Composition with Target Constraints

Marcelo Arenas
PUC Chile
marenas@ing.puc.cl

Ronald Fagin
IBM Research – Almaden
fagin@almaden.ibm.com

Alan Nash
Aleph One LLC

ABSTRACT

It is known that the composition of schema mappings, each specified by source-to-target tgds (st-tgds), can be specified by a second-order tgd (SO tgd). We consider the question of what happens when target constraints are allowed. Specifically, we consider the question of specifying the composition of *standard schema mappings* (those specified by st-tgds, target egds, and a weakly-acyclic set of target tgds). We show that SO tgds, even with the assistance of arbitrary source constraints and target constraints, cannot specify in general the composition of two standard schema mappings. Therefore, we introduce source-to-target second-order dependencies (st-SO dependencies), which are similar to SO tgds, but allow equations in the conclusion. We show that st-SO dependencies (along with target egds and target tgds) are sufficient to express the composition of every finite sequence of standard schema mappings, and further, every st-SO dependency specifies such a composition. In addition to this expressive power, we show that st-SO dependencies enjoy other desirable properties. In particular, they have a polynomial-time chase that generates a universal solution. This universal solution can be used to find the certain answers to unions of conjunctive queries in polynomial time.

It is easy to show that the composition of an arbitrary number of standard schema mappings is equivalent to the composition of only two standard schema mappings. We show that surprisingly, the analogous result holds also for schema mappings specified by just st-tgds (no target constraints). That is, the composition of an arbitrary number of such schema mappings is equivalent to the composition of only two such schema mappings. This is proven by showing that every SO tgd is equivalent to an unnested SO tgd (one where there is no nesting of function symbols). The language of unnested SO tgds is quite natural, and we show that unnested SO tgds are capable of specifying the composition of an arbitrary number of schema mappings, each specified by st-tgds. Similarly, we prove unnesting results for st-SO dependencies, with the same types of consequences.

The collapsing result for SO tgds gives us two alternative ways to deal with the composition of multiple schema mappings specified by st-tgds. First, we can replace the composition by a single schema mapping, specified by an unnested SO tgd. Second, we can replace the composition by the composition of only two schema mappings, each specified by st-tgds. A similar comment holds for the composition of standard schema mappings.

ICDT 2010, March 22–25, 2010, Lausanne, Switzerland.

Categories and Subject Descriptors

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

General Terms

Algorithms, Theory

Keywords

Metadata management, schema mapping, data exchange, composition, target constraint

1. INTRODUCTION

Schema mappings are high-level specifications that describe the relationship between two database schemas, a *source schema* and a *target schema*. Because of the crucial importance of schema mappings for data integration and data exchange (see the surveys [33, 34]), several different operators on schema mappings have been singled out as important objects of study [8]. One of the most fundamental is the composition operator, which combines successive schema mappings into a single schema mapping. The composition operator can play a useful role each time the target of a schema mapping is also the source of another schema mapping. This scenario occurs, for instance, in schema evolution, where a schema may undergo several successive changes. It also occurs in extract-transform-load (ETL) processes in which the output of a transformation may be the input to another [43]. The composition operator has been studied in depth [21, 37, 39, 40].

One of the most basic questions is: what is the language needed to express the composition of schema mappings? For example, if the schema mapping $\mathcal{M}_{12}$ is an *st-tgd mapping*, that is, a mapping specified by a finite set of the widely-studied *source-to-target tuple-generating dependencies (st-tgds)*, and the schema mapping $\mathcal{M}_{23}$ is also an st-tgd mapping, is the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ also an st-tgd mapping? Fagin et al. [21] showed that surprisingly, the answer is "No." In fact, they showed that it is necessary to pass to existential second-order logic to express this composition in general. Specifically, they defined a class of dependencies, which they call *second-order tgds (SO tgds)*, which are source-to-target, with existentially-quantified function symbols, and they showed that this is the "language of composition". That is, they showed that the composition of any number of st-tgd mappings can be specified by

an SO tgd. They also showed that every SO tgd specifies the composition of a finite number of st-tgd mappings. Thus, SO tgds are exactly the right language,

What happens if we allow not only source-to-target constraints, but also target constraints? Target constraints are important in practice; examples of important target constraints are those that specify the keys of target relations, and referential integrity constraints (or inclusion dependencies [9]). This paper is motivated by the question of how to express the compositions of schema mappings with target constraints. This question was first explored by Nash et al. [40], where an even more general class of constraints was studied: constraints expressed over the joint source and target schemas without any restrictions. Here we study a case intermediate between that studied by Fagin et al. in [21] and that studied by Nash et al. in [40]. Specifically, we study *standard schema mappings*, where the source-to-target constraints are st-tgds, and the target constraints consist of target equality-generating dependencies (t-egds) and a weakly acyclic set [20] of target tuple-generating dependencies (t-tgds). Standard schema mappings have a chase that is guaranteed to terminate in polynomial time. In fact, weak acyclicity was introduced in [20] in order to provide a fairly general sufficient condition for the chase to terminate in polynomial time (a slightly less general class was introduced in [15], under the name *constraints with stratified witness*, for the same purpose).

Standard schema mappings are a natural "sweet spot" between the schema mappings studied by Fagin et al. [20] (with only source-to-target constraints) and the schema mappings studied by Nash et al. [40] (with general constraints), for two reasons. The first reason is the importance of standard schema mappings. Source-to-target tgds are the natural and common backbone language of data exchange systems [18]. Furthermore, even though the notion of weakly acyclic sets of tgds was introduced only recently, it has now been studied extensively [1, 2, 3, 4, 6, 7, 10, 11, 13, 14, 19, 20, 24, 26, 27, 28, 29, 31, 32, 33, 38, 41, 42]. Among the important special cases of weakly acyclic sets of tgds are sets of full tgds (those with no existential quantifiers) and acyclic sets of inclusion dependencies [12], a large class that is common in practice. The second reason for our interest in standard schema mappings is that as we shall see, compositions of standard schema mappings have especially nice properties. Thus, the language of standard schema mappings is expressive enough to be useful in practice, and yet simple enough to allow nice properties, such as having a polynomial-time chase.

There are various inexpressibility results in [21] and [40] that show the inability of first-order logic to express compositions. Thus, each of these results say that there is a pair of schema mappings that are each specified by simple formulas in first-order logic, but where the composition cannot be expressed in first-order logic. In this paper, we show that some compositions cannot be expressed even in certain fragments of second-order logic. First, we show that SO tgds are not adequate to express the composition of an arbitrary pair of standard schema mappings. It turns out that this is quite easy to show. But what if we allow not only SO tgds, but also arbitrary source constraints and target constraints? This is a more delicate problem. By making use of a notion of locality from [5], we show that even these are not adequate to express the composition of an arbitrary pair of standard schema mappings.

Therefore, we introduce a richer class of dependencies, which we call *source-to-target second-order dependencies (st-SO dependencies)*. This class of dependencies is the source-to-target restriction of the class Sk$\forall$CQ$^{=}$ of dependencies introduced in [40]. Our st-SO dependencies differ from SO tgds in that st-SO dependencies may have not only relational atomic formulas $R(t_1, \ldots, t_n)$ in the conclusions, but also equalities $t_1 = t_2$. We show that st-SO dependencies are exactly the right extension of SO tgds for the purpose of expressing the composition of standard schema mappings. Specifically, we show that (1) the composition of standard schema mappings can be expressed by an st-SO dependency (along with target constraints), and (2) every st-SO dependency specifies the composition of some finite sequence of standard schema mappings. We note that a result analogous to (1), but for schema mappings that are not necessarily source-to-target, was obtained in [40] by using their class Sk$\forall$CQ$^{=}$ of dependencies. In fact, our proof of (1) is simply a variation of the proof in [40].

In addition, we show that st-SO dependencies enjoy other desirable properties. In particular, we show that they have a polynomial-time chase procedure. This chase procedure is novel, in that it has to keep track of constantly changing values of functions. As usual, the chase generates not just a solution, but a *universal solution* [20]. (Recall that a *solution* J for a source instance I with respect to a schema mapping $\mathcal{M}$ is a target instance where the pair (I, J) satisfies the constraints of $\mathcal{M}$, and a universal solution is a solution with a homomorphism to every solution.) The fact that the chase is guaranteed to terminate (whether in polynomial time or otherwise) implies that if there is a solution for a given source instance I, then there is a universal solution. The fact that the chase runs in polynomial time guarantees that there is a polynomial-time algorithm for deciding if there is a solution, and, if so, for producing a universal solution.

Let q be a query posed against the target schema. The *certain answers* for q on a source instance I, with respect to a schema mapping $\mathcal{M}$, are those tuples that appear in the answer $q(J)$ for every solution J for I. It is shown in [20] that if q is a union of conjunctive queries, and J^* is a universal solution for I, then the certain answers for q on I can be obtained by evaluating q on J^* and then keeping only those tuples formed entirely of values from I. Since the chase using an st-SO dependency can be carried out in polynomial time, it follows that we can obtain a universal solution in polynomial time, and so we can compute the certain answers to unions of conjunctive queries in polynomial time.

In addition to our results about st-SO dependencies, we also have some results directly about compositions of schema mappings. It is easy to show that the composition of an arbitrary number of standard schema mappings is equivalent to the composition of only two standard schema mappings. We show the surprising result that a similar result holds also for st-tgd mappings (no target constraints). That is, the composition of an arbitrary number of st-tgd mappings is equivalent to the composition of only two st-tgd mappings. This is proven by showing that every SO tgd is equivalent to an unnested SO tgd (one where there is no nesting of function symbols). We also prove a similar denesting result for st-SO dependencies. These denesting results are the most difficult results technically in the paper.

We feel that unnested dependencies are more natural, more readable, and easier to understand than nested dependencies. They are probably easier to use in practice. For example, it is easy to see that the "nested mappings" in [23] can be expressed by unnested SO tgds. We show that unnested SO tgds are also expressive enough to

specify the composition of an arbitrary number of st-tgd mappings. This was not even known for the composition of two st-tgd mappings. Thus, although it was shown in [21] that each unnested SO tgd specifies the composition of some pair of st-tgd mappings, the converse was not shown. In fact, for the composition of two st-tgd mappings, the composition construction in [21] can produce an SO tgd with nesting depth 2, not 1.

We close by discussing an application of our results. In practice, a composition of many schema mappings may arise (say, as the result of many steps of schema evolution). If these are st-tgd mappings, then there are several ways to "simplify" this composition. One solution is to replace the composition of many st-tgd mappings by a single schema mapping, specified by an unnested SO tgd. For another solution, we can remain within the language of st-tgds by replacing the composition of many st-tgd mappings by the composition of only two st-tgd mappings. A similar comment applies to the composition of many standard schema mappings.

2. PRELIMINARIES

A *schema* $\mathbf{R}$ is a finite set $\{R_1, \ldots, R_k\}$ of relation symbols, with each R_i having a fixed arity $n_i > 0$. Let $\mathbf{D}$ be a countably infinite domain. An *instance* I of $\mathbf{R}$ assigns to each relation symbol R_i of $\mathbf{R}$ a finite n_i-ary relation $R_i^I \subseteq \mathbf{D}^{n_i}$. The *domain* (or *active domain*) $\mathrm{dom}(I)$ of instance I is the set of all elements that occur in any of the relations R_i^I. We say that $R(a_1, \ldots, a_n)$ is a *fact* of I if $(a_1, \ldots, a_n) \in R^I$. We sometimes denote an instance by its set of facts.

