Logical Methods in Computer Science
Vol. 7 (3:13) 2011, pp. 1–38
www.lmcs-online.org

Submitted Nov. 8, 2010
Published Sep. 8, 2011

COMPOSITION WITH TARGET CONSTRAINTS *

MARCELO ARENAS [a], RONALD FAGIN [b], AND ALAN NASH [c]

[a] Pontificia Universidad Católica de Chile
e-mail address: marenas@ing.puc.cl

[b] IBM Research–Almaden
e-mail address: fagin@almaden.ibm.com

[c] 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.

1998 ACM Subject Classification: H.2.5.

Key words and phrases: Metadata management, schema mapping, data exchange, composition, target constraint.

* This is an extended version of [6]. The first two authors dedicate this paper to the memory of our co-author Alan Nash, who died tragically, before this final version of the paper was prepared.

[a,c] 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.

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.2168/LMCS-7 (3:13) 2011

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 [35, 36]), several different operators on schema mappings have been singled out as important objects of study [9]. 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 [45]. The composition operator has been studied in depth [23, 39, 41, 42].

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. [23] 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 [10]). This paper is motivated by the question of how to express the compositions of schema mappings that have target constraints. This question was first explored by Nash et al. [42], 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 [23] and that studied by Nash et al. in [42]. 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 [22] 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 [22] 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 [16], 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. [22] (with only source-to-target constraints) and the schema mappings studied by Nash et al. [42] (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 [20]. 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, 7, 8, 11, 12, 14, 15, 21, 22, 26, 28, 29, 30, 31, 33, 34, 35, 40, 43, 44]. 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 [13], 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 [23] and [42] that show the inability of first-order logic to express compositions. Thus, each of these results says 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 [42]. 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 [42] by using their class Sk$\forall$CQ$^{=}$ of dependencies. In fact, our proof of (1) is simply a variation of the proof in [42].

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* [22]. (Recall that a *solution* for a source instance I with respect to a schema mapping $\mathcal{M}$ is a target instance J 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 [22] 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 [25] 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 known even for the composition of two st-tgd mappings. Thus, although it was shown in [23] that each unnested SO tgd specifies the composition of a pair of st-tgd mappings, the converse was not shown. In fact, for the composition of two st-tgd mappings, the composition construction in [23] 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 approaches towards "simplifying" this composition. One approach is to replace the composition of many st-tgd mappings by a single schema mapping, specified by an unnested SO tgd. For another approach, 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 $R_i^I \subseteq \mathbf{D}^{n_i}$. We let $\mathrm{Inst}(\mathbf{R})$ be the set of instances of $\mathbf{R}$. 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 [22]. More precisely, let $\mathbf{C}$ and $\mathbf{N}$ be infinite and disjoint sets of constants and nulls, respectively, and take the domain $\mathbf{D}$ to be $\mathbf{C} \cup \mathbf{N}$. 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 same 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 same 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 K of $\mathbf{S} \cup \mathbf{T}$ satisfies α in the standard first-order logic sense. For every relation symbol $S \in \mathbf{S}$, relation S^K is defined as S^I, and for every relation symbol $T \in \mathbf{T}$, 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 $\mathcal{M}$ from $\mathbf{S}$ to $\mathbf{T}$ is said to be an *st-tgd mapping* if there exists a finite 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 finite set Σ_{st} of st-tgds and a finite 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 finite 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 $(T_1, i) \to (T_2, j)$ in G_Γ if there exist a t-tgd $\forall \bar{x}(\varphi(\bar{x}) \to \exists \bar{y}\, \psi(\bar{x}, \bar{y}))$ 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 $(T_1, i) \to^* (T_2, j)$ in G_Γ if there exist a t-tgd $\forall \bar{x}(\varphi(\bar{x}) \to \exists \bar{y}\, \psi(\bar{x}, \bar{y}))$ 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 [22].

Given a schema mapping $\mathcal{M}$, if a pair (I, J) belongs to it, then J is said to be a *solution* for I with respect to $\mathcal{M}$. A *universal solution* [22] 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.

Definition 2.1. **(**[23]**)** Given a source schema $\mathbf{S}$ and a target schema $\mathbf{T}$, a *second-order source-to-target tuple-generating dependency or SO tgd (from $\mathbf{S}$ to $\mathbf{T}$)* is a second-order formula of the form

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

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, …, 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, …, t_ℓ 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 [23, 42], 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 [23, 42] that this does not work properly. Instead, the solution in [23, 42] 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, [18]). It should be noticed that it is proven in [23] 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 [42] this is shown to be insufficient in the presence of unrestricted target constraints.

The class of SO tgds was introduced in [23] 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 [23] that the composition of an arbitrary number of st-tgd mappings is specified by an SO tgd, that SO tgds are closed under composition, and that every SO tgd specifies the composition of a finite number of st-tgd mappings.

3. A Negative Result: SO tgds are not Enough

As pointed out in the previous section, SO tgds were introduced in [23] to deal with the problem of composing schema mappings. 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 $\mathbf{S}_2$.

Let $\mathcal{M}_{13} = \mathcal{M}_{12} \circ \mathcal{M}_{23}$. We now show that $\mathcal{M}_{13}$ cannot be specified by an SO tgd σ. Assume that it were; we shall derive a contradiction. If I_1 is an arbitrary instance of $\mathbf{S}_1$, then there is I_3 such that $(I_1, I_3) \in \mathcal{M}_{13}$ (for example, we could take I_3 to be the result of chasing I_1 with σ as in [23]). However, let I_1 be the instance of $\mathbf{S}_1$ such that $P^{I_1} = \{(1,2),(1,3)\}$. Then 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 with respect to $\mathcal{M}_{12}$. □

From the previous example, we obtain the following proposition.

Proposition 3.1. *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 3.1 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, if $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ are as in Example 3.1, then the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ can be specified by a set of st-tgds together with some source constraints: specifically, $\mathcal{M}_{12} \circ \mathcal{M}_{23} = \mathcal{M}_{13}$, 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 3.2. *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 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 3.2, 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 [24, 27, 32, 37] 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 a notion of locality for data transformations. Before we give this definition, we give the standard recursive definition of the *quantifier rank* $qr(\phi)$ of an FO-formula ϕ.

- If φ is quantifier-free, then $qr(\varphi) = 0$.
- $qr(\neg\varphi) = qr(\varphi)$
- $qr(\varphi_1 \wedge \varphi_2) = \max\{qr(\varphi_1), qr(\varphi_2)\}$
- $qr(\forall x\varphi) = 1 + qr(\varphi)$

Following [5], we write $N_d^I(a) \equiv_k N_d^I(b)$ to mean 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 $N_d^I(b) \models \varphi$.

Definition 3.3. ([5]) Given a source schema $\mathbf{S}$ and a target schema $\mathbf{T}$, a mapping $\mathfrak{F} : \mathbf{S} \to \mathbf{T}$ 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 $\mathfrak{F}_{\mathrm{univ}}$ the transformation from $\mathbf{S}$ to $\mathbf{T}$, such that $\mathfrak{F}_{\mathrm{univ}}(I)$ is the *canonical universal solution* for I, which is obtained by doing a naive chase of I with Σ_{st}.

Proposition 3.4 ([5])**.** *For every st-tgd mapping, the transformation* $\mathfrak{F}_{\mathrm{univ}}$ *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 3.5. *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* $\mathfrak{F}^i_{\mathrm{univ}}$ *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:*

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

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

Lemma 3.5 is one of the key components in the proof of Theorem 3.2. We shall also utilize the next proposition (Proposition 3.6) in the proof of Theorem 3.2. Proposition 3.6 in a generalization of Proposition 7.2 of [19], which says that for the composition of two st-tgd mappings, the "chase of the chase" is a universal solution.

Proposition 3.6. *Let* $\mathcal{M}_1, \ldots, \mathcal{M}_k$ *be schema mappings, each specified by st-tgds, target egds, and target tgds. Let* $\mathcal{M} = \mathcal{M}_1 \circ \cdots \circ \mathcal{M}_k$*, and let* I *be a source instance for* $\mathcal{M}_1$*. Let* U *be a result (if it exists) of chasing* I *with* $\mathcal{M}_1$*, then chasing the result with* $\mathcal{M}_2$*, ..., and then chasing the result with* $\mathcal{M}_k$*. Then* U *is a universal solution for* I *with respect to* $\mathcal{M}$*.*

Proof. We use a simple trick from the proof of Proposition 7.2 in [19]. Define $\mathcal{M}'$ to be a schema mapping whose st-tgds consist of the st-tgds of $\mathcal{M}_1$, whose target egds consist of the union of the target egds of $\mathcal{M}_1, \ldots, \mathcal{M}_k$, and whose target tgds consist of all of the st-tgds of $\mathcal{M}_2, \ldots, \mathcal{M}_k$, along with all of the target tgds of $\mathcal{M}_1, \ldots, \mathcal{M}_k$. If I be a source instance for $\mathcal{M}_1$, and J is a target instance for $\mathcal{M}_k$, then it is easy to see that J is a solution for I with respect to $\mathcal{M}$ if and only if J is a solution for I with respect to $\mathcal{M}'$. Hence, $\mathcal{M}$

and $\mathcal{M}'$ are the same schema mapping semantically, in that they consist of the same set of (I, J) pairs. If I and U are as in the statement of the proposition, then it is easy to see that U is a result of a chase of I with $\mathcal{M}'$. So from Theorem 3.3 of [22], we have that U is a universal solution of I with respect to $\mathcal{M}'$. Since $\mathcal{M}$ and $\mathcal{M}'$ are the same schema mapping semantically, it follows that U is a universal solution of I with respect to $\mathcal{M}$. □

We need to say "if it exists" in the statement of Proposition 3.6, since there are two reasons that a result of the chase may not exist. First, a target egd may try to equate two constants during a chase. Second, target tgds might force an "infinite chase". These problems do not arise for the composition of two st-tgd mappings, the case considered in Proposition 7.2 of [19].

Proof of Theorem 3.2. 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

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

First, we show that $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ cannot be specified by an SO tgd. For the sake of contradiction, assume that σ_{13} is an SO tgd from $\mathbf{S}_1$ to $\mathbf{S}_3$ and that schema mapping $\mathcal{M}_{13} = (\mathbf{S}_1, \mathbf{S}_3, \sigma_{13})$ is the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, that is, $(I_1, I_3) \in \mathcal{M}_{12} \circ \mathcal{M}_{23}$ if and only if $(I_1, I_3) \models \sigma_{13}$.

