The Inverse of a Schema Mapping

Jorge Pérez

Department of Computer Science, Universidad de Chile
Blanco Encalada 2120, Santiago, Chile
`jperez@dcc.uchile.cl`

Abstract

The inversion of schema mappings has been identified as one of the fundamental operators for the development of a general framework for data exchange, data integration, and more generally, for metadata management. Given a mapping $\mathcal{M}$ from a schema $\mathbf{S}$ to a schema $\mathbf{T}$, *an inverse* of $\mathcal{M}$ is a new mapping that describes the *reverse* relationship from $\mathbf{T}$ to $\mathbf{S}$, and that is *semantically consistent* with the relationship previously established by $\mathcal{M}$. In practical scenarios, the inversion of a schema mapping can have several applications. For example, in a data exchange context, 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 $\mathcal{M}$.

The formalization of a clear semantics for the inverse operator has proved to be a very difficult task. In fact, during the last years, several alternative notions of inversion for schema mappings have been proposed in the literature. This chapter provides a survey on the different formalizations for the inverse operator and the main theoretical and practical results obtained so far. In particular, we present and compare the main proposals for inverting schema mappings that have been considered in the literature. For each one of them we present their formal semantics and characterizations of their existence. We also present algorithms to compute inverses and study the language needed to express such inverses.

1998 ACM Subject Classification H.2.5 [**Heterogeneous Databases**]: Data translation

Keywords and phrases Schema mappings, data exchange, inverse

Digital Object Identifier 10.4230/DFU.Vol5.10452

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 are of fundamental importance in data management today. In particular, they have proved to be the essential building block for several data-interoperability tasks such as data exchange, data integration and peer data management.

In recent years, the research on the schema mapping area has mainly focused on performing data-interoperability tasks using schema mappings. However, as Bernstein [12] pointed out, many information-system problems involve not only the design and integration of complex application artifacts, but also their subsequent manipulation. Notice that the creation of a schema mapping may imply considerable work by an expert who needs to know the semantics of the schema components. Only an expert can establish a meaningful high-level correspondence between those components. Thus, a schema mapping reflects the knowledge of the expert about the relationship between the schemas. This knowledge could, in principle, be reused beyond the interoperability tasks for which the mapping was initially created. Driven by these considerations, Bernstein [12] proposed a general framework for managing and reusing schema mappings.

Dummy Event this volume is based on.
Editor: Editor; pp. 1–28
Dagstuhl Publishing
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany

In Bernstein's framework [12], schema mappings are first class citizens, and high-level algebraic operators are used to manipulate and reuse them. One of the most fundamental operators in schema mapping management is the *inversion* of schema mappings. Given a mapping $\mathcal{M}$ from a schema $\mathbf{A}$ to a schema $\mathbf{B}$, *an inverse* of $\mathcal{M}$ is a new mapping that describes the *reverse* relationship from $\mathbf{B}$ to $\mathbf{A}$, and that is *semantically consistent* with the relationship previously established by $\mathcal{M}$. Notice that this is a very general idea of what an inverse of a schema mapping should be. In fact, even the formalization of a clear semantics for the inverse operator has proved to be a very difficult task [16, 17, 20, 21, 10, 11, 22, 6]. This chapter provides a survey on the different formalizations in the literature for the inverse operator on schema mappings and the study of the theoretical problems that arise. Before going into the details of these works, let us give a bit more intuition on how the inverse of schema mappings can be useful in practice.

In practical scenarios, the inversion of schema mappings can have several applications. In a data exchange context [18], 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) [14, 26]. In a PDMS, a peer can act as a data source, a mediator, or both, and the system relates peers by establishing mappings between the peers' schemas. Mappings between peers are usually *directional*, and are used to reformulate queries. For example, if there is a mapping $\mathcal{M}$ from peer P_1 to peer P_2 and a query over P_2, a PDMS can use $\mathcal{M}$ to reformulate the query by using P_1 as a source. Hence, an inverse of $\mathcal{M}$ would allow the PDMS to reformulate a query over P_1 in terms of P_2, thus considering this time P_2 as a source. Another application is schema evolution, where the inverse together with the *composition* play a crucial role [13, 23]. 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 intuitively obtained by inverting mapping $\mathcal{M}'$ and then composing the result with mapping $\mathcal{M}$.

As we have mentioned before, in the study of the inverse operator, one of the key issues is to provide a *good* semantics for this operator, which turned out to be a difficult problem. After defining a semantics, some of the important questions that need to be answered are:

Existence For which classes of mappings is the inverse guaranteed to exist?
Expressiveness What is the mapping language needed to specify an inverse?
Algorithmic How can we effectively construct an inverse?

In this chapter, we present and compare the main proposals for inverting schema mappings that have been considered in the literature, and for each one of them we present the main results regarding these three issues. In Section 2 we present the notion of inverse proposed by Fagin [16], that we call here *Fagin-inverse*[1], which is the first formal notion of inverse proposed in the literature. In Section 3 we present the notion of *quasi-inverse* [20, 21] which is obtained by relaxing the notion of Fagin-inverse. Section 4 presents the notions of *recovery* and *maximum recovery* [10, 11] which were proposed as alternative notions for inverting schema mappings. In Section 5 we present procedures to compute inverses and discuss expressiveness issues, more importantly, the issue of what the language needed to

[1] Fagin [17] named his notion just as *inverse* of a schema mapping. Since in this chapter we are introducing several 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 by Fagin [17].

express inverses is. In Section 6 we present a relaxation of the notions of recovery and maximum recovery based on certain answers, which gives alternative definitions of inverses when one is focused on retrieving information with a particular class of queries. In Section 7 we report on extensions to the previous notions that deal with incomplete information in source instances. Conclusions are presented in Section 8. We begin by giving a bit of general notation for the chapter.

Preliminary notions and notation

In the study of the inverse operator, we use a general notion of schema mapping (or just mapping in our context). We assume that a mapping from schema $\mathbf{S}$ to schema $\mathbf{T}$ is simply a set of pairs (I, J) where I is an instance of $\mathbf{S}$ and J is an instance of $\mathbf{T}$. As usual in data exchange, given a mapping $\mathcal{M}$ from $\mathbf{S}$ to $\mathbf{T}$ and an instance I of $\mathbf{S}$, we denote by $\text{Sol}_{\mathcal{M}}(I)$ the set of *solutions* for I under $\mathcal{M}$, that is $\text{Sol}_{\mathcal{M}}(I) = \{J \mid (I, J) \in \mathcal{M}\}$.

Notice that a mapping in this general setting is just a binary relation, and thus one can define some general operators over mappings that inherits from binary relations. One such particular operator that plays a crucial role in this chapter is mapping *composition*. Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$, and $\mathcal{M}'$ a mapping from $\mathbf{T}$ to $\mathbf{R}$. The composition of $\mathcal{M}$ and $\mathcal{M}'$, denoted by $\mathcal{M} \circ \mathcal{M}'$, is defined as the composition of binary relations, that is $\mathcal{M} \circ \mathcal{M}' = \{(I, K) \mid \text{there exists } J \text{ such that } (I, J) \in \mathcal{M} \text{ and } (J, K) \in \mathcal{M}'\}$ [33, 19].

We usually specify mappings by using logical languages. One particular language of special interest is the language of *source-to-target tuple-generating dependencies* (st-tgds) [18]. An st-tgd from $\mathbf{S}$ to $\mathbf{T}$ is a First-Order formula of the form

$$\forall \bar{x} \forall \bar{y} \big(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\ \psi(\bar{x}, \bar{z}) \big) \tag{1}$$

in which $\bar{x}$, $\bar{y}$ and $\bar{z}$ are tuple of variables, $\varphi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $\mathbf{S}$ (mentioning all the variables $\bar{x}$ and $\bar{y}$), and $\psi(\bar{x}, \bar{z})$ is a conjunction of relational atoms over $\mathbf{T}$ (mentioning all the variables $\bar{x}$ and $\bar{z}$). The left-hand side of the implication in formula (1) is called the *premise*, and the right-had side the *conclusion* of the st-tgd. For simplicity, we omit the universal quantifiers when writing st-tgds. That is, we just write $\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\ \psi(\bar{x}, \bar{z})$ for an st-tgd of the form (1). Given an st-tgd σ of the form $\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\ \psi(\bar{x}, \bar{z})$ from $\mathbf{S}$ to $\mathbf{T}$, and a pair (I, J) with I an instance of $\mathbf{S}$ and J an instance of $\mathbf{T}$, we say that (I, J) *satisfies* σ if for every pair of tuples $\bar{a}$, $\bar{b}$ such that I satisfies $\varphi(\bar{a}, \bar{b})$, there exists a tuple $\bar{c}$ such that J satisfies $\psi(\bar{a}, \bar{c})$.

We say that a mapping $\mathcal{M}$ from $\mathbf{S}$ to $\mathbf{T}$ is *specified* by a set Σ of st-tgds, if for every pair of instances I of $\mathbf{S}$ and J of $\mathbf{T}$ we have that $(I, J) \in \mathcal{M}$ if and only if (I, J) satisfies every st-tgd in Σ. We consider two types of values when defining mappings, constant and null values, and we assume the existence of a special predicate $\mathbf{C}(\cdot)$ to differentiate them. In particular, $\mathbf{C}(u)$ holds if and only if u is a constant value. As is usual in the data exchange context [18], when defining a mapping from $\mathbf{S}$ to $\mathbf{T}$ specified by st-tgds we assume that source instances (instances of $\mathbf{S}$) contain only constant values, while target instances (instances of $\mathbf{T}$) may contain constant and null values. Notice that an inverse of $\mathcal{M}$ is a mapping $\mathcal{M}'$ from $\mathbf{T}$ to $\mathbf{S}$, and thus $\mathcal{M}'$ has constant and null values in its source schema (schema $\mathbf{T}$), while only constants in its target schema (schema $\mathbf{S}$). In Section 7 we drop this assumption, and study inversion of mappings that may contain constant and nulls in source and target instances.

2 Fagin-inverse

The first notion of inverse in the literature was proposed by Fagin [16]. This notion is based on the algebraic intuition that a mapping composed with its inverse should be equal to the *identity*. Since we can unambiguously define the composition of two schema mappings (based on the composition of binary relations), we only needed to define a notion of identity for schema mappings.

Fagin was specially interested in defining an inverse for mappings specified by st-tgds thus, he defined an intuitive identity in terms of st-tgds as follows. Let $\mathbf{S}$ be a schema, and $\hat{\mathbf{S}} = \{\hat{R} \mid R \in \mathbf{S}\}$, that is, $\hat{\mathbf{S}}$ is a copy of $\mathbf{S}$. The set of *copying* st-tgds over $\mathbf{S}$ is defined as

$$\Sigma_{\mathbf{S}\text{-copy}} = \{\ R(x_1, \ldots, x_k) \to \hat{R}(x_1, \ldots, x_k) \mid R \text{ is a } k\text{-ary relation symbol in } \mathbf{S}\}.$$

The idea is that $\Sigma_{\mathbf{S}\text{-copy}}$ essentially copies every source relation from the source to the target. Notice that we need to use $\hat{\mathbf{S}} = \{\hat{R} \mid R \in \mathbf{S}\}$ in the definition of $\Sigma_{\mathbf{S}\text{-copy}}$ and not simply $\mathbf{S}$ since, otherwise, the semantics of the tgds would be trivial. Consider now the mapping $\mathcal{M}_{\mathbf{S}\text{-copy}}$ from $\mathbf{S}$ to $\hat{\mathbf{S}}$, which is specified by $\Sigma_{\mathbf{S}\text{-copy}}$. Given the definition of mapping $\mathcal{M}_{\mathbf{S}\text{-copy}}$, it its a natural identity in our context, and thus, the notion of Fagin-inverse is formulated as follows.

▶ **Definition 1** ([16]). Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$, and $\mathcal{M}'$ a mapping from $\mathbf{T}$ to $\hat{\mathbf{S}}$. Then $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$ if $\mathcal{M} \circ \mathcal{M}' = \mathcal{M}_{\mathbf{S}\text{-copy}}$.

Notice that in the above definition, a Fagin-inverse of a mapping from $\mathbf{S}$ to $\mathbf{T}$ is not a mapping from $\mathbf{T}$ to $\mathbf{S}$ but from $\mathbf{T}$ to $\hat{\mathbf{S}}$. This is because we were specially interested in defining the identity mapping with a set of st-tgds. But we can reformulate the above notion to use only schema $\mathbf{S}$. Let I, J be instances of $\mathbf{S}$ and $\hat{J}$ be a copy of J over schema $\hat{\mathbf{S}}$. Then we have that $(I, \hat{J}) \in \mathcal{M}_{\mathbf{S}\text{-copy}}$ if and only if $I \subseteq J$. Thus we can redefine the identity mapping as a mapping $\overline{\mathrm{Id}}_{\mathbf{S}}$ from $\mathbf{S}$ to $\mathbf{S}$ given by

$$\overline{\mathrm{Id}}_{\mathbf{S}} = \{(I, J) \mid I, J \text{ are instances of } \mathbf{S} \text{ and } I \subseteq J\}.$$

With this new identity mapping we can reformulate the notion of Fagin-inverse as follows.

▶ **Definition 2** ([16]). Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$, and $\mathcal{M}'$ a mapping from $\mathbf{T}$ to $\mathbf{S}$. Then $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$ if $\mathcal{M} \circ \mathcal{M}' = \overline{\mathrm{Id}}_{\mathbf{S}}$.