As is customary in the data exchange literature, we consider instances with two types of values: constants and nulls [20]. More precisely, let $\mathbf{C}$ and $\mathbf{N}$ be infinite and disjoint sets of constants and nulls, respectively, and assume that $\mathbf{D} = \mathbf{C} \cup \mathbf{N}$. If we refer to a schema $\mathbf{S}$ as a *source* schema, then we assume that for every instance I of $\mathbf{S}$, it holds that $\mathrm{dom}(I) \subseteq \mathbf{C}$. On the other hand, if we refer to a schema $\mathbf{T}$ as a *target* schema, then for every instance J of $\mathbf{S}$, it holds that $\mathrm{dom}(J) \subseteq \mathbf{C} \cup \mathbf{N}$. The distinction between constants and nulls is important in the definition of a homomorphism (which we give later).

2.1 Source-to-target and target dependencies

Fix a source schema $\mathbf{S}$ and a target schema $\mathbf{T}$, and assume that $\mathbf{S}$ and $\mathbf{T}$ do not have predicate symbols in common. Then a *source-to-target tuple-generating dependency (st-tgd)* is a first-order sentence of the form:

$$\forall \bar{x}\,(\varphi(\bar{x}) \rightarrow \exists \bar{y}\,\psi(\bar{x}, \bar{y})),$$

where $\varphi(\bar{x})$ is a conjunction of relational atoms over $\mathbf{S}$ and $\psi(\bar{x}, \bar{y})$ is a conjunction of relational atoms over $\mathbf{T}$. We assume a safety condition, that every member of $\bar{x}$ actually appears in a relational atom in $\varphi(\bar{x})$. A *target equality-generating dependency (t-egd)* is a first-order sentence of the form:

$$\forall \bar{x}\,(\varphi(\bar{x}) \rightarrow u = v),$$

where $\varphi(\bar{x})$ is a conjunction of relational atoms over $\mathbf{T}$ and u, v are among the variables mentioned in $\bar{x}$. We again assume the safety condition. In several of the examples we give in this paper, we shall make use of special t-egds called *key dependencies*, which say that one attribute of a binary relation is a key for that relation (of course, we could define more general key dependencies if we wanted). The key dependencies we consider are either of the form $R(x, y) \wedge R(x, z) \rightarrow y = z$ (which says that the first attribute is a key) or $S(y, x) \wedge S(z, x) \rightarrow y = z$ (which says that the second attribute is a key). Finally, a *target tuple-generating dependency (t-tgd)* is a first-order sentence of the form:

$$\forall \bar{x}\,(\varphi(\bar{x}) \rightarrow \exists \bar{y}\,\psi(\bar{x}, \bar{y})),$$

where both $\varphi(\bar{x})$ and $\psi(\bar{x}, \bar{y})$ are conjunctions of relational atoms over $\mathbf{T}$, and where we again assume the safety condition.

The notion of satisfaction of a t-egd α by a target instance J, denoted by $J \models \alpha$, is defined as the standard notion of satisfaction in first-order logic, and likewise for t-tgds. For the case of an st-tgd α, a source instance I and a target instance J, the pair (I, J) is said to satisfy α, denoted by $(I, J) \models \alpha$, if the following instance instance K of $\mathbf{S} \cup \mathbf{T}$ satisfies α in the standard first-order logic sense. For every relation name $S \in \mathbf{S}$, relation S^K is defined as S^I, and for every relation name $T \in \mathbf{T}$, relation T^K is defined as T^J. As usual, a set Σ_{st} of st-tgds is said to be satisfied by a pair (I, J), denoted by $(I, J) \models \Sigma_{st}$, if $(I, J) \models \alpha$ for every $\alpha \in \Sigma_{st}$ (and likewise for a set of t-egds and t-tgds).

2.2 Schema mappings

In general, a *schema mapping* from a source schema $\mathbf{S}$ to a target schema $\mathbf{T}$ is a set of pairs (I, J), where I is an instance of $\mathbf{S}$ and J is an instance of $\mathbf{T}$. In this paper, we restrict our attention to some classes of schema mappings that are *specified* in some logical formalisms. We may sometimes refer to two schema mappings with the same set of (I, J) pairs as *equivalent*, to capture the idea that the formulas that specify them are logically equivalent. A schema mapping from $\mathbf{S}$ to $\mathbf{T}$ is said to be an *st-tgd mapping* if there exists a set Σ_{st} of st-tgds such that (I, J) belongs to $\mathcal{M}$ if and only if $(I, J) \models \Sigma_{st}$, for every pair I, J of instances of $\mathbf{S}$ and $\mathbf{T}$, respectively. We use notation $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ to indicate that $\mathcal{M}$ is specified by Σ_{st}. Moreover, a schema mapping $\mathcal{M}$ from $\mathbf{S}$ to $\mathbf{T}$ is said to be a *standard schema mapping* if there exists a set Σ_{st} of st-tgds and a set Σ_t consisting of a set of t-egds and a *weakly-acyclic* set of t-tgds, such that (I, J) belongs to $\mathcal{M}$ if and only if $(I, J) \models \Sigma_{st}$ and $J \models \Sigma_t$, for every pair I, J of instances of $\mathbf{S}$ and $\mathbf{T}$, respectively; notation $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ is used in this case to indicate that $\mathcal{M}$ is specified by Σ_{st} and Σ_t. We occasionally allow a set Σ_s of source constraints in some of our schema mappings: we then use the notation $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_s, \Sigma_{st}, \Sigma_t)$.

To define the widely used notion of weak-acyclicity, we need to introduce some terminology. For a set Γ of t-tgds over $\mathbf{T}$, define the dependency graph G_Γ of Γ as follows.

- For every relation name T in $\mathbf{T}$ of arity n, and for every $i \in \{1, \ldots, n\}$, include a node (T, i) in G_Γ.
- Include an edge $(T_1, i) \rightarrow (T_2, j)$ in G_Γ if there exist a t-tgd $\forall \bar{x}(\varphi(\bar{x}) \rightarrow \exists \bar{y}\,\psi(\bar{x}, \bar{y}))$ in Γ and a variable x in $\bar{x}$ such that, x occurs in the i-th attribute of T_1 in a conjunct of φ and in the j-th attribute of T_2 in a conjunct of ψ.
- Include a *special* edge $(T_1, i) \rightarrow^* (T_2, j)$ in G_Γ if there exist a t-tgd $\forall \bar{x}(\varphi(\bar{x}) \rightarrow \exists \bar{y}\,\psi(\bar{x}, \bar{y}))$ in Γ and variables x, y in $\bar{x}$ and $\bar{y}$, respectively, such that x occurs in the i-th attribute of T_1 in a conjunct of φ and y occurs in the j-th attribute of T_2 in a conjunct of ψ.

Then set Γ of t-tgds is said to be *weakly acyclic* if its dependency graph G_Γ has no cycle through a special edge [20].

Given a schema mapping $\mathcal{M}$, if a pair (I, J) belongs to it, then J is said to be a *solution* for I under $\mathcal{M}$. A *universal solution* [20] for I is a solution with a homomorphism to every solution for I. A *homomorphism* from instance J_1 to instance J_2 is a function h from $\mathbf{C} \cup \mathbf{N}$ to $\mathbf{C} \cup \mathbf{N}$ such that (1) for each c in $\mathbf{C}$, we have that $h(c) = c$, and (2) whenever $R(a_1, \ldots, a_n)$ is a fact of J_1, then $R(h(a_1), \ldots, h(a_n))$ is a fact of J_2.

2.3 Second-order dependencies

In this paper, we also consider schema mappings that are specified by second-order dependencies. In the definition of these dependencies, the following terminology is used. Given a collection $\bar{x}$ of variables and a collection $\bar{f}$ of function symbols, a *term (based on $\bar{x}$ and $\bar{f}$) with depth of nesting d* is defined recursively as follows:

1. Every member of $\bar{x}$ and every 0-ary function symbol (constant symbol) of $\bar{f}$ is a term with depth of nesting 0.
2. If f is a k-ary function symbol in $\bar{f}$ with $k \geq 1$, and if $t_1, \ldots, t_k$ are terms, with maximum depth of nesting $d - 1$, then $f(t_1, \ldots, t_k)$ is a term with depth of nesting d.

Then, given a source schema $\mathbf{S}$ and a target schema $\mathbf{T}$, a second-order source-to-target tuple-generating dependency (SO tgd) from $\mathbf{S}$ to $\mathbf{T}$ is a second-order formula of the form [21]:

$$\exists \bar{f} \, (\forall \bar{x}_1(\varphi_1 \rightarrow \psi_1) \wedge \cdots \wedge \forall \bar{x}_n(\varphi_n \rightarrow \psi_n)),$$

where

1. Each member of $\bar{f}$ is a function symbol.
2. Each φ_i is a conjunction of
   - relational atomic formulas of the form $S(y_1, \ldots, y_k)$, where S is a k-ary relation symbol of $\mathbf{S}$ and $y_1, \ldots, y_k$ are (not necessarily distinct) variables in $\bar{x}_i$, and
   - equality atoms of the form $t = t'$, where t and t' are terms based on $\bar{x}_i$ and $\bar{f}$.
3. Each ψ_i is a conjunction of relational atomic formulas of the form $T(t_1, \ldots, t_\ell)$, where T is an ℓ-ary relation symbol of $\mathbf{T}$ and $t_1, \ldots, t_\ell$ are terms based on $\bar{x}_i$ and $\bar{f}$.
4. Each variable in $\bar{x}_i$ appears in some relational atomic formula of φ_i.

The fourth condition is the safety condition for SO tgds. Note that it is "built into" SO tgds that they are source-to-target. The depth of nesting of an SO tgd is the maximal depth of nesting of the terms that appear in it. We say that the SO tgd is *unnested* if its depth of nesting is at most 1. Thus, an unnested SO tgd can contain terms like $f(x)$, but not terms like $f(g(x))$.

As was noted in [21, 40], there is a subtlety in the semantics of SO tgds, namely, the semantics of existentially quantified function symbols. In particular, in deciding whether $(I, J) \models \sigma$, for an SO tgd σ, what should the domain and range of the functions instantiating the existentially quantified function symbols be? The obvious choice is to let the domain and range be the active domain of (I, J), but it is shown in [21, 40] that this does not work properly. Instead, the solution in [21, 40] is as follows. Let σ be an SO tgd from a source schema $\mathbf{S}$ to a target schema $\mathbf{T}$. Then given an instance I of $\mathbf{S}$ and an instance J of $\mathbf{T}$, instance (I, J) is converted into a structure $(U; I, J)$, which is just like (I, J) except that it has a *universe* U. The domain and range of the functions in σ is then taken to be U. The universe U is taken to be a countably infinite set that includes $\mathrm{dom}(I) \cup \mathrm{dom}(J)$. The intuition is that the universe contains the active domain along with an infinite set of nulls. Then (I, J) is said to satisfy σ, denoted by $(I, J) \models \sigma$, if $(U; I, J) \models \sigma$ under the standard notion of satisfaction in second-order logic (see, for example, [16]). It should be noticed that it is proven in [21] that in the case of SO tgds, instead of taking the universe U to be infinite, one can take it to be finite and "sufficiently large", whereas in [40] this is shown to be insufficient in the presence of unrestricted target constraints.

The class of SO tgds was introduced in [21] to deal with the problem of composing schema mappings. More specifically, given a schema mapping $\mathcal{M}_{12}$ from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$ and a schema mapping $\mathcal{M}_{23}$ from $\mathbf{S}_2$ to a schema $\mathbf{S}_3$, the composition of these two schemas, denoted by $\mathcal{M}_{12} \circ \mathcal{M}_{23}$, is defined as the schema mapping consisting of all pairs (I_1, I_3) of instances for which there exists an instance I_2 of $\mathbf{S}_2$ such that (I_1, I_2) belong to $\mathcal{M}_{12}$ and (I_2, I_3) belong to $\mathcal{M}_{23}$. It was shown in [21] that the composition of an arbitrary number of st-tgd mappings is defined by an SO tgd, and also that SO tgds are closed under composition.