From Theorem 8.2 in [23], we know that every SO tgd specifies the composition of a finite number of st-tgd mappings. Thus, given that $\mathcal{M}_{13}$ is 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 $\mathcal{M}'_1 \circ \ldots \circ \mathcal{M}'_{n-1}$ equals the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, that is, for every pair $(I_1, I_3) \in \mathrm{Inst}(\mathbf{S}_1) \times \mathrm{Inst}(\mathbf{S}_3)$:

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

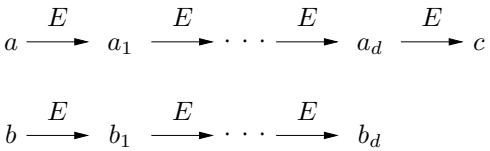
For every $i \in \{1, \ldots, n-1\}$, let $\mathfrak{F}^i_{\mathrm{univ}}$ be the canonical solution transformation for $\mathcal{M}'_i$, and assume that $\mathfrak{F} : \mathrm{Inst}(\mathbf{S}_1) \rightarrow \mathrm{Inst}(\mathbf{S}_3)$ is the transformation defined as

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

for every instance I_1 of $\mathbf{S}_1$. From Lemma 3.5, we have that $\mathfrak{F}$ is locally consistent under FO-equivalence. Thus, for $r = 1$ and $\ell = 1$, there exist $d, k \geq 0$ such that for every instance

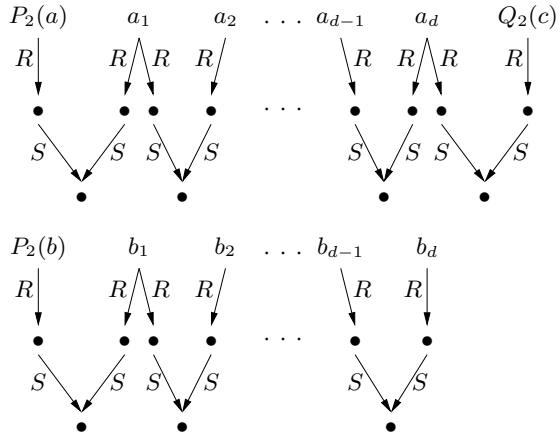
I_1 of $\mathbf{S}_1$ and for every $a, b \in \mathrm{dom}(I_1)$, if $N_d^{I_1}(a) \equiv_k N_d^{I_1}(b)$ then (1) $a \in \mathrm{dom}(\mathfrak{F}(I_1))$ if and only if $b \in \mathrm{dom}(\mathfrak{F}(I_1))$, and (2) $N_r^{\mathfrak{F}(I_1)}(a) \equiv_\ell N_r^{\mathfrak{F}(I_1)}(b)$.

Define an instance I_1 of $\mathbf{S}_1$ with domain $\{a, a_1, \ldots, a_d, b, b_1, \ldots, b_d, c\}$ as follows: $P_1^{I_1} = \{a, b\}$, $Q_1^{I_1} = \{c\}$, and E^{I_1} contains the tuples represented by the following figure:

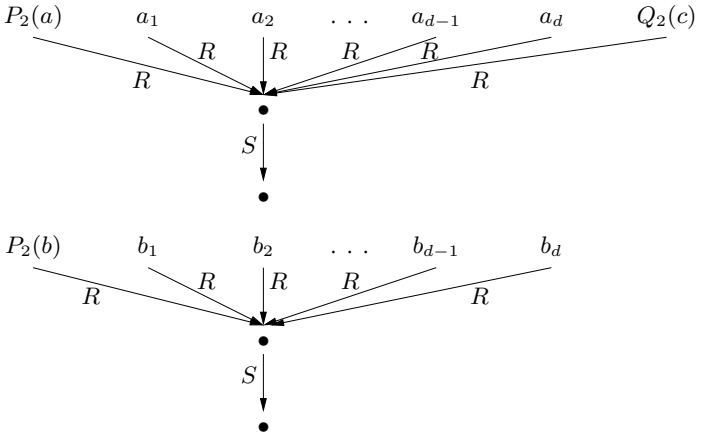


Thus, $E^{I_1} = \{(a, a_1), (a_1, a_2), \ldots, (a_{d-1}, a_d), (b, b_1), (b_1, b_2), \ldots, b_{d-1}, b_d)\}$. As shown in the figure, E^{I_1} is a union of two paths, one containing $d + 2$ elements with first element a and last element c, and another one containing $d+1$ elements with first element b. Observe that $N_d^{I_1}(a) \equiv_k N_d^{I_1}(b)$ since $N_d^{I_1}(a)$ is isomorphic to $N_d^{I_1}(b)$, with a and b treated as distinguished elements.

Let I_3 be the instance of $\mathbf{S}_3$ that contains only the tuple a in V (that is, $V^{I_3} = \{a\}$). Next we show that I_3 is a universal solution for I_1 with respect to $\mathcal{M}_{12} \circ \mathcal{M}_{23}$. A universal solution for I_1 in $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ can be constructed by first chasing with the set Σ_{12} of st-tgds:



(where each element in the figure that is not in $\mathrm{dom}(I_1)$ is a fresh null value, which is represented by a symbol $\bullet$ in the figure), then chasing with the set Σ_2 of t-egds:



and finally chasing with the set Σ_{23} of st-tgds. The result of this last step is I_3, since after chasing with Σ_{12} and Σ_2, we have that P_2 contains elements a and b, Q_2 contains element c, and a, c is the only pair of elements for which there exists an element z such that $P_2(a) \wedge R(a,z) \wedge R(c,z) \wedge Q_2(c)$ holds. We conclude that I_3 is a universal solution for I_1.

By Proposition 3.6, we know that $\mathfrak{F}(I_1)$ is a universal solution for I_1 with respect to $\mathcal{M}'_1 \circ \cdots \circ \mathcal{M}'_{n-1}$. Hence, $\mathfrak{F}(I_1)$ is a universal solution for I_1 with respect to $\mathcal{M}_{12} \circ \mathcal{M}_{23}$. Again by Proposition 3.6, we know that I_3 is also a universal solution for I_1 with respect to $\mathcal{M}_{12} \circ \mathcal{M}_{23}$. Therefore, since $V^{I_3} = \{a\}$, we conclude that $a \in V^{\mathfrak{F}(I_1)}$ and $b \notin V^{\mathfrak{F}(I_1)}$. Hence, we have that $a \in \mathrm{dom}(\mathfrak{F}(I_1))$ and $b \notin \mathrm{dom}(\mathfrak{F}(I_1))$, which contradicts the fact that $\mathfrak{F}$ is locally consistent under FO-equivalence and $N_d^{I_1}(a) \equiv_k N_d^{I_1}(b)$. This concludes the proof that $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ cannot be specified by an SO tgd.

To conclude the proof of the theorem, we need to show that $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ cannot be specified by an SO tgd together with some arbitrary source and target constraints. For the sake of contradiction, assume that schema mapping $\mathcal{M}_{13} = (\mathbf{S}_1, \mathbf{S}_3, \gamma_1, \gamma_{13}, \gamma_3)$ equals the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, where γ_1 is an arbitrary source constraint, γ_{13} is an SO tgd and γ_3 is an arbitrary target constraint. Given that $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ cannot be specified by an SO tgd, we have that either γ_1 is not trivial or γ_3 is not trivial, where an arbitrary constraint is said to be trivial if it allows all the possible instances. First assume that γ_1 is not trivial, and let I_1 be an instance of $\mathbf{S}_1$ such that I_1 does not satisfy γ_1 (I_1 is not allowed by γ_1). Let $\perp$ be a fresh null value and I_2 an instance of $\mathbf{S}_2$ defined as:

$$\begin{aligned} P_2^{I_2} &= P_1^{I_1}, \\ Q_2^{I_2} &= Q_1^{I_1}, \\ R^{I_2} &= \{(a,\perp) \mid \text{there exists } b \in \mathrm{dom}(I_1) \text{ such that } (a,b) \in E^{I_1} \text{ or } (b,a) \in E^{I_1}\}, \\ S^{I_2} &= \{(\perp,\perp)\}. \end{aligned}$$

Furthermore, let I_3 be an instance of $\mathbf{S}_3$ defined as $V^{I_3} = P_2^{I_2}$. It is easy to see that $(I_1, I_2) \models \Sigma_{12}$, $I_2 \models \Sigma_2$ and $(I_2, I_3) \models \Sigma_{23}$, which implies that $(I_1, I_3) \in \mathcal{M}_{12} \circ \mathcal{M}_{23}$. Thus, given that $\mathcal{M}_{13}$ is the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, we have that $(I_1, I_3) \in \mathcal{M}_{13}$. We conclude that I_1 satisfies γ_1, which contradicts our initial assumption.

Now suppose that γ_3 is not trivial, and let I_3 be an instance of $\mathbf{S}_3$ such that I_3 does not satisfy γ_3. Assume that I_1, I_2 are the empty instances of $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively. It is easy to see that $(I_1, I_2) \models \Sigma_{12}$, $I_2 \models \Sigma_2$ and $(I_2, I_3) \models \Sigma_{23}$, which implies that $(I_1, I_3) \in \mathcal{M}_{12} \circ \mathcal{M}_{23}$. Thus, given that $\mathcal{M}_{13}$ is the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, we have that $(I_1, I_3) \in \mathcal{M}_{13}$, and hence I_3 satisfies γ_3, which contradicts our initial assumption. This concludes the proof of the theorem. □

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 of the mappings being composed. 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 [42]. 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 Definition 2.1, 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 $\bar{f}$.

It is sometimes convenient to rewrite an st-SO dependency $\exists \bar{f}\,(\forall \bar{x}_1(\varphi_1 \rightarrow \psi_1) \wedge \cdots \wedge \forall \bar{x}_n(\varphi_n \rightarrow \psi_n))$ so that each conclusion ψ_i is either a conjunction of relational atomic formulas or a single equality of terms (this is possible because we can recursively replace $\forall \bar{x}_i(\varphi_i \rightarrow (\psi_i^1 \wedge \psi_i^2))$ by $\forall \bar{x}_i(\varphi_i \rightarrow \psi_i^1) \wedge \forall \bar{x}_i(\varphi_i \rightarrow \psi_i^2)$ without changing the meaning). Let Φ be the result of such a rewriting. If ψ_i is a conjunction of relational atomic formulas, then we refer to $\forall \bar{x}_i(\varphi_i \rightarrow \psi_i)$ as an *SO tgd part* of Φ, and if ψ_i is an equality $t = t'$, then we refer to $\forall \bar{x}_i(\varphi_i \rightarrow \psi_i)$ as an *SO egd part* of Φ.

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. As with SO tgds, it can be shown that the universe can be taken to be finite but "sufficiently large".

We shall show that the composition of a finite number of standard schema mappings can be specified by an st-SO dependency, together with t-egds and a weakly acyclic set of t-tgds. It is convenient to give these second-order schema mappings a name. To emphasize the similarity of these second-order schema mappings with the first-order case, we shall refer to these second-order 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 [23], the well-known chase process is extended so that it applies to an SO tgd Φ. If we define an *SO tgd part* of an SO tgd as we did for st-SO dependencies, then the idea of the chase with SO tgds 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 [42], 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 clarity, we keep the discussion here informal; it is not hard to convert this into a formal version. 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 $F(t_2(\bar{b}))$ 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 (and we say that the chase *succeeds*).

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.*

Proof. We first show that the chase of a ground instance I with Φ runs in time polynomial in the size of I (when Φ is held fixed). It is straightforward to show by induction on depth that the size of T is polynomial in the size of D. As we noted, during the chase with the SO egd parts, there are at most as many changes in the current value of F as the size of T. So only polynomially many changes occur in the values of F. For each such change, there is only a polynomial amount of work: the time needed to chase each SO egd part and update F if needed along with the time to check the invariant and update F if needed. Finally, since the SO tgd parts are source-to-target, it follows easily that the final portion of the chase, that is, chasing with the SO tgd parts, can also be done in polynomial time. Therefore, the entire chase can be carried out in polynomial time.

Assume for now that there is J such that $(I, J) \models \Phi$. If the existentially quantified function symbols of Φ are given by $\bar{f}$, then let $\bar{f}^0$ denote the instantiation of $\bar{f}$ that shows that $(I, J) \models \Phi$. For each member t of T, let t^0 be the value obtained by replacing the function symbols $\bar{f}$ by $\bar{f}^0$. By induction on the steps in the chase process, we can show that at all points during the chase process, if the current value of $F(t_1)$ equals the current value of $F(t_2)$, then necessarily $t_1^0 = t_2^0$. Thus, whenever we set two values $F(t_1)$ and $F(t_2)$ equal during the process of chasing with the SO egd parts, we are forced to do so. This is clear when two values are made equal because of the conclusion of an SO egd part. It is also true when two values are made equal because of maintaining the invariant, because the invariant is needed for the functions in $\bar{f}^0$ to be well-defined. For example, assume that f^0 and g^0 are unary functions in $\bar{f}^0$. If $f^0(a) = b$, then necessarily $g^0(f^0(a)) = g^0(b)$, and this is reflected by the requirement of the invariant that $F(g(f(a)) = F(g(b))$. Since the only time we make two values equal in the table for F is when we are forced to, it follows that if the chase process fails for I (because we try to set two constants to be equal), then there is no solution for I with respect to Φ. We now consider the other case, where the chase succeeds for I.

