{M}$ if $(I, I) \in e(\mathcal{M}) \circ e(\mathcal{M}')$, for every source instance I. Then given an extended recovery $\mathcal{M}'$ of $\mathcal{M}$, the mapping $\mathcal{M}'$ is said to be a *maximum extended recovery* of $\mathcal{M}$ if for every extended recovery $\mathcal{M}''$ of $\mathcal{M}$, it holds that $e(\mathcal{M}) \circ e(\mathcal{M}') \subseteq e(\mathcal{M}) \circ e(\mathcal{M}'')$ [17].

At a first glance, one may think that the notions of maximum recovery and maximum extended recovery are incomparable. Nevertheless, the next result shows that there is a tight connection between these two notions. In particular, it shows that the notion proposed in [17] can be defined in terms of the notion of maximum recovery.

Theorem 4.10 *A mapping $\mathcal{M}$ has a maximum extended recovery if and only if $e(\mathcal{M})$ has a maximum recovery. Moreover, $\mathcal{M}'$ is a maximum extended recovery of $\mathcal{M}$ if and only if $e(\mathcal{M}')$ is a maximum recovery of $e(\mathcal{M})$.*

In [17], it is proved that every mapping specified by a set of st-tgds and considering nulls in the source has a maximum extended recovery. It should be noticed that this result is also implied by Theorems 4.9 and 4.10.

Finally, another conclusion that can be drawn from the above result is that, all the machinery developed in [4, 5] for the notion of maximum recovery can be applied over maximum extended recoveries, and the extended semantics for mappings, thus giving a new insight about inverses of mappings with null values in the source.

4.4 Computing the inverse

Up to this point, we have introduced and compared three notions of inverse proposed in the literature, focusing mainly on the fundamental problem of the existence of such inverses. In this section, we study the problem of computing these inverses. More specifically, we present some of the algorithms that have been proposed in the literature for computing them, and we study the languages used in these algorithms to express these inverses.

Arguably, the most important problem to solve in this area is the problem of computing inverses of mappings specified by st-tgds. This problem has been studied for the case of Fagin-inverse [19, 20], quasi-inverse [19], maximum recovery [4, 3, 5] and maximum extended recovery [17, 18]. In this section, we start by presenting the algorithm proposed in [5] for computing maximum recoveries of mappings specified by st-tgds, which by the results of Sections 4.1 and 4.2 can also be used to compute Fagin-inverses and quasi-inverses for this class of mappings. Interestingly, this algorithm is based on *query rewriting*, which greatly simplifies the process of computing such inverses.

Let $\mathcal{M}$ be a mapping from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$ and Q a query over schema $\mathbf{R}_2$. Then a query Q' is said to be a *rewriting of Q over the source* if Q' is a query over $\mathbf{R}_1$ such that for every $I \in \text{Inst}(\mathbf{R}_1)$, it holds that $Q'(I) = \underline{\text{certain}}_{\mathcal{M}}(Q, I)$. That is, to obtain the set of certain answers of Q over I under $\mathcal{M}$, one just has to evaluate its rewriting Q' over instance I.

The computation of a rewriting of a conjunctive query is a basic step in the first algorithm presented in this section. This problem has been extensively studied in the database area [30, 31, 11, 1, 37] and, in particular, in the data integration context [23, 22, 29]. The following algorithm uses a query rewriting procedure QUERYREWRITING to compute a maximum recovery of a mapping $\mathcal{M}$ specified by a set Σ of st-tgds. In the algorithm, if $\bar{x} = (x_1, \ldots, x_k)$, then $\mathbf{C}(\bar{x})$ is a shorthand for $\mathbf{C}(x_1) \wedge \cdots \wedge \mathbf{C}(x_k)$.

```
Algorithm MAXIMUMRECOVERY($\mathcal{M}$)
Input: $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\Sigma$ is a set of st-tgds.
Output: $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$, where $\Sigma'$ is a set of CQ$^{\mathbf{C}}$-TO-UCQ$^=$ ts-dependencies and $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$.
1. Start with $\Sigma'$ as the empty set.
2. For every dependency of the form $\varphi(\bar{x}) \rightarrow \exists \bar{y}\, \psi(\bar{x}, \bar{y})$ in $\Sigma$, do the following:
  (a) Let $Q$ be the query defined by $\exists \bar{y}\, \psi(\bar{x}, \bar{y})$.
  (b) Use QUERYREWRITING($\mathcal{M}, Q$) to compute a formula $\alpha(\bar{x})$ in UCQ$^=$ that is a rewriting of $\exists \bar{y}\, \psi(\bar{x}, \bar{y})$ over the source.
```