3. A NEGATIVE RESULT: SO TGDS ARE NOT ENOUGH

In [21], SO tgds were introduced to deal with the problem of composing schema mappings. As we noted, it was proved in [21] that the composition of a finite number of st-tgd mappings can always be specified by an SO tgd, and also that SO tgds are closed under composition. Thus, SO tgds are a natural starting point for the study of languages for defining the composition of schema mappings with target constraints, which is the goal of this paper. Unfortunately, it can be easily proved that this language is not rich enough to be able to specify the composition of some simple schema mappings with target constraints. We now give an example.

EXAMPLE 3.1. Let $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12}, \Sigma_2)$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$, where $\mathbf{S}_1 = \{P(\cdot, \cdot)\}$, $\mathbf{S}_2 = \{R(\cdot, \cdot)\}$, $\mathbf{S}_3 = \{T(\cdot, \cdot)\}$ and

$$\begin{aligned} \Sigma_{12} &= \{P(x, y) \rightarrow R(x, y)\}, \\ \Sigma_2 &= \{R(x, y) \wedge R(x, z) \rightarrow y = z\}, \\ \Sigma_{23} &= \{R(x, y) \rightarrow T(x, y)\}. \end{aligned}$$

Notice that Σ_2 consists of a key dependency over $\mathbf{S}_2$.

If a schema mapping $\mathcal{M}$ from $\mathbf{S}_1$ to $\mathbf{S}_3$ is specified by an SO tgd, then for every instance I_1 of $\mathbf{S}_1$, there exists an instance I_3 of $\mathbf{S}_3$ such that $(I_1, I_3) \in \mathcal{M}$. On the contrary, for the instance I_1 of $\mathbf{S}_1$ such that $P^{I_1} = \{(1, 2), (1, 3)\}$, there is no instance I_3 of $\mathbf{S}_3$ such that $(I_1, I_3) \in \mathcal{M}_{12} \circ \mathcal{M}_{23}$, since I_1 does not have any solutions under $\mathcal{M}_{12}$. Thus, the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ cannot be specified by an SO tgd. □

From the previous example, we obtain the following proposition.

PROPOSITION 3.2. *There exist schema mappings* $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12}, \Sigma_2)$ *and* $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$*, where* Σ_{12} *and* Σ_{23} *are sets of st-tgds and* Σ_2 *is a set of key dependencies, such that* $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ *cannot be specified by an SO tgd.*

Proposition 3.2 does not rule out the possibility that the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ can be specified by using an SO tgd together with some source and target constraints. In fact, the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ can be specified by a set of st-tgds together with some source constraints: $\mathcal{M}_{12} \circ \mathcal{M}_{23} = \mathcal{M}_{13}$. where $\mathcal{M}_{13} = (\mathbf{S}_1, \mathbf{S}_3, \Sigma_1, \Sigma_{13})$ and

$$\begin{aligned} \Sigma_1 &= \{P(x,y) \wedge P(x,z) \rightarrow y = z\}, \\ \Sigma_{13} &= \{P(x,y) \rightarrow T(x,y)\}. \end{aligned}$$

A natural question is then whether the language of SO tgds together with source and target constraints is the right language for defining the composition of schema mappings with source and target constraints. Unfortunately, the following theorem shows that this is not the case.

THEOREM 3.3. *There exist schema mappings* $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12}, \Sigma_2)$ *and* $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$*, where* Σ_{12} *and* Σ_{23} *are sets of st-tgds and* Σ_2 *is a set of key dependencies, such that* $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ *cannot be specified by any schema mapping of the form* $(\mathbf{S}_1, \mathbf{S}_3, \sigma_1, \sigma_{13}, \sigma_3)$*, where* σ_1 *is an arbitrary source constraint,* σ_{13} *is an SO tgd, and* σ_3 *is an arbitrary target constraint.*

If we view a source constraint as a set of allowed source instances, then when we say that σ_1 is an "arbitrary source constraint" in Theorem 3.3, we mean that σ_1 allows an arbitrary set of source instances. A similar comment applies to σ_3 being an "arbitrary target constraint".

To prove this theorem, we use a notion of locality from [5]. Notions of locality [22, 25, 30, 35] have been widely used to prove inexpressibility results for first-order logic (FO) and some of its extensions. The intuition underlying those notions of locality is that FO cannot express properties (such as connectivity, cyclicity, etc.) that involve nontrivial recursive computations. The setting of locality is as follows. The *Gaifman graph* $\mathcal{G}(I)$ of an instance I of a schema $\mathbf{S}$ is the graph whose nodes are the elements of $\mathrm{dom}(I)$, and such that there exists an edge between a and b in $\mathcal{G}(I)$ if and only if a and b belong to the same tuple of a relation R^I, for some $R \in \mathbf{S}$. For example, if I is an undirected graph, then $\mathcal{G}(I)$ is I itself. The distance between two elements a and b in I is considered to be the distance between them in $\mathcal{G}(I)$. Given $a \in \mathrm{dom}(I)$, the instance $N_d^I(a)$, called the *d-neighborhood of a in I*, is defined as the restriction of I to the elements at distance at most d from a, with a treated as a distinguished element (a constant in the vocabulary).

The notion of neighborhood of a point is used in [5] to introduce the following notion of locality for data transformations. In this definition, $N_d^I(a) \equiv_k N_d^I(b)$ indicates that $N_d^I(a)$ and $N_d^I(b)$ agree on all FO-sentences of quantifier rank at most k, that is, for every FO-sentence φ of quantifier rank at most k, we have that $N_d^I(a) \models \varphi$ if and only if $N_d^I(b) \models \varphi$, where the quantifier rank of a formula φ is the maximum depth of quantifier nesting in it.

DEFINITION 3.4. ([5]). Given a source schema $\mathbf{S}$ and a target schema $\mathbf{T}$, a mapping $\mathfrak{F} : \mathbf{S} \rightarrow \mathbf{T}$ is *locally consistent under* FO*-equivalence* if for every $r, \ell \geq 0$ there exist $d, k \geq 0$ such that, for every instance I of $\mathbf{S}$ and $a, b \in \mathrm{dom}(I)$, if $N_d^I(a) \equiv_k N_d^I(b)$, then

1. $a \in \mathrm{dom}(\mathfrak{F}(I))$ if and only if $b \in \mathrm{dom}(\mathfrak{F}(I))$, and
2. $N_r^{\mathfrak{F}(I)}(a) \equiv_\ell N_r^{\mathfrak{F}(I)}(b)$.

For a fixed schema mapping $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$, we denote by $\mathfrak{F}_{\mathrm{can}}$ the transformation from $\mathbf{S}$ to $\mathbf{T}$, such that $\mathfrak{F}_{\mathrm{can}}(I)$ is the canonical universal solution for I. In [5], it was shown that $\mathfrak{F}_{\mathrm{can}}$ is locally consistent under FO-equivalence for schema mappings specified by st-tgds.

PROPOSITION 3.5 ([5]). *For every st-tgd mapping, the transformation* $\mathfrak{F}_{\mathrm{can}}$ *is locally consistent under* FO*-equivalence.*

The previous proposition can be easily extended to the case of a composition of a finite number of st-tgd mappings.

LEMMA 3.6. *Let* $n \geq 2$*. For every* $i \in [1, n-1]$*, let* $\mathcal{M}_i = (\mathbf{S}_i, \mathbf{S}_{i+1}, \Sigma_{i\,i+1})$ *be a schema mapping specified by a set* $\Sigma_{i\,i+1}$ *of st-tgds, and* $\mathfrak{F}_{\mathrm{can}}^i$ *be the canonical universal solution transformation for* $\mathcal{M}_i$*. Assume that* $\mathfrak{F}$ *is the transformation from* $\mathbf{S}_1$ *to* $\mathbf{S}_n$ *defined as:*

$$\mathfrak{F}(I_1) = \mathfrak{F}_{\mathrm{can}}^{n-1}(\cdots(\mathfrak{F}_{\mathrm{can}}^2(\mathfrak{F}_{\mathrm{can}}^1(I_1)))\cdots),$$

for every instance I_1 *of* $\mathbf{S}_1$*. Then* $\mathfrak{F}$ *is locally consistent under* FO*-equivalence.*

Lemma 3.6 is one of the key components in the proof of Theorem 3.3. We give here a sketch of the proof of Theorem 3.3.

PROOF SKETCH OF THEOREM 3.3. Let $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12}, \Sigma_2)$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$ be schema mappings, where:

$$\begin{aligned} \mathbf{S}_1 &= \{E(\cdot,\cdot), P_1(\cdot), Q_1(\cdot)\}, \\ \mathbf{S}_2 &= \{P_2(\cdot), Q_2(\cdot), R(\cdot,\cdot), S(\cdot,\cdot)\}, \\ \mathbf{S}_3 &= \{V(\cdot)\}, \end{aligned}$$

and

$$\begin{aligned} \Sigma_{12} = \{&P_1(x) \rightarrow P_2(x), \\ &Q_1(x) \rightarrow Q_2(x), \\ &E(x,y) \rightarrow \exists z_1 \exists z_2 \exists z_3\,(R(x,z_1) \wedge R(y,z_2) \wedge \\ &\qquad S(z_1,z_3) \wedge S(z_2,z_3))\}, \\ \Sigma_2 = \{&R(x,y) \wedge R(x,z) \rightarrow y = z, \\ &S(x,y) \wedge S(x,z) \rightarrow y = z, \\ &S(y,x) \wedge S(z,x) \rightarrow y = z\}, \\ \Sigma_{23} = \{&P_2(x) \wedge R(x,z) \wedge R(y,z) \wedge Q_2(y) \rightarrow V(x)\}. \end{aligned}$$

First, we note that for every instance I_1 of $\mathbf{S}_1$, there exists an instance I_3 of $\mathbf{S}_3$ such that $(I_1, I_3) \in \mathcal{M}_{12} \circ \mathcal{M}_{23}$, and for every instance I_3' of $\mathbf{S}_3$, there exists an instance I_1' of $\mathbf{S}_1$ such that $(I_1', I_3') \in \mathcal{M}_{12} \circ \mathcal{M}_{23}$. From these two properties, we conclude that source and target constraints cannot be used in defining the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$.

Thus, we need only show that $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ cannot be specified by an SO tgd. For the sake of contradiction, we assume that schema mapping $\mathcal{M}_{13} = (\mathbf{S}_1, \mathbf{S}_3, \sigma_{13})$ defines this composition, where σ_{13} is an SO tgd. From Theorem 8.2 in [21], we know that every SO tgd is equivalent to the composition of a finite number of st-tgd mappings. Thus, given that $\mathcal{M}_{13}$ defines the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, we have that there exist schema mappings $\mathcal{M}'_1 = (\mathbf{S}'_1, \mathbf{S}'_2, \Sigma'_{12})$, ..., $\mathcal{M}'_{n-1} = (\mathbf{S}'_{n-1}, \mathbf{S}'_n, \Sigma'_{n-1\,n})$ such that $n \geq 2$, $\mathbf{S}'_1 = \mathbf{S}_1$, $\mathbf{S}'_n = \mathbf{S}_3$, $\Sigma'_{i\,i+1}$ is a set of st-tgds for every $i \in \{1, \ldots, n-1\}$, and $\mathcal{M}'_1 \circ \ldots \circ \mathcal{M}'_{n-1}$ defines the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, that is, for every pair of instances $(I_1, I_3) \in \mathbf{S}_1 \times \mathbf{S}_3$:

$$(I_1, I_3) \in \mathcal{M}_{12} \circ \mathcal{M}_{23} \;\Leftrightarrow\; (I_1, I_3) \in \mathcal{M}'_1 \circ \ldots \circ \mathcal{M}'_{n-1}.$$

For every $i \in \{1, \ldots, n-1\}$, let $\mathfrak{F}^i_{\text{can}}$ be the canonical solution transformation for $\mathcal{M}'_i$, and assume that $\mathfrak{F}$ is the transformation defined as $\mathfrak{F}(I_1) = \mathfrak{F}^{n-1}_{\text{can}}(\cdots(\mathfrak{F}^2_{\text{can}}(\mathfrak{F}^1_{\text{can}}(I_1)))\cdots)$ for every instance I_1 of $\mathbf{S}_1$. From Lemma 3.6, we have that $\mathfrak{F}$ is locally consistent under FO-equivalence. In particular, we have that for $r = 1$ and $\ell = 1$, there exist $d, k \geq 0$ such that for every instance I_1 of $\mathbf{S}_1$ and for every $a, b \in \text{dom}(I_1)$, if $N^{I_1}_d(a) \equiv_k N^{I_1}_d(b)$, then $a \in \text{dom}(\mathfrak{F}(I_1))$ if and only if $b \in \text{dom}(\mathfrak{F}(I_1))$.