Assume that the chase succeeds for I. Let n be the maximal depth of nesting of function symbols in Φ. Use (the final values of) the table for F to define our functions $\bar{f}^0$ on the Herbrand universe of depth $n-1$. For example, if a is in the active domain D, and $F(f(a)) = c$, then let $f^0(a) = c$. Similarly, if a and b are in D, and $F(h(g(a), b)) = d$, then let $h^0(g^0(a), b) = d$. The invariant insures that the functions in $\bar{f}^0$ are well-defined on the Herbrand universe of depth $n-1$ (and the table then gives us the values of all members of the Herbrand universe of depth n). Our semantics requires the functions in $\bar{f}^0$ to be defined not just on the Herbrand universe of depth $n-1$, but on the entire universe. If f is a k-ary function symbol in $\bar{f}$, and if $c_1, \ldots, c_k$ are values such that $f^0(c_1, \ldots, c_k)$ is not already determined by the rules we have given, then let $f^0(c_1, \ldots, c_k)$ be arbitrary. The key point is that Φ refers only to terms in the Herbrand universe of depth n, so what happens outside of the Herbrand universe of depth n is irrelevant, as far as satisfaction of Φ is concerned. The chase with the SO egd parts force equalities among the values of the functions so that I (together with the choice of the functions) satisfies the SO egd parts of Φ. If J is the result of the chase, then the chase with the SO tgd parts force (I, J) to satisfy the SO tgd parts of Φ. Hence, J is a solution for I with respect to Φ, as desired.

It is also clear that if J' is an arbitrary solution for I with respect to Φ, then up to a replacement (not necessarily one-to-one) of nulls in J by other values (nulls or constants), every tuple of every relation that appears in J must appear in the corresponding relation of J' (since tuples are produced in the chase only if needed, and equalities are forced in the chase only if needed). But this means that there is a homomorphism from J into J', so J is a universal solution, as desired. □

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. The reason that chasing with the target dependencies requires only polynomial time is that the number of steps in this chase is polynomial, because of the weak acyclicity assumption (Theorem 3.9 of [22]). We therefore can extend Theorem 5.1 to apply to SO-standard schema mappings. We state this in the following corollary.

Corollary 5.1. Let $\mathcal{M}$ be an SO-standard schema mapping. The chase of a ground instance I with $\mathcal{M}$ runs in time polynomial in the size of I. The chase fails precisely if there is no solution for I with respect to $\mathcal{M}$. If the chase succeeds, then it produces a universal solution for I with respect to $\mathcal{M}$.

Note that in particular, Corollary 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.

As shown in [22], 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. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping, and let q be a k-ary query posed against the target schema $\mathbf{T}$. Denote by $q(J)$ the result of evaluating q on a target instance J. If I is a source instance, then the *certain answers of q on I with respect to $\mathcal{M}$*, denoted by $certain_{\mathcal{M}}(q, I)$, are the k-tuples t such that, for every solution J of I with respect to $\mathcal{M}$, we have that $t \in q(J)$. It should be noticed that if a source instance I does not have any solution with respect to the mapping $\mathcal{M}$, then $certain_{\mathcal{M}}(q, I) = \mathbf{D}^k$ (recall that $\mathbf{D}$ is the countably infinite domain from which the entries of tuples are taken),

as every k-tuple trivially satisfies the previous condition. In this case, we use the special symbol $\top$ to indicate that every k-tuple belongs to $certain_{\mathcal{M}}(q, I)$, that is, we say that $certain_{\mathcal{M}}(q, I) = \top$. If U is a universal solution for I with respect to $\mathcal{M}$, and q is a union of conjunctive queries, then it is shown in [22] that $certain_{\mathcal{M}}(q, I)$ equals $q(U)_{\downarrow}$, which is the result of evaluating q on U and then keeping only those tuples formed entirely of values from I (that is, tuples that do not contain nulls). The equality $certain_{\mathcal{M}}(q, I) = q(U)_{\downarrow}$ holds for arbitrarily specified schema mappings $\mathcal{M}$ (as long as such a universal solution U exists). Corollary 5.1 therefore has the following corollary, which is analogous to the same corollary in [23] for mappings specified by SO tgds.

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 $\mathbf{S}$, the set $certain_{\mathcal{M}}(q, I)$ can be computed in polynomial time (in the size of I).

Proof. Assume that the arity of query q is k, where $k \geq 0$. Then the polynomial-time algorithm to compute $certain_{\mathcal{M}}(q, I)$ works as follows. It first checks (using the polynomial-time algorithm of Corollary 5.1) whether I has a solution with respect to $\mathcal{M}$. If not, then $certain_{\mathcal{M}}(q, I) = \mathbf{D}^k$, and the algorithm returns symbol $\top$ to indicate that every tuple with k elements belongs to $certain_{\mathcal{M}}(q, I)$. Otherwise I has at least one solution with respect to $\mathcal{M}$, and the algorithm computes a universal solution U for I as in Corollary 5.1, and then it returns $q(U)_{\downarrow}$ (recall that, as discussed above, $certain_{\mathcal{M}}(q, I) = q(U)_{\downarrow}$). □

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 [19].

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 $(I, J) \models \sigma_{12}$. So by Proposition 6.2, we have that

$(I, J') \models \sigma_{12}$. Since σ_{12} specifies $\mathcal{M}_{12}$, we therefore have that $(I, J') \models \Sigma_{12} \cup \Sigma_2$. But this is 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, which we shall prove shortly, implies that $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is equivalent to an SO-standard schema mapping $\mathcal{M}_{13}$. In fact, it says that we can take the target constraints of $\mathcal{M}_{13}$ to be the set Σ_3 of target constraints of $\mathcal{M}_{23}$. Intuitively, this theorem tells us that 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 [42]. In fact, Theorem 6.3 and Proposition 6.4 were essentially established in [42] (see Theorems 6 and 9 and the paragraph after Theorem 10 in [42]), 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 [42], 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:

$$\begin{aligned} A(x, y) &\rightarrow C(x, y), \\ B(x) &\rightarrow \exists y\, C(x, y), \end{aligned} \tag{6.1}$$

Σ_2 consists of the following t-tgds:

$$\begin{aligned} C(x, y) \wedge C(y, z) &\rightarrow C(z, x), \\ C(x, y) &\rightarrow \exists z\, D(x, z), \\ C(x, x) &\rightarrow D(x, x), \\ D(x, y) &\rightarrow D(y, x), \end{aligned} \tag{6.2}$$

and Σ_{23} consists of the st-tgd:

$$D(x,y) \quad \rightarrow \quad \exists z\, E(x,y,z). \tag{6.3}$$

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. So we replace (6.1), (6.2) and (6.3) 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 $f_C(\bar{x}) = g_C(\bar{x})$:

$$\begin{aligned} A(x,y) &\rightarrow f_C(x,y) = g_C(x,y),\\ B(x) &\rightarrow f_C(x, f(x)) = g_C(x, f(x)). \end{aligned} \tag{6.4}$$

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:

$$\mathrm{dom}(x) \wedge \mathrm{dom}(y) \wedge \mathrm{dom}(z) \wedge f_C(x,y) = g_C(x,y) \wedge f_C(y,z) = g_C(y,z) \rightarrow f_C(z,x) = g_C(z,x), \tag{6.5}$$

$$\mathrm{dom}(x) \wedge \mathrm{dom}(y) \wedge f_C(x,y) = g_C(x,y) \rightarrow f_D(x, g(x,y)) = g_D(x, g(x,y)), \tag{6.6}$$

$$\mathrm{dom}(x) \wedge f_C(x,x) = g_C(x,x) \rightarrow f_D(x,x) = g_D(x,x), \tag{6.7}$$

$$\mathrm{dom}(x) \wedge \mathrm{dom}(y) \wedge f_D(x,y) = g_D(x,y) \rightarrow f_D(y,x) = g_D(y,x), \tag{6.8}$$

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 (6.4), 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 (6.6) 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 (6.5), (6.6) and (6.7) are

instantiated with all the possible combinations of these types of terms. For example, the following is one of the conjuncts of Ψ generated from formula (6.5):

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

while the following dependency is one of the conjuncts of Ψ generated from formula (6.7):

$$\begin{aligned}\mathrm{dom}(x) \wedge \mathrm{dom}(y) \wedge f_C(f(x), f(y)) = g_C(f(x), f(y)) \wedge \\ f(x) = f(y) \rightarrow f_D(f(x), f(y)) = g_D(f(x), f(y)). \quad (6.9)\end{aligned}$$

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 (6.8) is instantiated with all the possible combinations of these types of terms. For example, the following is one of the conjuncts of Ψ generated by this process:

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

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:

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

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 need to consider only terms where the nesting depth of functions is at most 2.

Example 6.1. 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:

$$B(x, y) \ \rightarrow \ \exists z\, B(y, z), \quad (6.10)$$

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 specifies 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 (6.10) is replaced by the dependency:

$$B(x, y) \quad \rightarrow \quad B(y, h(x, y)). \tag{6.11}$$

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 $f_B(x, y) = g_B(x, y)$:

$$A(x, y) \quad \rightarrow \quad f_B(x, y) = g_B(x, y). \tag{6.12}$$

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:

$$\mathrm{dom}(x) \wedge \mathrm{dom}(y) \wedge f_B(x, y) = g_B(x, y) \quad \rightarrow \quad f_B(y, h(x, y)) = g_B(y, h(x, y)), \tag{6.13}$$

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 (6.12), and then one can infer that $f_B(b, h(a, b)) = g_B(b, h(a, b))$ holds by considering dependency (6.13). 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 (6.11)). Thus, in this case it should be possible to infer that

$$f_B(h(a, b), h(b, h(a, b))) = g_B(h(a, b), h(b, h(a, b))) \tag{6.14}$$

holds from the dependencies in Ψ. However, if $\mathrm{dom}(h(a, b))$ does not hold, then one cannot infer equality (6.14) from dependency (6.13) and the fact that $f_B(b, h(a, b)) = g_B(b, h(a, b))$ holds. This forces one to instantiate dependency (6.13) with the terms that appear in the tuples that are generated by repeatedly applying (6.11). In particular, the following dependency is included as a conjunct of Ψ to be able to infer (6.14) from equality $f_B(b, h(a, b)) = g_B(b, h(a, b))$:

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

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:

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

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 6.5. *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 6.5 was essentially established in [42] (see Theorems 6 and 9 and the paragraph after Theorem 10 in [42]), since the class of st-SO dependencies corresponds to the source-to-target restriction of the class of Sk$\forall$CQ$^{=}$ dependencies introduced in [42].

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 6.6. *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 6.6. In this section, we show that SO-standard schema mappings are exactly the needed class, since the converse of Theorem 6.6 also holds. Specifically, we have the following theorem.

Theorem 6.7. *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 6.8. *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 6.8. 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; furthermore, by Theorem 6.8, 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 6.7 follows from Theorem 6.8. 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 6.8, 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 mapping, 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 6.8 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 of Theorem 8.2 in [23], that every SO tgd specifies the composition of a finite number of st-tgd mappings.