(c) Add dependency $\exists \bar{y}\, \psi(\bar{x}, \bar{y}) \wedge \mathbf{C}(\bar{x}) \rightarrow \alpha(\bar{x})$ to Σ'.

3. Return $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$. □

Theorem 4.11 ([4, 5]) *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a set of st-tgds. Then* MAXIMUMRECOVERY($\mathcal{M}$) *computes a maximum recovery of $\mathcal{M}$ in exponential time in the size of Σ, which is specified by a set of* $\text{CQ}^{\mathbf{C}}$-TO-$\text{UCQ}^{=}$ *dependencies. Moreover, if $\mathcal{M}$ is Fagin-invertible (quasi-invertible), then the output of* MAXIMUMRECOVERY($\mathcal{M}$) *is a Fagin-inverse (quasi-inverse) of $\mathcal{M}$.*

It is important to notice that the algorithm MAXIMUMRECOVERY returns a mapping that is a Fagin-inverse of an input mapping $\mathcal{M}$ whenever $\mathcal{M}$ is Fagin-invertible, but it does not check whether $\mathcal{M}$ indeed satisfies this condition (and likewise for the case of quasi-inverse). In fact, it is not immediately clear whether the problem of checking if a mapping given by a set of st-tgds has a Fagin-inverse is decidable. In [20], the authors solve this problem showing the following:

Theorem 4.12 ([20]) *The problem of verifying whether a mapping specified by a set of st-tgds is Fagin-invertible is coNP-complete.*

Interestingly, it is not known whether the previous problem is decidable for the case of the notion of quasi-inverse.

One of the interesting features of algorithm MAXIMUMRECOVERY is the use of query rewriting, as it allows to reuse in the computation of an inverse the large number of techniques developed to deal with the problem of query rewriting. However, one can identify two drawbacks in this procedure. First, algorithm MAXIMUMRECOVERY returns a mapping that is specified by a set of $\text{CQ}^{\mathbf{C}}$-TO-$\text{UCQ}^{=}$ dependencies. Unfortunately, this type of mappings are difficult to use in the data exchange context. In particular, it is not clear whether the standard chase procedure could be used to produce a single canonical target database in this case, thus making the process of exchanging data and answering queries much more complicated. Second, the output mapping of MAXIMUMRECOVERY can be of exponential size in the size of the input mapping. Thus, a natural question at this point is whether simpler and smaller inverse mappings can be computed. In the rest of this section, we show some negative results in this respect, and also some efforts to overcome these limitations by using more expressive mapping languages.

The languages needed to express Fagin-inverses and quasi-inverses are investigated in [19, 20]. In the respect, the first negative result proved in [19] is that there exist quasi-invertible mappings specified by st-tgds whose quasi-inverse cannot be specified by st-tgds. In fact, it is proved in [19] that the quasi-inverse of a mapping given by st-tgds can be specified by using $\text{CQ}^{\neq,\mathbf{C}}$-TO-UCQ dependencies, and that inequality, predicate $\mathbf{C}(\cdot)$ and disjunction are all unavoidable in this language in order to express such quasi-inverse. For the case of Fagin-inverse, it is shown in [19] that disjunctions are not needed, that is, the class of $\text{CQ}^{\neq,\mathbf{C}}$-TO-CQ dependencies is expressive enough to represent the Fagin-inverse of a Fagin-invertible mapping specified by a set of st-tgds. In [12, 20], it is proved a second negative result about the languages needed to express Fagin-inverses, namely that there is a family of Fagin-invertible mappings $\mathcal{M}$ specified by st-tgds such that the size of every Fagin-inverse of $\mathcal{M}$ specified by a set of $\text{CQ}^{\neq,\mathbf{C}}$-TO-CQ dependencies is exponential in the size of $\mathcal{M}$. Similar results are proved in [4, 5] for the case of maximum recoveries of mappings specified by st-tgds. More specifically, it is proved in [4] that the maximum recovery of a mapping given by st-tgds can be specified by using $\text{CQ}^{\mathbf{C}}$-TO-$\text{UCQ}^{=}$ dependencies, and that equality, predicate $\mathbf{C}(\cdot)$ and disjunction are all unavoidable in this language in order to express such maximum recovery. Moreover, it is proved in [5] that there is a family of mappings $\mathcal{M}$ specified by st-tgds such that the size of every maximum recovery of $\mathcal{M}$ specified by a set of $\text{CQ}^{\mathbf{C}}$-TO-$\text{UCQ}^{=}$ dependencies is exponential in the size of $\mathcal{M}$.