In the rest of the chapter we use Definition 2 for the notion of Fagin-inverse. Moreover, if mapping $\mathcal{M}$ has a Fagin-inverse, then we say that $\mathcal{M}$ is *Fagin-invertible*. It is important to notice that $\overline{\mathrm{Id}}_{\mathbf{S}}$ is not exactly the identity relation over instances of schema $\mathbf{S}$. In [17], Fagin formally justified the use of $\overline{\mathrm{Id}}_{\mathbf{S}}$ as the identity when inverting mappings specified by st-tgds, instead of the more natural $\mathrm{Id}_{\mathbf{S}} = \{(I, I) \mid I \text{ is an instance of } \mathbf{S}\}$. As Fagin proved, no composition of st-tgds can be equal to $\mathrm{Id}_{\mathbf{S}}$ (see Proposition 5.2 in [17]).

▶ **Example 3.** Let $\mathbf{S}$ be a source schema composed of a binary relation $A(\cdot, \cdot)$, and $\mathbf{T}$ a target schema with a ternary relation $B(\cdot, \cdot, \cdot)$. Consider now the mapping $\mathcal{M}$ from $\mathbf{S}$ to $\mathbf{T}$ specified by the st-tgd $A(x, y) \to B(x, x, y)$. Then the mapping $\mathcal{M}_1$ specified by $B(x, u, y) \to A(x, y)$ is a Fagin-inverse of $\mathcal{M}$.

To see why $\mathcal{M}_1$ is a Fagin-inverse of $\mathcal{M}$, assume that $(I, J) \in \mathcal{M} \circ \mathcal{M}_1$. We know that there exists an instance K of schema $\mathbf{T}$ such that $(I, K) \in \mathcal{M}$ and $(K, J) \in \mathcal{M}_1$. Then, if $A(a, b)$ is a fact in I with a and b arbitrary values, then $B(a, a, b)$ is a fact in K, which, by the definition of $\mathcal{M}_1$ implies that $A(a, b)$ is a fact in J. We have shown that every fact in I

is also a fact in J, and thus $I \subseteq J$. On the other hand, assume that $I \subseteq J$, and consider the instance L of $\mathbf{T}$ such that $B(a,a,b)$ is a fact in L if and only if $A(a,b)$ is a fact in I. Then it is straightforward that $(I,L) \in \mathcal{M}$ and $(L,J) \in \mathcal{M}_1$, which implies that $(I,J) \in \mathcal{M} \circ \mathcal{M}_1$. We have shown that $(I,J) \in \mathcal{M} \circ \mathcal{M}_1$ if and only if $I \subseteq J$, thus implying that $\mathcal{M} \circ \mathcal{M}_1 = \overline{\text{Id}}_{\mathbf{S}}$.

Consider now the mapping $\mathcal{M}_2$ specified by $B(u,x,y) \rightarrow A(x,y)$, and the mapping $\mathcal{M}_3$ specified by $B(x,x,y) \rightarrow A(x,y)$. Then both $\mathcal{M}_2$ and $\mathcal{M}_3$ are also Fagin-inverses of $\mathcal{M}$. This example shows that Fagin-inverses need not to be unique up to logical equivalence [17]. ◀

In the above example it was quite simple to obtain a Fagin-inverse. In fact, as mapping $\mathcal{M}_3$ shows, just *reversing the arrows* in the definition of $\mathcal{M}$ generates a Fagin-inverse. In the following examples, we show that Fagin-inverses are not always as easy to construct. In particular, Example 4 shows that reversing the arrows does not always produce a Fagin-inverse, and Example 5 shows that in some cases we need inequalities when specifying Fagin-inverses.

▶ **Example 4** ([17]). Consider a schema $\mathbf{S}$ with two unary relations $A(\cdot)$ and $B(\cdot)$, and a schema $\mathbf{T}$ with three unary relations $S(\cdot)$, $T(\cdot)$, and $U(\cdot)$. Let $\mathcal{M}$ be the mapping specified by the following set of st-tgds

$$\begin{array}{lcl} A(x) & \rightarrow & S(x) \\ A(x) & \rightarrow & T(x) \\ B(x) & \rightarrow & U(x) \\ B(x) & \rightarrow & T(x) \end{array}$$

Let $\mathcal{M}'$ be the mapping obtained from the specification of $\mathcal{M}$ by just reversing the arrows, that is, $\mathcal{M}'$ is specified by the set of tgds $S(x) \rightarrow A(x)$, $T(x) \rightarrow A(x)$, $U(x) \rightarrow B(x)$ and $T(x) \rightarrow B(x)$. It is easy to see that $\mathcal{M}'$ is not a Fagin-inverse of $\mathcal{M}$. Consider the instance $I = \{A(1)\}$. Then for every $K \in \text{Sol}_{\mathcal{M}}(I)$ we have that $T(1)$ is a fact in K. Thus, given that $T(x) \rightarrow B(x)$ is in the specification of $\mathcal{M}'$, we have that for every $J \in \text{Sol}_{\mathcal{M}'}(K)$ it holds that $B(1)$ is a fact in J. This implies that for every J such that $(I,J) \in \mathcal{M} \circ \mathcal{M}'$ it holds that $B(1)$ is a fact in J, and thus, $(I,I) \notin \mathcal{M} \circ \mathcal{M}'$, which shows that $\mathcal{M} \circ \mathcal{M}' \neq \overline{\text{Id}}_{\mathbf{S}}$.

The problem in this case is that dependencies $A(x) \rightarrow T(x)$ and $B(x) \rightarrow T(x)$ are somehow *mixing* the data of relations A and B in target relation T. Thus, to obtain a Fagin-inverse of $\mathcal{M}$, we cannot use T to recover the data of relations A and B. In fact, a Fagin-inverse of $\mathcal{M}$ can be constructed by using only target relations S and U as follows:

$$\begin{array}{lcl} S(x) & \rightarrow & A(x) \\ U(x) & \rightarrow & B(x) \end{array}$$

It can be easily shown that the mapping defined by the above dependencies is a Fagin-inverse of $\mathcal{M}$. ◀

▶ **Example 5** ([21]). Consider a schema $\mathbf{S}$ with a binary relation $A(\cdot,\cdot)$ and a unary relation $B(\cdot)$, and a schema $\mathbf{T}$ with a binary relation $S(\cdot,\cdot)$ and two unary relations $T(\cdot)$ and $U(\cdot)$. Let $\mathcal{M}$ be the mapping specified by the following set of st-tgds

$$\begin{array}{rcl} A(x,y) & \rightarrow & S(x,y) \\ B(x) & \rightarrow & S(x,x) \\ B(x) & \rightarrow & T(x) \\ A(x,x) & \rightarrow & U(x) \end{array}$$

Notice that in this case the mapping is translating tuples of the form $A(a, a)$ and $B(a)$ into the same target relation S, thus, as in Example 4, mapping $\mathcal{M}$ is somehow mixing the source information when translating it to the target. In this case we can solve this issue by using inequalities to specify a Fagin-inverse of $\mathcal{M}$. Consider the mapping $\mathcal{M}'$ specified by the following dependencies

$$\begin{array}{rcl} S(x,y) \wedge x \neq y & \rightarrow & A(x,y) \\ T(x) & \rightarrow & B(x) \\ U(x) & \rightarrow & A(x,x) \end{array}$$

Then, it can be shown that $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$. In fact, Fagin et al. showed [20, 21] that inequalities are strictly needed to specify Fagin-inverses of mappings given by st-tgds (we make this statement precise in Section 5). ◀

On the existence of Fagin-inverses

As we explained in the introduction, a first important question to answer for every definition of inverse of schema mappings, is for which class of mappings the inverse is guaranteed to exist. As we show next, there are several mappings specified by st-tgds that do not admit Fagin-inverses.

Consider the following mappings specified by st-tgds (in every case, source and target schemas are implicit in the dependencies).

$$\begin{array}{lrcl} \mathcal{M}_1: & A(x,y) & \rightarrow & S(x) \\ \\ \mathcal{M}_2: & A(x,y) & \rightarrow & S(x) \wedge T(y) \\ \\ \mathcal{M}_3: & A(x) & \rightarrow & S(x) \\ & B(x) & \rightarrow & S(x) \end{array} \tag{2}$$

As pointed out by Fagin [17], Fagin-invertibility for a mapping intuitively coincide with *no loss of information*. Thus, considering this intuition, none of the above mappings should be Fagin-invertible. For instance, mapping $\mathcal{M}_1$ is only transferring the first component of relation A from source to target, and thus, we are losing the second component when transferring the source data. In the case of $\mathcal{M}_2$, although it is actually transferring both components of A from source to target, these components are being stored in independent relations in the target thus loosing the relationships that they had in the source. For $\mathcal{M}_3$ the problem is a little bit different. In this case all the data in both A and B is being transferred but, since all the information is stored in the same relation in the target, it is impossible to reconstruct the initial source instances.

The question is how to formally prove that the above mappings have no Fagin-inverses? To answer this, Fagin [16] proposed a very simple condition that a mapping specified by st-tgds needs to satisfy in order to have a Fagin-inverse. This property is called the *unique-solutions property* and is formalized as follows.

▶ **Definition 6** ([16]). A mapping $\mathcal{M}$ from $\mathbf{S}$ to $\mathbf{T}$ satisfies the *unique-solutions property* if for every pair of instances I_1, I_2 of $\mathbf{S}$, it holds that $\text{Sol}_{\mathcal{M}}(I_1) = \text{Sol}_{\mathcal{M}}(I_2)$ implies $I_1 = I_2$.

▶ **Theorem 7** ([16]). *Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$ specified by st-tgds. If $\mathcal{M}$ has a Fagin-inverse then $\mathcal{M}$ satisfies the unique-solutions property.*

The proof of the theorem is very simple. Assume that we have a mapping $\mathcal{M}$ and that $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$. Now let I_1 and I_2 be instances such that $\text{Sol}_{\mathcal{M}}(I_1) = \text{Sol}_{\mathcal{M}}(I_2)$. Since $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$, we know that $\mathcal{M} \circ \mathcal{M}' = \overline{\text{Id}}_{\mathbf{S}}$ and thus $(I_2, I_2) \in \mathcal{M} \circ \mathcal{M}'$. This implies that there exists an instance K such that $(I_2, K) \in \mathcal{M}$ and $(K, I_2) \in \mathcal{M}'$. Now, since $\text{Sol}_{\mathcal{M}}(I_1) = \text{Sol}_{\mathcal{M}}(I_2)$ and $K \in \text{Sol}_{\mathcal{M}}(I_2)$, we have that $(I_1, K) \in \mathcal{M}$ and then $(I_1, I_2) \in \mathcal{M} \circ \mathcal{M}' = \overline{\text{Id}}_{\mathbf{S}}$ which implies that $I_1 \subseteq I_2$. With a symmetric argument we can show that $I_2 \subseteq I_1$ and thus $I_1 = I_2$.

With this tool we can formally prove that the mappings in (2) have no Fagin-inverses. For the case of $\mathcal{M}_1$, consider instances $I_1 = \{A(1,2)\}$ and $I_2 = \{A(1,3)\}$. The instances are different but $\text{Sol}_{\mathcal{M}_1}(I_1) = \text{Sol}_{\mathcal{M}_2}(I_2)$. For the case of $\mathcal{M}_2$ we can use instances $I_1 = \{A(1,2), A(3,4)\}$ and $I_2 = \{A(1,4), A(3,2)\}$ which satisfy that $\text{Sol}_{\mathcal{M}_2}(I_1) = \text{Sol}_{\mathcal{M}_2}(I_2)$. For the mapping $\mathcal{M}_3$ and the instances $I_1 = \{A(1)\}$ and $I_2 = \{B(1)\}$, we have that $\text{Sol}_{\mathcal{M}_3}(I_1) = \text{Sol}_{\mathcal{M}_3}(I_2)$. Thus neither $\mathcal{M}_1$ nor $\mathcal{M}_2$ nor $\mathcal{M}_3$ satisfy the unique-solutions property which implies that they have no Fagin-inverse.

The natural question at this point is whether the unique-solutions property is also a sufficient condition to test Fagin-invertibility for mappings specified by st-tgds. It can be shown that it is not [21][2]. Fortunately, Fagin et al. [20] introduced another property, called the *subset property*, which characterizes Fagin-invertibility for the case of mappings specified by st-tgds.

▶ **Definition 8** ([20]). A mapping $\mathcal{M}$ from $\mathbf{S}$ to $\mathbf{T}$ satisfies the *subset property* if for every pair of instances I_1, I_2 of $\mathbf{S}$ we have that $\text{Sol}_{\mathcal{M}}(I_1) \subseteq \text{Sol}_{\mathcal{M}}(I_2)$ implies $I_2 \subseteq I_1$.

▶ **Theorem 9** ([20]). *Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$ specified by st-tgds. Then $\mathcal{M}$ has a Fagin-inverse if and only if $\mathcal{M}$ satisfies the subset property.*

We have shown that there are several mappings specified by st-tgds that have no Fagin-inverse, thus the question at this point is whether we can find some relaxed notions that can give natural and useful reverse mappings when Fagin-inverses do not exist. In the next two sections we introduce the notions of *quasi-inverse* [20, 21] and *maximum recovery* [10, 11] proposed to deal with this issue.

3 Quasi-inverse

As we have shown in the previous section, there are many simple mappings specified by st-tgds that do not possess Fagin-inverses. Nevertheless, in many cases there are very simple and natural ways of specifying useful reverse mappings. Thus, there is a need for a weaker notion of inverse to handle these cases. Towards solving this problem, Fagin et al. [20] proposed the notion of a quasi-inverse of a schema mapping.

