Composing Schema Mappings: Second-Order Dependencies to the Rescue

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

Phokion G. Kolaitis*
UC Santa Cruz
kolaitis@cs.ucsc.edu

Lucian Popa
IBM Almaden Research Center
lucian@almaden.ibm.com

Wang-Chiew Tan
UC Santa Cruz
wctan@cs.ucsc.edu

ABSTRACT

A schema mapping is a specification that describes how data structured under one schema (the source schema) is to be transformed into data structured under a different schema (the target schema). Schema mappings play a key role in numerous areas of database systems, including database design, information integration, and model management. A fundamental problem in this context is composing schema mappings: given two successive schema mappings, derive a schema mapping between the source schema of the first and the target schema of the second that has the same effect as applying successively the two schema mappings.

In this paper, we give a rigorous semantics to the composition of schema mappings and investigate the definability and computational complexity of the composition of two schema mappings. We first study the important case of schema mappings in which the specification is given by a finite set of source-to-target tuple-generating dependencies (source-to-target tgds). We show that the composition of a finite set of full source-to-target tgds with a finite set of tgds is always definable by a finite set of source-to-target tgds, but the composition of a finite set of source-to-target tgds with a finite set of full source-to-target tgds may not be definable by any set (finite or infinite) of source-to-target tgds; furthermore, it may not be definable by any formula of least fixed-point logic, and the associated composition query may be NP-complete. After this, we introduce a class of existential second-order formulas with function symbols, which we call second-order tgds, and make a case that they are the "right" language for composing schema mappings. To this effect, we show that the composition of finite sets of source-to-target tgds is always definable by a second-order tgd. Moreover, the composition of second-order tgds is also definable by a second-order tgd. Our second-order tgds allow equalities, even though the "obvious" way to define them does not require equalities. Allowing equalities in second-order tgds turns out to be of the essence, because we show that second-order tgds without equalities are not sufficiently expressive to define even the composition of finite sets of source-to-target tgds. Finally, we show that second-order tgds possess good properties for data exchange. In particular, the chase procedure can be extended to second-order tgds so that it produces polynomial-time computable universal solutions in data exchange settings specified by second-order tgds.

*Research partially supported by NSF Grant No. IIS-9907419

1. Introduction & Summary of Results

The problem of transforming data structured under one schema into data structured under a different schema is an old, but persistent, problem arising in several different areas of database management systems. In recent years, this problem has received considerable attention in the context of information integration, where data from various heterogeneous sources has to be transformed into data structured under a mediated schema. To achieve interoperability, data-sharing architectures use *schema mappings* to describe how data is to be transformed from one representation to another. These schema mappings are typically specified using high-level declarative formalisms that make it possible to describe the correspondence between different schemas at a logical level, without having to specify physical details that may be relevant only for the implementation (run-time) phase. In particular, declarative schema mappings in the form of GAV (global-as-view), LAV (local-as-view), and, more generally, GLAV (global-and-local-as-view) assertions have been used in *data integration* systems [15]. Similarly, source-to-target tuple-generating dependencies (source-to-target tgds) have been used for specifying data exchange between a relational source and a relational target [10, 11]; moreover, nested (XML-style) source-to-target dependencies have been used in the Clio data exchange system [19].

The extensive use of schema mappings has motivated the need to develop a framework for managing these schema mappings and other related meta-data. Bernstein [5] has introduced such a framework, called *model management*, in which the main abstractions are schemas and mappings between schemas, as well as operators for manipulating schemas and mappings. One of the most fundamental operators in this framework 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 peer-to-peer data management systems, such as the Piazza System [13], and in extract-transform-load (ETL) processes in which the output of a transformation may be input to another [20]. A model management system should be able to figure out automatically how to compose two or more successive schema mappings into a single schema mapping between the first schema and the last schema in a way that captures the interaction of the schema mappings in the entire sequence. The resulting single schema mapping can then be used during the run-time phase for various purposes, such as query answering and data exchange, potentially with significant performance benefits.

Bernstein's approach provides a rich conceptual framework for model management. The next stage in the development of this framework is to provide a rigorous and meaningful semantics of the basic model management operators and to investigate the properties of this semantics. As pointed out by Bernstein [5], while the semantics of the match operator have been worked out to a certain extent, the semantics of other basic operators, including the composition operator, "are less well developed". The problem of composing schema mappings has the following general formulation: given a schema mapping $\mathcal{M}_{12}$ from schema $\mathbf{S}_1$ to schema $\mathbf{S}_2$, and a schema mapping $\mathcal{M}_{23}$ from schema $\mathbf{S}_2$ to schema $\mathbf{S}_3$, derive a schema mapping $\mathcal{M}_{13}$ from schema $\mathbf{S}_1$ to schema $\mathbf{S}_3$ that is "equivalent" to the successive application of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$. Thus, providing semantics to the composition operator amounts to making precise what "equivalence" means in this context. Madhavan and Halevy [17] were the first to propose a semantics of the composition operator. To this effect, they defined the semantics of the composition operator relative to a class $\mathcal{Q}$ of queries over the schema $\mathbf{S}_3$ by stipulating that "equivalence" means that, for every query q in $\mathcal{Q}$, the certain answers of q in $\mathcal{M}_{13}$ coincide with the certain answers of q that would be obtained by successively applying the two schema mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$. They then established a number of results for this semantics in the case in which the schema mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ are specified by source-to-target tgds (that is, GLAV assertions), and the class $\mathcal{Q}$ is the class of all conjunctive queries over $\mathbf{S}_3$. The semantics of the composition operator proposed by Madhavan and Halevy [17] is a significant first step, but it suffers from certain drawbacks that seem to be caused by the fact that this semantics is given relative to a class of queries. To begin with, the set of formulas specifying a composition $\mathcal{M}_{13}$ of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ relative to a class $\mathcal{Q}$ of queries need not be unique up to logical equivalence. Moreover, this semantics is rather fragile, because a schema mapping $\mathcal{M}_{13}$ may be a composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ with respect to a class $\mathcal{Q}$ of queries, but fail to be a composition of these two schema mappings with respect to a slightly larger class $\mathcal{Q}'$ of queries.