To obtain a contradiction, and thus conclude the proof, it is enough to show an instance I_1 of $\mathbf{S}_1$ and elements $a, b \in \text{dom}(I_1)$ such that $N^{I_1}_d(a) \equiv_k N^{I_1}_d(b)$, $a \in \text{dom}(\mathfrak{F}(I_1))$ and $b \notin \text{dom}(\mathfrak{F}(I_1))$. Such an instance is defined as follows: $P^{I_1}_1 = \{a, b\}$, $Q^{I_1}_1 = \{c\}$, and E^{I_1} contains the following tuples:

$$a \xrightarrow{E} a_1 \xrightarrow{E} \cdots \xrightarrow{E} a_d \xrightarrow{E} c$$

$$b \xrightarrow{E} b_1 \xrightarrow{E} \cdots \xrightarrow{E} b_d$$

As shown in the figure, E^{I_1} is a union of two paths, one containing $d+2$ elements with first element a and last element c, and another one containing $d+1$ elements with first element b. Observe that $N^{I_1}_d(a) \equiv_k N^{I_1}_d(b)$ since $N^{I_1}_d(a)$ is isomorphic to $N^{I_1}_d(b)$, with a and b treated as distinguished elements. But not only that, it is also possible to prove that $a \in \text{dom}(\mathfrak{F}(I_1))$ and $b \notin \text{dom}(\mathfrak{F}(I_1))$, which leads us to the aforementioned contradiction. □

4. SOURCE-TO-TARGET SO DEPENDENCIES

In Section 3, we showed that SO tgds, even with the assistance of arbitrary source constraints and arbitrary target constraints, cannot always be used to specify the composition of mappings with target constraints, even if only key dependencies are allowed as target constraints. In this paper, we define a richer class, called source-to-target SO dependencies (st-SO dependencies). This class of dependencies is the source-to-target restriction of the class Sk$\forall$CQ$^=$ of dependencies introduced in [40]. We show that st-SO dependencies (together with appropriate target constraints) are the right extension of SO tgds for the purpose of expressing the composition of standard schema mappings. The definition of st-SO dependencies is exactly like the definition of SO tgds in Section 2, except that condition (3) is changed to:

3. Each ψ_i is a conjunction of
   - relational atomic formulas of the form $T(t_1, \ldots, t_\ell)$, where T is an ℓ-ary relation symbol of $\mathbf{T}$ and $t_1, \ldots, t_\ell$ are terms based on $\bar{x}_i$ and $\bar{f}$, and
   - equality atoms of the form $t = t'$, where t and t' are terms based on $\bar{x}_i$ and $\bar{f}$.

Let Φ be the st-SO dependency $\exists \bar{f}\,(\forall \bar{x}_1(\varphi_1 \rightarrow \psi_1) \wedge \cdots \wedge \forall \bar{x}_n(\varphi_n \rightarrow \psi_n))$. From now on, we say that $\forall \bar{x}_i(\varphi_i \rightarrow \psi_i)$ is an *SO tgd part* of Φ if ψ_i is a conjunction of relational atomic formulas of the form $T(t_1, \ldots, t_\ell)$, and we say that $\forall \bar{x}_i(\varphi_i \rightarrow \psi_i)$ is an *SO egd part* of Φ if ψ_i is an equality atom of the form $t = t'$. Note that if γ_t is the conjunction of the SO tgd parts of Φ, and γ_e is the conjunction of the SO egd parts of Φ, then Φ is equivalent to the formula $\exists \bar{f}(\gamma_t \wedge \gamma_e)$.

We adopt the same convention for the semantics of st-SO dependencies as was given in Section 2 for SO tgds, by assuming the existence of a countably infinite universe that includes the active domain.

We shall show that the composition of a finite number of standard schema mappings is given by a schema mapping specified by an st-SO dependency, together with t-egds and a weakly acyclic set of t-tgds. It is convenient to give these latter schema mappings a name. To emphasize the similarity of these second-order schema mappings with the first-order case, we shall refer to these schema mappings as *SO-standard*. Thus, an SO-standard schema mapping is one that is specified by an st-SO dependency, together with t-egds and a weakly acyclic set of t-tgds.

Note that st-SO dependencies, like SO tgds, are closed under conjunction. That is, the conjunction of two st-SO dependencies is equivalent to a single st-SO dependency. This is why we define an SO-standard schema mapping to have only one st-SO dependency, not several. Note also that every finite set of st-tgds can be expressed with an SO tgd, and so with an st-SO dependency. In particular, every standard schema mapping is an SO-standard schema mapping.

5. THE CHASE FOR ST-SO DEPENDENCIES

In [21], the well-known chase process is extended so that it applies to an SO tgd Φ. The idea is that each SO tgd part of Φ is treated like a tgd (of course, the conclusion contains Skolem functions rather than existential quantifiers). In deciding whether the premise of the SO tgd part is instantiated in the instance being chased, two terms are treated as equal precisely if they are syntactically identical. So a premise containing the equality atom $f(x) = g(y)$ automatically fails to hold over an instance, and a premise containing the equality atom $f(g(x)) = f(g(y))$ automatically fails to hold over an instance unless the instantiation of x equals the instantiation of y.

In this section, we discuss how the chase can be extended to apply to an st-SO dependency. We note that in [40], a chase procedure for the dependencies studied there (which are like ours but not necessarily source-to-target) was introduced. However, their chase was not procedural, in that their chase procedure says to set terms t_1 and t_2 to be equal when the dependencies logically imply that $t_1 = t_2$. Because of our source-to-target restriction, we are able to give an explicit, polynomial-time procedure for equating terms.

For space reasons, we keep the discussion here informal. In chasing

an instance I with an st-SO dependency Φ, we chase first with all of the SO egd parts of Φ, and then we chase with all of the SO tgd parts of Φ. We no longer consider two terms to be equal precisely if they are syntactically identical, since an SO egd part may force, say, $f(0)$ and $g(1)$ to be equal, even though $f(0)$ and $g(1)$ are not syntactically identical.

Given a source instance I and an st-SO dependency Φ, we now describe how to chase I with the SO egd parts of Φ. Let D be the active domain of I (by our assumptions, D consists of constants only). Let n be the maximal depth of nesting over all terms that appear in Φ. Let $\bar{f}$ consist of the function symbols that appear in Φ. Let T be the set of terms based on D and $\bar{f}$ that have depth of nesting at most n. This set T is sometimes called the *Herbrand universe* (with respect to D and $\bar{f}$) of depth n. It is straightforward to see (by induction on depth) that the size of T is polynomial in the size of D, for a fixed choice of Φ. We note that if we define T' to be the subset of T that consists of all terms $t(\bar{a})$, where $t(\bar{x})$ is a subterm of Φ, and $\bar{a}$ is the result of replacing members of $\bar{x}$ by values in D, then we could work just as well with T' as with T in defining the chase. However, the proofs are easier to give using T instead of T'.

We now define a function F with domain the members of T. The values $F(t)$ are stored in a table that is updated repeatedly during the chase process. If a is a member of D, then the initial value of $F(a)$ is a itself (in fact, the value of $F(a)$ will never change for members a of D). If t is a member of T that is not in D (so that t is of the form $f(t_1, \ldots, t_k)$ for some function symbol f), then $F(t)$ is initially taken to be a new null value. As we change F, we shall maintain the invariant that if $f(t_1, \ldots, t_k)$ and $f(t'_1, \ldots, t'_k)$ are members of T where $F(t_i) = F(t'_i)$, for $1 \leq i \leq k$, then $F(f(t_1, \ldots, t_k)) = F(f(t'_1, \ldots, t'_k))$. This is certainly true initially, since F is initially one-to-one on members of T.

Let N be the set of all of the new null values (the values initially assigned to $F(t)$ when t is not in D). We create an ordering $\prec$ on $D \cup N$, where the members of D are an initial segment of the ordering $\prec$, followed by the members of N.

We now begin chasing I with the SO egd parts of Φ, to change the values of F. Whenever t is a member of T such that we replace a current value of $F(t)$ by a new value during the chase process, we will always replace the current value of $F(t)$ by a value that is lower in the ordering $\prec$. If $s_1(\bar{y}_1) = s_2(\bar{y}_2)$ is an equality in the premise of an SO egd part of Φ, then the equality $s_1(\bar{e}_1) = s_2(\bar{e}_2)$ evaluates to "true" where $\bar{e}_1$ and $\bar{e}_2$ consist of members of D, precisely if the current value of $F(s_1(\bar{e}_1))$ equals the current value of $F(s_2(\bar{e}_2))$. Each time an equality $t_1(\bar{a}) = t_2(\bar{b})$ is forced (because of an SO egd part with conclusion $t_1(\bar{x}) = t_2(\bar{y})$), and the current value of $F(t_1(\bar{a}))$ does not equal the current value of $F(t_2(\bar{b}))$ we proceed as follows. Let c_1 be the smaller of these two values and let c_2 be the larger of these two values in our ordering $\prec$. If c_2 is a constant, then the chase fails and halts. Otherwise, for every member s of T where the current value of $F(s)$ is c_2, change the value so that the new value of $F(s)$ is c_1. Note that under this change, the new value of $F(t_1(\bar{a}))$ and the new value of of $F(t_2(\bar{b}))$ are the same (namely, c_1).

These changes in F may propagate new changes in F, which we need to make in order to maintain the invariant. Assume that as a result of our changes in F so far, there are terms $f(t_1, \ldots, t_k)$ and $f(t'_1, \ldots, t'_k)$ in T where $F(t_i) = F(t'_i)$, for $1 \leq i \leq k$, but $F(f(t_1, \ldots, t_k))$ and $F(f(t'_1, \ldots, t'_k))$ are different. As before, let c_1 be the smaller of these two values and let c_2 be the larger of these two values in our ordering $\prec$. If c_2 is a constant, then the chase fails and halts. Otherwise, for every member s of T where the current value of $F(s)$ is c_2, change the value so that the new value of $F(s)$ is c_1. Note that under this change, the new value of $F(f(t_1, \ldots, t_k))$ and the new value of $F(f(t'_1, \ldots, t'_k))$ are the same (namely, c_1). Continue this process until no more changes occur. It is easy to see that we have maintained our invariant. Continue chasing with SO egd parts until no more changes occur. Note that at most as many changes can occur as the size of T, since every time a change occurs, there are strictly fewer values of $F(t)$ as t ranges over T. This is the key reason why the chase runs in polynomial time.

Once F has stabilized, so that no more changes are caused by chasing with the SO egd parts of Φ, then chase I with the SO tgd parts of Φ. If $s_1(\bar{y}_1) = s_2(\bar{y}_2)$ is an equality in the premise of an SO tgd part of Φ, then the equality $s_1(\bar{e}_1) = s_2(\bar{e}_2)$ evaluates to "true" where $\bar{e}_1$ and $\bar{e}_2$ consist of members of I, precisely if $F(s_1(\bar{e}_1)) = F(s_2(\bar{e}_2))$. These chase steps produce the target relation J that is taken to be the result of the chase.