Intuitively, the notion of quasi-inverse is obtained from the notion of Fagin-inverse by not differentiating between source instances that are equivalent for data-exchange purposes. Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$, and define the equivalence relation $\sim_{\mathcal{M}}$ between instances of $\mathbf{S}$ as follows: $I_1 \sim_{\mathcal{M}} I_2$ if and only if $\text{Sol}_{\mathcal{M}}(I_1) = \text{Sol}_{\mathcal{M}}(I_2)$. That is, I_1 and I_2 are considered equivalent if they have the same space of solutions under $\mathcal{M}$. For instance, for the mapping $\mathcal{M}$ specified by the st-tgds $A(x,y) \rightarrow S(x)$, and the instances $I_1 = \{A(1,2)\}$ and $I_2 = \{A(1,3)\}$, we have that $I_1 \sim_{\mathcal{M}} I_2$.

[2] Fagin proved that for LAV mappings, that is, mappings specified by st-tgds in which only one atom is mentioned in the premises of dependencies, the unique-solutions property is necessary and sufficient to characterize Fagin-invertibility [17].

Informally, $\mathcal{M}'$ is a quasi-inverse of $\mathcal{M}$ if the equation $\mathcal{M} \circ \mathcal{M}' = \overline{\mathrm{Id}}_{\mathbf{S}}$ holds *modulo* $\sim_{\mathcal{M}}$. To make this statement precise, let us introduce some notation. Let D be a binary relation over instances of a source schema $\mathbf{S}$ (that is, a mapping from $\mathbf{S}$ to $\mathbf{S}$), and let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to a schema $\mathbf{T}$. Then we define the relation $D[\sim_{\mathcal{M}}]$ as follows:

$$D[\sim_{\mathcal{M}}] = \{(I_1, I_2) \mid \text{ there exists } I_1' \text{ and } I_2' \text{ such that } I_1 \sim_{\mathcal{M}} I_1', I_2 \sim_{\mathcal{M}} I_2' \text{ and } (I_1', I_2') \in D\}.$$

Now we can formally introduce the notion of quasi-inverse of a schema mapping.

▶ **Definition 10** ([20]). Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$, and $\mathcal{M}'$ a mapping from $\mathbf{T}$ to $\mathbf{S}$. Then $\mathcal{M}'$ is a *quasi-inverse* of $\mathcal{M}$ if $(\mathcal{M} \circ \mathcal{M}')[\sim_{\mathcal{M}}] = \overline{\mathrm{Id}}_{\mathbf{S}}[\sim_{\mathcal{M}}]$.

▶ **Example 11.** Consider a source schema $\mathbf{S} = \{A(\cdot,\cdot)\}$ and a target schema $\mathbf{T} = \{S(\cdot)\}$, and let $\mathcal{M}$ be the mapping from $\mathbf{S}$ to $\mathbf{T}$ specified by $A(x, y) \to S(x)$. We showed in the previous section that $\mathcal{M}$ has no Fagin-inverse. Consider now the mapping $\mathcal{M}'$ specified by $S(x) \to \exists u\, A(x, u)$. We now show that $\mathcal{M}'$ is a quasi-inverse of $\mathcal{M}$. To see why this is the case, consider first the inclusion:

$$\overline{\mathrm{Id}}_{\mathbf{S}}[\sim_{\mathcal{M}}] \quad \subseteq \quad (\mathcal{M} \circ \mathcal{M}')[\sim_{\mathcal{M}}]. \tag{3}$$

If $(I_1, I_2) \in \overline{\mathrm{Id}}_{\mathbf{S}}[\sim_{\mathcal{M}}]$, then there exist instances I_1', I_2' of $\mathbf{S}$ such that $I_1 \sim_{\mathcal{M}} I_1'$, $I_2 \sim_{\mathcal{M}} I_2'$ and $(I_1', I_2') \in \overline{\mathrm{Id}}_{\mathbf{S}}$. Thus, we have that $I_1' \subseteq I_2'$. Let J_1' be an instance of $\mathbf{T}$ such that $S(a)$ is a fact in J_1' if and only if $A(a, b)$ is a fact in I_1' (for arbitrary values a and b). Then we have that $(I_1', J_1') \in \mathcal{M}$, and also that $(J_1', I_1') \in \mathcal{M}'$ by the definitions of $\mathcal{M}$ and $\mathcal{M}'$. Moreover, given that $I_1' \subseteq I_2'$ we have that (J_1', I_2') also satisfies the tgds defining $\mathcal{M}'$, and thus $(J_1', I_2') \in \mathcal{M}'$. Hence, we conclude that $(I_1', I_2') \in (\mathcal{M} \circ \mathcal{M}')$, which implies that $(I_1, I_2) \in (\mathcal{M} \circ \mathcal{M}')[\sim_{\mathcal{M}}]$ (since $I_1 \sim_{\mathcal{M}} I_1'$ and $I_2 \sim_{\mathcal{M}} I_2'$). Thus, we have shown that inclusion (3) holds, and it only remains to prove that the following inclusion holds:

$$(\mathcal{M} \circ \mathcal{M}')[\sim_{\mathcal{M}}] \quad \subseteq \quad \overline{\mathrm{Id}}_{\mathbf{S}}[\sim_{\mathcal{M}}]. \tag{4}$$

If $(I_1, I_2) \in (\mathcal{M} \circ \mathcal{M}')[\sim_{\mathcal{M}}]$, then there exist instances I_1', I_2' of $\mathbf{S}$ such that $I_1 \sim_{\mathcal{M}} I_1'$, $I_2 \sim_{\mathcal{M}} I_2'$ and $(I_1', I_2') \in (\mathcal{M} \circ \mathcal{M}')$. Thus, we have that there exists an instance K of $\mathbf{T}$ such that $(I_1', K) \in \mathcal{M}$ and $(K, I_2') \in \mathcal{M}'$. By the definitions of $\mathcal{M}$ and $\mathcal{M}'$ we conclude that for every fact $A(a, b)$ in I_1', there exists an element c such that $A(a, c)$ is a fact in I_2'. From this last property we conclude that the instance $I^\star = I_1' \cup I_2'$ is such that $I^\star \sim_{\mathcal{M}} I_2'$. Moreover, since $I_2 \sim_{\mathcal{M}} I_2'$ and $I_2' \sim_{\mathcal{M}} I^\star$, we have that $I_2 \sim_{\mathcal{M}} I^\star$. Notice that $I_1' \subseteq I^\star$, and thus we have that $I_1 \sim_{\mathcal{M}} I_1'$, $I_2 \sim_{\mathcal{M}} I^\star$ and $(I_1', I^\star) \in \overline{\mathrm{Id}}_{\mathbf{S}}$, which implies that $(I_1, I_2) \in \overline{\mathrm{Id}}_{\mathbf{S}}[\sim_{\mathcal{M}}]$. Thus, we have shown that (4) holds, which proves that $\mathcal{M}'$ is a quasi-inverse of $\mathcal{M}$. ◀

As the previous example shows, there are mappings that are not Fagin-invertible but have a quasi-inverse. This, plus the following result, show that the notion of quasi-inverse is a strict generalization of the notion of Fagin-inverse. In particular, the result shows that if a mapping has a Fagin-inverse, then the notions of Fagin-inverse and quasi-inverse coincide.

▶ **Theorem 12** ([20]). *Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$ specified by st-tgds, and assume that $\mathcal{M}$ has a Fagin-inverse. Then $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$ if and only if $\mathcal{M}'$ is a quasi-inverse of $\mathcal{M}$.*

It is not difficult to see why the theorem holds. Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$, and assume that $\mathcal{M}$ is specified by st-tgds and has a Fagin-inverse. If $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$ then $\mathcal{M} \circ \mathcal{M}' = \overline{\mathrm{Id}}_{\mathbf{S}}$, which implies that $(\mathcal{M} \circ \mathcal{M}')[\sim_{\mathcal{M}}] = \overline{\mathrm{Id}}_{\mathbf{S}}[\sim_{\mathcal{M}}]$, and thus $\mathcal{M}'$ is a quasi-inverse of $\mathcal{M}$. Assume now that $\mathcal{M}'$ is a quasi-inverse of $\mathcal{M}$, that is,

$(\mathcal{M} \circ \mathcal{M}')[\sim_{\mathcal{M}}] = \overline{\mathrm{Id}}_{\mathbf{S}}[\sim_{\mathcal{M}}]$. Given that $\mathcal{M}$ has a Fagin-inverse, from Theorem 7 we know that $\mathcal{M}$ satisfies the unique-solutions property. Thus, we have that for every pair of instances I_1, I_2 of $\mathbf{S}$, it holds that if $I_1 \sim_{\mathcal{M}} I_2$ (or equivalently $\mathrm{Sol}_{\mathcal{M}}(I_1) = \mathrm{Sol}_{\mathcal{M}}(I_2)$) then $I_1 = I_2$. Notice that this implies that $(\mathcal{M} \circ \mathcal{M}')[\sim_{\mathcal{M}}] = \mathcal{M} \circ \mathcal{M}'$ and $\overline{\mathrm{Id}}_{\mathbf{S}}[\sim_{\mathcal{M}}] = \overline{\mathrm{Id}}_{\mathbf{S}}$. Thus, since $(\mathcal{M} \circ \mathcal{M}')[\sim_{\mathcal{M}}] = \overline{\mathrm{Id}}_{\mathbf{S}}[\sim_{\mathcal{M}}]$ we obtain that $\mathcal{M} \circ \mathcal{M}' = \overline{\mathrm{Id}}_{\mathbf{S}}$ which implies that $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$.

On the existence of quasi-inverses

Consider the mappings in (2) in the previous section. In Example 11 we showed that for mapping $\mathcal{M}_1$ specified by $A(x,y) \to S(x)$, the mapping specified by $S(x) \to \exists u\ A(x,u)$ is a quasi-inverse. Consider $\mathcal{M}_2$, which is specified by $A(x,y) \to S(x) \wedge T(y)$. In this case it can be proved that the mapping specified by $S(x) \wedge T(y) \to \exists u\ A(x,u) \wedge \exists v\ A(v,y)$ is a quasi-inverse of $\mathcal{M}_2$. Moreover, for the case of mapping $\mathcal{M}_3$ specified by $A(x) \to S(x)$ and $B(x) \to S(x)$, it can be proved that $S(x) \to A(x) \vee B(x)$ is a quasi-inverse. Although none of these mappings admit a Fagin-inverse, all of them admit a quasi-inverse. This gives rise to the interesting question of whether every mapping specified by sets of st-tgds has a quasi-inverse. Unfortunately, it was shown by Fagin et al. [20, 21] that the answer is negative. To formalize this result we next introduce a property that characterizes when a mapping specified by st-tgds has a quasi-inverse. This property is obtained by relaxing the subset-property that characterizes Fagin-inverses.

▶ **Definition 13** ([20]). A mapping $\mathcal{M}$ from $\mathbf{S}$ to $\mathbf{T}$ satisfies the $(\sim_{\mathcal{M}})$*-subset property*, when for every pair I_1, I_2 of instances of $\mathbf{S}$, if $\mathrm{Sol}_{\mathcal{M}}(I_1) \subseteq \mathrm{Sol}_{\mathcal{M}}(I_2)$ then there exist instances I_1' and I_2' such that $I_1 \sim_{\mathcal{M}} I_1'$, $I_2 \sim_{\mathcal{M}} I_2'$ and $I_2' \subseteq I_1'$.

▶ **Theorem 14** ([20]). *Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$ specified by st-tgds. Then $\mathcal{M}$ has a quasi-inverse if and only if $\mathcal{M}$ satisfies the $(\sim_{\mathcal{M}})$-subset property.*

With the above tool we can show that there are mappings specified by st-tgds that do not have a quasi-inverse.

▶ **Example 15** ([20, 21]). Consider a source schema $\mathbf{S}$ consisting of a binary relation $A(\cdot,\cdot)$, a target schema $\mathbf{T}$ consisting of a binary relation $S(\cdot,\cdot)$ and a unary relation $T(\cdot)$, and the mapping $\mathcal{M}$ from $\mathbf{S}$ to $\mathbf{T}$ specified by the st-tgd

$$A(x,z) \wedge A(z,y) \quad \to \quad S(x,y) \wedge T(z). \tag{5}$$

Fagin et al. showed [20, 21] that $\mathcal{M}$ does not satisfy the $(\sim_{\mathcal{M}})$-subset property, from which follows that $\mathcal{M}$ has no quasi-inverse. We next show why $\mathcal{M}$ does not satisfy the $(\sim_{\mathcal{M}})$-subset property.

Let I_1, I_2 be instances of $\mathbf{S}$ such that:

$$\begin{aligned} I_1 &= \{A(1,4), A(4,3), A(1,2), A(2,5), A(4,2)\} \\ I_2 &= \{A(1,2), A(2,3)\} \end{aligned}$$

Moreover, let J_1, J_2 be the following instances over $\mathbf{T}$:

$$\begin{aligned} J_1 &= \{S(1,3), S(1,2), S(1,5), S(4,5), T(2), T(4)\} \\ J_2 &= \{S(1,3), T(2)\} \end{aligned}$$

That is, J_1 and J_2 are the *canonical solutions* [18, 3] of I_1 and I_2, respectively. In this case, given the definition of $\mathcal{M}$, it is not difficult to see that they are also *minimal* and characterizes their space of solutions. For instance, we have that $K \in \text{Sol}_{\mathcal{M}}(I_1)$ if and only if $J_1 \subseteq K$. Similarly for I_2 we have that $K \in \text{Sol}_{\mathcal{M}}(I_2)$ if and only if $J_2 \subseteq K$. Thus, given that $J_2 \subseteq J_1$, we conclude that $\text{Sol}_{\mathcal{M}}(I_1) \subseteq \text{Sol}_{\mathcal{M}}(I_2)$. Next we show that there are no instances I_1', I_2' of $\mathbf{S}$ such that $I_1 \sim_{\mathcal{M}} I_1'$, $I_2 \sim_{\mathcal{M}} I_2'$ and $I_2' \subseteq I_1'$, which implies that $\mathcal{M}$ does not satisfy the $(\sim_{\mathcal{M}})$-subset property.