In this paper, we first introduce a different semantics for the composition operator and then investigate the definability and computational complexity of the composition of schema mappings under this new semantics. Unlike the semantics proposed by Madhavan and Halevy [17], our semantics does not carry along a class of queries as a parameter. Specifically, we focus on the space of instances of schema mappings and define a schema mapping $\mathcal{M}_{13}$ to be a composition of two schema mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ if the space of instances of $\mathcal{M}_{13}$ is the set-theoretic composition of the spaces of instances of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, where these spaces are viewed as binary relations between source instances and target instances. One advantage of this approach is that the set of formulas defining a composition $\mathcal{M}_{13}$ of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is unique up to logical equivalence; thus, we can refer to such a schema mapping $\mathcal{M}_{13}$ as *the* composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$. Moreover, our semantics is robust, since it is defined in terms of the schema mappings alone and without a reference to a set of queries. In fact, it is easy to see that the composition (in our sense) of two schema mappings is a composition of these two schema mappings in the sense of Madhavan and Halevy relative to *every* class of queries.

We explore in depth the properties of the composition of schema mappings specified by a finite set of source-to-target tuple-generating dependencies (source-to-target tgds). A natural question to ask is whether the composition of two such schema mappings can also be specified by a finite set of source-to-target tgds; if not, in what logical formalism can it be actually expressed? On the positive side, we show that the composition of a finite set of full source-to-target tgds with a finite set of source-to-target tgds is always definable by a finite set of source-to-target tgds (a source-to-target tgd is *full* if no existentially quantified variables occur in the tgd). On the negative side, however, we show that the composition of a finite set of source-to-target tgds with a finite set of full source-to-target tgds may not be definable by any set (finite or infinite) of source-to-target tgds. It should be noted that Madhavan and Halevy [17] showed that under their semantics the composition of finite sets of source-to-target tgds may not be specifiable by a finite set of source-to-target tgds, but it does not appear that their paper considers the possibility that even an infinite set of tgds will not suffice. We also show that the composition of a finite set of source-to-target tgds with a finite set of full source-to-target tgds may not even be definable in the finite-variable infinitary logic $L^{\omega}_{\infty\omega}$, which implies that it is not definable in least fixed-point logic LFP; moreover, the associated composition query may be NP-complete.

To ameliorate these negative results, we introduce a class of existential second-order formulas with function symbols, called *second-order tgds*, which express source-to-target constraints and subsume the class of (first-order) source-to-target tgds. We make a case that second-order tgds are the *right* language both for specifying schema mappings and for composing such schema mappings. To this effect, we show that the composition of finite sets of source-to-target tgds is always definable by a second-order tgd. Moreover, the composition of second-order tgds is also definable by a second-order tgd, and we give an algorithm that, given two schema mappings specified by second-order tgds, outputs a second-order tgd that defines the composition. We also show that second-order tgds possess good properties for data exchange. In particular, the chase procedure can be extended to second-order tgds so that it produces polynomial-time computable universal solutions in data exchange settings specified by second-order tgds. As a result, in such data exchange settings the certain answers of conjunctive queries can be computed in polynomial time.

It should be pointed out that arriving at the *right* concept of second-order tgds is a rather delicate matter. Indeed, at first one may consider the class of second-order formulas that are obtained from first-order tgds by Skolemizing the existential first-order quantifiers into existentially quantified function symbols. This process gives rise to a class of existential second-order formulas with no equalities. Therefore, the "obvious" way to define second-order tgds is with formulas with no equalities. Interestingly enough, however, we show that second-order tgds without equalities are *not* sufficiently expressive to define the composition query of finite sets of (first-order) source-to-target tgds.

Our second-order tgds are not too strong a fragment of second-order logic, and also not too weak a fragment. If they were weaker (by not having equalities), then, as we just noted, they would not be sufficiently expressive to capture the composition query of finite sets of first-order tgds. At the same time, they are not so strong that we cannot carry out the chase process efficiently. To appreciate this point, let us consider the computation of the certain answers of conjunctive queries. As we noted earlier, if the data exchange setting is defined by second-order tgds, then the certain answers of every conjunctive query can be computed in polynomial time (by doing the chase). By contrast, when the source schema is described in terms of the target schema by means of arbitrary first-order views, there are conjunctive queries for which computing the certain answers is an undecidable problem [1]. Thus, our second-order tgds form a fragment of second-order logic that in some ways is more well-behaved than first-order logic.

2. The Semantics of Composition

In this section, we define basic concepts and the meaning of composing two schema mappings.

A *schema* is a finite sequence $\mathbf{R} = \langle R_1, \dots, R_k \rangle$ of relation symbols, each of a fixed arity. An *instance* I *(over the schema* $\mathbf{R}$) is a sequence $\langle R_1^I, \dots, R_k^I \rangle$ such that each R_i^I is a finite relation of the same arity as R_i. We call R_i^I the R_i*-relation of* I. We shall often abuse the notation and use R_i to denote both the relation symbol and the relation R_i^I that interprets it.

Let $\mathbf{S} = \langle S_1, \dots, S_n \rangle$ and $\mathbf{T} = \langle T_1, \dots, T_m \rangle$ be two schemas with no relation symbols in common. We write $\langle \mathbf{S}, \mathbf{T} \rangle$ to denote the schema $\langle S_1, \dots, S_n, T_1, \dots, T_m \rangle$. If I is an instance over $\mathbf{S}$ and J is an instance over $\mathbf{T}$, then we write $\langle I, J \rangle$ for the instance K over the schema $\langle \mathbf{S}, \mathbf{T} \rangle$ such that $S_i^K = S_i^I$ and $T_j^K = T_j^J$, for $1 \le i \le n$ and $1 \le j \le m$.

Definition 2.1. A *schema mapping* (or, in short, *mapping*) is a triple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, where $\mathbf{S}$ and $\mathbf{T}$ are schemas with no relation symbols in common and Σ_{st} is a set of formulas of some logical formalism over $\langle \mathbf{S}, \mathbf{T} \rangle$.

If $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ is a schema mapping, then an *instance of* $\mathcal{M}$ is an instance $\langle I, J \rangle$ over $\langle \mathbf{S}, \mathbf{T} \rangle$ that satisfies every formula in the set Σ_{st}.

We write Inst($\mathcal{M}$) to denote the set of all instances $\langle I, J \rangle$ of $\mathcal{M}$. Moreover, if $\langle I, J \rangle \in \text{Inst}(\mathcal{M})$, then we say that J is a *solution of* I *under* $\mathcal{M}$. $\square$

Several remarks are in order now. In the sequel, if $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ is a schema mapping, we will often refer to $\mathbf{S}$ as the *source schema* and to $\mathbf{T}$ as the *target schema*. The formulas in the set Σ_{st} express *constraints* that an instance $\langle I, J \rangle$ over the schema $\langle \mathbf{S}, \mathbf{T} \rangle$ must satisfy. We assume that the logical formalisms considered have the property that the satisfaction relation between formulas and instances is *preserved under isomorphisms*, which means that if an instance satisfies a formula, then every isomorphic instance also satisfies that formula. This is a mild condition that is true of all standard logical formalisms, such as first-order logic, second-order logic, fixed-point logics, and infinitary logics. An immediate consequence of this property is that Inst($\mathcal{M}$) is *closed under isomorphisms*; that is, if $\langle I, J \rangle \in \text{Inst}(\mathcal{M})$ and $\langle I', J' \rangle$ is isomorphic to $\langle I, J \rangle$, then also $\langle I', J' \rangle \in \text{Inst}(\mathcal{M})$.