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:

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

which 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 $\forall \bar{x}\,(\varphi \rightarrow t_1 = t_2)$. Thus, 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 6.8. 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 6.9. *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 6.8 follows immediately from Lemma 6.9 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 6.9 (the unnested case) rather than the general case that we dealt with in proving Theorem 6.8.

7. Collapsing Results: Nesting is Not Necessary

Recall that we say that an st-SO dependency or SO tgd is *unnested* if its depth of nesting is at most 1. Thus, an unnested st-SO dependency or SO tgd can contain terms like $f(x)$, but not terms like $f(g(x))$. 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

$$\exists f \exists g \forall x \forall y (P(x,y) \wedge (f(g(x)) = y) \rightarrow Q(x,y)) \tag{7.1}$$

The "obvious" way to denest (7.1) is to introduce a new variable z and rewrite (7.1) as

$$\exists f \exists g \forall x \forall y \forall z (P(x,y) \wedge (g(x) = z) \wedge (f(z) = y) \rightarrow Q(x,y)) \tag{7.2}$$

However, the formula (7.2) 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 (7.2)).

It should be mentioned that in [38], 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 [38] 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 [38]. However, no safety condition is imposed on the premises of SkSTDs in [38] 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 (7.2) is a valid constraint according to [38].

Before giving the proofs of Theorems 7.1 and 7.2, we present and discuss some corollaries of these theorems.

Corollary 7.1. 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 [23]) that the composition of a finite number of st-tgd mappings can be specified by an SO tgd (thus, Corollary 7.1 says that we can replace "SO tgd" in Theorem 8.1 in [23] by "unnested SO tgd"). Corollary 7.1 follows immediately from the result we just cited (Theorem 8.1 in [23]) 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 [23] 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 [23] 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 [25] can be expressed by unnested SO tgds. Corollary 7.1 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 7.2. The composition of a finite number of st-tgd mappings is equivalent to the composition of two st-tgd mappings.

This follows from Corollary 7.1 and the fact (which is a special case of Theorem 8.4 of [23]) 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 7.1 and 7.2 follow from Theorem 7.2.

Corollary 7.3. 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 7.4. 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 7.3 and Lemma 6.9 that we can slightly strengthen Corollary 7.4 as follows.

Corollary 7.5. 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 7.4 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 7.4, the reason that Corollary 7.2 is quite unexpected is that there is no obvious way to deal with all of the st-tgds in the intermediate schema mappings.

Corollary 7.3, unlike Corollary 7.4, 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 7.1, we can replace this composition by a single schema mapping, specified by an unnested SO tgd. Second, by Corollary 7.2, we can replace the composition by the composition of only two st-tgd mappings. Similarly, by using Corollaries 7.3 and 7.4, we have two alternative ways to deal with the composition of a large number of standard schema mappings.

We now provide the proofs of Theorems 7.1 and 7.2.

Proof of Theorem 7.1. In this proof, we use the following terminology. Given a term t, recursively define the set of non-atomic sub-terms of t, denoted by non-atomic(t), and the list of variables of t, denoted by list-var(t), as follows: (1) if $t = x$, where x is a variable, then non-atomic(t) $= \emptyset$ and list-var(t) $= [x]$; (2) if $t = f(t_1, t_2, \ldots, t_n)$, where t_1, t_2, ..., t_n are terms, then

$$\text{non-atomic}(t) \quad = \quad \{f(t_1, \ldots, t_n)\} \cup \bigcup_{i=1}^{n} \text{non-atomic}(t_i)$$

and list-var(t) = list-var[t_1]·list-var[t_2]·...·list-var[t_n], where $L_1 \cdot L_2$ is the result of appending L_2 to L_1. For example, if $t = f(x, h(z, y, g(x)))$, then non-atomic(t) $= \{f(x, h(z, y, g(x))), h(z, y, g(x)), g(x)\}$ and list-var($f(x, h(z, y, g(x)))$) $= [x, z, y, x]$. Moreover, consider a term replacement skel($\cdot$) that describes the *skeleton* of a term. For example, if $t = f(x, g(y))$, then skel(t) is $f(\_, g(\_))$, as this shows what are the functions that have been included in t and how they have been nested in this term.[1] More precisely, recursively define skel($\cdot$) as follows: (1) skel(x) = _ for every variable x, and (2) skel($f(t_1, \ldots, t_n)$) = $f(\text{skel}(t_1), \ldots, \text{skel}(t_n))$, for every n-ary symbol f and terms t_1, ..., t_n. For example, skel($f(x, h(z, y, g(x)))$) = $f(\_, h(\_, \_, g(\_)))$. Finally, by considering function skel($\cdot$), define a second term replacement $\xi(\cdot)$ as follows:

$$\xi(t) = \begin{cases} t & \text{if } t \text{ is a variable} \\ \xi_{\text{skel}(t)}(x_1, \ldots, x_n) & \text{if } t \text{ is a non-atomic term and list-var}(t) = [x_1, \ldots, x_n] \end{cases} \tag{7.3}$$

[1] It should be noticed that a similar term replacement was used in [17] to eliminate function expressions from a logic program. However, the term replacement used in [17] considers only terms without nesting of function symbols.

For example, if $t = f(x, h(z, y, g(x)))$, then $\xi(t) = \xi_{f(\_,h(\_,\_,g(\_)))}(x, z, y, x)$.

Given an st-SO dependency σ from a source schema $\mathbf{S}$ to a target schema $\mathbf{T}$, next we show how to construct an unnested st-SO dependency $\sigma^\star$ from $\mathbf{S}$ to $\mathbf{T}$ such that σ and $\sigma^\star$ are equivalent. Let t_1, ..., t_ℓ be the non-atomic terms t such that there exists an atomic formula mentioned in σ of the form either $t = t'$ or $t' = t$ or $R(t_1, \ldots, t_{i-i}, t, t_{i+1}, \ldots, t_k)$ (where k is the arity of R and $i \in \{1, \ldots, k\}$), let $\mathcal{H}(\sigma) = \{t_1, \ldots, t_\ell\}$, and let $\mathcal{ST}(\sigma)$ be the set of all non-atomic sub-terms of t_1, ..., t_ℓ, that is, $\mathcal{ST}(\sigma) = \bigcup_{i=1}^{\ell} \text{non-atomic}(t_i)$. Then define $\Xi(\sigma)$ as the following set of function symbols:

$$\Xi(\sigma) \quad = \quad \{\xi_{\text{skel}(t)} \mid t \in \mathcal{ST}(\sigma)\}.$$

For example, if σ is the following st-SO dependency:

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

then $\mathcal{H}(\sigma)$ is the set $\{f(x, g(y)), g(f(y, g(x)))\}$. Thus, given that non-atomic$(f(x, g(y))) = \{f(x, g(y)), g(y)\}$ and non-atomic$(g(f(y, g(x)))) = \{g(f(y, g(x))), f(y, g(x)), g(x)\}$, we have that $\mathcal{ST}(\sigma) = \{f(x, g(y)), g(y), g(f(y, g(x))), f(y, g(x)), g(x)\}$, and

$$\Xi(\sigma) \quad = \quad \{\xi_{f(\_,g(\_))},\ \xi_{g(\_)},\ \xi_{g(f(\_,g(\_)))}\},$$

Note that two members of $\mathcal{ST}(\sigma)$, namely $g(y)$ and $g(x)$, have the same skeleton $g(\_)$, as do $f(x, g(y))$ and $f(y, g(x))$, which have the same skeleton $f(\_, g(\_))$. The set $\Xi(\sigma)$ plays a fundamental role in the definition of the st-SO dependency $\sigma^\star$. More precisely, assume that $\Xi(\sigma) = \{\chi_1, \chi_2, \ldots, \chi_m\}$. Then $\sigma^\star$ is defined as:

$$\exists \chi_1 \exists \chi_2 \cdots \exists \chi_m\, \Psi,$$

where Ψ is defined as the conjunction of the following dependencies. For every conjunct $\forall \bar{x}\, (\varphi \rightarrow \psi)$ of σ, the st-SO dependency $\sigma^\star$ contains a conjunct $\forall \bar{x}(\varphi' \rightarrow \psi')$, where (1) φ' is obtained from φ by replacing every non-atomic term $t \in \mathcal{ST}(\sigma)$ by $\xi(t)$, and (2) ψ' is obtained from ψ by replacing every non-atomic term $t \in \mathcal{ST}(\sigma)$ by $\xi(t)$. Furthermore, for every pair of (non-necessarily distinct) terms $t, t' \in \mathcal{ST}(\sigma)$, if $t = f(t_1, \ldots, t_n)$ and $t' = f(t'_1, \ldots, t'_n)$, the following procedure is executed to obtain a set $\Gamma_{t,t'}$ of dependencies, and then $\bigwedge \Gamma_{t,t'}$ is included as a conjunct of $\sigma^\star$. First, replace each occurrence of a variable in t by a fresh variable to obtain a term $s = f(s_1, \ldots, s_n)$, and replace each occurrence of a variable in t' by a fresh variable to obtain a term $s' = f(s'_1, \ldots, s'_n)$ (in particular, s and s' have no variables in common). Assume that x_1, ..., x_p is the set of variables mentioned in s, s'. Second, as in the proof of Proposition 6.4, let dom$(\cdot)$ be a formula that defines the domain of the instances of $\mathbf{S}$. Finally, let $\Gamma_{t,t'}$ be a set of dependencies obtained from the dependency:

$$\forall x_1 \cdots \forall x_p \left[\left(\bigwedge_{i=1}^{p} \text{dom}(x_i) \right) \wedge \left(\bigwedge_{j=1}^{n} \xi(s_i) = \xi(s'_i) \right) \quad \rightarrow \quad \xi(s) = \xi(s') \right] \tag{7.4}$$

by repeatedly using the equivalences:

$$((\alpha \vee \beta) \wedge \gamma) \rightarrow \delta \quad \equiv \quad ((\alpha \wedge \gamma) \rightarrow \delta) \wedge ((\beta \wedge \gamma) \rightarrow \delta), \tag{7.5}$$

$$(\exists x\, \alpha) \rightarrow \beta \quad \equiv \quad \forall x\, (\alpha \rightarrow \beta) \qquad \text{if } x \text{ is not mentioned in } \beta, \tag{7.6}$$

until all the disjunctions and existential quantifications in the left-hand side of (7.4) have been eliminated.