For the sake of contradiction, assume that there exist instances I_1', I_2' of $\mathbf{S}$ such that $I_1 \sim_{\mathcal{M}} I_1'$, $I_2 \sim_{\mathcal{M}} I_2'$ and $I_2' \subseteq I_1'$, and let J_1', J_2' be the canonical solutions for I_1' and I_2' under $\mathcal{M}$, respectively. That is

$$\begin{aligned} J_1' &= \{S(a,b) \mid \text{ there exists } c \text{ s.t. } A(a,c), A(c,b) \in I_1'\} \cup \\ & \{T(c) \mid \text{ there exist } a,b \text{ s.t. } A(a,c), A(c,b) \in I_1'\}, \\ J_2' &= \{S(a,b) \mid \text{ there exists } c \text{ s.t. } A(a,c), A(c,b) \in I_2'\} \cup \\ & \{T(c) \mid \text{ there exist } a,b \text{ s.t. } A(a,c), A(c,b) \in I_2'\}, \end{aligned}$$

Given that $I_2 \sim_{\mathcal{M}} I_2'$, we have that $\text{Sol}_{\mathcal{M}}(I_2) = \text{Sol}_{\mathcal{M}}(I_2')$ and, therefore, $J_2 = J_2'$ by the definition of $\mathcal{M}$. Thus, given that $S(1,3) \in J_2$, we have that $S(1,3) \in J_2'$ and, hence, there exists an element m such that $A(1,m), A(m,3) \in I_2'$. Notice that this implies that $T(m)$ should be a fact in J_2' and then since $J_2' = J_2$, we obtain that m must be equal to 2. Thus, we have that $A(1,2), A(2,3) \in I_2'$. Now given that $I_2' \subseteq I_1'$, we conclude that:

$$A(2,3) \quad \in \quad I_1'. \tag{6}$$

Given that $I_1 \sim_{\mathcal{M}} I_1'$, we have that $\text{Sol}_{\mathcal{M}}(I_1) = \text{Sol}_{\mathcal{M}}(I_1')$ and, therefore, $J_1 = J_1'$ by the definition of $\mathcal{M}$. Thus, given that $S(4,5) \in J_1$, we have that $S(4,5) \in J_1'$ and, hence, there exists an element n such that $A(4,n), A(n,5) \in I_1'$. Notice that this implies that $T(n)$ should be a fact in J_1' and then since $J_1' = J_1$, we obtain that n must be equal to either 2 or 4. We show that in both cases we obtain a contradiction. Assume that $n = 4$, then we have that $A(4,4) \in I_1'$ implying that $S(4,4) \in J_1'$ which leads to a contradiction since $J_1 = J_1'$ and $S(4,4) \notin J_1$. Assume that $n = 2$, then $A(4,2), A(2,5) \in I_1'$. But we know from (6) that $A(2,3) \in I_1'$ concluding that $S(4,3) \in J_1'$ (since $A(4,2) \in I_1'$), from which we obtain a contradiction since $J_1 = J_2'$ and $S(4,3) \notin J_1$. ◀

Although numerous non-Fagin-invertible schema mappings possess natural and useful quasi-inverses, the previous example shows that there still exist simple mappings specified by st-tgds that have no quasi-inverse. This leaves as an open problem the issue of finding a notion of inversion for st-tgds which ensures that every mapping in this class is invertible. This is the main motivation for the introduction of the notion of inversion discussed in the following section.

4 Recovery and Maximum Recovery

In this section we introduce the notions of recovery and maximum recovery proposed by Arenas et al. [10] as alternative notions for inverting mappings. As we show in this section, the notion of maximum recovery strictly generalizes the notion of Fagin-inverses, but has the desirable property that every mapping specified by st-tgds admits a maximum recovery, thus solving the open problem left by the notion of quasi-inverse.

Let us start by considering the mapping $\mathcal{M}$ in Example 15, that is, $\mathcal{M}$ is specified by $A(x,z) \wedge A(z,y) \rightarrow S(x,y) \wedge T(z)$. Notice that although mapping $\mathcal{M}$ does not have a quasi-inverse, there is a very natural reverse mapping in this case. Consider the mapping

$\mathcal{M}'$ defined by the tgds

$$\begin{array}{rcl} S(x,y) & \rightarrow & \exists u\ \big(A(x,u) \wedge A(u,y)\big) \\ T(z) & \rightarrow & \exists v \exists w\ \big(A(v,z) \wedge A(z,w)\big) \end{array}$$

$\mathcal{M}'$ is essentially doing its *best effort* to recover the data initially stored in the source schema. This is the main intuition behind the notions of recovery and maximum recovery. Intuitively, a recovery of $\mathcal{M}$ is a mapping that is capable of recovering *sound data* with respect to $\mathcal{M}$, and a maximum recovery of $\mathcal{M}$ is a mapping that is capable to recover *the maximum amount of sound data* with respect to $\mathcal{M}$. We next formalize both notions.

Let $\mathcal{M}$ be a mapping from a schema $\mathbf{S}$ to a schema $\mathbf{T}$, and $\text{Id}_{\mathbf{S}}$ the *identity mapping* over $\mathbf{S}$, that is,

$$\text{Id}_{\mathbf{S}} = \{(I,I) \mid I \text{ is an instance of } \mathbf{S}\}.$$

Notice the difference between $\overline{\text{Id}}_{\mathbf{S}}$ and $\text{Id}_{\mathbf{S}}$; mapping $\text{Id}_{\mathbf{S}}$ is the classical identity of binary relations. When trying to invert $\mathcal{M}$, the ideal would be to find a mapping $\mathcal{M}'$ from $\mathbf{T}$ to $\mathbf{S}$ such that, $\mathcal{M} \circ \mathcal{M}' = \text{Id}_{\mathbf{S}}$. If such a mapping exists, we know that if we use $\mathcal{M}$ to exchange data, the application of $\mathcal{M}'$ gives as result exactly the initial source instance. Unfortunately, in most cases this ideal is impossible to reach. For example, it is impossible to obtain such an inverse if $\mathcal{M}$ is specified by a set of st-tgds [16]. The main problem with such an ideal definition of inverse is that, in general, no matter what $\mathcal{M}'$ we choose, we will have not one but many solutions for a source instance under $\mathcal{M} \circ \mathcal{M}'$.

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

▶ **Definition 16** ([10])**.** Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$ and $\mathcal{M}'$ a mapping from $\mathbf{T}$ to $\mathbf{S}$. Then $\mathcal{M}'$ is a *recovery* of $\mathcal{M}$ if $(I,I) \in \mathcal{M} \circ \mathcal{M}'$ for every instance I of $\mathbf{S}$.

▶ **Example 17.** Let $\mathbf{S} = \{A(\cdot,\cdot)\}$, and $\mathbf{T} = \{S(\cdot,\cdot), T(\cdot)\}$. Consider the mapping $\mathcal{M}$ in Example 15, that is $\mathcal{M}$ is a mapping from $\mathbf{S}$ to $\mathbf{T}$ specified by the following st-tgd:

$$A(x,z) \wedge A(z,y) \quad \rightarrow \quad S(x,y) \wedge T(z), \tag{7}$$

Let $\mathcal{M}_1$ be a mapping from $\mathbf{T}$ to $\mathbf{S}$ specified by tgd:

$$S(x,y) \quad \rightarrow \quad \exists u \big(A(x,u) \wedge A(u,y)\big).$$

It is straightforward to prove that $\mathcal{M}_1$ is a recovery of $\mathcal{M}$. Let I be an arbitrary instance of $\mathbf{S}$, and J the *canonical solution* [18] for I, that is,

$$\begin{array}{rcl} J & = & \{S(a,b) \mid \text{ there exists } c \text{ s.t. } A(a,c), A(c,b) \in I\} \ \cup \\ & & \{T(c) \mid \text{ there exist } a,b \text{ s.t. } A(a,c), A(c,b) \in I\}. \end{array}$$

Then in this case we have that $(I,J) \in \mathcal{M}$ and $(J,I) \in \mathcal{M}_1$, from which we conclude that $(I,I) \in \mathcal{M} \circ \mathcal{M}_1$. This implies that $\mathcal{M}_1$ is a recovery of $\mathcal{M}$. Similarly, if $\mathcal{M}_2$ is a mapping from $\mathbf{T}$ to $\mathbf{S}$ specified by tgd:

$$T(z) \quad \rightarrow \quad \exists v \exists w \big(A(v,z) \wedge A(z,w)\big),$$

then we also have that $\mathcal{M}_2$ is a recovery of $\mathcal{M}$. On the other hand, if $\mathcal{M}_3$ is a mapping from $\mathbf{T}$ to $\mathbf{S}$ specified by tgd:

$$S(x,y) \wedge T(z) \quad \rightarrow \quad A(x,z) \wedge A(z,y), \tag{8}$$

then we have that $\mathcal{M}_3$ is not a recovery of $\mathcal{M}$. To see why this is the case, consider the instance $I = \{A(1,1), A(2,2)\}$. Next we show that $(I,I) \notin \mathcal{M} \circ \mathcal{M}_3$. ForL the sake of contradiction, assume that $(I,I) \in \mathcal{M} \circ \mathcal{M}_3$, and let K be an instance such that $(I,K) \in \mathcal{M}$ and $(K,I) \in \mathcal{M}_3$. Given that (I,K) satisfies st-tgd (7), we have that $S(1,1), S(2,2)$ and $T(1), T(2)$ are facts in K. But then given that (K,I) satisfies tgd (8), we conclude that $A(1,2)$, and $A(2,1)$ are facts in I, which is a contradiction. ◀

Being a recovery is a sound but mild requirement. Indeed, a schema mapping $\mathcal{M}$ from $\mathbf{S}$ to $\mathbf{T}$ always has as recoveries, for example, mappings $\mathcal{M}_1 = \{(J,I) \mid J$ is an instance of $\mathbf{T}$ and I is an instance of $\mathbf{S}\}$, and $\mathcal{M}_2 = \mathcal{M}^{-1} = \{(J,I) \mid (I,J) \in \mathcal{M}\}$. If one has to choose between $\mathcal{M}_1$ and $\mathcal{M}_2$ as a recovery of $\mathcal{M}$, then one would probably choose $\mathcal{M}_2$ since the space of possible solutions for a source instance I under $\mathcal{M} \circ \mathcal{M}_2$ is smaller than under $\mathcal{M} \circ \mathcal{M}_1$. In fact, if there exists a mapping $\mathcal{M}_3$ such that $\mathcal{M} \circ \mathcal{M}_3 = \mathrm{Id}_{\mathbf{S}}$, then one would definitely prefer $\mathcal{M}_3$ over $\mathcal{M}_1$ and $\mathcal{M}_2$.

In general, if $\mathcal{M}'$ is a recovery of $\mathcal{M}$, then the smaller the space of solutions generated by $\mathcal{M} \circ \mathcal{M}'$, the more informative $\mathcal{M}'$ is about the initial source instances. This notion induces an *order* among recoveries. If $\mathcal{M}_1$ and $\mathcal{M}_2$ are recoveries of $\mathcal{M}$ and $\mathcal{M} \circ \mathcal{M}_2 \subseteq \mathcal{M} \circ \mathcal{M}_1$ then we say that $\mathcal{M}_2$ is *more(-or-equally) informative* than $\mathcal{M}_1$ as a recovery of $\mathcal{M}$. This naturally gives rise to the notion of maximum recovery. If for a mapping $\mathcal{M}$ there exists a recovery $\mathcal{M}'$ that is more informative than any other recovery of $\mathcal{M}$, then $\mathcal{M}'$ is the best option to bring exchanged data back, among all the recoveries. Intuitively, such a mapping $\mathcal{M}'$ recovers the maximum amount of sound information. Such a mapping $\mathcal{M}'$ is called a maximum recovery of $\mathcal{M}$.

▶ **Definition 18** ([10]). Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$, and $\mathcal{M}'$ a mapping from $\mathbf{T}$ to $\mathbf{S}$. Then $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$ if:
1. $\mathcal{M}'$ is a recovery of $\mathcal{M}$, and
2. for every recovery $\mathcal{M}''$ of $\mathcal{M}$, it holds that $\mathcal{M} \circ \mathcal{M}' \subseteq \mathcal{M} \circ \mathcal{M}''$.

Notice that the definition of maximum recovery implies a quantification over all the possible recoveries of a mapping $\mathcal{M}$. Thus, the process of proving that a particular mapping is indeed a maximum recovery of $\mathcal{M}$ seems to be very a difficult task (compare it with the definitions of Fagin-inverse, quasi-inverse and recovery). Fortunately, Arenas et al. [11] provide a toolbox to deal with maximum recoveries. In particular, the following general characterization is useful to prove that a mapping is a maximum recovery of another mapping.

▶ **Theorem 19** ([11]). *Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$ and $\mathcal{M}'$ a mapping from $\mathbf{T}$ to $\mathbf{S}$. Then $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$ if and only if $\mathcal{M} = \mathcal{M} \circ \mathcal{M}' \circ \mathcal{M}$.*

▶ **Example 20.** Let $\mathbf{S} = \{A(\cdot,\cdot)\}$, and $\mathbf{T} = \{S(\cdot,\cdot), T(\cdot)\}$. Consider again the mapping $\mathcal{M}$ in Example 15 specified by:

$$A(x,z) \wedge A(z,y) \quad \rightarrow \quad S(x,y) \wedge T(z),$$

and let $\mathcal{M}'$ be the mapping from $\mathbf{T}$ to $\mathbf{S}$ specified by the following tgds:

$$\begin{aligned} S(x,y) \quad &\rightarrow \quad \exists u \, \big(A(x,u) \wedge A(u,y)\big), \\ T(z) \quad &\rightarrow \quad \exists v \exists w \, \big(A(u,z) \wedge A(z,w)\big), \end{aligned}$$

Next we use Theorem 19 to show that $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$. Given that $\mathcal{M}'$ is a recovery of $\mathcal{M}$ (see Example 17), we have that $\mathcal{M} \subseteq \mathcal{M} \circ \mathcal{M}' \circ \mathcal{M}$. Thus, by using Theorem 19, in order to show that $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$, we only need to show that $\mathcal{M} \circ \mathcal{M}' \circ \mathcal{M} \subseteq \mathcal{M}$.