At this level of generality, some of the formulas in Σ_{st} may be just over the source schema $\mathbf{S}$ and others may be just over the target schema $\mathbf{T}$; thus, the set Σ_{st} may include constraints over the source schema $\mathbf{S}$ alone or over the target schema $\mathbf{T}$ alone. We note that, although the term "schema mapping" or "mapping" has been used earlier in the literature (for instance, in [17, 18]), it is a bit of a misnomer, as a schema mapping is not a mapping in the traditional mathematical sense, but actually it is a schema (although partitioned in two parts) together with a set of constraints. Nonetheless, a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ gives rise to a mapping such that, given an instance I over $\mathbf{S}$, it associates the set J of all instances over $\mathbf{T}$ that are solutions for I under $\mathcal{M}$. Note also that the terminology "J is a solution for I" comes from [10, 11], where J is a solution to the *data exchange problem* associated with the mapping $\mathcal{M}$ and the source instance I.

If P_1 and P_2 are two binary relations, then, by definition, the *composition* $P_1 \circ P_2$ of P_1 and P_2 is the binary relation

$$P_1 \circ P_2 = \{(x, y) : (\exists z)((x, z) \in P_1 \wedge (z, y) \in P_2)\}.$$

Clearly, if $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ is a schema mapping, then Inst($\mathcal{M}$) is a binary relation between instances over $\mathbf{S}$ and instances over $\mathbf{T}$. In what follows, we define the concept of a *composition of two schema mappings* $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ using the composition of the binary relations Inst($\mathcal{M}_{12}$) and Inst($\mathcal{M}_{23}$).

Definition 2.2. Let $\mathcal{M}_{12}$=($\mathbf{S}_1$, $\mathbf{S}_2$, Σ_{12}) and $\mathcal{M}_{23}$=($\mathbf{S}_2$, $\mathbf{S}_3$, Σ_{23}) be two schema mappings such that the schemas $\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3$ have no relation symbol in common pairwise. A schema mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_3, \Sigma_{13})$ is a *composition of* $\mathcal{M}_{12}$ *and* $\mathcal{M}_{23}$ if

$$\text{Inst}(\mathcal{M}) = \text{Inst}(\mathcal{M}_{12}) \circ \text{Inst}(\mathcal{M}_{23}),$$

which means that Inst($\mathcal{M}$) = $\{\langle I_1, I_3 \rangle \mid$ there exists I_2 such that $\langle I_1, I_2 \rangle \in \text{Inst}(\mathcal{M}_{12})$ and $\langle I_2, I_3 \rangle \in \text{Inst}(\mathcal{M}_{23})\}$.

Since Inst($\mathcal{M}_{12}$) and Inst($\mathcal{M}_{23}$) are closed under isomorphisms, their composition Inst($\mathcal{M}_{12}$)$\circ$Inst($\mathcal{M}_{23}$) is also closed under isomorphisms. Consequently, the class $\mathrm{Inst}(\mathcal{M}_{12}) \circ \mathrm{Inst}(\mathcal{M}_{23})$ can be identified with the following query[1], which we call the *composition query of* $\mathcal{M}_{12}$ *and* $\mathcal{M}_{23}$: given two instances I_1 and I_3, is $\langle I_1, I_3\rangle \in \mathrm{Inst}(\mathcal{M}_{12}) \circ \mathrm{Inst}(\mathcal{M}_{23})$? Note that, according to Definition 2.2, asserting that $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_3, \Sigma_{13})$ is a *composition of* $\mathcal{M}_{12}$ *and* $\mathcal{M}_{23}$ amounts to saying that the composition query of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is exactly the set of instances over $\langle \mathbf{S}_1, \mathbf{S}_3\rangle$ that satisfy Σ_{13}. In other words, this means that the composition query of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is definable by the formulas in the set Σ_{13}.

It is well known and easy to see that every query is definable by an infinitary disjunction of first-order sentences. Specifically, for each finite structure satisfying the query, we construct a first-order formula that defines the structure up to isomorphism and then take the disjunction of all these formulas. This infinitary sentence defines the query. Moreover, every query is definable by a set of first-order formulas. Indeed, for each finite structure that does not satisfy the query, we construct the negation of the first-order formula that defines the structure up to isomorphism and then form the set of all such formulas. Note that this is an infinite set of first-order formulas, unless the query is satisfied by all but finitely many non-isomorphic instances. It follows that a composition of two schema mappings always exists, since, given two schema mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, we can obtain a composition $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_3, \Sigma_{13})$ of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ by taking Σ_{13} to be the singleton consisting of the infinitary formula that defines the composition query of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$. Alternatively, we could take Σ_{13} to be the (usually infinite) set of first-order formulas that defines the composition query of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$. Furthermore, this composition is unique up to logical equivalence in the sense that if $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_3, \Sigma)$ and $\mathcal{M}' = (\mathbf{S}_1, \mathbf{S}_3, \Sigma')$ are both compositions of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, then Σ and Σ' are logically equivalent. For this reason, from now on we will refer to *the* composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, and will denote it by $\mathcal{M}_{12} \circ \mathcal{M}_{23}$.

Since the composition query is always definable both by an infinitary disjunction of first-order formulas and by an infinite set of first-order formulas, it is natural to investigate when the composition of two schema mappings is definable in less expressive, but more tractable, logical formalisms. It is also natural to investigate whether the composition of two schema mappings is definable in the same logical formalism that is used to define these two schema mappings. We embark on this investigation in the next section.