In view of the above negative results, Arenas et al. explore in [3] the possibility of using a more expressive language for representing inverses. In particular, they explore the possibility of using some extensions of the class of SO-tgds to express this operator. In fact, Arenas et al. provide in [3] a polynomial-time algorithm that given a mapping $\mathcal{M}$ specified by a set of st-tgds, returns a maximum recovery of $\mathcal{M}$, which is specified in a language that extends SO-tgds (see [3] for a precise definition of this language). It should be noticed that the algorithm presented in [3] was designed to compute maximum recoveries of mappings specified in languages beyond st-tgds, such as the language of *nested mappings* [21] and plain SO-tgds (see Section 5 for a definition of the class of plain SO-tgds). Thus, the algorithm proposed in [3] can also be used to compute in polynomial time Fagin-inverses (quasi-inverses) of Fagin-invertible (quasi-invertible) mappings specified by st-tgds, nested mappings and plain SO-tgds. Interestingly, a similar approach was used in [18] to provide a polynomial-time algorithm for computing the maximum extended recovery for the case of mappings defined by st-tgds.

5 Query-based notions of composition and inverse

As we have discussed in the previous sections, to express the composition and the inverse of schema mappings given by st-tgds, one usually needs mapping languages that are more expressive than st-tgds, and that do not have the same good properties for data exchange as st-tgds.

As a way to overcome this limitation, some weaker notions of composition and inversion have been proposed in the recent years, which are based on the idea that in practice one may be interested in querying exchanged data by using only a particular class of queries. In this section, we review these notions.

5.1 A query-based notion of composition

In this section, we study the notion of *composition w.r.t. conjunctive queries* (CQ-composition for short) introduced by Madhavan and Halevy [32]. This semantics for composition can be defined in terms of the notion of *conjunctive-query equivalence* of mappings that was introduced in [32] for studying CQ-composition and generalized in [15] when studying optimization of schema mappings. Two mappings $\mathcal{M}$ and $\mathcal{M}'$ from $\mathbf{S}$ to $\mathbf{T}$ are said to be *equivalent w.r.t. conjunctive queries*, denoted by $\mathcal{M} \equiv_{\text{CQ}} \mathcal{M}'$, if for every conjunctive query Q, the set of certain answers of Q under $\mathcal{M}$ coincides with the set of certain answers of Q under $\mathcal{M}'$. Formally, $\mathcal{M} \equiv_{\text{CQ}} \mathcal{M}'$ if for every conjunctive query Q over $\mathbf{T}$ and every instance I of $\mathbf{S}$, it holds that $\underline{\text{certain}}_{\mathcal{M}}(Q, I) = \underline{\text{certain}}_{\mathcal{M}'}(Q, I)$. Then CQ-composition can be defined as follows: $\mathcal{M}_3$ is a CQ-composition of $\mathcal{M}_1$ and $\mathcal{M}_2$ if $\mathcal{M}_3 \equiv_{\text{CQ}} \mathcal{M}_1 \circ \mathcal{M}_2$.

A fundamental question about the notion of CQ-composition is whether the class of st-tgds is closed under this notion. This problem was implicitly studied by Fagin et al. [15] in the context of schema mapping optimization. In [15], the authors consider the problem of whether a mapping specified by an SO-tgd is CQ-equivalent to a mapping specified by st-tgds. Thus, given that the composition of a finite number of mappings given by st-tgds can be defined by an SO-tgd [16], the latter problem is a reformulation of the problem of testing whether st-tgds are closed under CQ-composition. In fact, by using the results and the examples in [15], one can easily construct mappings $\mathcal{M}_1$ and $\mathcal{M}_2$ given by st-tgds such that the CQ-composition of $\mathcal{M}_1$ and $\mathcal{M}_2$ is not definable by a finite set of st-tgds.