Let $(I, J) \in \mathcal{M} \circ \mathcal{M}' \circ \mathcal{M}$. To prove that $(I, J) \in \mathcal{M}$, we need to show that (I, J) satisfies the st-tgd that specifies $\mathcal{M}$. Let $A(a, b)$ and $A(b, c)$ be facts in I, with a, b, c arbitrary elements. Then we need to prove that $S(a, c), T(b) \in J$. To prove this, first notice that given that $(I, J) \in \mathcal{M} \circ \mathcal{M}' \circ \mathcal{M}$, there exist instances K of $\mathbf{T}$ and L of $\mathbf{S}$ such that $(I, K) \in \mathcal{M}$, $(K, L) \in \mathcal{M}'$ and $(L, J) \in \mathcal{M}$. Thus, given that $A(a, b), A(b, c) \in I$ and $(I, K) \in \mathcal{M}$, we conclude that $S(a, c), T(b) \in K$. Hence, from the definition of $\mathcal{M}'$ and the fact that $(K, L) \in \mathcal{M}'$, we conclude that there exist elements d, e and f such that

$$A(a, d), A(d, c), A(e, b), A(b, f) \quad \in \quad L.$$

Therefore, given that $(L, J) \in \mathcal{M}$, we conclude that $S(a, c), T(b) \in J$ which was to be shown. ◀

As for the case of the quasi-inverse, it can be shown that the notion of maximum recovery strictly generalizes the notion of Fagin-inverse for mappings specified by st-tgds.

▶ **Theorem 21** ([10]). *Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$ specified by st-tgds, and assume that $\mathcal{M}$ has a Fagin-inverse. Then, $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$ if and only if $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$.*

The relationship between maximum recoveries and quasi-inverses is a little bit more complicated and is given in the following result.

▶ **Theorem 22** ([10]). *Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$ specified by st-tgds, and assume that $\mathcal{M}$ has a quasi-inverse.*

1. *If $\mathcal{M}'$ is a maximum-recovery of $\mathcal{M}$ then $\mathcal{M}'$ is a quasi-inverse of $\mathcal{M}$.*
2. *If $\mathcal{M}'$ is a quasi-inverse and a recovery of $\mathcal{M}$, then $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$.*

On the existence of maximum recoveries

One of the main results regarding maximum recoveries is that they exist for every mapping specified by st-tgds [10]. To show this, we next introduce the notion of *witness solution* that can be used to characterize when a general mapping has a maximum recovery.

▶ **Definition 23** ([10]). Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$ and I an instance of $\mathbf{S}$. Then an instance $J \in \mathrm{Sol}_{\mathcal{M}}(I)$ is a *witness solution* for I under $\mathcal{M}$, if for every other instance I' such that $J \in \mathrm{Sol}_{\mathcal{M}}(I')$ it holds that $\mathrm{Sol}_{\mathcal{M}}(I) \subseteq \mathrm{Sol}_{\mathcal{M}}(I')$.

Witness solutions are in a sense identifiers for spaces of solutions. In particular, if there are two instances I_1 and I_2 that share a witness solution, then $\mathrm{Sol}_{\mathcal{M}}(I_1) = \mathrm{Sol}_{\mathcal{M}}(I_2)$. Arenas et al. [10], proved the following general characterization of the existence of maximum recoveries.

▶ **Theorem 24** ([10]). *Let $\mathcal{M}$ be a general mapping from $\mathbf{S}$ to $\mathbf{T}$ (not necessarily specified by st-tgds). Then $\mathcal{M}$ has a maximum recovery if and only if every instance I of $\mathbf{S}$ has a witness solution under $\mathcal{M}$.*

It should be noticed that as opposed to the characterizations for the existence of Fagin-inverses and quasi-inverse shown in Theorems 9 and 14, respectively, the characterization for maximum recoveries can be applied to general mappings, not necessarily specified by st-tgds. We can now use Theorem 24 to show that mappings specified my st-tgds always have maximum recovery. For this we need to recall the notion of *universal solutions* in data exchange [18]. Universal solutions were introduced as desirable solutions for data exchange. Essentially, a universal solution for an instance I under a mapping $\mathcal{M}$ is, in a precise sense, the most general solution for I and can be *embedded* in any other solution for I [18]. In particular, for mappings specified by st-tgds, universal solutions can be obtained by using the *chase* procedure (see Chapter 1, for a comprehensive study of the chase procedure). In our context, the two most important properties of universal solutions are stated in the following lemma.

▶ **Lemma 25** ([10, 18]). *Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$ specified by st-tgds.*
1. *If J is a universal solution for I under $\mathcal{M}$ then J is a witness solution for I under $\mathcal{M}$.*
2. *For every instance I of $\mathbf{S}$ there exists a universal solution for I under $\mathcal{M}$.*

Then from Theorem 24 and Lemma 25 we obtain the following.

▶ **Corollary 26** ([10]). *Every mapping $\mathcal{M}$ specified by st-tgds has a maximum recovery.*

5 Computing Inverses

Up to this point we have presented three alternative notions for inverting mappings. For every one of them we have discussed their formal definitions, characterizations and the existence problem. But we have not discussed the most important practical problem regarding inverses of schema mappings: how to compute an inverse. In this section we present a general algorithm that can be used to compute all the notions of inverses introduced so far. We also discuss expressiveness issues. In particular, what is the language needed to express these inverses which is directly related to the language used in the output of the algorithm.

5.1 An algorithm for inverting mappings

The algorithm presented in this section is based on *query rewriting* and thus we first introduce the necessary terminology and some preliminary results. A fundamental notion in this section is the notion of *certain answers*. Given a mapping $\mathcal{M}$ from $\mathbf{S}$ to $\mathbf{T}$, a source instance I, and a query $Q_{\mathbf{T}}$ over $\mathbf{T}$, the set of *certain answers* of $Q_{\mathbf{T}}$ over I, denoted by $\text{certain}_{\mathcal{M}}(Q_{\mathbf{T}}, I)$ is the set

$$\text{certain}_{\mathcal{M}}(Q_{\mathbf{T}}, I) = \bigcap_{J \in \text{Sol}_{\mathcal{M}}(I)} Q_{\mathbf{T}}(J).$$

That is, a tuple $\bar{a}$ is a certain answer if $\bar{a} \in Q_{\mathbf{T}}(J)$ for every solution J of I. With the notion of certain answers we can define the notion of *source rewritability*. Given a mapping $\mathcal{M}$ from $\mathbf{S}$ to $\mathbf{T}$, and a query $Q_{\mathbf{T}}$ over $\mathbf{T}$, we say that $Q_{\mathbf{S}}$ over $\mathbf{S}$ is a *source rewriting* of $Q_{\mathbf{T}}$ under $\mathcal{M}$ if for every instance I of $\mathbf{S}$ it holds that

$$Q_{\mathbf{S}}(I) = \text{certain}_{\mathcal{M}}(Q_{\mathbf{T}}, I).$$

That is, if $Q_{\mathbf{S}}$ is a source rewriting of $Q_{\mathbf{T}}$, in order to compute the certain answers of $Q_{\mathbf{T}}$ one only needs to compute $Q_{\mathbf{S}}(I)$.

The computation of a source 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, 15, 1, 35] and, in particular, in the data integration context [24, 25, 29]. It can be shown that given a mapping $\mathcal{M}$ from $\mathbf{S}$ to $\mathbf{T}$ specified by a set of st-tgds, and a conjunctive query $Q_{\mathbf{T}}$ over $\mathbf{T}$, then a rewriting of $Q_{\mathbf{T}}$ over the source can always be expressed as a union of conjunctive queries with equality predicates (UCQ$^=$). As an example, consider a mapping given by the following tgds:

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

and let $Q_{\mathbf{T}}$ be the target query given by formula $S(x,y)$. Then a rewriting of $Q_{\mathbf{T}}$ over the source is given by $A(x,y) \vee (B(x) \wedge x = y)$, which is a query in UCQ$^=$. Notice that in this rewriting, we do need disjunction and the equality $x = y$. Moreover, it is known that source rewritings of conjunctive queries can be computed in exponential time. We formalize the above discussion in the following lemma.

▶ **Lemma 27** ([11])**.** *There exists a procedure* Source-Rew *that given a set Σ of st-tgds from $\mathbf{S}$ to $\mathbf{T}$, and a conjunctive query $Q_{\mathbf{T}}$ over $\mathbf{T}$, computes (in exponential time) a query in* UCQ$^=$ *which is a source rewriting of $Q_{\mathbf{T}}$ under the mapping $\mathcal{M}$ specified by Σ.*

The following algorithm, proposed in [11], uses procedure Source-Rew to compute inverses. In particular, the algorithm computes a maximum recovery of the input mapping. In the algorithm we use the special predicate $\mathbf{C}(\cdot)$ that differentiates constant values from labeled null values (that is $\mathbf{C}(u)$ holds if and only if u is a constant value). We also use $\mathbf{C}(\bar{x})$, with $\bar{x}$ a tuple of variables $(x_1, \ldots, x_k)$, as a shorthand of $\mathbf{C}(x_1) \wedge \cdots \wedge \mathbf{C}(x_k)$.

```
Algorithm Inverse
Input: A mapping $\mathcal{M}$ from $\mathbf{S}$ to $\mathbf{T}$ specified by a set $\Sigma$ of st-tgds.
Output: A mapping $\mathcal{M}'$ from $\mathbf{T}$ to $\mathbf{S}$ specified by a set $\Sigma'$ of tgds with disjunctions, equalities
and predicate $\mathbf{C}(\cdot)$.
1. Start with $\Sigma'$ as the empty set.
2. For every st-tgd $\varphi(\bar{x}) \rightarrow \exists \bar{y}\, \psi(\bar{x}, \bar{y})$ in $\Sigma$, do the following:
   a. Let $Q_{\mathbf{T}}$ be the conjunctive query defined by formula $\exists \bar{y}\, \psi(\bar{x}, \bar{y})$.
   b. Use Source-Rew to compute a formula $\alpha(\bar{x})$ in UCQ$^=$ that is a source rewriting of
      $Q_{\mathbf{T}}$ under mapping $\mathcal{M}$.
   c. Add dependency $\exists \bar{y}\, \psi(\bar{x}, \bar{y}) \wedge \mathbf{C}(\bar{x}) \rightarrow \alpha(\bar{x})$ to $\Sigma'$.
3. Return the mapping $\mathcal{M}'$ from $\mathbf{T}$ to $\mathbf{S}$ specified by $\Sigma'$.    ◀
```

▶ **Example 28.** Let $\mathbf{S} = \{A(\cdot,\cdot), B(\cdot)\}$, and $\mathbf{T} = \{S(\cdot,\cdot)\}$, and let $\mathcal{M}$ be the mapping for $\mathbf{S}$ to $\mathbf{T}$ specified by the st-tgds

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

With input $\mathcal{M}$, algorithm Inverse first considers the st-tgd $A(x,y) \rightarrow S(x,y)$ and computes a source rewriting of $S(x,y)$. From the discussion previous to the algorithm we know that $A(x,y) \vee (B(x) \wedge x = y)$ is a source rewriting of $S(x,y)$. Thus the algorithm includes in Σ' the dependency $S(x,y) \wedge \mathbf{C}(x) \wedge \mathbf{C}(y) \rightarrow A(x,y) \vee (B(x) \wedge x = y)$. Then the algorithm considers dependency $B(x) \rightarrow S(x,x)$ and computes a source rewriting of $S(x,x)$ which is given by the source query $A(x,x) \vee B(x)$. Then the algorithm includes dependency

$S(x,x) \wedge \mathbf{C}(x) \to A(x,x) \vee B(x)$ in Σ'. Finally, the output of the algorithm is the mapping $\mathcal{M}'$ specified by the dependencies

$$\begin{aligned} S(x,y) \wedge \mathbf{C}(x) \wedge \mathbf{C}(y) &\to A(x,y) \vee (B(x) \wedge x = y), \\ S(x,x) \wedge \mathbf{C}(x) &\to A(x,x) \vee B(x). \end{aligned}$$

◀

▶ **Theorem 29** ([6, 11]). *Let $\mathcal{M}$ be a mapping specified by st-tgds. Then with input $\mathcal{M}$, algorithm* Inverse *computes a maximum recovery of $\mathcal{M}$.*

By Theorems 21 and 22 we obtain the following corollary regarding the computation of Fagin-inverses and quasi-inverses.

▶ **Corollary 30.** *Let $\mathcal{M}$ be a mapping specified by st-tgds. If $\mathcal{M}$ has a Fagin-inverse (quasi-inverse), then with input $\mathcal{M}$, algorithm* Inverse *computes a Fagin-inverse (quasi-inverse) of $\mathcal{M}$.*

In general, the set Σ' constructed in algorithm Inverse is of exponential size. Notice that this directly depends on the size of the source rewritings computed by Source-Rew which are in general exponential. Nevertheless, there are cases for which this process can be done more efficiently. In particular, if mapping $\mathcal{M}$ is specified by a set of st-tgds that do not use existential quantification in the conclusions of dependencies, also called *full st-tgds* [18], then Step (2b) of algorithm Inverse can be accomplished in polynomial time [11, 34].