3. Composing Source-to-Target TGDs

Let $\mathbf{S}$ be a source schema and $\mathbf{T}$ a target schema. A *source-to-target tuple-generating dependency (source-to-target tgd)* is a first-order formula of the form

$$\forall \mathbf{x}(\phi_S(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi_T(\mathbf{x}, \mathbf{y})),$$

where $\phi_S(\mathbf{x})$ is a conjunction of atomic formulas over $\mathbf{S}$ and $\psi_T(\mathbf{x}, \mathbf{y})$ is a conjunction of atomic formulas over $\mathbf{T}$. We assume that every variable in $\mathbf{x}$ appears in ϕ_S. A *full source-to-target tuple-generating dependency (full source-to-target tgd)* is a source-to-target tgd of the form

$$\forall \mathbf{x}(\phi_S(\mathbf{x}) \rightarrow \psi_T(\mathbf{x})),$$

where $\phi_S(\mathbf{x})$ is a conjunction of atomic formulas over $\mathbf{S}$ and $\psi_T(\mathbf{x})$ is a conjunction of atomic formulas over $\mathbf{T}$. We again assume that every variable in $\mathbf{x}$ appears in ϕ_S. Source-to-target tgds have been used to formalize data exchange [10, 11]. Moreover, they have been used in data integration scenarios under the name of GLAV assertions [15].

Note that every full source-to-target tgd is logically equivalent to a finite set of full source-to-target tgds each of which has a single atom in its right-hand side. Specifically, a full source-to-target tgd of the form $\forall \mathbf{x}(\phi_S(\mathbf{x}) \rightarrow \wedge_{i=1}^{k} R_i(\mathbf{x}_i))$ is equivalent to the set consisting of the full source-to-target tgds $\forall \mathbf{x}(\phi_S(\mathbf{x}) \rightarrow R_i(\mathbf{x}_i))$, for $i = 1, \ldots, k$. In contrast, this property fails for arbitrary source-to-target tgds, since the existential quantifiers may bind variables used across different atomic formulas.

Example 3.1. Consider the following three schemas $\mathbf{S}_1$, $\mathbf{S}_2$ and $\mathbf{S}_3$. Schema $\mathbf{S}_1$ consists of a single binary relation symbol `Takes`, that associates student names with the courses they take. Schema $\mathbf{S}_2$ consists of a similar binary relation symbol $\texttt{Takes}_1$ and of an additional binary relation symbol `Student` that associates each student name with a student id. Schema $\mathbf{S}_3$ consists of one binary relation symbol `Enrollment` that associates student ids with the courses the students take. Consider now the schema mappings $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$, where

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

These three formulas are source-to-target tgds. The second formula in Σ_{12} is an example of a source-to-target tgd that is not full, while the other two formulas are full source-to-target tgds. The first mapping, associated with the set Σ_{12} of formulas, requires that "copies" of the tuples in `Takes` must exist in $\texttt{Takes}_1$ and, moreover, that each student name must be associated with some student id (s) in `Student`. The second mapping, associated with the formula in Σ_{23}, requires that pairs of student id and course must exist in the relation `Enrollment`, provided that they are associated with the same student name.

Define I_1 by letting $\texttt{Takes}^{I_1} = \{(\mathrm{Alice}, \mathrm{Math}), (\mathrm{Alice}, \mathrm{Art})\}$. Define I_2 by letting $\texttt{Takes}_1^{I_2} = \texttt{Takes}^{I_1}$ and $\texttt{Student}^{I_2} = \{(\mathrm{Alice},\ 1234)\}$. Here 1234 is Alice's student id. Define I_3 by letting $\texttt{Enrollment}^{I_3} = \{(1234, \mathrm{Math}), (1234, \mathrm{Art})\}$. It is easy to verify that $\langle I_1, I_2\rangle \in \mathrm{Inst}(\mathcal{M}_{12})$ and that $\langle I_2, I_3\rangle \in \mathrm{Inst}(\mathcal{M}_{23})$. Hence, $\langle I_1, I_3\rangle \in \mathrm{Inst}(\mathcal{M}_{12}) \circ \mathrm{Inst}(\mathcal{M}_{23})$. One of the main problems that we study in this paper is how to find, and in what language, a schema mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_3, \Sigma_{13})$ that is a composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, according to Definition 2.2. In other words, we will be looking for Σ_{13} (involving only $\mathbf{S}_1$ and $\mathbf{S}_3$) such that an instance $\langle I_1, I_3\rangle$ is in $\mathrm{Inst}(\mathcal{M}_{12}) \circ \mathrm{Inst}(\mathcal{M}_{23})$ if and only if $\langle I_1, I_3\rangle$ satisfies Σ_{13}. $\square$

In this section, we will investigate the definability and com-

[1]Recall that a *query* is a class of finite structures that is closed under isomorphisms [6].

putational complexity of the composition of two schema mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ in which the dependencies Σ_{12} and Σ_{23} are finite sets of source-to-target tgds. We will first show that if Σ_{12} and Σ_{23} are finite sets of full source-to-target tgds, then the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is also definable by a finite set of full source-to-target tgds. Furthermore, if Σ_{12} is a finite set of full source-to-target tgds and Σ_{23} is a finite set of source-to-target tgds (not necessarily full), then the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is definable by a finite set of source-to-target tgds. In turn, this implies that the associated composition query is polynomial-time computable. In contrast, if both Σ_{12} and Σ_{23} are finite sets of arbitrary source-to-target tgds (not necessarily full), then the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ may not even be first-order definable, and the associated composition query may be NP-complete.

3.1 Positive Results

Our first positive result shows the good behavior of the composition of mappings, each of which is defined by finite sets of full source-to-target tgds.

Proposition 3.2. *Let $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$ be two schema mappings such that Σ_{12} and Σ_{23} are finite sets of full source-to-target tgds. Then the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is definable by a finite set of full source-to-target tgds. Consequently the composition query of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is a polynomial-time query.*

Proof. *(Sketch)* Without loss of generality, we may assume that each full source-to-target tgd in Σ_{12} has a single atom in its right-hand side. The composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is the schema mapping $(\mathbf{S}_1, \mathbf{S}_3, \Sigma_{13})$, where Σ_{13} is constructed as follows. For every full source-to-target tgd in Σ_{23} of the form $\forall \mathbf{x}((R_1(\mathbf{x_1}) \wedge ... \wedge R_k(\mathbf{x_k})) \rightarrow S(\mathbf{x_0}))$, and for every full source-to-target tgd in Σ_{12} of the form $\forall \mathbf{z}(\phi(\mathbf{z}) \rightarrow R_i(\mathbf{z_i}))$, replace each atom $R_i(\mathbf{x}_i)$ in the first tgd by the formula $\phi(\mathbf{z}_i/\mathbf{x}_i)$, where $\phi(\mathbf{z}_i/\mathbf{x}_i)$ is the result of letting $\mathbf{x}_i$ play the role of $\mathbf{z}_i$ in ϕ. After all possible such replacements have been performed, this process associates with each full source-to-target tgd in Σ_{23} a finite set of full source-to-target tgds from $\mathbf{S}_1$ to $\mathbf{S}_3$. Then Σ_{13} is the union of all these sets, and it is a finite set, since Σ_{23} is a finite set. It is clear that if $\langle I_1, I_3\rangle$ is in $\mathrm{Inst}(\mathcal{M}_{12}) \circ \mathrm{Inst}(\mathcal{M}_{23})$, then $\langle I_1, I_3\rangle$ satisfies Σ_{13}. As for the converse, it is not hard to verify that if $\langle I_1, I_3\rangle$ satisfies Σ_{13}, and if I_2 is the result of chasing I_1 with Σ_{12}, then $\langle I_1, I_2\rangle \in \mathrm{Inst}(\mathcal{M}_{12})$ and $\langle I_2, I_3\rangle \in \mathrm{Inst}(\mathcal{M}_{23})$, and so $\langle I_1, I_3\rangle \in \mathrm{Inst}(\mathcal{M}_{12}) \circ \mathrm{Inst}(\mathcal{M}_{23})$. $\square$