Example 7.6. Let us give the intuition behind the definition of $\sigma^\star$ through an example. Assume that σ is the following st-SO dependency:

$$\exists f \exists g \left[\forall x \left(A(x) \rightarrow (T(x, f(g(x))) \wedge f(x) = g(x)) \right) \ \wedge \ \forall x \left(B(x) \rightarrow U(x, g(f(x))) \right) \right]. \quad (7.7)$$

Then we have that $\mathcal{ST}(\sigma) = \{f(g(x)), g(x), g(f(x)), f(x)\}$ and $\Xi(\sigma) = \{\xi_{f(g(\_))}, \xi_{g(\_)}, \xi_{g(f(\_))}, \xi_{f(\_)}\}$. Intuitively, $\xi_{f(\_)}$ and $\xi_{g(\_)}$ are used to represent functions f and g, respectively, and $\xi_{f(g(\_))}$ and $\xi_{g(f(\_))}$ are used to represent the composition functions $(g \circ f)$ and $(f \circ g)$, respectively, thus eliminating the nesting of functions from σ. More precisely, the st-SO dependency $\sigma^\star$ is defined as:

$$\exists \xi_{f(g(\_))} \exists \xi_{g(\_)} \exists \xi_{g(f(\_))} \exists \xi_{f(\_)} \, \Psi,$$

where Ψ is defined as the conjunction of the following dependencies. First, given that $\forall x \, (A(x) \rightarrow (T(x, f(g(x))) \wedge f(x) = g(x)))$ and $\forall x \, (B(x) \rightarrow U(x, g(f(x))))$ are the conjuncts of σ, the following dependencies are conjuncts of Ψ:

$$\forall x \, (A(x) \rightarrow (T(x, \xi_{f(g(\_))}(x)) \wedge \xi_{f(\_)}(x) = \xi_{g(\_)}(x))),$$
$$\forall x \, (B(x) \rightarrow U(x, \xi_{g(f(\_))}(x))).$$

Furthermore, for every pair t, t' of (non-necessarily distinct) terms from $\mathcal{ST}(\sigma)$, if either $t = f(t_1)$ and $t' = f(t'_1)$ or $t = g(t_1)$ and $t' = g(t'_1)$, then the following conjuncts are included in Ψ. Assume that $t = t' = f(g(x))$. First, each occurrence of a variable in these terms is replaced by a fresh variable, generating the terms $s = f(g(u))$ and $s' = f(g(v))$. Second, given that the source schema consists of the unary predicates A and B, formula $\mathrm{dom}(x)$ is defined as $A(x) \vee B(x)$ (that is, $\mathrm{dom}(x)$ holds if x is in the domain of a source instance). Finally, assuming that $s_1 = g(u)$ and $s'_1 = g(v)$, let α be the following dependency:

$$\forall u \forall v \left[\mathrm{dom}(u) \wedge \mathrm{dom}(v) \wedge \xi(s_1) = \xi(s'_1) \ \rightarrow \ \xi(s) = \xi(s') \right],$$

that is, α is:

$$\forall u \forall v \left[(A(u) \vee B(u)) \wedge (A(v) \vee B(v)) \wedge \xi_{g(\_)}(u) = \xi_{g(\_)}(v) \ \rightarrow \ \xi_{f(g(\_))}(u) = \xi_{f(g(\_))}(v) \right].$$

Then the set $\Gamma_{t,t'}$ consists of the following dependencies:

$$\forall u \forall v \left[A(u) \wedge A(v) \wedge \xi_{g(\_)}(u) = \xi_{g(\_)}(v) \ \rightarrow \ \xi_{f(g(\_))}(u) = \xi_{f(g(\_))}(v) \right],$$
$$\forall u \forall v \left[A(u) \wedge B(v) \wedge \xi_{g(\_)}(u) = \xi_{g(\_)}(v) \ \rightarrow \ \xi_{f(g(\_))}(u) = \xi_{f(g(\_))}(v) \right],$$
$$\forall u \forall v \left[B(u) \wedge A(v) \wedge \xi_{g(\_)}(u) = \xi_{g(\_)}(v) \ \rightarrow \ \xi_{f(g(\_))}(u) = \xi_{f(g(\_))}(v) \right],$$
$$\forall u \forall v \left[B(u) \wedge B(v) \wedge \xi_{g(\_)}(u) = \xi_{g(\_)}(v) \ \rightarrow \ \xi_{f(g(\_))}(u) = \xi_{f(g(\_))}(v) \right],$$

and each one of these four dependencies is a conjunct of $\sigma^\star$. It is important to notice that these dependencies make explicit some properties that are implicit in σ. Given that f and g are function symbols in σ, we know that if $g(u) = g(v)$, then $f(g(u)) = f(g(v))$. But this property does not immediately hold for $\xi_{f(g(\_))}$ and $\xi_{g(\_)}$ and, thus, we have to include the above four conjuncts into $\sigma^\star$ to enforce it. It should also be noticed that the formula

dom(·) 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.

To give more intuition about the definition of dependency $\sigma^\star$, we also consider the case $t = f(g(x))$ and $t' = f(x)$. As above, we start by replacing each occurrence of a variable in these terms by a fresh variable, generating the terms $s = f(s_1)$ and $s' = f(s'_1)$, where $s_1 = g(u)$ and $s'_1 = v$. Then we define a dependency β as:

$$\beta \;=\; \forall u \forall v \left[(A(u) \vee B(u)) \wedge (A(v) \vee B(v)) \wedge \xi_{g(\_)}(u) = v \;\rightarrow\; \xi_{f(g(\_))}(u) = \xi_{f(\_)}(v) \right],$$

and, therefore, in this case the set $\Gamma_{t,t'}$ consists of the following dependencies:

$$\forall u \forall v \left[A(u) \wedge A(v) \wedge \xi_{g(\_)}(u) = v \;\rightarrow\; \xi_{f(g(\_))}(u) = \xi_{f(\_)}(v) \right],$$

$$\forall u \forall v \left[A(u) \wedge B(v) \wedge \xi_{g(\_)}(u) = v \;\rightarrow\; \xi_{f(g(\_))}(u) = \xi_{f(\_)}(v) \right],$$

$$\forall u \forall v \left[B(u) \wedge A(v) \wedge \xi_{g(\_)}(u) = v \;\rightarrow\; \xi_{f(g(\_))}(u) = \xi_{f(\_)}(v) \right],$$

$$\forall u \forall v \left[B(u) \wedge B(v) \wedge \xi_{g(\_)}(u) = v \;\rightarrow\; \xi_{f(g(\_))}(u) = \xi_{f(\_)}(v) \right].$$

As in the previous case, each one of the dependencies of $\Gamma_{t,t'}$ is a conjunct of $\sigma^\star$. It is important to notice that these dependencies make explicit the fact that in σ, if $g(u) = v$, then $f(g(u)) = f(v)$.

For the st-SO dependency (7.7), we took dom(u) to be $A(u) \vee B(u)$. We then made use of (7.5) to eliminate the disjunction in $A(u) \vee B(u)$. If the left-hand side of the first conjunct of (7.7) had been $P(x, y)$ instead of $A(x)$, we would have taken dom(u) to be $\exists w P(u, w) \vee \exists w P(w, u) \vee B(u)$. We then would have made use not only of (7.5), but also (7.6), to eliminate the disjunctions and existential quantifiers in $\exists w P(u, w) \vee \exists w P(w, u) \vee B(u)$. □

We now prove that $\sigma \Leftrightarrow \sigma^\star$, that is, that σ and $\sigma^\star$ are equivalent.

($\Rightarrow$) If $(I, J) \models \sigma$, then it is straightforward to prove that $(I, J) \models \sigma^\star$ (the interpretation of each function symbol in $\Xi(\sigma)$ is defined from the corresponding composition of the interpretations of the function symbols from σ).

($\Leftarrow$) Assume that $\Xi(\sigma) = \{\chi_1, \ldots, \chi_m\}$ and that $(I, J) \models \sigma^\star$ with the instantiations χ_1^0, …, χ_m^0 of χ_1, …, χ_m. Moreover, assume that f_1, …, f_ℓ are the function symbols mentioned in σ. To show that $(I, J) \models \sigma$, we first need to define from χ_1^0, …, χ_m^0 the instantiations f_1^0, …, f_ℓ^0 of function symbols f_1, …, f_ℓ, and then we have to show that (I, J) satisfies all the conjuncts of σ with these instantiations.

Given that $(I, J) \models \sigma^\star$, we have by definition of satisfaction for st-SO dependencies that there exists a countably infinite universe[2] U such that (1) U is the union of dom$(I) \cup$ dom(J) and a set of nulls, and (2) $(U; I, J)$ satisfies $\sigma^\star$ in the standard second-order logic sense. Assume that $\perp$ is a fresh null value ($\perp \notin U$) and that the arity of function symbol f_i is k_i ($1 \leq i \leq \ell$). Then the domain of each one of the functions f_1^0, …, f_ℓ^0 is defined to be $U \cup \{\perp\}$, and for every $(a_1, \ldots, a_{k_i}) \in (U \cup \{\perp\})^{k_i}$, we define $f_i^0(a_1, \ldots, a_{k_i})$ as follows. If there exist $f_i(t_1, \ldots, t_{k_i}) \in \mathcal{ST}(\sigma)$ and tuples $\bar{b}_1$, …, $\bar{b}_{k_i}$ such that for every $i \in \{1, \ldots, k_i\}$:

[2]As noted earlier, the universe can even be taken to be finite, but we do not need this.

- t_i is a variable, $\bar{b}_i = (a_i)$ and $a_i \in \mathrm{dom}(I)$; or
- t_i is a non-atomic term, $a_i = \xi^0_{\mathrm{skel}(t_i)}(\bar{b}_i)$ and $\bar{b}_i \subseteq \mathrm{dom}(I)$ (that is, every element mentioned in $\bar{b}_i$ is in $\mathrm{dom}(I)$);

then $f^0_i(a_1, \ldots, a_{k_i})$ is defined as $\xi^0_{\mathrm{skel}(f_i(t_1,\ldots,t_{k_i}))}(\bar{b}_1, \ldots, \bar{b}_{k_i})$. Otherwise, $f^0_i(a_1, \ldots, a_{k_i})$ is defined as $\perp$ (in particular, if $a_i = \perp$ for some $i \in \{1, \ldots, k_i\}$, then $f^0_i(a_1, \ldots, a_{k_i}) = \perp$).

Before showing that all the conjuncts of σ are satisfied by (I, J) under the instantiations f^0_1, ..., f^0_ℓ of function symbols f_1, ..., f_ℓ, we need to show that these functions are well defined. That is, we have to show that if by using the above definition, one has different ways of assigning a value to $f^0_i(a_1, \ldots, a_{k_i})$, then all these ways assign the same value to $f^0_i(a_1, \ldots, a_{k_i})$. In order to prove this, we need to consider several cases. In this proof, we consider only one of these cases, as the other ones can be handled in the same way. Assume that for $i \in \{1, \ldots, \ell\}$ and elements a_1, ..., a_{k_i} from $U \cup \{\perp\}$, it holds that (1) $f_i(t_1, \ldots, t_{k_i}) \in \mathcal{ST}(\sigma)$, (2) $a_i = \xi^0_{\mathrm{skel}(t_i)}(\bar{b}_i)$ and $\bar{b}_i \subseteq \mathrm{dom}(I)$, for every $i \in \{1, \ldots, k_i\}$, (3) $f_i(s_1, \ldots, s_{k_i}) \in \mathcal{ST}(\sigma)$, and (4) $a_i = \xi^0_{\mathrm{skel}(s_i)}(\bar{c}_i)$ and $\bar{c}_i \subseteq \mathrm{dom}(I)$, for every $i \in \{1, \ldots, k_i\}$. Then we have to prove that:

$$\xi^0_{\mathrm{skel}(f_i(t_1,\ldots,t_{k_i}))}(\bar{b}_1, \ldots, \bar{b}_{k_i}) \quad = \quad \xi^0_{\mathrm{skel}(f_i(s_1,\ldots,s_{k_i}))}(\bar{c}_1, \ldots, \bar{c}_{k_i}). \tag{7.8}$$

Given a tuple $\bar{x} = (x_1, \ldots, x_p)$ of variables, let $\mathrm{dom}(\bar{x})$ be a shorthand for $\mathrm{dom}(x_1) \wedge \cdots \wedge \mathrm{dom}(x_p)$. By definition of $\sigma^\star$ and the fact that $(I, J) \models \sigma^\star$, we have that (I, J) satisfies the following instantiated dependency:

$$\left(\bigwedge_{i=1}^{k_i} \mathrm{dom}(\bar{b}_i)\right) \wedge \left(\bigwedge_{i=1}^{k_i} \mathrm{dom}(\bar{c}_i)\right) \wedge \left(\bigwedge_{i=1}^{k_i} \xi_{\mathrm{skel}(t_i)}(\bar{b}_i) = \xi_{\mathrm{skel}(s_i)}(\bar{c}_i)\right) \rightarrow$$
$$\xi_{\mathrm{skel}(f_i(t_1,\ldots,t_{k_i}))}(\bar{b}_1, \ldots, \bar{b}_{k_i}) = \xi_{\mathrm{skel}(f_i(s_1,\ldots,s_{k_i}))}(\bar{c}_1, \ldots, \bar{c}_{k_i}).$$

Thus, we conclude that (7.8) holds, since for every $i \in \{1, \ldots, k_i\}$, it holds that $\xi^0_{\mathrm{skel}(t_i)}(\bar{b}_i) = a_i = \xi^0_{\mathrm{skel}(s_i)}(\bar{c}_i)$, $\bar{b}_i \subseteq \mathrm{dom}(I)$ and $\bar{c}_i \subseteq \mathrm{dom}(I)$.