For the case of the Fagin-inverse, Arenas et al. [7] proposed an alternative algorithm that uses *target rewritings*. Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$ specified by st-tgds, and $Q_{\mathbf{S}}$ a conjunctive query over $\mathbf{S}$. Then, a query $Q_{\mathbf{T}}$ is a *target rewriting* of $Q_{\mathbf{S}}$ under $\mathcal{M}$ if $\text{certain}_{\mathcal{M}}(Q_{\mathbf{T}}, I) = Q_{\mathbf{S}}(I)$ for every source instance I. That is, $Q_{\mathbf{T}}$ is a target rewriting of $Q_{\mathbf{S}}$ if and only if $Q_{\mathbf{S}}$ is a source rewriting of $Q_{\mathbf{T}}$. Although the notions of source and target rewriting are tightly related, their associated algorithmic problems are not equivalent. For example, as opposed to the case of source rewritings, for a conjunctive query $Q_{\mathbf{S}}$ a target rewriting does not always exist [7]. Nevertheless, Arenas et al. [7] showed that if $\mathcal{M}$ is a mapping specified by st-tgds that has a Fagin-inverse, then every conjunctive source query is target rewritable. Moreover, it can be proved that a target rewriting can be computed in exponential time and can be expressed as a union of conjunctive queries with equalities and inequalities ($\text{UCQ}^{=,\neq}$) [7, 34]. We formalize this in the following lemma.

▶ **Lemma 31** ([7, 34]). *There exists a procedure* Target-Rew *that given a set Σ of st-tgds from $\mathbf{S}$ to $\mathbf{T}$, and a conjunctive query $Q_{\mathbf{S}}$ over $\mathbf{S}$ that is target rewritable, computes (in exponential time) a query in $\text{UCQ}^{=,\neq}$ which is a target rewriting of $Q_{\mathbf{S}}$ under the mapping $\mathcal{M}$ specified by Σ.*

With this procedure we can present the following algorithm to compute Fagin-inverses (which is implicit in the work by Arenas et al. [7](Proposition 5.3)). In the algorithm we also use the following property. A tgd from $\mathbf{S}$ to $\mathbf{T}$ of the form $\varphi_1(\bar{x}) \vee \varphi_2(\bar{x}) \to \psi(\bar{x})$ is equivalent to the set of tgds $\{\varphi_1(\bar{x}) \to \psi(\bar{x}), \varphi_2(\bar{x}) \to \psi(\bar{x})\}$. That is, one can always eliminate disjunctions from the premises of tgds. Another property that we use is that equalities in the premises of tgds can always be eliminated by making the necessary variable replacements. That is, the dependency $\varphi(\bar{x}) \wedge x = y \to \psi(\bar{x})$ is equivalent to $\varphi(\bar{x}') \to \psi(\bar{x}')$ where $\bar{x}'$ is the tuple obtained from $\bar{x}$ by replacing every occurrence of y by x.

Algorithm Fagin-Inverse

Input: A mapping $\mathcal{M}$ from $\mathbf{S}$ to $\mathbf{T}$ specified by a set Σ of st-tgds that has a Fagin-inverse.

Output: A mapping $\mathcal{M}'$ from $\mathbf{T}$ to $\mathbf{S}$ specified by a set Σ' of tgds with inequalities and predicate $\mathbf{C}(\cdot)$.

```
1. Start with $\Sigma'$ as the empty set.
2. For every source $k$-ary relation symbol $R$ do the following:
   a. Let $x = (x_1, \ldots, x_k)$ be a $k$-tuple of distinct variables, and $Q_{\mathbf{S}}$ the conjunctive query
      defined by formula $R(\bar{x})$.
   b. Use Target-Rew to compute a formula $\alpha(\bar{x})$ in UCQ$^{=,\neq}$ that is a target rewriting
      of $Q_{\mathbf{S}}$ under mapping $\mathcal{M}$.
   c. For every disjunct $\beta(\bar{x})$ of $\alpha(\bar{x})$ add dependency $\beta(\bar{x}) \wedge \mathbf{C}(\bar{x}) \to R(\bar{x})$ to $\Sigma'$.
3. Eliminate all the equality predicates in $\Sigma'$ by making the necessary variable replacements
   (and eliminating the remaining predicates $\mathbf{C}(x)$ for every replaced variable $x$).
4. Return the mapping $\mathcal{M}'$ from $\mathbf{T}$ to $\mathbf{S}$ specified by $\Sigma'$.
```
◀

▶ **Example 32.** Let $\mathbf{S} = \{A(\cdot,\cdot), B(\cdot)\}$ and $\mathbf{T} = \{S(\cdot,\cdot), T(\cdot), U(\cdot)\}$, and let $\mathcal{M}$ be the mapping in Example 5, that is, $\mathcal{M}$ is specified by the set of st-tgds

$$\begin{aligned} A(x,y) &\to S(x,y) \\ B(x) &\to S(x,x) \\ B(x) &\to T(x) \\ A(x,x) &\to U(x) \end{aligned}$$

With input $\mathcal{M}$, algorithm Fagin-Inverse first considers relation symbol A and in Step (2b) and computes a target rewriting of $A(x,y)$. It can be shown that the query given by $(S(x,y) \wedge x \neq y) \vee (U(x) \wedge x = y)$ is a target rewriting of $A(x,y)$. Then in Step (2c) the algorithm adds dependencies

$$\begin{aligned} S(x,y) \wedge x \neq y \wedge \mathbf{C}(x) \wedge \mathbf{C}(y) &\to A(x,y) \\ U(x) \wedge x = y \wedge \mathbf{C}(x) \wedge \mathbf{C}(y) &\to A(x,y) \end{aligned}$$

to the set Σ'. Then the algorithm considers relation symbol B and computes a target rewriting of $B(x)$. It can be proved that $T(x)$ is a target rewriting in this case, thus, the algorithm adds dependency $T(x) \wedge \mathbf{C}(x) \to B(x)$. Finally, in Step (4) the algorithm eliminates the equalities by variable replacements to obtain the set of dependencies

$$\begin{aligned} S(x,y) \wedge x \neq y \wedge \mathbf{C}(x) \wedge \mathbf{C}(y) &\to A(x,y) \\ U(x) \wedge \mathbf{C}(x) &\to A(x,x) \\ T(x) \wedge \mathbf{C}(x) &\to B(x) \end{aligned}$$

Notice that the obtained mapping is almost exactly the mapping that is claimed to be a Fagin-inverse of $\mathcal{M}$ in Example 5. ◀

The correctness of algorithm Fagin-Inverse is stated in the following theorem.

▶ **Theorem 33** ([7, 34]). *Let $\mathcal{M}$ be a mapping specified by st-tgds that has a Fagin-inverse. Then with input $\mathcal{M}$, algorithm* Fagin-Inverse *computes a Fagin-inverse of $\mathcal{M}$.*

It should be pointed out that the first algorithms proposed to compute quasi-inverses [21], and maximum recoveries [10], used ad-hoc techniques and were far more complicated that the one that we presented in this section. The algorithms that we have presented were proposed by Arenas et al. [11, 6, 7] and are based on query rewriting procedures which makes them suitable for optimizations and can be benefited from the vast amount of work on query rewriting in the data integration and data exchange contexts. Fagin et al. [21] also proposed a simple algorithm to compute Fagin-inverses based on the *chase* procedure. We do not explain all the details of this algorithm but describe the main idea. We assume some familiarity with the chase procedure for tgds (see Chapter 1 for details on the chase procedure). For every source atom $R(\bar{x})$, the algorithm by Fagin et al. [21] considers all the atoms obtained by considering all possible combinations of equalities among the variables in $\bar{x}$. Those atoms are called *prime atoms* in [21]. For example, for a source relation $R(\cdot,\cdot,\cdot)$ the algorithm considers the prime atoms $R(x_1,x_1,x_1)$, $R(x_1,x_1,x_2)$, $R(x_1,x_2,x_2)$, $R(x_1,x_2,x_1)$, and $R(x_1,x_2,x_3)$. Assume that a Fagin-inverse is to be computed for a mapping $\mathcal{M}$ specified by a set Σ of st-tgds. Moreover, given a prime atom α, let $\text{chase}_\Sigma(\alpha)$ be the result of chasing α with Σ. Then for every prime atom α the algorithm generates a formula σ_α of the form $\beta \wedge \delta \wedge \gamma \rightarrow \alpha$, where β is the conjunction of all the target atoms in $\text{chase}_\Sigma(\alpha)$, δ is a conjunction of inequalities of the form $x \neq y$ for every pair of different variables mentioned in α, and γ is a conjunction of formulas $\mathbf{C}(x)$ for every variable x mentioned in α. Fagin et al. [21] proved that if $\mathcal{M}$ has a Fagin-inverse, then the set of formulas $\{\sigma_\alpha \mid \alpha \text{ is a prime source atom}\}$ specifies a Fagin-inverse of $\mathcal{M}$ [21].

5.2 Languages for expressing inverses

In this section we study the question of what the language needed to express inverses is. In particular we survey the results in the literature that justify the languages used as output in the algorithms of the previous section. In particular a first result which is immediately obtained from the algorithms is the following.

▶ **Theorem 34** ([20, 10]). *Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$ specified by st-tgds.*

1. *$\mathcal{M}$ has a maximum recovery specified by a set of tgds from $\mathbf{T}$ to $\mathbf{S}$ with disjunctions and equalities in the conclusions and predicate $\mathbf{C}(\cdot)$ in the premises.*
2. *If $\mathcal{M}$ has a Fagin-inverse (quasi-inverse), then there exists a Fagin-inverse (quasi-inverse) of $\mathcal{M}$ specified by a set of tgds from $\mathbf{T}$ to $\mathbf{S}$ with disjunctions and equalities in the conclusions and predicate $\mathbf{C}(\cdot)$ in the premises.*
3. *If $\mathcal{M}$ has a Fagin-inverse, there exists a Fagin-inverse of $\mathcal{M}$ specified by a set of tgds from $\mathbf{T}$ to $\mathbf{S}$ with inequalities and predicate $\mathbf{C}(\cdot)$ in the premises.*

Parts 1) and 2) of the theorem follow from algorithm Inverse, while part 3) follows from algorithm Fagin-Inverse. Fagin et al. [20, 21] use a slightly different language to specify quasi-inverses of mappings specified by st-tgds. In particular, they use tgds with inequalities and predicate $\mathbf{C}(\cdot)$ in the premises and disjunctions (without equalities) in the conclusions. It is not difficult to see that in the output of algorithm Inverse one can replace the equalities in the conclusions of dependencies by inequalities in the premises as is outlined in the following example.

▶ **Example 35.** Consider the mapping $\mathcal{M}$ in Example 28, that is, $\mathcal{M}$ is specified by the dependencies $A(x,y) \rightarrow S(x,y)$ and $B(x) \rightarrow S(x,x)$. In that example, we compute a maximum recovery $\mathcal{M}'$ of $\mathcal{M}$ specified by the set of dependencies

$$\begin{aligned} S(x,y) \wedge \mathbf{C}(x) \wedge \mathbf{C}(y) \quad &\rightarrow \quad A(x,y) \vee (B(x) \wedge x = y), \\ S(x,x) \wedge \mathbf{C}(x) \quad &\rightarrow \quad A(x,x) \vee B(x). \end{aligned} \tag{9}$$

Notice that from (9) we can generate two formulas depending on whether $x = y$ or $x \neq y$

obtaining the set

$$\begin{array}{rcll} S(x,y) \wedge \mathbf{C}(x) \wedge \mathbf{C}(y) \wedge x \neq y & \rightarrow & A(x,y), & \\ S(x,y) \wedge \mathbf{C}(x) \wedge \mathbf{C}(y) \wedge x = y & \rightarrow & A(x,y) \vee B(x), & (10) \\ S(x,x) \wedge \mathbf{C}(x) & \rightarrow & A(x,x) \vee B(x). & (11) \end{array}$$

Finally we can use variable substitutions to eliminate the equality in (10). In that case the obtained dependency is equivalent to (11), and thus the final set of dependencies is

$$\begin{array}{rcl} S(x,y) \wedge \mathbf{C}(x) \wedge \mathbf{C}(y) \wedge x \neq y & \rightarrow & A(x,y), \\ S(x,x) \wedge \mathbf{C}(x) & \rightarrow & A(x,x) \vee B(x). \end{array}$$

◀

It was shown by Arenas et al. [6](Lemma 4.2) that if from the output of algorithm Inverse we eliminate the equalities with the process outlined in the above example, then the obtained mapping is still a maximum recovery of $\mathcal{M}$. By a different procedure Fagin et al. [20] showed that for mappings specified by st-tgds that has a quasi-inverse, there exists a quasi inverse specified by a set of tgds with inequalities and predicate $\mathbf{C}(\cdot)$ in the premises and disjunctions in the conclusions. Thus we have the following.

▶ **Theorem 36** ([6, 20]). *Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$ specified by st-tgds.*

1. *$\mathcal{M}$ has a maximum recovery specified by a set of tgds from $\mathbf{T}$ to $\mathbf{S}$ with inequalities and predicate $\mathbf{C}(\cdot)$ in the premises and disjunctions in the conclusions.*
2. *If $\mathcal{M}$ has a quasi-inverse, then there exists a quasi-inverse of $\mathcal{M}$ specified by a set of tgds from $\mathbf{T}$ to $\mathbf{S}$ with inequalities and predicate $\mathbf{C}(\cdot)$ in the premises and disjunctions in the conclusions.*

We know what languages are sufficient to specify inverses but, are all the features of these languages strictly needed to specify inverses? For example, do we really need disjunctions to specify maximum recoveries and quasi-inverses? Do we really need predicate $\mathbf{C}(\cdot)$ to specify Fagin-inverses? In what follows we answer these questions.

The first result that we report was proved by Fagin et al. [20, 21], and states that predicate $\mathbf{C}(\cdot)$ is strictly necessary to specify Fagin-inverses.