A special case of this proposition appeared in [4, Lemma 2.3]. An inspection of the proof of Proposition 3.2 shows that the same construction yields the following result.

Proposition 3.3. *Let $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$ be two schema mappings such that Σ_{12} is a finite set of full source-to-target tgds and Σ_{23} is a finite set of source-to-target tgds. Then the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is definable by a finite set of source-to-target tgds. Consequently, the composition query of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is a polynomial-time query.*

3.2 Negative Results

We now present a series of negative results associated with the composition of schema mappings specified by source-to-target tgds.

Proposition 3.4. *There exist schema mappings $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$ such that Σ_{12} is a finite set of source-to-target tgds, Σ_{23} is a finite set of full source-to-target tgds, and the following hold for the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$:*

1. *$\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is not definable by any finite set of source-to-target tgds, but it is definable by an infinite set of source-to-target tgds.*
2. *$\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is definable by a first-order formula. Consequently, the composition query of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is a polynomial-time query.*

Proof. *(Sketch)* The two schema mappings that we use to prove the proposition are the schema mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ of Example 3.1. Consider, for each $k \geq 1$, the following source-to-target tgd:

$$\begin{aligned}\phi_k :\quad & \forall n \forall c_1 \ldots \forall c_k (\texttt{Takes}(n, c_1) \wedge \ldots \wedge \texttt{Takes}(n, c_k) \rightarrow \\ & \exists s(\texttt{Enrollment}(s, c_1) \wedge \ldots \wedge \texttt{Enrollment}(s, c_k)))\end{aligned}$$

It is easy to verify that the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is given by the schema mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_3, \Sigma_{13})$, where Σ_{13} is the infinite set $\{\phi_1, \ldots, \phi_k, \ldots\}$ of source-to-target tgds. It is also easy to see that Σ_{13} is not equivalent to any finite subset of it. In fact, by using techniques from [10], we can show that Σ_{13} is not equivalent to any finite set of source-to-target tgds. The details are omitted.

Finally, it is easy to verify that the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ of the two schema mappings is definable by the first-order formula $\forall n \exists s \forall c(\texttt{Takes}(n, c) \rightarrow \texttt{Enrollment}(s, c))$. $\square$

We have just given an example in which the composition is definable by an infinite set of source-to-target tgds, but it is not definable by any finite set of source-to-target tgds. There is also a different example in which the composition is not definable even by an infinite set of source-to-target tgds. This is stated in the next result, which amplifies the limitations of the language of source-to-target tgds with respect to composition. A proof appears in Section 4, after we develop the necessary machinery.

Proposition 3.5. *There exist schema mappings $\mathcal{M}_{12}$=($\mathbf{S}_1$, $\mathbf{S}_2$, Σ_{12}) and $\mathcal{M}_{23}$=($\mathbf{S}_2$, $\mathbf{S}_3$, Σ_{23}) such that Σ_{12} consists of a single source-to-target tgd, Σ_{23} is a finite set of full source-to-target tgds, and the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ cannot be defined by any finite or infinite set of source-to-target tgds.*

In the example given in Proposition 3.4, the composition query is polynomial-time computable, since it is first-order. In what follows, we will show that there are schema mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ such that Σ_{12} is a finite set of source-to-target tgds, Σ_{23} consists of a single full source-to-target tgd, but the composition query for $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is NP-complete. Furthermore, this composition query is not definable by any formula of the finite-variable infinitary logic $L^{\omega}_{\infty\omega}$, which is a powerful formalism that subsumes least fixed-point logic LFP (hence, it subsumes first-order logic and Datalog) on finite structures (see [2]).

THEOREM 3.6. *There exist schema mappings $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$ such that Σ_{12} is a finite set of source-to-target tgds, Σ_{23} consists of one full source-to-target tgd, and the following hold for the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$:*

1. *The composition query of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is NP-complete.*
2. *The composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is not definable by any formula of $L^{\omega}_{\infty\omega}$, and hence of least fixed-point logic LFP.*

PROOF. *(Sketch)* Later (Proposition 3.8), we shall show that the composition query of schema mappings definable by finite sets of source-to-target tgds is always in NP. As we now describe, NP-hardness can be obtained by a reduction of 3-COLORABILITY to the composition query of two schema mappings. The schema $\mathbf{S}_1$ consists of a single binary relation symbol E, the schema $\mathbf{S}_2$ consists of two binary relation symbols C and F, and the schema $\mathbf{S}_3$ consists of one binary relation symbol D. The set Σ_{12} consists of the following two source-to-target tgds:

$$\forall x \forall y (E(x,y) \rightarrow \exists u \exists v (C(x,u) \wedge C(y,v)))$$
$$\forall x \forall y (E(x,y) \rightarrow F(x,y)).$$

Finally, Σ_{23} consists of a single full source-to-target tgd:

$$\forall x \forall y \forall u \forall v (C(x,u) \wedge C(y,v) \wedge F(x,y) \rightarrow D(u,v)).$$

Let I_3 be the instance over the schema $\mathbf{S}_3$ with

$$D = \{(r,g),(g,r),(b,r),(r,b),(g,b),(b,g)\}.$$

In words, D contains all pairs of different colors among the three colors r, g, and b. Let $\mathbf{G} = (V,E)$ be a graph and let I_1 be the instance over $\mathbf{S}_1$ consisting of the edge relation E of $\mathbf{G}$. It is not hard to verify that $\mathbf{G}$ is 3-colorable if and only if $\langle I_1, I_3 \rangle \in \mathrm{Inst}(\mathcal{M}_{12}) \circ \mathrm{Inst}(\mathcal{M}_{23})$.