We have the following theorem about the chase process.

THEOREM 5.1. *Let Φ be a fixed st-SO dependency. The chase of a ground instance I with Φ runs in time polynomial in the size of I. The chase fails precisely if there is no solution for I with respect to Φ. If the chase succeeds, then it produces a universal solution for I with respect to Φ.*

Note that in particular, Theorem 5.1 tells us that there is a polynomial-time algorithm for determining, given a source instance I, whether there is a solution for I, and if so, producing a universal solution for I.

Because there is a polynomial-time chase for st-SO dependencies, there is also a polynomial-time chase for SO-standard schema mappings: first, chase with the st-SO dependency, and then with the target dependencies (where there is a polynomial-time chase because of the weak-acyclicity assumption). It follows that there is a polynomial-time algorithm for obtaining a universal solution for an SO-standard schema mapping (if there is a solution).

As shown in [20], we can use a universal solution to obtain the certain answers to unions of conjunctive queries in polynomial time. We now recall the definition of the certain answers. Given a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, an instance I over the source schema $\mathbf{S}$ and a k-ary query q posed against the target schema $\mathbf{T}$, the *certain answers of q on I with respect to $\mathcal{M}$*, denoted by $certain_{\mathcal{M}}(q, I)$, is the set of all k-tuples t of values from I such that, for every solution J of I under $\mathcal{M}$, we have that $t \in q(J)$, where $q(J)$ is the result of evaluating q on J. If J is a universal solution for I under $\mathcal{M}$, and q is a union of conjunctive queries, then it is shown in [20] that $certain_{\mathcal{M}}(q, I)$ equals $q(J)_{\downarrow}$, which is the result of evaluating q on J and then keeping only those tuples formed entirely of values from I (that is, tuples that do not contain nulls). The equality $certain_{\mathcal{M}}(q, I) = q(J)_{\downarrow}$ holds for arbitrarily specified schema mappings $\mathcal{M}$ (as long as such a universal solution J exists). We thereby obtain the following corollary to Theorem 5.1, which is analogous to the same corollary in [21] for mappings specified by

SO tgds (except that in the case of SO tgds, there is always a solution for every ground instance I).

COROLLARY 5.2. *Let $\mathcal{M}$ be an SO-standard schema mapping. Let q be a union of conjunctive queries over the target schema $\mathbf{T}$. Then for every ground instance I over $\mathbf{S}$ such that there is a solution for I with respect to Φ, the set certain$_{\mathcal{M}}(q, I)$ can be computed in polynomial time (in the size of I).*

6. A POSITIVE RESULT: SO-STANDARD SCHEMA MAPPINGS ARE THE NEEDED CLASS

In this section, we show that SO-standard schema mappings (those specified by an st-SO dependency, along with target constraints consisting of t-egds and a weakly-acyclic set of t-tgds) exactly correspond to the composition of standard schema mappings.

6.1 Using SO-standard schema mappings to define compositions

Before we show that the composition of an arbitrary number of standard schema mappings is equivalent to an SO-standard schema mapping, we first show that target constraints are needed (that is, st-SO dependencies by themselves are not enough). In fact, the next proposition says that st-SO dependencies, without target constraints, are not capable of specifying even schema mappings specified by st-tgds and a set of key dependencies.

PROPOSITION 6.1. *There exists a schema mapping $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12}, \Sigma_2)$, where Σ_{12} is a set of st-tgds and Σ_2 is a set of key dependencies, such that $\mathcal{M}_{12}$ cannot be specified by an st-SO dependency.*

As we shall see, we get an easy proof of Proposition 6.1 by using the following simple proposition, which is analogous to the same result for st-tgds [17].

PROPOSITION 6.2. *Let σ_{12} be an st-SO dependency, let I be a source instance, and let J be a target instance. If $(I, J) \models \sigma_{12}$ and $J \subseteq J'$, then $(I, J') \models \sigma_{12}$.*

PROOF OF PROPOSITION 6.1. Let $\mathbf{S}_1 = \{S(\cdot, \cdot)\}$, $\mathbf{S}_2 = \{T(\cdot, \cdot)\}$ and $\Sigma_{12} = \{S(x, y) \rightarrow T(x, y)\}$, and assume that Σ_2 consists of the single key dependency $T(x, y) \wedge T(x, z) \rightarrow y = z$. By way of contradiction, assume that $\mathcal{M}_{12}$ can be specified by an st-SO dependency σ_{12}. Let $I = \{S(1, 2)\}$, $J = \{T(1, 2)\}$ and $J' = \{T(1, 2), T(1, 3)\}$. Given that $(I, J) \models \Sigma_{12} \cup \Sigma_2$, and σ_{12} specifies $\mathcal{M}_{12}$, we have that $(I, J) \models \sigma_{12}$. Hence, by Proposition 6.2, we have that $(I, J') \models \sigma_{12}$ and, thus, $(I, J') \models \Sigma_{12} \cup \Sigma_2$ since σ_{12} specifies $\mathcal{M}_{12}$. Thus, we obtain a contradiction, since $J' \not\models \Sigma_2$. □

Let $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ be standard schema mappings. The previous negative result implies that st-SO dependencies by themselves cannot necessarily specify the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$. Our next theorem implies that $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is equivalent to an SO-standard schema mapping $\mathcal{M}_{13}$. In fact, we can take the target constraints of $\mathcal{M}_{13}$ to be the target constraints of $\mathcal{M}_{23}$. Thus, intuitively, st-SO dependencies are expressive enough to capture the intermediate target constraints in a composition.

THEOREM 6.3. *Let $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12}, \Sigma_2)$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23}, \Sigma_3)$ be standard schema mappings (so that Σ_{12}, Σ_{23} are sets of st-tgds, and Σ_i $(i = 2, 3)$ is the union of a set of t-egds and a weakly-acyclic set of t-tgds). Then there exists an st-SO dependency σ_{13} such that the mapping $\mathcal{M}_{13} = (\mathbf{S}_1, \mathbf{S}_3, \sigma_{13}, \Sigma_3)$ is equivalent to the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$.*

In Section 6.2, we show that the composition of SO-standard schema mappings is also an SO-standard schema mapping. By combining this result with Theorem 6.3 (and using the simple fact, noted earlier, that every standard schema mapping is an SO-standard schema mapping), we obtain our desired result, namely, that the composition of a finite number of standard schema mappings is equivalent to an SO-standard schema mapping.

It is straightforward to show that Theorem 6.3 is a consequence of the following proposition.

PROPOSITION 6.4. *Let $\mathcal{M}_{12}$ be a standard schema mapping, and let $\mathcal{M}_{23}$ be an st-tgd mapping (no target constraints). Then the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ can be specified by an st-SO dependency.*

As pointed out in Section 4, the class of st-SO dependencies corresponds to the source-to-target restriction of the class of Sk$\forall$CQ$^{=}$ dependencies introduced in [40]. In fact, Theorem 6.3 and Proposition 6.4 were essentially established in [40] (see Theorems 6 and 9 and the paragraph after Theorem 10 in [40]), but they are restated and clarified here for the sake of completeness. We also show here how Proposition 6.4 is proved, which is a straightforward adaptation of the proofs of Theorems 6 and 9 in [40], and the comments in the paragraph after Theorem 10 to handle a weakly-acyclic set of target tgds.

We now demonstrate, by example, how an st-SO dependency σ_{13} is obtained from $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ in Proposition 6.4 (it will be clear how to extend from the example to the general case). Assume that $\mathbf{S}_1 = \{A(\cdot, \cdot), B(\cdot)\}$, $\mathbf{S}_2 = \{C(\cdot, \cdot), D(\cdot, \cdot)\}$, $\mathbf{S}_3 = \{E(\cdot, \cdot)\}$. Furthermore, suppose that Σ_{12} consists of the following st-tgds:

$$\begin{aligned} A(x, y) &\rightarrow C(x, y), \\ B(x) &\rightarrow \exists y\, C(x, y), \end{aligned} \tag{1}$$

Σ_2 consists of the following t-tgds:

$$\begin{aligned} C(x, y) \wedge C(y, z) &\rightarrow C(z, x), \\ C(x, y) &\rightarrow \exists z\, D(x, z), \\ C(x, x) &\rightarrow D(x, x), \\ D(x, y) &\rightarrow D(y, x), \end{aligned} \tag{2}$$

and Σ_{23} consists of the st-tgd:

$$D(x, y) \rightarrow \exists z\, E(x, y, z). \tag{3}$$

To obtain σ_{13}, we first Skolemize each dependency in Σ_{12}, Σ_2 and Σ_{23} to obtain the sets $\mathcal{E}(\Sigma_{12})$, $\mathcal{E}(\Sigma_2)$ and $\mathcal{E}(\Sigma_{23})$ of dependencies,

respectively. In this process, we use distinct Skolem functions for distinct dependencies. In particular, we replace (1), (2) and (3) by:

$$\begin{aligned} B(x) &\rightarrow C(x, f(x)), \\ C(x,y) &\rightarrow D(x, g(x,y)), \\ D(x,y) &\rightarrow E(x,y,h(x,y)), \end{aligned}$$

respectively. Then for predicates C and D, we introduce functions f_C, g_C, f_D and g_D, where f_C, g_C have the same arity as C, and where f_D, g_D have the same arity as D, and we define σ_{13} as:

$$\exists f \exists g \exists h \exists f_C \exists g_C \exists f_D \exists g_D \, \Psi,$$

where f, g and h are the Skolem functions introduced above and Ψ is a conjunction of a set of dependencies defined as follows. As predicate C cannot be mentioned in Ψ, functions f_C and g_C are used to replace it: the equality $f_C(\bar{a}) = g_C(\bar{a})$ is used to indicate that $C(\bar{a})$ holds. Thus, the first two conjuncts of Ψ are generated from $\mathcal{E}(\Sigma_{12})$ by replacing $C(\bar{x})$ by $f_C(\bar{x}) = g_C(\bar{x})$:

$$\begin{aligned} A(x,y) &\rightarrow f_C(x,y) = g_C(x,y), \\ B(x) &\rightarrow f_C(x, f(x)) = g_C(x, f(x)). \end{aligned} \tag{4}$$

Similarly, functions f_D and g_D are used to replace predicate D, and the dependencies in $\mathcal{E}(\Sigma_2)$ are used to generate the following conjuncts of Ψ:

$$\begin{aligned} &\mathrm{dom}(x) \wedge \mathrm{dom}(y) \wedge \mathrm{dom}(z) \wedge f_C(x,y) = g_C(x,y) \wedge \\ &\qquad f_C(y,z) = g_C(y,z) \rightarrow f_C(z,x) = g_C(z,x), \end{aligned} \tag{5}$$

$$\begin{aligned} &\mathrm{dom}(x) \wedge \mathrm{dom}(y) \wedge \\ &\quad f_C(x,y) = g_C(x,y) \rightarrow f_D(x, g(x,y)) = g_D(x, g(x,y)), \end{aligned} \tag{6}$$

$$\mathrm{dom}(x) \wedge f_C(x,x) = g_C(x,x) \rightarrow f_D(x,x) = g_D(x,x), \tag{7}$$

$$\begin{aligned} &\mathrm{dom}(x) \wedge \mathrm{dom}(y) \wedge \\ &\qquad f_D(x,y) = g_D(x,y) \rightarrow f_D(y,x) = g_D(y,x), \end{aligned} \tag{8}$$

where $\mathrm{dom}(\cdot)$ is a formula that defines the domain of the instances of $\mathbf{S}_1$, that is, $\mathrm{dom}(x)$ is $\exists y \, A(x,y) \vee \exists z \, A(z,x) \vee B(x)$. This predicate is included in the previous dependencies to satisfy the safety condition of st-SO dependencies, namely, that every variable mentioned in a term has to be mentioned in a source predicate. We then use the standard approach for eliminating disjunctions in a premise (for example, $\varphi_1 \vee \varphi_2 \rightarrow \psi$ can be replaced by the two formulas $\varphi_1 \rightarrow \psi$ and $\varphi_2 \rightarrow \psi$).