▶ **Theorem 37** (Necessity of $\mathbf{C}(\cdot)$ [20]). *There exists a mapping $\mathcal{M}$ specified by st-tgds that has a Fagin-inverse but does not have a Fagin-inverse specified by tgds with inequalities in the premises and disjunctions in the conclusions (without using predicate $\mathbf{C}(\cdot)$).*

▶ **Example 38.** Consider the mapping $\mathcal{M}$ specified by the st-tgds $A(x,y) \rightarrow \exists z \big(S(x,z) \wedge S(z,y)\big)$. It can be shown that $\mathcal{M}$ is Fagin-invertible. In fact, the mapping $\mathcal{M}'$ specified by $S(x,z) \wedge S(z,y) \wedge \mathbf{C}(x) \wedge \mathbf{C}(y) \rightarrow A(x,y)$ is a Fagin-inverse of $\mathcal{M}$. Fagin et al. [20] show that $\mathcal{M}$ does not have a Fagin-inverse that does not use $\mathbf{C}(\cdot)$. To see the intuition of the failure, lets show that if we delete the $\mathbf{C}(\cdot)$ predicates in the definition of $\mathcal{M}'$, the resulting mapping is no longer a Fagin-inverse of $\mathcal{M}$. Thus consider the mapping $\mathcal{M}''$ specified by $S(x,z) \wedge S(z,y) \rightarrow A(x,y)$ and assume that $\mathcal{M}''$ is a Fagin-Inverse of $\mathcal{M}$. Then for every source instance I we have that $(I,I) \in \mathcal{M} \circ \mathcal{M}''$. Now consider the instance $I = \{A(1,2), A(2,1)\}$. Since $(I,I) \in \mathcal{M} \circ \mathcal{M}''$ we know that there exists an instance K such that $(I,K) \in \mathcal{M}$ and $(K,I) \in \mathcal{M}''$. Thus, by the definition of $\mathcal{M}$, we have that there exists elements a, b such that $S(1,a), S(a,2), S(2,b), S(b,1) \in K$. Then by definition of $\mathcal{M}''$ we have that $A(a,b), A(b,a) \in I$ and thus, either $a = 1$ and $b = 2$, or $a = 2$ and $b = 1$. Assume first that $a = 1$ and $b = 2$, then we have that $S(1,1), S(2,2) \in K$ which implies that

$A(1,1), A(2,2) \in I$ which is a contradiction. If we assume that $a = 2$ and $b = 1$ we obtain the same contradiction. Notice that we cannot obtain this contradiction with $\mathcal{M}'$ since I has as solution under $\mathcal{M}$ the instance $K' = \{A(1,n), A(n,2), A(2,m), A(m,2)\}$ with n and m different null values. Moreover, K' has I as solution under $\mathcal{M}'$ and thus $(I, I) \in \mathcal{M} \circ \mathcal{M}'$. ◀

One can prove a stronger result for the case of maximum recoveries, namely that predicate $\mathbf{C}(\cdot)$ is needed even if we allow the full power of First-Order logic.

▶ **Theorem 39** (Necessity of $\mathbf{C}(\cdot)$ for maximum recoveries [10]). *There exists a mapping $\mathcal{M}$ specified by st-tgds that has no maximum recovery specified by First-Order sentences that do not use predicate $\mathbf{C}(\cdot)$.*

The following result shows that we need either inequalities in the premises or equalities in the conclusions of dependencies in order to specify Fagin-inverses. This immediately implies the necessity of these features to specify quasi-inverses and maximum recoveries.

▶ **Theorem 40** (Necessity of either $=$ or $\neq$ [20]). *There exists a mapping $\mathcal{M}$ specified by st-tgds that has a Fagin-inverse but does not have a Fagin-inverse specified by tgds with predicate $\mathbf{C}(\cdot)$ in the premises and disjunctions in the conclusions.*

Fagin et al. [21] use the mapping $\mathcal{M}$ in Example 32 to show the necessity of either inequalities in the premises of equalities in the conclusions to specify Fagin-inverses. The only remaining property that we need to prove is that disjunctions are necessary for quasi-inverses and maximum recoveries.

▶ **Theorem 41** (Necessity of $\vee$ [34, 20]).
1. *There exists a mapping $\mathcal{M}$ specified by st-tgds that has no maximum recovery specified by tgds with predicate $\mathbf{C}(\cdot)$ in the premises and equalities in the conclusions.*
2. *There exists a mapping $\mathcal{M}$ specified by st-tgds that has a quasi-inverse but has no quasi-inverse specified by tgds with inequalities and predicate $\mathbf{C}(\cdot)$ in the premises.*

6 Query Language-Based Inverses of Schema Mappings

In the data exchange scenario, the standard procedure used to exchange data with a mapping is based on the *chase* procedure [18] (See Chapter 1 for a comprehensive study of the chase procedure in data exchange). More precisely, given a mapping $\mathcal{M}$ and a source database I, a *canonical* translation of I according to $\mathcal{M}$ is computed by *chasing* I with the set of dependencies defining $\mathcal{M}$ [18]. Thus, when computing an inverse of $\mathcal{M}$, it would be desirable from a practical point of view to obtain a mapping $\mathcal{M}'$ where the chase procedure can be used to exchange data. Unfortunately, the notions of inverse that we have introduced in the previous sections, express inverses in some mapping languages which include features that are difficult to use in practice. The most important of those issues is the use of disjunctions in the conclusion of the mapping rules.

To provide a solution for the aforementioned issue, Arenas et. al [6] introduce a query-language based notion of inverse called $\mathcal{C}$-maximum recovery, with $\mathcal{C}$ a class of queries. The idea is that when one focuses on particular query languages one can obtain inverses with better properties regarding the languages needed to specify these inverses. In particular, Arenas et al. [6] proved that when one focuses on conjunctive queries, one can obtain inverses that can be expressed in a *chaseable* language.

The main intuition behind the notion proposed by Arenas et al. [6] is to use queries to measure the amount of information that a mapping $\mathcal{M}'$ can recover with respect to a mapping $\mathcal{M}$. Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$, $\mathcal{M}'$ a mapping from $\mathbf{T}$ to $\mathbf{S}$. Notice that $\mathcal{M} \circ \mathcal{M}'$ is a mapping that goes from $\mathbf{S}$ to $\mathbf{T}$ and then to $\mathbf{S}$ again. Thus one can measure the amount of information recovered by $\mathcal{M}'$ by using queries over $\mathbf{S}$. Let Q be a query over $\mathbf{S}$, then we say that $\mathcal{M}'$ recovers sound information w.r.t. Q under $\mathcal{M}$ if for every instance I it holds that

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

Thus, by posing the query Q over the space of solutions for I under $\mathcal{M} \circ \mathcal{M}'$, one can only obtain tuples that are already in the evaluation of Q over the original instance I. This notion can be generalized to a class $\mathcal{C}$ of queries, which gives rise to the notion of $\mathcal{C}$-recovery.

▶ **Definition 42** ([6])**.** Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$, $\mathcal{M}'$ a mappings from $\mathbf{T}$ to $\mathbf{S}$, and $\mathcal{C}$ a class of queries over $\mathbf{S}$. Then $\mathcal{M}'$ is a $\mathcal{C}$-recovery of $\mathcal{M}$ if for every query $Q \in \mathcal{C}$ and every source instance I it holds that

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

As for the definition of maximum recovery, one can compare different $\mathcal{C}$-recoveries. Let $\mathcal{M}'$ and $\mathcal{M}''$ be $\mathcal{C}$-recoveries of $\mathcal{M}$, and suppose that for every query $Q \in \mathcal{C}$ and source instance I, it holds that

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

Clearly, the mapping $\mathcal{M}'$ is better than $\mathcal{M}''$ to recover information w.r.t. queries in $\mathcal{C}$, since $\text{certain}_{\mathcal{M} \circ \mathcal{M}'}(Q, I)$ is *closer* to $Q(I)$ than $\text{certain}_{\mathcal{M} \circ \mathcal{M}''}(Q, I)$. This discussion naturally gives rise to the notion of $\mathcal{C}$-maximum recovery.

▶ **Definition 43** ([6])**.** Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$, and $\mathcal{C}$ a class of queries over $\mathbf{S}$. Then $\mathcal{M}'$ is a $\mathcal{C}$-maximum recovery of $\mathcal{M}$ if

1. $\mathcal{M}'$ is a $\mathcal{C}$-recovery of $\mathcal{M}$, and
2. for every $\mathcal{C}$-recovery $\mathcal{M}''$ of $\mathcal{M}$, it holds that $\text{certain}_{\mathcal{M} \circ \mathcal{M}''}(Q, I) \subseteq \text{certain}_{\mathcal{M} \circ \mathcal{M}'}(Q, I)$.

Before stating some general results regarding $\mathcal{C}$-maximum recoveries and the relationship with the notions presented in the previous sections, let us show some examples on what is the influence of the class $\mathcal{C}$ of queries in the notion of $\mathcal{C}$-maximum recovery. Before presenting the example, we note that if $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$, then $\mathcal{M}'$ is a $\mathcal{C}$-maximum recovery of $\mathcal{M}$ for every class $\mathcal{C}$ of queries [8]. This is not difficult to show given the set of tools for maximum recoveries proposed by Arenas et al. [11] (see [34, 8] for details on the relationship between maximum recoveries and $\mathcal{C}$-maximum recoveries).

▶ **Example 44.** Let $\mathcal{M}$ be specified by these two st-tgds:

$$A(x, y) \ \rightarrow \ R(x, y), \qquad B(x) \ \rightarrow \ R(x, x).$$

It can be shown that mapping $\mathcal{M}_1$ specified by dependency:

$$R(x, y) \quad \rightarrow \quad A(x, y) \ \vee \ \big(B(x) \wedge x = y\big)$$

is a UCQ-maximum recovery of $\mathcal{M}$ (in fact $\mathcal{M}_1$ is a maximum recovery of $\mathcal{M}$). To specify $\mathcal{M}_1$, we have used a disjunction in the conclusion of the dependency. This disjunction is unavoidable if we use UCQ to retrieve information [8]. On the other hand, if we focus on

CQ to retrieve information, then, intuitively, there is no need for disjunctions in the right-hand side of the rules as conjunctive queries cannot extract disjunctive information. In fact, it can be shown that a CQ-maximum recovery of $\mathcal{M}$ is specified by dependency:

$$R(x,y) \wedge x \neq y \quad \rightarrow \quad A(x,y).$$

◀

The example suggests that the notion of CQ-maximum recovery is a strict generalization of the notion of UCQ-maximum recovery. More importantly, it shows that for different choices of the class of queries used, we obtain different notions of inverses of schema mappings. The following results show that one can actually characterize the notions of Fagin-inverse and quasi-inverse for particular classes of queries. In the theorem we use $\text{UCQ}^{\neq}$ to denote the class of unions of conjunctive queries with inequalities.

▶ **Theorem 45** ([6, 8]). *Let $\mathcal{M}$ be a mapping specified by a set of st-tgds.*

1. *Assume that $\mathcal{M}$ has a Fagin-inverse. Then $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$ if and only if $\mathcal{M}'$ is a $\text{UCQ}^{\neq}$-maximum recovery of $\mathcal{M}$.*
2. *Assume that $\mathcal{M}$ has a quasi-inverse. There exists a class $\mathcal{C}_{\mathcal{M}}$ that depends on $\mathcal{M}$ such that $\mathcal{M}'$ is a quasi-inverse of $\mathcal{M}$ if and only if $\mathcal{M}'$ is a $\mathcal{C}_{\mathcal{M}}$-maximum recovery of $\mathcal{M}$.*

Arenas et al. [6, 8] provide several tools to work with $\mathcal{C}$-maximum recoveries including characterizations for the mappings that admit $\mathcal{C}$-maximum recoveries and a general necessary and sufficient condition for the existence of $\mathcal{C}$-maximum recoveries. We refer the reader to [34, 8] for a comprehensive study of $\mathcal{C}$-maximum recoveries, and in particular, for a definition of the class $\mathcal{C}_{\mathcal{M}}$ used in part 2) of Theorem 45.

The language of CQ-maximum recoveries

Arenas et al. [6] study several properties about $\mathcal{C}$-maximum recoveries when one focuses on CQ as the class $\mathcal{C}$ of queries. In particular, they provide an algorithm to compute CQ-maximum recoveries for st-tgds showing the following theorem.

▶ **Theorem 46** ([6]). *Every mapping specified by a set of st-tgds has a* CQ*-maximum recovery, which is specified by a set of tgds with inequalities and predicate* $\mathbf{C}(\cdot)$ *in the premises.*

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 a CQ-maximum recovery can be specified in a mapping language with good properties for data exchange (which is not the case for quasi-inverses and maximum recoveries).

The algorithm proposed by Arenas et al. [6] to compute CQ-maximum recoveries is based on the algorithm for computing maximum recoveries reported in the previous section. After computing a maximum recovery, the algorithm does a post-processing step to eliminate the disjunctions in the conclusions of the dependencies by using a notion of *conjunctive-query products* [6, 34]. Given two conjunctive queries Q_1 and Q_2, the product query $Q_1 \times Q_2$ is, intuitively, the closest conjunctive query to both Q_1 and Q_2 in terms of *homomorphisms.* Let us to introduce some terminology to formalize this notion.

Let Q_1 and Q_2 be two n-ary conjunctive queries, and assume that $\bar{x}$ is the tuple of free variables of Q_1 and Q_2. The *product* of Q_1 and Q_2, denoted by $Q_1 \times Q_2$, is defined as a k-ary conjunctive query (with $k \leq n$) constructed as follows. Let $f(\cdot,\cdot)$ be a one-to-one function from pairs of variables to variables such that:

1. $f(x,x) = x$ for every variable x in $\bar{x}$, and
2. $f(y,z)$ is a fresh variable (mentioned neither in Q_1 nor in Q_2) in any other case.