The above reduction of 3-COLORABILITY to the composition query of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ belongs to a class of weak polynomial-time reductions known as *quantifier-free* reductions, since the instance $\langle I_1, I_3 \rangle$ of the composition query can be defined from the instance $\mathbf{G} = (V,E)$ using quantifier-free formulas (see Immerman [14] for the precise definitions). Dawar [7] showed that 3-COLORABILITY is not expressible in the finite-variable infinitary logic $L^{\omega}_{\infty\omega}$. It follows that the composition query of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is not expressible in $L^{\omega}_{\infty\omega}$. ☐

Note that the formula

$$\forall x \forall y (E(x,y) \rightarrow \exists u \exists v (C(x,u) \wedge C(y,v)))$$

is the only source-to-target tgd in Σ_{12} in the proof of Theorem 3.6 that is not a full source-to-target tgd. Although two existential quantifiers occur in this source-to-target tgd, it is equivalent to the set of the two source-to-target tgds

$$\forall x \forall y (E(x,y) \rightarrow \exists u C(x,u)), \quad \forall x \forall y (E(x,y) \rightarrow \exists v C(y,v)).$$

Thus, the preceding Theorem 3.6 and Proposition 3.3 yield a sharp boundary on the definability of the composition of schema mappings specified by finite sets of source-to-target tgds. More precisely, the composition of a finite set of full source-to-target tgds with a finite set of source-to-target tgds is always definable by a first-order formula (and, in fact, definable by a finite conjunction of source-to-target tgds), while the composition of a finite set of source-to-target tgds having at most one existential quantifier with a set consisting of one full source-to-target tgd may not even be $L^{\omega}_{\infty\omega}$-definable. Similarly, the computational complexity of the associated composition query may jump from solvable in polynomial time to NP-complete.

Let $\mathbf{S}$ be a schema and I, J two instances over $\mathbf{S}$. A function h is a *homomorphism* from I to J if for every relation symbol R in $\mathbf{S}$ and every tuple $(a_1, \ldots, a_n) \in R^I$, we have that $(h(a_1), \ldots, h(a_n)) \in R^J$. The HOMOMORPHISM PROBLEM over the schema $\mathbf{S}$ is the following decision problem: given two instances I and J of $\mathbf{S}$, is there a homomorphism from I to J? This is a fundamental algorithmic problem because, as shown by Feder and Vardi [12], all constraint satisfaction problems can be identified with homomorphism problems. In particular, 3-SAT and 3-COLORABILITY are special cases of the HOMOMORPHISM PROBLEM over suitable schemas. A slight modification of the proof of the preceding Theorem 3.6 shows that for every schema $\mathbf{S}$, the HOMOMORPHISM PROBLEM over $\mathbf{S}$ has a simple quantifier-free reduction to the composition query of two schema mappings specified by finite sets of source-to-target tgds.

PROPOSITION 3.7. *For every schema $\mathbf{S} = \langle R_1, \ldots, R_m \rangle$, there are schema mappings $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$ such that Σ_{12} is a finite set of source-to-target tgds and Σ_{23} is a finite set of full source-to-target tgds, with the property that the* HOMOMORPHISM PROBLEM *over $\mathbf{S}$ has a quantifier-free reduction to the composition query of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$.*

PROOF. The schema $\mathbf{S}_1$ is the same as the schema $\mathbf{S} = \langle R_1, \ldots, R_m \rangle$. The schema $\mathbf{S_2}$ is $\langle H, T_1, ..., T_m \rangle$, where H is a binary relation symbol and each T_i has the same arity as R_i, $1 \leq i \leq m$. The schema $\mathbf{S_3}$ is $\langle P_1, \ldots, P_m \rangle$, where each P_i has the same arity as R_i, $1 \leq i \leq m$. The dependencies in Σ_{12} and Σ_{23} are as follows:

$$\begin{aligned}
\Sigma_{12} = \{ \ & \forall x_1 ... \forall x_{k_1} (R_1(x_1, ..., x_{k_1}) \rightarrow \\
& \quad \exists y_1 ... \exists y_{k_1} \ (H(x_1, y_1) \wedge ... \wedge H(x_{k_1}, y_{k_1}))), \\
& \vdots \\
& \forall x_1 ... \forall x_{k_m} (R_m(x_1, ..., x_{k_m}) \rightarrow \\
& \quad \exists y_1 ... \exists y_{k_m} \ (H(x_1, y_1) \wedge ... \wedge H(x_{k_m}, y_{k_m}))), \\
& \forall \mathbf{x} (R_1(\mathbf{x}) \rightarrow T_1(\mathbf{x})), \\
& \vdots \\
& \forall \mathbf{x} (R_m(\mathbf{x}) \rightarrow T_m(\mathbf{x})) \ \}
\end{aligned}$$

$$\begin{aligned}
\Sigma_{23} = \{ \ & \forall x_1 \forall y_1 ... \forall x_{k_1} \forall y_{k_1} \\
& \quad ((H(x_1, y_1) \wedge ... \wedge H(x_{k_1}, y_{k_1}) \\
& \qquad \wedge T_1(x_1, ..., x_{k_1})) \rightarrow P_1(y_1, ..., y_{k_1})), \\
& \vdots
\end{aligned}$$

$$\forall x_1 \forall y_1 ... \forall x_{k_m} \forall y_{k_m} \\ ((H(x_1, y_1) \wedge ... \wedge H(x_{k_m}, y_{k_m}) \\ \wedge T_m(x_1, ..., x_{k_m})) \rightarrow P_m(y_1, ..., y_{k_m}))\ \}.$$

Let $I = \langle R_1^I, ..., R_m^I \rangle$ and $J = \langle R_1^J, \ldots, R_m^J \rangle$ be two instances of $\mathbf{S}$. Since $\mathbf{S}_1$ is the same as $\mathbf{S}$, we have that I is an instance of $\mathbf{S}_1$. Let J' be the instance over $\mathbf{S}_3$ where $P_i^{J'} = R_i^J$, for $1 \leq i \leq m$. (Thus, J' is the same as J except that the relation names reflect schema $\mathbf{S}_3$ rather than $\mathbf{S}_1$.) It is easy to verify that there is a homomorphism from I to J if and only if $\langle I, J' \rangle$ is in $\text{Inst}(\mathcal{M}_{12}) \circ \text{Inst}(\mathcal{M}_{23})$. $\square$

The next result establishes an upper bound on the computational complexity of the composition query associated with two schema mappings specified by finite sets of source-to-target tgds. It also shows that the composition of two such mappings is always definable by an existential second-order formula.