A second fundamental question about the notion of CQ-composition is what is the right language to express it. Although this problem is still open, in the rest of this section we shed light on this issue. By the results in [16], we know that the language of SO-tgds is enough to represent the CQ-composition of st-tgds. However, as motivated by the following example, some features of SO-tgds are not needed to express the CQ-composition of mappings given by st-tgds.

Example 5.1. (from [16]) Consider a schema $\mathbf{R}_1$ consisting of one unary relation $\texttt{Emp}$ that stores employee names, a schema $\mathbf{R}_2$ consisting of a binary relation $\texttt{Mgr}_\texttt{1}$ that assigns a manager to each employee, and a schema $\mathbf{R}_3$ consisting of a binary relation $\texttt{Mgr}$ intended to be a copy of $\texttt{Mgr}_1$ and of a unary relation $\texttt{SelfMgr}$, that stores employees that are manager of themselves. Consider now mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ specified by the following sets of st-tgds:

$$\begin{aligned}\Sigma_{12} &= \{\, \texttt{Emp}(e) \rightarrow \exists m\, \texttt{Mgr}_\texttt{1}(e,m) \,\},\\ \Sigma_{23} &= \{\, \texttt{Mgr}_\texttt{1}(e,m) \rightarrow \texttt{Mgr}(e,m),\\ &\qquad \texttt{Mgr}_\texttt{1}(e,e) \rightarrow \texttt{SelfMgr}(e) \,\}.\end{aligned}$$

Mapping $\mathcal{M}_{12}$ intuitively states that every employee must be associated with a manager. Mapping $\mathcal{M}_{23}$ requires that a copy of every tuple in $\texttt{Mgr}_\texttt{1}$ must exists in $\texttt{Mgr}$, and creates a tuple in $\texttt{SelfMgr}$ whenever an employee is the manager of her/himself. It was shown in [16] that the mapping $\mathcal{M}_{13}$ given by the following SO-tgd:

$$\begin{aligned}\exists f \big(&\forall e (\texttt{Emp}(e) \rightarrow \texttt{Mgr}(e, f(e))) \wedge\\ &\forall e (\texttt{Emp}(e) \wedge e = f(e) \rightarrow \texttt{SelfMgr}(e))\big) \qquad (2)\end{aligned}$$

represents the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$. Moreover, the authors prove in [16] that the equality in the above formula is strictly necessary to represent that composition. However, it is not difficult to prove that the mapping $\mathcal{M}'_{13}$ given by the following formula:

$$\exists f \big(\forall e (\texttt{Emp}(e) \rightarrow \texttt{Mgr}(e, f(e)))\big) \qquad (3)$$

is CQ-equivalent to $\mathcal{M}_{13}$, and thus, $\mathcal{M}'_{13}$ is a CQ-composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$. $\square$

We say that formula (3) is a *plain SO-tgd*. Formally, a plain SO-tgd from $\mathbf{R}_1$ to $\mathbf{R}_2$ is an SO-tgd satisfying the following restrictions: (1) equality atoms are not allowed, and (2) nesting of functions is not allowed. Notice that, just as SO-tgds, this language is closed under conjunction and, thus, we talk about a mapping specified by a plain SO-tgd (instead of a set of plain SO-tgds). The following result shows that even though the language of plain SO-tgds is less expressive than the language of SO-tgds, they are equally expressive in terms of CQ-equivalence.

Lemma 5.2 *For every SO-tgd σ, there exists a plain SO-tgd σ' such that $\sigma \equiv_{\text{CQ}} \sigma'$.*

It is easy to see that every mapping specified by a set of st-tgds can be specified with a plain SO-tgd. Moreover, the following theorem shows that this language is closed under CQ-composition, thus showing that this class of dependencies has good properties within the framework of CQ-equivalence.