Notice that if an equality $f_C(a, f(a)) = g_C(a, f(a))$ can be inferred by using dependency (4), then we know that $C(a, f(a))$ holds. Thus, since $D(a, g(a, f(a)))$ can be obtained from $C(a, f(a))$ and the dependency $C(x,y) \rightarrow D(x, g(x,y))$, it should be possible to infer that $f_D(a, g(a, f(a))) = g_D(a, g(a, f(a)))$ holds by using the fact that $f_C(a, f(a)) = g_C(a, f(a))$ holds and the dependencies in Ψ. However, if $\mathrm{dom}(f(a))$ does not hold, then $f_C(a, f(a)) = g_C(a, f(a))$ does not satisfy the premise of dependency (6) and, therefore, $f_D(a, g(a, f(a))) = g_D(a, g(a, f(a)))$ cannot be inferred by using this dependency. To overcome this limitation, we also instantiate the above four dependencies with the terms that appear in the tuples that are generated by repeatedly applying the formulas in $\mathcal{E}(\Sigma_2)$. More precisely, it is possible to infer that only terms of the form x and $f(y)$ need to be considered for the case of predicate C and, thus, dependencies (5), (6) and (7) are instantiated with all the possible combinations of this type of terms. For example, the following is one of the conjuncts of Ψ generated from formula (5):

$$\begin{aligned} &\mathrm{dom}(x) \wedge \mathrm{dom}(y) \wedge \mathrm{dom}(z) \wedge \\ &f_C(f(x), y) = g_C(f(x), y) \wedge f_C(y, f(z)) = g_C(y, f(z)) \rightarrow \\ &\qquad f_C(f(z), f(x)) = g_C(f(z), f(x)), \end{aligned}$$

while the following dependency is one of the conjuncts of Ψ generated from formula (7):

$$\begin{aligned} &\mathrm{dom}(x) \wedge \mathrm{dom}(y) \wedge f_C(f(x), f(y)) = g_C(f(x), f(y)) \wedge \\ &\qquad f(x) = f(y) \rightarrow f_D(f(x), f(y)) = g_D(f(x), f(y)). \end{aligned}$$

Notice that in the previous dependency we have included the equality $f(x) = f(y)$, as it can be the case that $f(a) = f(b)$ holds for distinct elements a and b. Similarly, it is possible to infer that only terms of the form x, $f(y)$, $g(x,y)$, $g(x, f(y))$, $g(f(x), y)$ and $g(f(x), f(y))$ need to be considered for the case of predicate D. Thus, dependency (8) is instantiated with all the possible combinations of this type of terms. For example, the following is one of the conjuncts of Ψ generated by this process:

$$\begin{aligned} &\mathrm{dom}(x) \wedge \mathrm{dom}(y) \wedge \mathrm{dom}(z) \wedge \\ &\quad f_D(f(x), g(f(y), z)) = g_D(f(x), g(f(y), z)) \rightarrow \\ &\qquad f_D(g(f(y), z), f(x)) = g_D(g(f(y), z), f(x)). \end{aligned}$$

Finally, the last conjuncts of Ψ are generated from dependency $D(x,y) \rightarrow E(x,y,h(x,y))$ as above. For example, the following are two of these conjuncts:

$$\begin{aligned} &\mathrm{dom}(x) \wedge \mathrm{dom}(y) \wedge f_D(x,y) = g_D(x,y) \rightarrow E(x,y,h(x,y)), \\ &\mathrm{dom}(x) \wedge \mathrm{dom}(y) \wedge \mathrm{dom}(z) \wedge \\ &\quad f_D(f(x), g(f(y), f(z))) = g_D(f(x), g(f(y), f(z))) \rightarrow \\ &\qquad E(f(x), g(f(y), f(z)), h(f(x), g(f(y), f(z)))). \end{aligned}$$

It is important to notice that the weak acyclicity of Σ_2 guarantees that the above process terminates. That is, we need only consider terms up to a certain fixed depth of nesting. In particular, in the above example, we only need to consider terms where the nesting depth of functions is at most 2.

EXAMPLE 6.5. We conclude this section by showing why weak acyclicity is necessary to guarantee the termination of the above process. Assume that $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12}, \Sigma_2)$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$, where $\mathbf{S}_1 = \{A(\cdot,\cdot)\}$, $\mathbf{S}_2 = \{B(\cdot,\cdot)\}$, $\mathbf{S}_3 = \{C(\cdot,\cdot)\}$, Σ_{12} consists of the following st-tgd:

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

Σ_2 consists of the following t-tgd:

$$B(x,y) \rightarrow \exists z \, B(y,z), \tag{9}$$

and Σ_{23} consists of the st-tgd:

$$B(x,y) \rightarrow C(x,y).$$

Notice that $\mathcal{M}_{12}$ is not a standard schema mapping, as Σ_2 is not weakly acyclic.

In order to obtain an st-SO dependency σ_{13} that defines the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, the above process first Skolemizes each dependency in Σ_{12}, Σ_2 and Σ_{23} to obtain the sets $\mathcal{E}(\Sigma_{12})$, $\mathcal{E}(\Sigma_2)$ and $\mathcal{E}(\Sigma_{23})$ of dependencies, respectively. In particular, the t-tgd (9) is replaced by the dependency:

$$B(x,y) \rightarrow B(y, h(x,y)). \tag{10}$$

Then binary functions f_B and g_B are introduced, and σ_{13} is defined as $\exists h \exists f_B \exists g_B\, \Psi$, where Ψ is a conjunction of a set of dependencies defined as follows. The first conjunct of Ψ is generated from $\mathcal{E}(\Sigma_{12})$ by replacing $B(x,y)$ by $f_B(x,y) = g_B(x,y)$:

$$A(x,y) \quad \rightarrow \quad f_B(x,y) = g_B(x,y). \tag{11}$$

Then functions f_B and g_B are used to eliminate predicate B from $\mathcal{E}(\Sigma_2)$. In particular, the following conjunct is included in Ψ:

$$\begin{aligned}\mathrm{dom}(x) \wedge \mathrm{dom}(y) \wedge f_B(x,y) = g_B(x,y) \;\rightarrow \\ f_B(y, h(x,y)) = g_B(y, h(x,y)),\end{aligned} \tag{12}$$

where $\mathrm{dom}(\cdot)$ is a formula that defines the domain of the instances of $\mathbf{S}_1$, that is, $\mathrm{dom}(x)$ is $\exists u\, A(x,u) \vee \exists v\, A(v,x)$. As mentioned above, predicate $\mathrm{dom}(\cdot)$ is included in the previous dependency to satisfy the safety condition of st-SO dependencies.

It should be noticed if (a,b) is a tuple in A, one can infer that $f_B(a,b) = g_B(a,b)$ holds by considering dependency (11), and then one can infer that $f_B(b, h(a,b)) = g_B(b, h(a,b))$ holds by considering dependency (12). By definition of σ_{13}, this implies that $B(b, h(a,b))$ holds, from which one concludes that $B(h(a,b), h(b, h(a,b)))$ also holds (from dependency (10)). Thus, in this case it should be possible to infer that

$$f_B(h(a,b), h(b, h(a,b))) = g_B(h(a,b), h(b, h(a,b))) \tag{13}$$

holds from the dependencies in Ψ. However, if $\mathrm{dom}(h(a,b))$ does not hold, then one cannot infer equality (13) from dependency (12) and the fact that $f_B(b, h(a,b)) = g_B(b, h(a,b))$ holds. This forces one to instantiate dependency (12) with the terms that appear in the tuples that are generated by repeatedly applying (10). In particular, the following dependency is included as a conjunct of Ψ to be able to infer (13) from equality $f_B(b, h(a,b)) = g_B(b, h(a,b))$:

$$\begin{aligned}\mathrm{dom}(x) \wedge \mathrm{dom}(y) \wedge f_B(x, h(x,y)) = g_B(x, h(x,y)) \;\rightarrow \\ f_B(h(x,y), h(x, h(x,y))) = g_B(h(x,y), h(x, h(x,y))).\end{aligned}$$

The previous dependencies are used to deal with the terms where the nesting depth of functions is at most 2. But given that Σ_2 is not weakly acyclic, one also needs to deal with the terms where the nesting depth of functions is 3, which forces one to include the following dependency as a conjunct of Ψ:

$$\begin{aligned}&\mathrm{dom}(x) \wedge \mathrm{dom}(y) \wedge \\ &f_B(h(x,y), h(x, h(x,y))) = g_B(h(x,y), h(x, h(x,y))) \;\rightarrow \\ &\quad f_B(h(x, h(x,y)), h(h(x,y), h(x, h(x,y)))) \;= \\ &\quad\quad g_B(h(x, h(x,y)), h(h(x,y), h(x, h(x,y)))).\end{aligned}$$

It is not difficult to see that the process does not terminate in this case, as from the preceding dependency one needs to generate a formula to deal with the terms where the nesting depth of functions is 4, which in turn has to be used to generate a dependency to deal with nesting depth 5, and so on. □

6.2 Composability of SO-standard schema mappings

The next theorem implies that the composition of SO-standard schema mappings is an SO-standard schema mapping. This is the final step we need to show that the composition of a finite number of standard schema mappings is given by an SO-standard schema mapping.

THEOREM 6.6. *For every pair $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \sigma_{12}, \Sigma_2)$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \sigma_{23}, \Sigma_3)$ of schema mappings, where σ_{12}, σ_{23} are st-SO dependencies and Σ_i $(i = 2, 3)$ is the union of a set of t-egds and a weakly acyclic set of t-tgds, there exists an st-SO dependency σ_{13} such that the schema mapping $\mathcal{M}_{13} = (\mathbf{S}_1, \mathbf{S}_3, \sigma_{13}, \Sigma_3)$ is equivalent to the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$.*

Note that, just as in Theorem 6.3, the set Σ_3 used in $\mathcal{M}_{23}$ is also used in $\mathcal{M}_{13}$. Theorem 6.6 was essentially established in [40] (see Theorems 6 and 9 and the paragraph after Theorem 10 in [40]), since the class of st-SO dependencies corresponds to the source-to-target restriction of the class of Sk$\forall$CQ$^{=}$ dependencies introduced in [40].

As pointed out in Section 6.1, the previous result is fundamental to showing that SO-standard schema mappings can define the composition of standard schema mappings, since from the combination of this result with Theorem 6.3 (and using the simple fact that every standard schema mapping is an SO-standard schema mapping), we obtain the following theorem as a consequence.

THEOREM 6.7. *The composition of a finite number of standard schema mappings is equivalent to an SO-standard schema mapping.*

6.3 SO-standard schema mappings are exactly the needed class

We have introduced st-SO dependencies (and SO-standard schema mappings) because of Theorem 6.7. In this section, we show that SO-standard schema mappings are exactly the needed class, since the converse of Theorem 6.7 also holds. Specifically, we have the following theorem.

THEOREM 6.8. *Every SO-standard schema mapping is equivalent to the composition of a finite number of standard schema mappings.*

This is proven by showing the following:

THEOREM 6.9. *Every schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \sigma_{st})$, where σ_{st} is an st-SO dependency, is equivalent to the composition of a finite number of schema mappings, each specified by st-tgds and t-egds.*

Note that, somewhat surprisingly, we do not need to make use of a weakly acyclic set of t-tgds (or any t-tgds at all) in Theorem 6.9. In particular, let $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ be as in Proposition 6.4 (where the specification of $\mathcal{M}_{12}$ may make use of a weakly-acyclic set of t-tgds). By Proposition 6.4, the composition is given by a schema mapping $\mathcal{M}_{13}$ specified by an st-SO dependency, and by Theorem 6.9, we know that $\mathcal{M}_{13}$ is the composition of a finite number of schema mappings, each specified by st-tgds and t-egds (no t-tgds). So $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ needs no t-tgds to specify it, even though $\mathcal{M}_{12}$ makes use of t-tgds.