Proposition 3.8. *If $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$ are schema mappings such that Σ_{12} and Σ_{23} are finite sets of source-to-target tgds, then the composition query of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is in NP. Consequently, the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is definable by an existential second-order formula.*

Proof. *(Sketch)* To establish membership in NP, it suffices to show that if $\langle I_1, I_3 \rangle \in \text{Inst}(\mathcal{M}_{12}) \circ \text{Inst}(\mathcal{M}_{23})$, then there is an instance I_2 over $\mathbf{S}_2$ that has size polynomial in the sizes of I_1 and I_3, and is such that $\langle I_1, I_2 \rangle \models \Sigma_{12}$ and $\langle I_2, I_3 \rangle \models \Sigma_{23}$. This can be proven by taking I_2 to be a homomorphic image of a *canonical universal solution* as defined in [10]. The details are omitted.

The fact that the composition query of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is in NP implies, by Fagin's Theorem [8], that the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is definable by an existential second-order formula. $\square$

We conclude this section by showing that the results of Proposition 3.8 may fail dramatically for schema mappings specified by arbitrary first-order formulas.

Proposition 3.9. *There are schema mappings $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$ such that Σ_{12} consists of a single first-order formula, Σ_{23} is the empty set, and the composition query of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is undecidable.*

Proof. *(Sketch)* We define $\mathcal{M}_{12}$ in such a way that $\langle I_1, I_2 \rangle \in \text{Inst}(\mathcal{M}_{12})$ precisely when I_1 is the encoding of a Turing machine and I_2 represents a terminating computation of that Turing machine (thus, Σ_{12} consists of a first-order formula that expresses this connection). We let the schema $\mathbf{S}_3$ consist of, say, a single unary relation symbol, and let Σ_{23} be the empty set. So, the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ consists of all $\langle I_1, I_3 \rangle$ where I_1 is the encoding of a halting Turing machine, and I_3 is arbitrary. The result follows from the fact that it is undecidable to determine if a Turing machine is halting. $\square$

4. Second-Order TGDs

We have seen in the previous section that the composition of two schema mappings specified by finite sets of source-to-target tgds may not be definable by a set (finite or infinite) of source-to-target tgds. From Proposition 3.8, however, we know that such a composition is always definable by an existential second-order formula. We shall show in this section that, in fact, the composition of schema mappings specified by finite sets of source-to-target tgds is always definable by a restricted form of existential second-order formulas, which we call *second-order tuple-generating dependencies (SO tgds)*. Intuitively, an SO tgd is a source-to-target tgd suitably extended with existentially quantified functions and equalities. Furthermore, an SO tgd is capable of defining the composition of two schema mappings that are specified by SO tgds. In other words, SO tgds are *closed under composition*. Moreover, we shall show in Section 5 that SO tgds possess good properties for data exchange. All these properties justify SO tgds as the right language for representing schema mappings and for composing schema mappings.

Example 4.1. The proof of Proposition 3.4 shows that for the two schema mappings of Example 3.1 there is no finite set of source-to-target tgds that can define the composition. In contrast, the following formula is an SO tgd that defines this composition:

$$\exists f (\ \forall n \forall c\, (\texttt{Takes}(n, c) \rightarrow \texttt{Enrollment}(f(n), c))\ ) \qquad (1)$$

In this formula, f is a function symbol that associates each student name n with a student id $f(n)$. The SO tgd states that whenever a student name n is associated with a course c in Takes, then the corresponding student id $f(n)$ is associated with c in Enrollment. This is independent of how many courses a student takes: if student name n is associated with courses $c_1, \ldots, c_k$ in Takes, then $f(n)$ is associated with all of $c_1, \ldots, c_k$ in Enrollment.

To verify that (1) does indeed define the composition, assume first that $\langle I_1, I_3 \rangle \in \text{Inst}(\mathcal{M}_{12}) \circ \text{Inst}(\mathcal{M}_{23})$. Then there is I_2 over $\mathbf{S}_2$ such that $\langle I_1, I_2 \rangle \models \Sigma_{12}$ and $\langle I_2, I_3 \rangle \models \Sigma_{23}$. We construct a function f as follows. For each n such that (n, c) is in $\texttt{Takes}^{I_1}$, we set $f(n) = s$, where s is such that (n, s) is in $\texttt{Student}^{I_2}$ (such s is guaranteed to exist according to the second source-to-target tgd in Σ_{12}, and we pick one such s). It is immediate that $\langle I_1, I_3 \rangle$ satisfies the SO tgd when the existentially quantified function symbol is instantiated with the constructed f. Conversely, assume that $\langle I_1, I_3 \rangle$ satisfies the SO tgd. Then there is a function f such that for every (n, c) in $\texttt{Takes}^{I_1}$ we have that $(f(n), c)$ is in $\texttt{Enrollment}^{I_3}$. Let I_2 be such that $\texttt{Student}^{I_2} = \{(n, f(n) \mid (n, c) \in \texttt{Takes}^{I_1}\}$ and $\texttt{Takes}_1^{I_2} = \texttt{Takes}^{I_1}$. It can be verified that $\langle I_1, I_2 \rangle \models \Sigma_{12}$ and $\langle I_2, I_3 \rangle \models \Sigma_{23}$. $\square$

Example 4.2. This example illustrates a slightly more complex form of a second-order tgd that contains equalities between terms. Consider the following three schemas $\mathbf{S}_1$, $\mathbf{S}_2$ and $\mathbf{S}_3$. Schema $\mathbf{S}_1$ consists of a single unary relation symbol Emp of employees. Schema $\mathbf{S}_2$ consists of one binary relation symbol $\texttt{Mgr}_1$, that associates each employee with a manager. Schema $\mathbf{S}_3$ consists of a similar binary relation symbol Mgr and an additional unary relation symbol SelfMgr, intended to store employees who are their own managers.

Consider now the schema mappings $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$, where

$$\begin{array}{rcl}\Sigma_{12} & = & \{\ \forall e\,(\mathtt{Emp}(e) \rightarrow \exists m \mathtt{Mgr}_1(e,m))\ \} \\ \Sigma_{23} & = & \{\ \forall e \forall m\,(\mathtt{Mgr}_1(e,m) \rightarrow \mathtt{Mgr}(e,m)), \\ & & \quad \forall e(\mathtt{Mgr}_1(e,e) \rightarrow \mathtt{SelfMgr}(e))\ \}.\end{array}$$

It is easy to verify that the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is $\mathcal{M}_{13}$, where Σ_{13} is the following second-order tgd:

$$\begin{array}{l}\exists f(\ \forall e(\mathtt{Emp}(e) \rightarrow \mathtt{Mgr}(e, f(e))) \wedge \\ \quad \forall e(\mathtt{Emp}(e) \wedge (e = f(e)) \rightarrow \mathtt{SelfMgr}(e))).\end{array}$$

In fact, we shall derive this later when we give a composition algorithm. □

We will use this example shortly to show that equalities in SO tgds are strictly necessary for the purposes of composition, and also to give a proof for the earlier Proposition 3.5.