Theorem 5.3 *Let $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ be mappings specified by plain SO-tgds. Then the* CQ*-composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ can be specified with a plain SO-tgd.*

Thus, the CQ-composition of a finite number of mappings, each specified by a set of st-tgds, is definable by a plain SO-tgd. It should be noticed that Theorem 5.3 is a consequence of Lemma 5.2 and the fact that the class of SO-tgds is closed under composition [16].

Besides the above mentioned results, the language of plain SO-tgds also has good properties regarding inversion. In particular, it is proved in [3] that every plain SO-tgd has a maximum recovery, and, moreover, it is given in that paper a polynomial-time algorithm to compute it. Thus, it can be argued that this class of dependencies is more suitable for inversion than SO-tgds, as there exist SO-tgds that do not admit maximum recoveries.

5.2 A query-based notion of inverse

In [3], the authors propose an alternative notion of inverse by focusing on conjunctive queries. In particular, the authors first define the notion of CQ-*recovery* as follows. A mapping $\mathcal{M}'$ is a CQ-recovery of $\mathcal{M}$ if for every instance I and conjunctive query Q, it holds that

$$\underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}'}(Q, I) \quad \subseteq \quad Q(I).$$

Intuitively, this equation states that $\mathcal{M}'$ *recovers sound information* for $\mathcal{M}$ w.r.t. conjunctive queries since for every instance I, by posing a conjunctive query Q against the space of solutions for I under $\mathcal{M} \circ \mathcal{M}'$, one can only recover data that is already in the evaluation of Q over I. A CQ-*maximum recovery* is then defined as a mapping that recovers the maximum amount of sound information w.r.t. conjunctive queries. Formally, a CQ-recovery $\mathcal{M}'$ of $\mathcal{M}$ is a CQ-maximum recovery of $\mathcal{M}$ if for every other CQ-recovery $\mathcal{M}''$ of $\mathcal{M}$, it holds that

$$\underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}''}(Q, I) \quad \subseteq \quad \underline{\text{certain}}_{\mathcal{M}\circ\mathcal{M}'}(Q, I),$$

for every instance I and conjunctive query Q.

In [3], the authors study several properties about CQ-maximum recoveries. In particular, they provide an algorithm to compute CQ-maximum recoveries for st-tgds showing the following:

Theorem 5.4 ([3]) *Every mapping specified by a set of st-tgds has a* CQ*-maximum recovery, which is specified by a set of* $\mathrm{CQ}^{\mathbf{C},\neq}$-TO-CQ *dependencies.*

Notice that the language needed to express CQ-maximum recoveries of st-tgds has the same good properties as st-tgds for data exchange. In particular, the language is *chaseable* in the sense that the standard chase procedure can be used to obtain a canonical solution. Thus, compared to the notions of Fagin-inverse, quasi-inverse, and maximum recovery, the notion of CQ-maximum recovery has two advantages: (1) every mapping specified by st-tgds has a CQ-maximum recovery (which is not the case for Fagin-inverses and quasi-inverses), and (2) such recovery can be specified in a mapping language with good properties for data exchange (which is not the case for quasi-inverses and maximum recovery).

In [3], the authors also study the minimality of the language used to express CQ-maximum recoveries, showing that inequalities and predicate $\mathbf{C}(\cdot)$ are both needed to express the CQ-maximum recoveries of mappings specified by st-tgds.

6 Future Work

As many information-system problems involve not only the design and integration of complex application artifacts, but also their subsequent manipulation, the definition and implementation of some operators for meta data management has been identified as a fundamental issue to be solved [7]. In particular, composition and inverse have been identified as two of the fundamental operators to be studied in this area, as they can serve as building blocks of many other operators [33, 35]. In this paper, we have presented some of the results that have been obtained in the recent years about the composition and inversion of schema mappings.

Many problems remain open in this area. Up to now, XML schema mapping languages have been proposed and studied [6, 2, 38], but little attention has been paid to the formal study of XML schema mapping operators. For the case of composition, a first insight has been given in [2], showing that the previous results for the relational model are not directly applicable over XML. Inversion of XML schema mappings remains an unexplored field.

Regarding the relational model, we believe that the future effort has to be focused in providing a unifying framework for these operators, one that permits the successful application of them. A natural question, for instance, is whether there exists a schema mapping language that is closed under both composition and inverse. Needless to say, this unified framework will permit the modeling of more complex algebraic operators for schema mappings.