We now show how Theorem 6.8 follows from Theorem 6.9. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \sigma_{st}, \Sigma_t)$ be an SO-standard schema mapping (where

σ_{st} is an st-SO dependency, and Σ_t is the union of a set of t-egds and a weakly acyclic set of t-tgds). Let $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \sigma_{st})$, where we discard Σ_t from $\mathcal{M}$. By Theorem 6.9, where the role of $\mathcal{M}$ is played by $\mathcal{M}'$, we know that there are schema mappings $\mathcal{M}_1, \ldots, \mathcal{M}_k$, each specified by st-tgds and t-egds, such that $\mathcal{M}' = \mathcal{M}_1 \circ \cdots \circ \mathcal{M}_k$. Assume that $\mathcal{M}_k = (\mathbf{S}', \mathbf{T}, \sigma_{st}, T_k)$, with T_k consisting only of t-egds. Let $\mathcal{M}'_k = (\mathbf{S}', \mathbf{T}, \sigma_{st}, T_k \cup \Sigma_t)$. Then $\mathcal{M}_1, \ldots, \mathcal{M}_{k-1}, \mathcal{M}'_k$ are standard schema mappings ($\mathcal{M}'_k$ is a standard schema mappings, since its only t-tgds are those in Σ_t). Since $(\mathbf{S}, \mathbf{T}, \sigma_{st}) = \mathcal{M}_1 \circ \cdots \circ \mathcal{M}_k$, it follows easily that $(\mathbf{S}, \mathbf{T}, \sigma_{st}, \Sigma_t) = \mathcal{M}_1 \circ \cdots \circ \mathcal{M}_{k-1} \circ \mathcal{M}'_k$. Thus, $\mathcal{M} = \mathcal{M}_1 \circ \cdots \circ \mathcal{M}_{k-1} \circ \mathcal{M}'_k$.

We now demonstrate, by example, how Theorem 6.9 is proved (again, it will be clear how to extend from the example to the general case). Our proof is an extension of the proof in [21] that every SO tgd specifies the composition of a finite number of st-tgd mappings (see Theorem 8.4 in [21]).

Assume that $\mathbf{S} = \{S(\cdot)\}$, $\mathbf{T} = \{T(\cdot,\cdot)\}$, $\Sigma_t = \emptyset$ and σ_{st} is the following st-SO dependency:

$$\exists f \exists g\, [\forall x\, (S(x) \rightarrow T(f(g(x)), g(f(x)))) \wedge \\ \forall x \forall y\, (S(x) \wedge S(y) \wedge f(x) = f(y) \rightarrow g(x) = g(y))].$$

Next we construct schema mappings $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12}, \Sigma_2)$, $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23}, \Sigma_3)$ and $\mathcal{M}_{34} = (\mathbf{S}_3, \mathbf{S}_4, \Sigma_{34})$ such that (1) $\mathbf{S}_1 = \mathbf{S}$, (2) $\mathbf{S}_4 = \mathbf{T}$, (3) Σ_{12}, Σ_{23} and Σ_{34} are sets of st-tgds, (4) Σ_2 and Σ_3 are set of t-egds, and (5) the mapping specified by σ_{st} is equivalent to $\mathcal{M}_{12} \circ \mathcal{M}_{23} \circ \mathcal{M}_{34}$.

Define $\mathbf{S}_2$ as $\{R_1(\cdot), F_1(\cdot,\cdot), G_1(\cdot,\cdot)\}$ and Σ_{12} to consist of the following st-tgds:

$$\begin{aligned} S(x) &\rightarrow R_1(x), \\ S(x) &\rightarrow \exists y\, F_1(x,y), \\ S(x) &\rightarrow \exists y\, G_1(x,y). \end{aligned}$$

Intuitively, we take R_1 to copy S, we take $F_1(x,y)$ to encode $f(x) = y$, and we take $G_1(x,y)$ to encode $g(x) = y$. In particular, the second and third dependencies have the effect of guaranteeing that $f(x)$ and $g(x)$ are defined for every element x in S, respectively.

Given that Σ_{12} cannot guarantee that F_1 and G_1 each define a single image for every element in S, we let Σ_2 consist of the following t-egds:

$$\begin{aligned} F_1(x,y) \wedge F_1(x,z) &\rightarrow y = z, \\ G_1(x,y) \wedge G_1(x,z) &\rightarrow y = z, \end{aligned}$$

that guarantee that F_1 and G_1 encode functions. In the same way, define $\mathbf{S}_3$ as $\{R_2(\cdot), F_2(\cdot,\cdot), G_2(\cdot,\cdot)\}$ and Σ_{23} to consist of the following st-tgds:

$$\begin{aligned} R_1(x) &\rightarrow R_2(x), \\ F_1(x,y) &\rightarrow F_2(x,y), \\ G_1(x,y) &\rightarrow G_2(x,y), \\ F_1(x,y) &\rightarrow \exists z\, G_2(y,z), \\ G_1(x,y) &\rightarrow \exists z\, F_2(y,z). \end{aligned}$$

Intuitively, we take R_2 to copy R_1, F_2 to copy F_1, and G_2 to copy G_1, and we include the fourth dependency to guarantee that $g(y)$ is defined for all y in the range of f, and we include the fifth dependency to guarantee that $f(y)$ is defined for all y in the range of g. Also as in the previous case, we include in Σ_3 two t-egds that guarantee that F_2 and G_2 are indeed functions:

$$\begin{aligned} F_2(x,y) \wedge F_2(x,z) &\rightarrow y = z, \\ G_2(x,y) \wedge G_2(x,z) &\rightarrow y = z. \end{aligned}$$

Given that at this point, we have predicates that encode the values of all the terms that are used in σ_{st}, we also include in Σ_3 dependencies that encode the conjuncts of σ_{st} of the form $\forall \bar{x}\, (\varphi \rightarrow t_1 = t_2)$. More precisely, in this case we include in Σ_3 the following t-egd that encodes the conjunct $\forall x \forall y\, (S(x) \wedge S(y) \wedge f(x) = f(y) \rightarrow g(x) = g(y))$:

$$R_2(x) \wedge R_2(y) \wedge F_2(x,z) \wedge \\ F_2(y,z) \wedge G_2(x,u) \wedge G_2(y,v) \rightarrow u = v.$$

Finally, we use R_2, F_2 and G_2 to encode the remaining conjuncts of σ_{st}, which indicate how to populate the target relations of σ_{st}. Thus, we define Σ_{34} to consist of the following st-tgd:

$$R_2(x) \wedge G_2(x,y_1) \wedge F_2(y_1,y_2) \wedge \\ F_2(x,z_1) \wedge G_2(z_1,z_2) \rightarrow T(y_2,z_2).$$

This concludes the demonstration by example of how to prove Theorem 6.9. This demonstration gives, as a special case (when the st-SO dependency is unnested) the following lemma (where we note also the number of schema mappings that are composed).

LEMMA 6.10. *Every schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \sigma_{st})$, where σ_{st} is an unnested st-SO dependency, is equivalent to the composition of two schema mappings, each specified by st-tgds and t-egds.*

We note that Theorem 6.9 follows immediately from Lemma 6.10 and the fact, as we show later, that every st-SO dependency is equivalent to an unnested st-SO dependency, Therefore, we really needed to prove only Lemma 6.10 (the unnested case) rather than the general case that we dealt with in proving Theorem 6.9.

7. COLLAPSING RESULTS: NESTING IS NOT NECESSARY

In this section, we present collapsing results about the depth of nesting of function symbols in st-SO dependencies and SO tgds. Specifically, we prove the following two theorems.

THEOREM 7.1. *Every st-SO dependency is equivalent to an unnested st-SO dependency.*

THEOREM 7.2. *Every SO tgd is equivalent to an unnested SO tgd*

These two results, especially the second one, are the most technically difficult results in the paper. Both results are surprising, since the "obvious" way to try to denest, which we now describe, does not work. Consider for example the SO tgd

$$\exists f \exists g \forall x \forall y (P(x,y) \wedge (f(g(x)) = y) \rightarrow Q(x,y)) \qquad (14)$$

The "obvious" way to denest (14) is to introduce a new variable z and rewrite (14) as

$$\exists f \exists g \forall x \forall y \forall z (P(x,y) \wedge (g(x) = z) \wedge (f(z) = y) \rightarrow Q(x,y)) \quad (15)$$

However, the formula (15) is not an SO tgd, since it violates the safety condition (because the variable z does not appear in $P(x,y)$, the only relational atomic formula in the premise of (15)).

It should be mentioned that in [36], Libkin and Sirangelo introduce the second-order language of Skolemized STDs (SkSTDs), and study some of its fundamental properties. In particular, it is shown in [36] that this language is closed under composition if the premises of SkSTDs are restricted to be conjunctive queries. Interestingly, this fragment of SkSTDs is similar to the language of SO tgds but does not allow nesting of functions, which may lead one to think that Theorem 7.2 can be deduced from the results in [36]. However, no safety condition is imposed on the premises of SkSTDs in [36] and, thus, nesting of functions is not needed in this language as it can be eliminated in the "obvious" way shown above. In fact, dependency (15) is a valid constraint according to [36].

We now present and discuss two corollaries of Theorem 7.2.

COROLLARY 7.3. *The composition of a finite number of st-tgd mappings can be specified by an unnested SO tgd.*

This is a strengthening of the result (Theorem 8.1 in [21]) that the composition of a finite number of st-tgd mappings can be specified by an SO tgd (thus, we replace "SO tgd" by "unnested SO tgd"). Corollary 7.3 follows immediately from the result we just cited (Theorem 8.1 in [21]) and our Theorem 7.2. It was not even known before that the composition of two st-tgd mappings can be specified by an unnested SO tgd. Thus, although it was shown in [21] that each unnested SO tgd specifies the composition of some pair of st-tgd mappings, the converse was not shown. In fact, for the composition of two st-tgd mappings, the composition construction in [21] produces an SO tgd whose depth of nesting can be 2, not 1.

We feel that nested dependencies are difficult to understand (just think about an equality like $f(g(x), h(f(x,y))) = g(f(x, h(y)))$), and probably also difficult to use in practice. On the other hand, unnested dependencies seem to be more natural and readable. For example, it is easy to see that the "nested mappings" in [23] can be expressed by unnested SO tgds. Corollary 7.3 tells us that unnested SO tgds are also expressive enough to specify the composition of an arbitrary number of st-tgd mappings.

Theorem 7.2 has as another corollary the following collapsing result about the number of compositions of st-tgd mappings.

COROLLARY 7.4. *The composition of a finite number of st-tgd mappings is equivalent to the composition of two st-tgd mappings.*

This follows from Corollary 7.3 and the fact (which is a special case of Theorem 8.4 of [21]) that a schema mapping specified by an unnested SO tgd is equivalent to the composition of two st-tgd mappings.

The next two corollaries follow from Theorem 7.1 just as Corollaries 7.3 and 7.4 follow from Theorem 7.2.