Now we move to the proof that all the conjuncts of σ are satisfied by (I, J) under the instantiations f^0_1, ..., f^0_ℓ of function symbols f_1, ..., f_ℓ. In this proof, we need the following claim, where we use the following terminology. Given a non-atomic term $t = f_i(t_1, \ldots, t_{k_i})$ based on variables x_1, ..., x_k and function symbols f_1, ..., f_ℓ, and given a variable substitution $\rho : \{x_1, \ldots, x_k\} \rightarrow (U \cup \{\perp\})$, the evaluation of ρ over t is recursively defined as $\rho(t) = f^0_i(\rho(t_1), \ldots, \rho(t_{k_i}))$.

Claim 7.7. Let $t \in \mathcal{ST}(\sigma)$ such that list-var$(t) = [x_1, \ldots, x_k]$. Then for every variable substitution $\rho : \{x_1, \ldots, x_k\} \rightarrow \mathrm{dom}(I)$, it holds that $\rho(t) = \xi^0_{\mathrm{skel}(t)}(\rho(x_1), \ldots, \rho(x_k))$.

Proof. By induction on the depth of nesting of functions in t.

- Base case: If the depth of nesting of functions in t is 1, then $t = f_i(x_1, \ldots, x_k)$ and $k = k_i$. Then we have that $\rho(f_i(x_1, \ldots, x_k)) = f^0_i(\rho(x_1), \ldots, \rho(x_k))$. But given that $\rho(x_j) \in \mathrm{dom}(I)$ for every $j \in \{1, \ldots, k\}$, we have by definition of f^0_i that $f^0_i(\rho(x_1), \ldots, \rho(x_k)) = \xi^0_{\mathrm{skel}(f_i(x_1,\ldots,x_k))}(\rho(x_1), \ldots, \rho(x_k))$. Thus, we conclude that $\rho(t) = \xi^0_{\mathrm{skel}(t)}(\rho(x_1), \ldots, \rho(x_k))$.
- Inductive step: Assume that the depth of nesting of functions in t is p, and that the property holds for every term with depth of nesting of functions smaller than p. In this case, we have that $t = f_i(t_1, \ldots, t_{k_i})$. Thus, we have that $\rho(t) = f^0_i(\rho(t_1), \ldots, \rho(t_{k_i}))$. If t_i is a variable, then we have that $\rho(t_i) \in \mathrm{dom}(I)$ since $\rho : \{x_1, \ldots, x_k\} \rightarrow \mathrm{dom}(I)$.

On the other hand, if t_i is a non-atomic term such that list-var$(t_i) = [u_1, \ldots, u_q]$ (with $[u_1, \ldots, u_q]$ a sub-list of $[x_1, \ldots, x_k]$, that is, $[u_1, \ldots, u_q]$ consisting of consecutive elements of $[x_1, \ldots, x_k]$), then given that the depth of nesting of functions in t_i is smaller than p, we have by induction hypothesis that $\rho(t_i) = \xi^0_{\text{skel}(t_i)}(\rho(u_1), \ldots, \rho(u_q))$. Thus, given that $\rho : \{x_1, \ldots, x_k\} \to \text{dom}(I)$, we have by definition of f^0_i that $f^0_i(\rho(t_1), \ldots, \rho(t_{k_i})) = \xi^0_{\text{skel}(f_i(t_1,\ldots,t_{k_i}))}(\rho(x_1), \ldots, \rho(x_k))$ and, therefore, $\rho(t) = \xi^0_{\text{skel}(t)}(\rho(x_1), \ldots, \rho(x_k))$. This concludes the proof of the claim. □

We finally have all the necessary ingredients to prove that $(I, J) \models \sigma$. More precisely, we show next that all the conjuncts of σ are satisfied by (I, J) under the instantiations f^0_1, ..., f^0_ℓ of function symbols f_1, ..., f_ℓ. Let

$$\forall x_1 \cdots \forall x_k \forall y_1 \cdots \forall y_m \, (\varphi(x_1, \ldots, x_k, y_1, \ldots, y_m) \to \psi(x_1, \ldots, x_k)) \tag{7.9}$$

be a conjunct of σ, and let ρ be a variable substitution with domain $\{x_1, \ldots, x_k, y_1, \ldots, y_m\}$ and range contained in dom(I), and assume that $I \models \varphi(\rho(x_1), \ldots, \rho(x_k), \rho(y_1), \ldots, \rho(y_m))$ with the instantiations f^0_1, ..., f^0_ℓ. Next we show that $J \models \psi(\rho(x_1), \ldots, \rho(x_k))$. Assume that

$$\forall x_1 \cdots \forall x_k \forall y_1 \cdots \forall y_m \, (\varphi'(x_1, \ldots, x_k, y_1, \ldots, y_m) \to \psi'(x_1, \ldots, x_k))$$

is the conjunct of $\sigma^\star$ obtained from (7.9) by replacing every non-atomic term $t \in \mathcal{ST}(\sigma)$ by $\xi(t)$. Then given that the range of ρ is contained in dom(I), we have by Claim 7.7 and definition of $\sigma^\star$ that $I \models \varphi'(\rho(x_1), \ldots, \rho(x_k), \rho(y_1), \ldots, \rho(y_m))$ with the instantiations χ^0_1, ..., χ^0_m of the function symbols χ_1, ..., χ_m (recall that $\Xi(\sigma) = \{\chi_1, \ldots, \chi_m\}$). Thus, we conclude that $J \models \psi'(\rho(x_1), \ldots, \rho(x_k))$ with the instantiations χ^0_1, ..., χ^0_m (as (I, J) satisfies the conjuncts of $\sigma^\star$ with these instantiations). Therefore, again by Claim 7.7 and definition of $\sigma^\star$, we have that $J \models \psi(\rho(x_1), \ldots, \rho(x_k))$ with the instantiations f^0_1, ..., f^0_ℓ, which was to be shown. This concludes the proof of the theorem. □

We now move to the proof of Theorem 7.2. The following lemma will be used in this proof.

Lemma 7.3. *For every SO tgd σ of nesting depth 2, there exists an unnested SO tgd $\sigma^\star$ that is equivalent to σ.*

Proof. In this proof, we extensively used the terminology defined in the proof of Theorem 7.1. Besides, we say that a term t is an i-term if the depth of nesting of function symbols in t is i. For example, $f(x, y)$ is a 1-term while $g(f(x, y), z)$ is a 2-term.

Let σ be an SO tgd from a source schema $\mathbf{S}$ to a target schema $\mathbf{T}$. Assume that the depth of nesting of function symbols in every term mentioned in σ is at most 2, and that f_1, ..., f_ℓ are the function symbols mentioned in σ. Then define a set Θ of dependencies as follows. For every conjunct α of σ, we include the following dependencies as elements of Θ. Assume that t_1, ..., t_m are the 1-terms t for which there exists a 2-term t' mentioned in α such that $t \in$ non-atomic(t'). For example, if α is the following dependency:

$$\forall x \forall y \, (S(x, y) \wedge f(x) = g(f(x), f(y)) \to T(f(g(x, x)))), \tag{7.10}$$

then $f(x)$, $f(y)$ and $g(x, x)$ are the only 1-terms satisfying the preceding condition. Furthermore, for every $i \in \{1, \ldots, m\}$, define T_i as the set $\{x, f_1(\bar{x}_1), \ldots, f_\ell(\bar{x}_\ell)\}$ of terms, where: (1) x is a variable, (2) each $\bar{x}_j$ $(1 \leq j \leq \ell)$ is a tuple of pairwise distinct variables, (3) for every $j \in \{1, \ldots, \ell\}$, we have that x is not mentioned in $\bar{x}_j$, and (4) for every pair j, k of distinct values in $\{1, \ldots, \ell\}$, we have that $\bar{x}_j$ and $\bar{x}_k$ do not have variables in common.

Besides, assume that for every pair i, j of distinct values in $\{1, \ldots, m\}$, we have that T_i and T_j do not have variables in common, and for every $i \in \{1, \ldots, m\}$, we have that T_i and α do not have variables in common. For example, assuming that $t_1 = f(x)$, $t_2 = f(y)$ and $t_3 = g(x, x)$ for the case of conjunct (7.10), we have that:

$$\begin{aligned} T_1 &= \{u_1, f(u_2), g(u_3, u_4)\}, \\ T_2 &= \{u_5, f(u_6), g(u_7, u_8)\}, \\ T_3 &= \{u_9, f(u_{10}), g(u_{11}, u_{12})\}, \end{aligned}$$

satisfy the preceding conditions.

Assume that α is $\forall \bar{x}\,(\varphi \rightarrow \psi)$. Then for every $s_1 \in T_1$, ..., $s_m \in T_m$, define dependency $\theta_{s_1,\ldots,s_m}$ as:

$$\forall \bar{x} \forall y_1 \cdots \forall y_n \left[\left(\left(\bigwedge_{i=1}^{n} \mathrm{dom}(y_i) \right) \wedge \left(\bigwedge_{j=1}^{m} \xi(t_j) = \xi(s_j) \right) \wedge \varphi' \right) \rightarrow \psi' \right],$$

where ξ is defined as in the proof of Theorem 7.1 (see (7.3)), y_1, ..., y_n are the variables mentioned in the terms s_1, ..., s_m, and φ', ψ' are obtained from φ and ψ, respectively, by replacing as follows the 1-terms and 2-terms of these formulas. Every 1-term t mentioned in φ (resp. ψ) is replaced by $\xi(t)$ in φ' (resp. ψ'). Furthermore, every 2-term $t = f(t'_1, \ldots, t'_p)$ mentioned in φ (resp. ψ) is replaced by $\xi(f(t''_1, \ldots, t''_p))$ in φ' (resp. ψ'), where t''_i $(1 \leq i \leq p)$ is defined as follows. If t'_i is a 1-term, and so t'_i is t_j for some j in $\{1, \ldots, m\}$, then let t''_i be s_j. If t'_i is a variable v, then let t''_i be v.

Finally, for each dependency $\theta_{s_1,\ldots,s_m}$, let $\Theta_{s_1,\ldots,s_m}$ be a set of dependencies obtained from $\theta_{s_1,\ldots,s_m}$ by repeatedly using the equivalences:

$$\begin{aligned} ((\alpha \vee \beta) \wedge \gamma) \rightarrow \delta &\equiv ((\alpha \wedge \gamma) \rightarrow \delta) \wedge ((\beta \wedge \gamma) \rightarrow \delta), \\ (\exists x\, \alpha) \rightarrow \beta &\equiv \forall x\, (\alpha \rightarrow \beta) \qquad \text{if } x \text{ is not mentioned in } \beta, \end{aligned}$$

until all the disjunctions and existential quantifications in the left-hand side of $\theta_{s_1,\ldots,s_m}$ have been eliminated. Then all the dependencies in $\Theta_{s_1,\ldots,s_m}$ are included in Θ.