Acknowledgments

We would like to thank L. Libkin for many useful comments. The authors were supported by: Arenas - Fondecyt grant 1090565; Pérez - Conicyt Ph.D. Scholarship.

References

[1] S. Abiteboul and O. Duschka. Complexity of Answering Queries Using Materialized Views. In *PODS*, pages 254–263, 1998.

[2] S. Amano, L. Libkin, and F .Murlak. XML schema mappings. In *PODS*, pages 33-42, 2009.

[3] 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.

[4] M. Arenas, J. Pérez, and C. Riveros. The recovery of a schema mapping: bringing exchanged data back. In *PODS*, pages 13–22, 2008.

[5] M. Arenas, J. Pérez, and C. Riveros. The recovery of a schema mapping: bringing exchanged data back. To appear in *TODS*, 2009.

[6] M. Arenas and L. Libkin XML data exchange: Consistency and query answering. *JACM*, 55(2), 2008.

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

[8] P. A. Bernstein, S. Melnik. Model management 2.0: manipulating richer mappings. In *SIGMOD*, pages 1-12, 2007.

[9] P. A. Bernstein, T. Green, S. Melnik, and A. Nash. Implementing mapping composition, VLDB J. 17(2): 333-353, 2008.

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

[11] O. Duschka, M. Genesereth. Answering Recursive Queries Using Views. In *PODS*, pages 109–116, 1997

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

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

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

[15] R. Fagin, P. G. Kolaitis, A. Nash, L. Popa. Towards a theory of schema-mapping optimization. In *PODS*, pages 33–42, 2008.

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

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

[18] R. Fagin, P. Kolaitis, L. Popa, and W.-C. Tan. Reverse data exchange: coping with nulls. Extended version of [17], submitted for publication.

[19] R. Fagin, P. Kolaitis, L. Popa, and W.-C. Tan. Quasi-inverses of schema mappings. In *TODS*, 33(2), 2008.

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

[21] A. Fuxman, M. Hernández, H. Ho, R. Miller, P. Papotti, L. Popa Nested Mappings: Schema Mapping Reloaded In *VLDB*, pages 67–78, 2006

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

[23] A. Halevy. Theory of Answering Queries using Views. SIGMOD Record 29(1), pages 40–47, 2000.

[24] A. Halevy, Z. Ives, J. Madhavan, P. Mork, D. Suciu, I. Tatarinov. The Piazza Peer Data Management System. *IEEE TKDE* 16(7):787–798 (2004)

[25] T. Imielinski and W. Lipski Jr. Incomplete information in relational databases. *JACM*, 31(4):761–791, 1984.

[26] L. Libkin. Elements of Finite Model Theory. Springer, 2004.

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

[28] L. Libkin, C. Sirangelo. Data Exchange and Schema Mappings in Open and Closed Worlds In *PODS*, pages 139–148, 2008.

[29] M. Lenzerini. Data Integration: A Theoretical Perspective.. In *PODS*, pages 233–246, 2002.

[30] A. Levy, A. Mendelzon, Y. Sagiv and D. Srivastava. Answering Queries Using Views. In *PODS*, pages 95–104, 1995.

[31] A. Levy, A. Rajaraman and J. Ordille. Querying Heterogeneous Information Sources using Source Descriptions. In *VLDB*, pages 251–262, 1996.

[32] J. Madhavan and A. Y. Halevy. Composing mappings among data sources. In *VLDB*, pages 572–583, 2003.

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

[34] S. Melnik, A. Adya, P. A. Bernstein. Compiling mappings to bridge applications and databases. In *TODS* 33(4), 2008.

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

[36] A. Nash, P. A. Bernstein, S. Melnik. Composition of mappings given by embedded dependencies. In *TODS* 32(1), 2007.

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

[38] J. F. Terwilliger, P. A. Bernstein, and S. Melnik. Full-Fidelity Flexible Object-Oriented XML Access. In *VLDB*, pages 1030–1041, 2009.

[39] M. Y. Vardi. The Complexity of Relational Query Languages. In *STOC*, pages 137–146, 1982.