Then for every pair of atoms $R(y_1,\ldots,y_m)$ in Q_1 and $R(z_1,\ldots,z_m)$ in Q_2, the atom $R(f(y_1,z_1),\ldots,f(y_m,z_m))$ is included as a conjunct in the query $Q_1 \times Q_2$. Furthermore, the set of free variables of $Q_1 \times Q_2$ is the set of variables from $\bar{x}$ that are mentioned in $Q_1 \times Q_2$. For example, consider conjunctive queries:

$$\begin{aligned} Q_1(x_1,x_2) &\ :\ P(x_1,x_2) \wedge R(x_1,x_1), \\ Q_2(x_1,x_2) &\ :\ \exists y\,(P(x_1,y) \wedge R(x_2,x_2)). \end{aligned}$$

Then we have that $Q_1 \times Q_2$ is the conjunctive query:

$$(Q_1 \times Q_2)(x_1) \ :\ \exists z_1 \exists z_2\,(P(x_1,z_1) \wedge R(z_2,z_2)).$$

In this case, we have used a function f such that $f(x_1,x_1) = x_1$, $f(x_2,y) = z_1$, and $f(x_1,x_2) = z_2$. As shown in the example, the free variables of $Q_1 \times Q_2$ do not necessarily coincide with the free variables of Q_1 and Q_2. The definition of the product of queries is motivated by the standard notion of *Cartesian product* of graphs. In fact, if Q_1 and Q_2 are Boolean queries constructed by using a single binary relation $E(\cdot,\cdot)$, then the product $Q_1 \times Q_2$ exactly resembles the graph-theoretical Cartesian product [27].

The product of queries is the key ingredient in the algorithm CQ-Max-Recovery proposed by Arenas et al. [6] to compute CQ-maximum recoveries. Given a mapping $\mathcal{M}$ specified by st-tgds, CQ-Max-Recovery first uses algorithm Inverse to compute a maximum recovery $\mathcal{M}'$ of $\mathcal{M}$. Then it eliminates equalities in the conclusions of the dependencies defining $\mathcal{M}'$ by adding the necessary inequalities in the premises of the dependencies (as outlined in Example 35). Finally, the algorithm replaces the remaining disjunctions $Q_1 \vee Q_2 \vee \cdots \vee Q_k$ in the conclusions of the tgds, by the conjunctive query $Q_1 \times Q_2 \times \cdots \times Q_k$. The final output of CQ-Max-Recovery is a set of tgds with inequalities and predicate $\mathbf{C}(\cdot)$ in the premises (without disjunctions in the conclusions).

▶ **Example 47.** Assume that the output of Inverse contains the dependency

$$A(x,y) \wedge \mathbf{C}(x) \wedge \mathbf{C}(y) \ \rightarrow\ \big(P(x,y) \wedge R(x,x)\big) \ \vee\ \exists z\,\big(P(x,z) \wedge R(y,y)\big).$$

Then algorithm CQ-Max-Recovery replaces this dependency by

$$A(x,y) \wedge \mathbf{C}(x) \wedge \mathbf{C}(y) \ \rightarrow\ \exists u \exists v\,\big(P(x,u) \wedge R(v,v)\big),$$

since $\exists u \exists v\,(P(x,u) \wedge R(v,v))$ is the product of $P(x,y) \wedge R(x,x)$ and $\exists z\,\big(P(x,z) \wedge R(y,y)\big)$. ◀

Arenas et al. [6] 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. Arenas et al. [8] also show that the class CQ is *optimal* to obtain the desired result of a notion of inverse with good properties for data exchange. In particular, if one uses either $\text{CQ}^{\neq}$ or $\text{UCQ}^{\neq}$ in the definition of $\mathcal{C}$-maximum recovery, then the language needed to express inverses is no longer *chaseable* [8].

7 Inversion in the Presence of Null Values in Source Instances

Fagin et al. [22] 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 contain null values. Most of the results regarding inverses that we have reported so far are proved for the case of mappings in which the source instances contain only constant values while target instances may contain constant and null values. Fagin et al. [22] 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 (for a comprehensive study of the notion of extended inverse see the work by Fagin et al. [22]).

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 [28]. In fact, as shown by Fagin et al. [22], if one treats nulls in that way, the existence of a maximum recovery for mappings given by st-tgds is no longer guaranteed.

▶ **Example 48.** Consider a source schema $\mathbf{S} = \{A(\cdot), B(\cdot)\}$ and a target schema $\mathbf{T} = \{S(\cdot)\}$, and let $\mathcal{M}$ be a mapping specified by the st-tgds

$$\begin{aligned} A(x) &\rightarrow \exists u S(u) \\ B(x) &\rightarrow S(x) \end{aligned}$$

From Theorem 24, we know that if source instances only contain constant values, then $\mathcal{M}$ has a maximum recovery. This property holds since, under this assumption, every source instance I has a witness solution (see Definition 23 and Theorem 24). For example, for the instance $I = \{A(1)\}$ the target instance $J = \{S(n)\}$, with n a null value, is a witness solution of I. In fact, if I' is any source instance such that $J \in \text{Sol}_{\mathcal{M}}(I)$ then $\text{Sol}_{\mathcal{M}}(I) \subseteq \text{Sol}_{\mathcal{M}}(I')$. Assume now that instances of $\mathbf{S}$ may contain constant and null values. Then we have that J is no longer a witness solution of I under $\mathcal{M}$. To see this consider the source instance $I' = \{B(n)\}$. Then we have that $J \in \text{Sol}_{\mathcal{M}}(I')$ but, for example $J' = \{S(2)\}$ is a solution for I but not for I', therefore $\text{Sol}_{\mathcal{M}}(I) \not\subseteq \text{Sol}_{\mathcal{M}}(I')$, and thus J is not a witness solution of I. In fact, it can be proved that if source instances may contain null values then I has no witness solution under $\mathcal{M}$ implying that $\mathcal{M}$ has no maximum recovery if null are allowed in the source. ◀

Notice that in the above example, nulls in the source are considered as constants when evaluating the tgds. Since nulls should not be treated naively when exchanging data, Fagin et al. [22] proposed a new way to deal with null values based on homomorphisms. Recall that given instances I and I' containing constant and null values, a homomorphism from I to I' is a function h that is the identity over constant values, maps nulls to constants or null values, and is such that if $R(a_1, \dots, a_k)$ is a fact in I, then $R(h(a_1), \dots, h(a_k))$ is a fact in I'. Intuitively, in order to treat null values and constants differently, Fagin et al. [22] *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, the authors introduce the following concept.

▶ **Definition 49** ([22]). Let $\mathcal{M}$ be a mapping. The *homomorphic extension* of $\mathcal{M}$, denoted by $e(\mathcal{M})$, is the mapping

$$e(\mathcal{M}) = \{(I,J) \mid \text{ there exist } I', J' \text{ such that } (I',J') \in \mathcal{M} \text{ and there exist homomorphisms from } I \text{ to } I' \text{ and from } J' \text{ to } J\}.$$

The idea is that 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 [22]. The following result shows that with this new semantics one can avoid anomalies as the one shown in Example 48.

▶ **Theorem 50** ([22]). *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. [22] go a step further by introducing new notions of inverse for mappings that consider nulls in the source. More specifically, the authors introduce the following definitions

▶ **Definition 51** ([22]). Let $\mathcal{M}$ be a mapping from $\mathbf{S}$ to $\mathbf{T}$, in which source and target instances may contain null values. Mapping $\mathcal{M}'$ is an *extended recovery* of $\mathcal{M}$ if $(I,I) \in e(\mathcal{M}) \circ e(\mathcal{M}')$, for every instance I of $\mathbf{S}$. Then given an extended recovery $\mathcal{M}'$ of $\mathcal{M}$, the mapping $\mathcal{M}'$ is a *maximum extended recovery* of $\mathcal{M}$ if for every extended recovery $\mathcal{M}''$ of $\mathcal{M}$, it holds that $e(\mathcal{M}) \circ e(\mathcal{M}') \subseteq e(\mathcal{M}) \circ e(\mathcal{M}'')$.

At a first glance, one may think that the notions of maximum recovery and maximum extended recovery are incomparable. Nevertheless, as shown by Arenas et al. [5] there is a tight connection between these two notions.

▶ **Theorem 52** ([5]). *Let $\mathcal{M}$ be a mapping that may have nulls values in source and target instances. Then $\mathcal{M}$ 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 $e(\mathcal{M})$.*

One of the main result of Fagin et al. [22] regarding maximum extended recoveries is that every mapping specified by st-tgds having nulls in source and target instances has a maximum extended recovery. This result is implied by Theorems 50 and 52, and we formalize it in the following theorem.

▶ **Theorem 53** ([22]). *Let $\mathcal{M}$ be a mapping specified by st-tgds in which source and target instances may contain null values. Then $\mathcal{M}$ has a maximum extended recovery.*

It was left as an open problem to identify what is the exact language needed to express maximum extended recoveries [22]. In fact, it is even open whether maximum extended recoveries can be specified if the full power of First-Order logic is allowed to construct mappings.

8 Conclusions

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 schema mappings has been identified as a fundamental issue to be solved [12, 13]. Nowadays, the community recognizes the need to develop techniques to manipulate these mappings' specifications and, in particular, the inverse of a schema mapping has been identified as one of the fundamental operators to be studied in this area. In this chapter, we have surveyed the main definition for the inverse operator proposed in the literature and the results that have been obtained in the last years.

One very important and challenging problem is the interplay between the inverse operator and other schema mapping operators, in particular the composition of schema mappings [33, 19]. Arenas et al. [9] proved that the mapping that results from composing mappings specified by st-tgds is not always invertible (even considering the relaxed notion of CQ-maximum recovery). This opens the question on good notions for inversion and composition of schema mappings, and a language for expressing mappings, suitable to deal with the interplay between the two operators [5]. A first attempt and a partial solution for this problem was given in [9].

The definition of the appropriate semantics for the inverse operator has proven to be a non-trivial task, in which many *sensible* decisions had to be taken. In fact, the answer to each of these decisions has given rise to different semantics for the inverse operator. Some general questions that one might want to answer include whether we want inverses that are guaranteed to be *consistent* in a general scenario, or do we settle for relaxed notions that allow only answering conjunctive queries (or other restricted classes of queries)? Certainly, the latter question involves a tradeoff, since the more general operators usually require more expressive languages, and their computation is more complex.

The spread of new semantics for the schema mappings, either modified semantics for mappings specified by standard logical specifications over the relational model [32, 22], or mappings for data models beyond the relational model, such as XML, which need different mapping specification languages [4, 2, 36], originates several challenges. Under these new scenarios, previously defined mapping operators have to be re-studied. This shows the importance of having general notions of inverse that are not tied to a particular schema mapping semantic, language or data model. Among the notions that we have presented, only the notion of maximum recovery is defined in a general setting and can be applied to abstract mappings independent of the mapping specification language, the semantics used for logical specifications, or the data model used. Nevertheless, there is not yet clear consensus about which semantics for the inverse operator is the appropriate one in general, and we think that particular applications would need to use different inverses depending on their specific needs. We hope the definitions and results presented in this chapter would be useful to compare the proposals for inverses, their characteristics, and their applicability in different contexts.

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

4 M. Arenas and L. Libkin. XML data exchange: Consistency and query answering. *J. ACM*, 55(2), 2008.

5 M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. Composition and inversion of schema mappings. *SIGMOD Record*, 38(3):17–28, 2009.

6 M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. Inverting schema mappings: Bridging the gap between theory and practice. *PVLDB*, 2(1):1018–1029, 2009.

7 M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. Foundations of schema mapping management. In *PODS*, pages 227–238, 2010.

8 M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. Query language based inverses of schema mappings: Semantics, computation, and closure properties. *VLDB J.*, 21(6):823–842, 2012.

9 M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. The language of plain SO-tgds: Composition, inversion and structural properties. *J. Comput. Syst. Sci.*, 79(6):763–784, 2013.

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

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

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

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

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

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

16 R. Fagin. Inverting schema mappings. In *PODS*, pages 50–59, 2006.

17 R. Fagin. Inverting schema mappings. *TODS*, 32(4), 2007.

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

19 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.

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

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

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

23 R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. Schema mapping evolution through composition and inversion. In Z. Bellahsene, A. Bonifati, and E. Rahm, editors, *Schema Matching and Mapping*, pages 191–222. Springer, 2011.

24 A. Halevy. Theory of answering queries using views. *SIGMOD Record*, 29(1):40–47, 2000.

25 A. Halevy. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294, 2001.

26 A. Halevy, Z. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov. The piazza peer data management system. *IEEE Trans. Knowl. Data Eng.*, 16(7):787–798, 2004.

27 P. Hell and J. Nešetřil. *Graphs and Homomorphisms.* Oxford University Press, 2004.

28 T. Imielinski and W. Lipski. Incomplete information in relational databases. *Journal of the ACM*, 31(4):761–791, 1984.

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

33 S. Melnik. *Generic Model Management: concepts and Algorithms*, volume 2967 of *Lecture Notes in Computer Science.* Springer, 2004.

34 J. Pérez. *Schema Mapping Management in Data Exchange Systems.* PhD thesis, Escuela de Ingeniería, Pontificia Universidad Católica de Chile, 2011.

35 R. Pottinger and A. Halevy. Minicon: A scalable algorithm for answering queries using views. *VLDB J.*, 10(2-3):182–198, 2001.

36 J. Terwilliger, P. Bernstein, and S. Melnik. Full-fidelity flexible object-oriented XML access. *PVLDB*, 2(1):1030–1041, 2009.