Example 7.8. Let us give the intuition behind the definition of Θ through an example. Assume that α is the following conjunct of SO tgd σ:

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

Then, as mentioned above, we have that $t_1 = f(x)$, $t_2 = f(y)$, $t_3 = g(x, x)$ are the 1-terms t for which there exists a 2-term t' in α such that $t \in \text{non-atomic}(t')$. Furthermore, as also mentioned above, we can assume that:

$$\begin{aligned} T_1 &= \{u_1, f(u_2), g(u_3, u_4)\}, \\ T_2 &= \{u_5, f(u_6), g(u_7, u_8)\}, \\ T_3 &= \{u_9, f(u_{10}), g(u_{11}, u_{12})\}. \end{aligned}$$

Then for every $s_1 \in T_1$, $s_2 \in T_2$ and $s_3 \in T_3$, we have to compute the formula θ_{s_1,s_2,s_3}, and then to include all the dependencies of Θ_{s_1,s_2,s_3} as elements of Θ. Assume that $s_1 = u_1$,

$s_2 = g(u_7, u_8)$ and $s_3 = f(u_{10})$. Then θ_{s_1,s_2,s_3} is the following dependency:

$$\forall x \forall y \forall u_1 \forall u_7 \forall u_8 \forall u_{10} \Big[\Big(S(x,y) \wedge \mathrm{dom}(u_1) \wedge \mathrm{dom}(u_7) \wedge \mathrm{dom}(u_8) \wedge \mathrm{dom}(u_{10}) \wedge \xi_{f(\_)}(x) = u_1 \wedge \xi_{f(\_)}(y) = \xi_{g(\_,\_)}(u_7, u_8) \wedge \xi_{g(\_,\_)}(x,x) = \xi_{f(\_)}(u_{10}) \wedge \xi_{f(\_)}(x) = \xi_{g(\_,g(\_,\_))}(u_1, u_7, u_8) \Big) \rightarrow T(\xi_{f(f(\_))}(u_{10})) \Big].$$

We note that equalities $\xi_{f(\_)}(x) = u_1$, $\xi_{f(\_)}(y) = \xi_{g(\_,\_)}(u_7, u_8)$ and $\xi_{g(\_,\_)}(x,x) = \xi_{f(\_)}(u_{10})$ correspond to $\xi(t_1) = \xi(s_1)$, $\xi(t_2) = \xi(s_2)$ and $\xi(t_3) = \xi(s_3)$ in the definition of dependency θ_{s_1,s_2,s_3}, respectively. Furthermore, we note that equality $\xi_{f(\_)}(x) = \xi_{g(\_,g(\_,\_))}(u_1, u_7, u_8)$ is generated as follows from equality $f(x) = g(f(x), f(y))$ in α. The 1-term $f(x)$ in the left-hand side of this equality is replaced by $\xi(f(x)) = \xi_{f(\_)}(x)$ in θ_{s_1,s_2,s_3}, and the 2-term $g(f(x), f(y))$ is replaced by $\xi(g(u_1, g(u_7, u_8))) = \xi_{g(\_,g(\_,\_))}(u_1, u_7, u_8)$ in θ_{s_1,s_2,s_3} (since $t_1 = f(x)$, $s_1 = u_1$, $t_2 = f(y)$ and $s_2 = g(u_7, u_8)$). Finally, we note that $T(\xi_{f(f(\_))}(u_{10}))$ is obtained by replacing the 2-term $f(g(x,x))$ by $\xi(f(f(u_{10}))) = \xi_{f(f(\_))}(u_{10})$ (since $t_3 = g(x,x)$ and $s_3 = f(u_{10})$).

Given that $\mathrm{dom}(x) = \exists u S(x,u) \vee \exists v S(v,x)$ in this case, we have that the following dependency is one of the elements of Θ_{s_1,s_2,s_3}:

$$\forall x \forall y \forall u_1 \forall u_7 \forall u_8 \forall u_{10} \forall z_1 \forall z_7 \forall z_8 \forall z_{10} \Big[\Big(S(x,y) \wedge S(u_1,z_1) \wedge S(z_7,u_7) \wedge S(z_8,u_8) \wedge S(u_{10},z_{10}) \wedge \xi_{f(\_)}(x) = u_1 \wedge \xi_{f(\_)}(y) = \xi_{g(\_,\_)}(u_7,u_8) \wedge \xi_{g(\_,\_)}(x,x) = \xi_{f(\_)}(u_{10}) \wedge \xi_{f(\_)}(x) = \xi_{g(\_,g(\_,\_))}(u_1,u_7,u_8) \Big) \rightarrow T(\xi_{f(f(\_))}(u_{10})) \Big].$$

As in the proof of Theorem 7.1, function symbols $\xi_{f(\_)}$, $\xi_{g(\_,\_)}$ are used to represent functions f and g, respectively, and $\xi_{g(\_,g(\_,\_))}$, $\xi_{f(f(\_))}$ are used to represent functions $g(x, g(y,z))$ and $f(f(x))$, respectively, thus eliminating the nesting of functions from α. It is important to notice that the preceding dependency makes explicit some properties that are implicit in α. Given that f and g are function symbols in α, we know that if $f(x) = u_1$, $f(y) = g(u_7, u_8)$, $g(x,x) = f(u_{10})$, $f(x) = g(f(x), f(y))$ and $T(f(g(x,x)))$ hold, then $f(x) = g(u_1, g(u_7, u_8))$ and $T(f(f(u_{10})))$ also hold. But this property is not immediately true for $\xi_{g(\_,g(\_,\_))}$ and $\xi_{f(f(\_))}$, and, thus, we have to include the preceding dependency in Θ to enforce it. □

Assume that $\chi_1, \ldots, \chi_k$ are the function symbols mentioned in Θ. Then unnested SO tgd $\sigma^\star$ is defined as:

$$\exists \chi_1 \cdots \exists \chi_k \, (\bigwedge \Theta).$$

Next we show that σ and $\sigma^\star$ are equivalent.

($\Rightarrow$) If $(I,J) \models \sigma$, then it is straightforward to prove that $(I,J) \models \sigma^\star$ (the interpretation of each function symbol mentioned in Θ is defined from the corresponding composition of the interpretations of the function symbols from σ).

($\Leftarrow$) Assume that $(I,J) \models \sigma^\star$ with the instantiations $\chi_1^0, \ldots, \chi_k^0$ of the function symbols $\chi_1, \ldots, \chi_k$. To show that $(I,J) \models \sigma$, we first need to define from $\chi_1^0, \ldots, \chi_k^0$ the instantiations $f_1^0, \ldots, f_\ell^0$ of the function symbols $f_1, \ldots, f_\ell$, and then we have to show that (I,J) satisfies all the conjuncts of σ with these instantiations.

Let $\mathcal{FT}(I)$ be the set of all pairs $(f_i, \bar{a})$ such that (1) $i \in \{1, \ldots, \ell\}$, (2) $\bar{a}$ is a tuple of elements from $\mathrm{dom}(I)$, and (3) the length of $\bar{a}$ is the same as the arity of function symbol f_i. Furthermore, let $<$ be an arbitrary linear order over $\mathcal{FT}(I)$, and define $\mathcal{DT}(J)$ as the set of elements $a \in \mathrm{dom}(J)$ such that $a \in \mathrm{dom}(I)$ or $a = \xi^0_{f_i(\_,\ldots,\_)}(\bar{a})$ for some $i \in \{1, \ldots, \ell\}$ and tuple $\bar{a}$ of elements from $\mathrm{dom}(I)$. The sets $\mathcal{FT}(I)$ and $\mathcal{DT}(J)$ are used to define a substitution κ, which in turn is used to define the interpretations of function symbols f_1, …, f_ℓ. More precisely, for every $a \in \mathcal{DT}(J)$, define:

$$\kappa(a) = \begin{cases} (\_, a) & \text{if } a \in \mathrm{dom}(I) \\ (f_j(\_, \ldots, \_), \bar{b}) & \text{if } a \notin \mathrm{dom}(I) \text{ and } (f_j, \bar{b}) = \min_{<}\{(f_k, \bar{c}) \in \mathcal{FT}(I) \mid \xi^0_{f_k(\_,\ldots,\_)}(\bar{c}) = a\} \end{cases}$$

Note that in the second case in the definition of $\kappa(a)$, the set over which the min is taken is nonempty, because $a \in \mathcal{DT}(J)$ and $a \notin \mathrm{dom}(I)$.

Define the instantiations f^0_1, …, f^0_ℓ of function symbols f_1, …, f_ℓ as follows. Given that $(I, J) \models \sigma^\star$, we have by definition of satisfaction for SO tgds that there exists a countably infinite universe[3] U such that (1) U is the union of $\mathrm{dom}(I) \cup \mathrm{dom}(J)$ and a set of nulls, and (2) $(U; I, J)$ satisfies $\sigma^\star$ in the standard second-order logic sense. Assume that $\perp$ is a fresh null value ($\perp \notin U$) and that the arity of function symbol f_i is k_i ($1 \leq i \leq \ell$). Then the domain of each one of the functions f^0_1, …, f^0_ℓ is defined to be $U \cup \{\perp\}$, and for every $(a_1, \ldots, a_{k_i}) \in (U \cup \{\perp\})^{k_i}$, we define:

$$f^0_i(a_1, \ldots, a_{k_i}) = \begin{cases} \xi^0_{f_i(w_1,\ldots,w_{k_i})}(\bar{b}_1, \ldots, \bar{b}_{k_i}) & \text{if for every } i \in \{1, \ldots, k_i\}, \text{ it holds that} \\ & \quad a_i \in \mathcal{DT}(J) \text{ and } \kappa(a_i) = (w_i, \bar{b}_i) \\ \perp & \text{otherwise} \end{cases}$$

Notice that if $a_i = \perp$ in the definition above, for some $i \in \{1, \ldots, k_i\}$, then $f^0_i(a_1, \ldots, a_{k_i}) = \perp$.

Next we show that (I, J) satisfies[4] every conjunct of σ with the instantiations f^0_1, …, f^0_ℓ of function symbols f_1, …, f_ℓ. But before doing this, we give an example that shows how the strategy of the proof works.

Example 7.9. Consider again conjunct (7.10). To prove that (I, J) satisfies this conjunct under the preceding definition of the functions in σ, we have to prove that if:

$$I \quad \models \quad S(a, b) \wedge f(a) = g(f(a), f(b)),$$

then $J \models T(f(g(a, a)))$. In order to prove this, we first need to figure out what the values of $f^0(a)$, $f^0(b)$, $g^0(f^0(a), f^0(b))$ and $f^0(g^0(a, a))$ are. Given that $a, b \in \mathrm{dom}(I)$, we have that $f^0(a) = \xi^0_{f(\_)}(a)$ and $f^0(b) = \xi^0_{f(\_)}(b)$. The definition of $g^0(f^0(a), f^0(b))$ depends on whether $\xi^0_{f(\_)}(a)$ and $\xi^0_{f(\_)}(b)$ belong to $\mathrm{dom}(I)$. Assume that $\xi^0_{f(\_)}(a) = a_1$ and $\xi^0_{f(\_)}(b) = b_1$, where $a_1 \in \mathrm{dom}(I)$ and $b_1 \notin \mathrm{dom}(I)$. Then by the preceding definition, we have to compute $\kappa(a_1)$ and $\kappa(b_1)$ in order to compute the value of $g^0(f^0(a), f^0(b))$. We have that $\kappa(a_1) = (\_, a_1)$ since $a_1 \in \mathrm{dom}(I)$, and we assume for this example that $\kappa(b_1) = (g(\_, \_), (c_1, c_2))$, where $c_1, c_2 \in \mathrm{dom}(I)$. That is, we assume that $\xi^0_{g(\_,\_)}(c_1, c_2) = b_1$ and $(g, (c_1, c_2))$ is the

[3]Again, the universe can even be taken to be finite, but we do not need this.