COROLLARY 7.5. *The composition of a finite number of standard schema mappings can be specified by an unnested st-SO dependency, along with t-egds and a weakly acyclic set of t-tgds.*

COROLLARY 7.6. *The composition of a finite number of standard schema mappings is equivalent to the composition of two standard schema mappings.*

In fact, it follows from Corollary 7.5 and Lemma 6.10 that we can slightly strengthen Corollary 7.6 as follows

COROLLARY 7.7. *The composition of a finite number of standard schema mappings is equivalent to the composition $\mathcal{M}_1 \circ \mathcal{M}_2$ of two standard schema mappings $\mathcal{M}_1$ and $\mathcal{M}_2$, where the target constraints of $\mathcal{M}_1$ are only t-egds (no t-tgds).*

Corollary 7.6 has a direct, almost trivial proof that does not use our heavy machinery, as we now show. Let $\mathcal{M}_{12}$, $\mathcal{M}_{23}$, ..., $\mathcal{M}_{k-1\,k}$ be standard schema mappings. Define $\mathcal{M}'_{12}$ to have source schema the same as $\mathcal{M}_{12}$, target schema equal to the union of the target schemas of $\mathcal{M}_{12}$, ..., $\mathcal{M}_{k-2\,k-1}$, and constraints equal to the union of the constraints of $\mathcal{M}_{12}$, ..., $\mathcal{M}_{k-2\,k-1}$. Because all of the schemas are disjoint, it is easy to see that $\mathcal{M}'_{12}$ is a standard schema mapping (note that the st-tgds of $\mathcal{M}_{23}$, ..., $\mathcal{M}_{k-2\,k-1}$ are now being treated as t-tgds of $\mathcal{M}'_{12}$). Then it is clear that

$$\mathcal{M}_{12} \circ \mathcal{M}_{23} \circ \ldots \circ \mathcal{M}_{k-1\,k} = \mathcal{M}'_{12} \circ \mathcal{M}_{k-1\,k}.$$

In contrast to Corollary 7.6, the reason that Corollary 7.4 is quite unexpected is that there is no obvious way to deal with all of the st-tgds in the intermediate schema mappings.

Corollary 7.5, unlike Corollary 7.6, does not seem to have a simple direct proof that avoids the machinery of Theorem 7.1. This is because our construction of the composition of two standard schema mappings produces an st-SO dependency whose nesting depth can be arbitrarily large.

Based on our collapsing results, there are two alternative ways to deal with the composition of multiple st-tgd mappings. First, by Corollary 7.3, we can replace this composition by a single schema mapping, specified by an unnested SO tgd. Second, by Corollary 7.4, we can replace the composition by the composition of only two st-tgd mappings. Similarly, by using Corollaries 7.5 and 7.6, we have two alternative ways to deal with the composition of a large number of standard schema mappings.

8. CONCLUDING REMARKS

We have investigated the question of what language is needed to specify the composition of schema mappings with target constraints. In particular, we showed that *st-SO dependencies* (along with appropriate target constraints) are exactly the right language for specifying the composition of *standard schema mappings* (those specified by st-tgds, target egds, and a weakly-acyclic set of target tgds). By contrast, we showed that SO tgds, even with arbitrary source and target constraints, are not rich enough to be

able to specify in general the composition of two standard schema mappings. In addition to their expressive power, we also showed that st-SO dependencies enjoy other desirable properties. In particular, they have a polynomial-time chase that generates a universal solution. which can be used to find the certain answers to unions of conjunctive queries in polynomial time.

We proved the surprising results that SO tgds and st-SO dependencies can be denested: that is, each such dependency is equivalent to another dependency of that type with no nested function symbols. These denesting results can be used to "collapse" multiple compositions of schema mappings into the composition of two schema mappings of that type. In particular, we obtain the unexpected result that the composition of an arbitrary number of st-tgd mappings is equivalent to the composition of only two st-tgd mappings.

Our results gave us two ways to "simplify" the composition of an arbitrary number of st-tgd mappings. First, we could replace the composition by a single schema mapping, specified by an unnested SO tgd. Second, we could replace the composition by the composition of only two st-tgd schema mappings. A similar comment applies to the composition of an arbitrary number of standard schema mappings.

Acknowledgments

The authors are grateful to Phokion Kolaitis for helpful discussions. Most of the work on this paper was done while Alan Nash was at IBM Research – Almaden and Marcelo Arenas was a visitor at IBM Research – Almaden. Marcelo Arenas was also supported by FONDECYT grant 1090565.

9. REFERENCES

[1] F. Afrati and N. Kiourtis. Query Answering Using Views in the Presence of Dependencies. In *New Trends in Information Integration (NTII)*, pages 8–11, 2008.

[2] F. Afrati and P. Kolaitis. Repair Checking in Inconsistent Databases: Algorithms and Complexity. In *International Conference on Database Theory (ICDT)*, pages 31–41, 2009.

[3] F. Afrati, C. Li, and V. Pavlaki. Data Exchange in the Presence of Arithmetic Comparisons. In *Extending Data Base Technology (EDBT)*, pages 487–498, 2008.

[4] F. Afrati, C. Li, and V. Pavlaki. Data Exchange: Query Answering for Incomplete Data Sources. In *3rd International Conference on Scalable Information Systems*, 2009.

[5] M. Arenas, P. Barceló, R. Fagin, and L. Libkin. Locally Consistent Transformations and Query Answering in Data Exchange. In *Proceedings of the 23rd ACM Symposium on Principles of Database Systems, PODS'04*, pages 229–240, 2004.

[6] M. Arenas, J. Pérez, and C. Riveros. The Recovery of a Schema Mapping: Bringing Back Exchanged Data. In *Proceedings of the 27th ACM Symposium on Principles of Database Systems, PODS'08*, pages 13–22, 2008.

[7] P. Barceló. Logical Foundations of Relational Data Exchange. *SIGMOD Record*, 38(1):49–58, 2009.

[8] P. A. Bernstein. Applying Model Management to Classical Meta-Data Problems. In *Conference on Innovative Data Systems Research (CIDR)*, pages 209–220, 2003.

[9] M. Casanova, R. Fagin, and C. Papadimitriou. Inclusion Dependencies and their Interaction with Functional Dependencies. *J. Computer and System Sciences*, 20(1):29–59, 1984.

[10] B. Cautis, A. Deutsch, and N. Onose. Querying Data Sources that Export Infinite Sets of Views. In *International Conference on Database Theory (ICDT)*, pages 84–97, 2009.

[11] R. Chirkova and M. Genesereth. Equivalence of SQL Queries in Presence of Embedded Dependencies. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 217–226, 2009.

[12] S. S. Cosmadakis and P. C. Kanellakis. Functional and Inclusion Dependencies: A Graph Theoretic Approach. In *Advances in Computing Research*, volume 3, pages 163–184. 1986.

[13] A. Deutsch, A. Nash, and J. Remmel. The Chase Revisited. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 149–158, 2008.

[14] A. Deutsch, L. Popa, and V. Tannen. Query Reformulation with Constraints. *SIGMOD Record*, 35(1):65–73, 2006.

[15] A. Deutsch and V. Tannen. Reformulation of XML Queries and Constraints. In *International Conference on Database Theory (ICDT)*, pages 225–241, 2003.

[16] H. B. Enderton. *A Mathematical Introduction to Logic: Second Edition*. Academic Press, 2001.

[17] R. Fagin. Inverting Schema Mappings. *ACM Trans. Database Syst.*, 32(4), 2007.

[18] R. Fagin, L. Haas, M. Hernandez, R. Miller, L. Popa, and Y. Velegrakis. Clio: Schema mapping creation and data exchange. In A. Borgida, V. Chaudhri, P. Giorgini, and E. Yu, editors, *Conceptual Modeling: Foundations and Applications, Essays in Honor of John Mylopoulos*, volume 5600 of *Lecture Notes in Computer Science*, pages 198–236. Springer-Verlag, 2009.

[19] R. Fagin, P. Kolaitis, A. Nash, and L. Popa. Towards a Theory of Schema-Mapping Optimization. In *Proceedings of the 27th ACM Symposium on Principles of Database Systems, PODS'08*, pages 33–42, 2008.

[20] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. *Theoretical Computer Science*, 336:89–124, 2005. Preliminary version in *Proc. 2003 International Conference on Database Theory*, pp. 207–224.

[21] R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Composing schema mappings: Second-order dependencies to the rescue. *ACM Trans. Database Syst.*, 30(4):994–1055, 2005.

[22] R. Fagin, L. Stockmeyer, and M. Vardi. On monadic NP vs monadic coNP. *Information and Computation*, 120(1):78–92, 1995.

[23] A. Fuxman, M. A. Hernández, C. T. H. Ho, R. J. Miller, P. Papotti, and L. Popa. Nested Mappings: Schema Mapping Reloaded. In *Very Large Data Bases (VLDB)*, pages 67–78, 2006.

[24] A. Fuxman, P. Kolaitis, R. Miller, and W.-C. Tan. Peer Data Exchange. *ACM Transactions on Database Systems*, 31(4):1454–1498, 2006.

[25] H. Gaifman. On local and non-local properties. In *Proceedings Herbrand Symposium Logic Colloquium, North Holland, 1982*, pages 105–135, 1982.

[26] G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. On Reconciling Data Exchange, Data Integration, and Peer Data Management. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 133–142, 2007.

[27] G. Gottlob and A. Nash. Efficient Core Computation in Data Exchange. *Journal of the ACM*, 55(2), 2008.

[28] G. Gottlob and S. Szeider. Fixed-Parameter Algorithms For Artificial Intelligence, Constraint Satisfaction and Database Problems. *Computer Journal*, 51(3):303–325, 2008.

[29] T. Green, G. Karvounarakis, Z. Ives, and V. Tannen. Update Exchange with Mappings and Provenance. In *International Conference on Very Large Data Bases (VLDB)*, pages 675–686, 2007.

[30] W. P. Hanf. Model-theoretic methods in the study of elementary logic. In *The Theory of Models; Addison, Henkin, and Tarski, eds., North Holland 1965*, pages 132–145, 1965.

[31] A. Hernich and N. Schweikardt. CWA-solutions for Data Exchange Settings with Target Dependencies. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 113–122, 2007.

[32] G. Karvounaraki and V. Tannen. Conjunctive Queries and Mappings with Unequalities. Technical Report MS-CIS-08-37, University of Pennsylvania, 2008.

[33] P. G. Kolaitis. Schema Mappings, Data Exchange, and Metadata Management. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 61–75, 2005.

[34] M. Lenzerini. Data integration: a theoretical perspective. In *Proceedings of the 21st ACM Symposium on Principles of Database Systems, PODS'02*, pages 233–246, 2002.

[35] L. Libkin. *Elements of Finite Model Theory*. Springer-Verlag, 1st edition, 2004.

[36] L. Libkin and C. Sirangelo. Data exchange and schema mappings in open and closed worlds. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 139–148, 2008.

[37] J. Madhavan and A. Y. Halevy. Composing Mappings Among Data Sources. In *International Conference on Very Large Data Bases (VLDB)*, pages 572–583, 2003.

[38] M. Meier. Towards Rule-Based Minimization of RDF Graphs under Constraints. In *Web Reasoning and Rule Systems*, volume 5341 of *Lecture Notes in Computer Science*, pages 89–103. Springer-Verlag, 2008.

[39] S. Melnik. *Generic Model Management: Concepts and Algorithms*, volume 2967 of *Lecture Notes in Computer Science*. Springer, 2004.

[40] A. Nash, P. A. Bernstein, and S. Melnik. Composition of Mappings Given by Embedded Dependencies. In *Proceedings of the 24th ACM Symposium on Principles of Database Systems, PODS'05*, pages 172–183, 2005.

[41] P. Papotti and R. Tortone. Schema Exchange: Generic Mappings for Transformaing Data and Metadata. *Data and Knowledge Engineering*, 68(7):665–682, 2009.

[42] B. ten Cate and P. Kolaitis. Structural Characterizations of Schema-Mapping Languages. In *International Conference on Database Theory (ICDT)*, pages 63–72, 2009.

[43] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos. On the Logical Modeling of ETL Processes. In *International Conference on Advanced Information Systems Engineering (CAiSE)*, pages 782–786, 2002.