[4]When we say that (I, J) satisfies a formula, we mean that $(U; I, J)$ satisfies the formula. Similarly, when we say that I satisfies a formula, we mean that $(U; I)$ satisfies the formula, and likewise for J satisfying a formula.

smallest element $(h, \bar{d})$ in $\mathcal{FT}(I)$, according to the linear order $<$, satisfying the condition $\xi^0_{h(\_,\ldots,\_)}(\bar{d}) = b_1$. Thus, by the preceding definition, we have that:

$$g^0(f^0(a), f^0(b)) \;=\; g^0(a_1, b_1) \;=\; \xi^0_{g(\_,g(\_,\_))}(a_1, c_1, c_2).$$

Finally, we also need to know what the value of $f^0(g^0(a,a))$ is. By the preceding definition, we know that $g^0(a,a) = \xi^0_{g(\_,\_)}(a,a)$. Assume that $\xi^0_{g(\_,\_)}(a,a) = d_1$ with $d_1 \notin \mathrm{dom}(I)$. Then, as in the previous case, we need to compute $\kappa(d_1)$ in order to compute $f^0(g^0(a,a))$. Assume for this example that $\kappa(d_1) = (f(\_), d_2)$, where $d_2 \in \mathrm{dom}(I)$. That is, assume that $\xi_{f(\_)}(d_2) = d_1$ and (f, d_2) is the smallest element $(h, \bar{d})$ in $\mathcal{FT}(I)$, according to the linear order $<$, satisfying the condition $\xi^0_{h(\_,\ldots,\_)}(\bar{d}) = d_1$. Thus, by the preceding definition, we have that:

$$f^0(g^0(a,a)) \;=\; f^0(d_1) \;=\; \xi^0_{f(f(\_))}(d_2). \tag{7.11}$$

Therefore, from the previous discussion and the fact that that $I \models f(a) = g(f(a), f(b))$, we conclude that:

$$\begin{aligned} I \;\models\; & S(a,b) \wedge \mathrm{dom}(a_1) \wedge \mathrm{dom}(c_1) \wedge \mathrm{dom}(c_2) \wedge \mathrm{dom}(d_2) \wedge \xi_{f(\_)}(a) = a_1 \;\wedge \\ & \xi_{f(\_)}(b) = \xi_{g(\_,\_)}(c_1, c_2) \wedge \xi_{g(\_,\_)}(a,a) = \xi_{f(\_)}(d_2) \wedge \xi_{f(\_)}(a) = \xi_{g(\_,g(\_,\_))}(a_1, c_1, c_2). \end{aligned}$$

Hence, given that we assume that $(I, J) \models \sigma^\star$ and the following dependency is one of the formulas θ_{s_1,s_2,s_3} (see Example 7.8):

$$\begin{aligned} \forall x \forall y \forall u_1 \forall u_7 \forall u_8 \forall u_{10} \Big[\Big(& S(x,y) \wedge \mathrm{dom}(u_1) \wedge \mathrm{dom}(u_7) \wedge \mathrm{dom}(u_8) \wedge \mathrm{dom}(u_{10}) \;\wedge \\ & \xi_{f(\_)}(x) = u_1 \wedge \xi_{f(\_)}(y) = \xi_{g(\_,\_)}(u_7, u_8) \wedge \xi_{g(\_,\_)}(x,x) = \xi_{f(\_)}(u_{10}) \;\wedge \\ & \xi_{f(\_)}(x) = \xi_{g(\_,g(\_,\_))}(u_1, u_7, u_8) \Big) \;\rightarrow\; T(\xi_{f(f(\_))}(u_{10})) \Big], \end{aligned}$$

we conclude that $J \models T(\xi_{f(f(\_))}(d_2))$. But we know from (7.11) that $f^0(g^0(a,a)) = \xi^0_{f(f(\_))}(d_2)$ and, thus, we have that $J \models T(f(g(a,a)))$, which was to be shown. □

In general, we have to show that if $\forall \bar{x}\,(\varphi(\bar{x}) \rightarrow \psi(\bar{x}))$ is a conjunct of σ and $I \models \varphi(\bar{a})$ with the instantiations f^0_1, …, f^0_ℓ of the function symbols f_1, …, f_ℓ, then $J \models \psi(\bar{a})$ with these instantiations. It is straightforward but lengthy to generalize the strategy shown in the previous example to this case. In particular, given that in the construction of $\sigma^\star$ we consider all the possible cases for substitution κ, the previous strategy can be applied in general. This concludes the proof of the lemma. □

Proof of Theorem 7.2. The theorem is proved by induction on the nesting depth n of an SO tgd. If $n = 1$, then the property trivially holds, and if $n = 2$, then the property holds by Lemma 7.3. Thus, let σ be an SO tgd from a source schema $\mathbf{S}$ to a target schema $\mathbf{T}$, and assume that the nesting depth of σ is $n \geq 3$. Moreover, assume that the theorem holds for every SO tgd σ' of nesting depth $n' < n$.

By Theorem 8.4 in [23], we know that there exist schema mappings $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, …, $\mathcal{M}_{n+1}$ such that σ specifies $\mathcal{M}_1 \circ \mathcal{M}_2 \circ \mathcal{M}_3 \circ \cdots \circ \mathcal{M}_{n+1}$ and every mapping $\mathcal{M}_i$ $(1 \leq i \leq n+1)$ is specified by a set of st-tgds. For every $i \in \{1, \ldots, n+1\}$, let σ_i be an unnested SO tgd that specifies $\mathcal{M}_i$. We know, by the definition of the algorithm Compose in [23], that there exists an SO tgd σ_{12} that specifies the composition of the schema mappings specified by σ_1

and σ_2 and whose nesting depth is at most 2. By Lemma 7.3, we have that there exists an unnested SO tgd $\sigma_{12}^{\star}$ that is equivalent to σ_{12}. Thus, by considering again the definition of the algorithm Compose in [23], we have that there exists an SO tgd σ_{13} such that σ_{13} specifies the composition of the schema mappings specified by $\sigma_{12}^{\star}$ and σ_3 and whose nesting depth is at most 2. Hence, by considering again Lemma 7.3, we conclude that there exists an unnested SO tgd $\sigma_{13}^{\star}$ that specifies the composition of the schema mappings specified by $\sigma_{12}^{\star}$ and σ_3 and, thus, also specifies $\mathcal{M}_1 \circ \mathcal{M}_2 \circ \mathcal{M}_3$. Finally, by considering again the definition of algorithm Compose in [23], we have that there exists an SO tgd σ' such that: (1) σ' specifies the composition of the mappings specified by $\sigma_{13}^{\star}$, σ_4, ..., σ_{n+1}; and (2) the depth of nesting of σ' is at most $n-1$ (since $\sigma_{13}^{\star}$, σ_4, ..., σ_{n+1} are all unnested SO tgds). Therefore, we conclude that there exists an SO tgd σ' that is equivalent to σ and whose depth of nesting is at most $n-1$. But then by induction hypothesis, there exists an unnested SO tgd $\sigma^{\star}$ that is equivalent to σ' and, hence, there exists an unnested SO tgd $\sigma^{\star}$ that is equivalent to σ. This concludes the proof of the theorem. □

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 result 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. Marcelo Arenas was partially supported by FONDECYT grant 1090565.

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 *23rd ACM Symposium on Principles of Database Systems (PODS)*, pages 229–240, 2004.

[6] M. Arenas, R. Fagin, and A. Nash. Composition with Target Constraints. In *13th International Conference on Database Theory (ICDT)*, pages 129–142, 2010.

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

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

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

[10] 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.

[11] 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.

[12] R. Chirkova and M. Genesereth. Equivalence of SQL Queries in Presence of Embedded Dependencies. In *28th ACM Symposium on Principles of Database Systems (PODS)*, pages 217–226, 2009.

[13] 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.

[14] A. Deutsch, A. Nash, and J. Remmel. The Chase Revisited. In *27th ACM Symposium on Principles of Database Systems (PODS)*, pages 149–158, 2008.

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

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

[17] O. M. Duschka, M. R. Genesereth, and A. Y. Levy. Recursive Query Plans for Data Integration. *J. Log. Program.*, 43(1):49–73, 2000.

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

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

[20] 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.

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

[22] 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 *International Conference on Database Theory (ICDT)*, pages 207–224, 2003.

[23] 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.

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

[25] 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.

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

[27] H. Gaifman. On Local and Non-local Properties. In *Herbrand Symposium Logic Colloquium, North Holland*, pages 105–135, 1982.
[28] G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. On Reconciling Data Exchange, Data Integration, and Peer Data Management. In *26th ACM Symposium on Principles of Database Systems (PODS)*, pages 133–142, 2007.
[29] G. Gottlob and A. Nash. Efficient Core Computation in Data Exchange. *Journal of the ACM*, 55(2), 2008.
[30] G. Gottlob and S. Szeider. Fixed-Parameter Algorithms For Artificial Intelligence, Constraint Satisfaction and Database Problems. *Computer Journal*, 51(3):303–325, 2008.
[31] 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.
[32] W. P. Hanf. Model-theoretic Methods in the Study of Elementary Logic. In *The Theory of Models; Addison, Henkin, and Tarski, eds., North Holland*, pages 132–145, 1965.
[33] A. Hernich and N. Schweikardt. CWA-solutions for Data Exchange Settings with Target Dependencies. In *26th ACM Symposium on Principles of Database Systems (PODS)*, pages 113–122, 2007.
[34] G. Karvounaraki and V. Tannen. Conjunctive Queries and Mappings with Unequalities. Technical Report MS-CIS-08-37, University of Pennsylvania, 2008.
[35] P. G. Kolaitis. Schema Mappings, Data Exchange, and Metadata Management. In *24th ACM Symposium on Principles of Database Systems (PODS)*, pages 61–75, 2005.
[36] M. Lenzerini. Data Integration: A Theoretical Perspective. In *21st ACM Symposium on Principles of Database Systems (PODS)*, pages 233–246, 2002.
[37] L. Libkin. *Elements of Finite Model Theory*. Springer-Verlag, 1st edition, 2004.
[38] L. Libkin and C. Sirangelo. Data Exchange and Schema Mappings in Open and Closed Worlds. In *27th ACM Symposium on Principles of Database Systems (PODS)*, pages 139–148, 2008.
[39] J. Madhavan and A. Y. Halevy. Composing Mappings Among Data Sources. In *International Conference on Very Large Data Bases (VLDB)*, pages 572–583, 2003.
[40] 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.
[41] S. Melnik. *Generic Model Management: Concepts and Algorithms*, volume 2967 of *Lecture Notes in Computer Science*. Springer, 2004.
[42] A. Nash, P. A. Bernstein, and S. Melnik. Composition of Mappings Given by Embedded Dependencies. In *24th ACM Symposium on Principles of Database Systems (PODS)*, pages 172–183, 2005.
[43] P. Papotti and R. Torlone. Schema Exchange: Generic Mappings for Transformaing Data and Metadata. *Data and Knowledge Engineering*, 68(7):665–682, 2009.
[44] B. ten Cate and P. Kolaitis. Structural Characterizations of Schema-Mapping Languages. In *International Conference on Database Theory (ICDT)*, pages 63–72, 2009.
[45] 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.