Before we formally define SO tgds, we need to define *terms*. Given a collection $\mathbf{x}$ of variables and a collection $\mathbf{F}$ of function symbols, a *term (based on* $\mathbf{x}$ *and* $\mathbf{F}$*)* is defined recursively as follows:

1. Every variable in $\mathbf{x}$ is a term.
2. If f is a k-ary function symbol in $\mathbf{F}$ and $t_1, \ldots, t_k$ are terms, then $f(t_1, \ldots, t_k)$ is a term.

We now give the precise definition of an SO tgd.

Definition 4.3. Let $\mathbf{S}$ be a source schema and $\mathbf{T}$ a target schema. A *second-order tuple-generating dependency (SO tgd)* is a formula of the form:

$$\exists f_1 \ldots \exists f_m\,((\forall \mathbf{x_1}(\phi_1 \rightarrow \psi_1)) \wedge \ldots \wedge (\forall \mathbf{x_n}(\phi_n \rightarrow \psi_n)))$$

where

1. Each f_i is a function symbol.
2. Each ϕ_i is a conjunction of
   • atomic formulas of the form $R(y_1, ..., y_k)$, where R is a k-ary relation symbol of schema $\mathbf{S}$ and $y_1, \ldots, y_k$ are variables in $\mathbf{x_i}$, not necessarily distinct,
   and
   • equalities of the form $t = t'$ where t and t' are terms based on $\mathbf{x_i}$ and $\{f_1, \ldots, f_m\}$.
3. Each ψ_i is a conjunction of atomic formulas $S(t_1, ..., t_l)$ where S is an l-ary relation symbol of schema $\mathbf{T}$ and $t_1, \ldots, t_l$ are terms based on $\mathbf{x_i}$ and $\{f_1, \ldots, f_m\}$.
4. Each variable in $\mathbf{x_i}$ is a safe term with respect to ϕ_i and $\{f_1, \ldots, f_m\}$, where:
   A *safe term with respect to* ϕ_i *and* $\{f_1, \ldots, f_m\}$ is defined recursively as one of the following: (a) a variable x occurring in a relational atomic formula of ϕ_i, (b) a variable x occurring in an equality of the form $x = t$ or $t = x$ of ϕ_i where t is a safe term with respect to ϕ_i and $\{f_1, \ldots, f_m\}$, or (c) a term $f(t_1, \ldots, t_k)$ where f is in $\{f_1, \ldots, f_m\}$ and $t_1, \ldots, t_k$ are safe terms with respect to ϕ_i and $\{f_1, \ldots, f_m\}$.

The fourth condition is a "safety" assumption that makes the second-order tgds *domain independent* (so that their truth does not depend on any underlying domain, but only on the "active domain" of elements that appear in tuples in the source). In the case of first-order tgds, where equalities are not present, this condition becomes a simpler one by requiring that every universally quantified variable appear in one of the relational atomic formulas in the left-hand side of the tgd. This condition is not always made explicit in the literature in the definition of first-order tgds, although it should be. It was first made explicit for dependencies by Fagin [9], who observed that this condition makes the dependencies domain independent.

The following formula is a valid example of a second-order tgd where all universally quantified variables are safe:

$$\exists f \forall x \forall y \forall z (R(x) \wedge (y = f(z)) \wedge (z = x) \rightarrow S(x, y, z))$$

More concretely, x is safe because it appears inside R; z is safe because it is equated with x; and y is safe because it is equated with $g(z)$, which is safe because z is safe. In contrast, the following two formulas are not valid second-order tgds:

$$\exists f \forall x \forall y \forall z \forall w (R(x) \wedge (y = f(z)) \wedge (z = w) \rightarrow S(x, y, z))$$

$$\exists f \exists h \forall x \forall y \forall z (R(x) \wedge (h(y) = f(z)) \wedge (z = x) \rightarrow S(x, y, z))$$

In the first case, w, z and y are not safe, while in the second case, y is not safe.

Several remarks are in order now. First, it is easy to verify that SO tgds are closed under conjunction. That is, a finite set of SO tgds is always equivalent to a single SO tgd. For this reason, when we consider schema mappings specified by SO tgds, it is enough to restrict our attention to the case where the set Σ_{st} consists of one SO tgd. We will then identify the singleton set Σ_{st} with the SO tgd itself, and refer to Σ_{st} as an SO tgd.

Next, we describe a "normal form" for SO tgds. For this, let us define the *nesting depth* of a term t, denoted by $\mathit{nest}(t)$, as follows. If t is a variable, then $\mathit{nest}(t) = 0$. If t is of the form $f(t_1, \ldots, t_k)$, then $\mathit{nest}(t) = 1 + \max\{\mathit{nest}(t_1), \ldots, \mathit{nest}(t_k)\}$. Let us say that a term t is *nested* if $\mathit{nest}(t) \geq 2$. Thus, $f(x, g(x))$ is nested, while $f(x, y)$ is not nested. It is easy to see that every SO tgd is logically equivalent to an SO tgd containing only non-nested terms. Intuitively, this is because we can make use of equalities to transform a given SO tgd to an equivalent SO tgd without nested terms. For example, consider the SO tgd

$$\exists f \exists g \forall x (R(x) \rightarrow S(f(x, g(x)))),$$

which has the nested term $f(x, g(x))$. This SO tgd is equivalent to the SO tgd

$$\exists f \exists g \forall x \forall y ((R(x) \wedge (y = g(x))) \rightarrow S(f(x, y))),$$

which has no nested terms. Thus, we could have restricted the definition of SO tgds by allowing only non-nested terms to appear; for the sake of naturalness, however, we choose to allow nested terms.

Finally, it should not come as a surprise that every (first-order) source-to-target tgd is equivalent to an SO tgd. In fact, it is easy to see that every source-to-target tgd is equivalent to an SO tgd without equalities. Specifically, let σ be the source-to-target tgd

$$\forall x_1 \ldots \forall x_m(\phi_S(x_1, \ldots, x_m) \rightarrow \\ \exists y_1 \ldots \exists y_n \psi_T(x_1, \ldots, x_m, y_1, \ldots, y_n)).$$

Then σ is equivalent to the following SO tgd without equalities, which is obtained by Skolemizing σ:

$$\exists f_1 \ldots \exists f_n \forall x_1 \ldots \forall x_m(\phi_S(x_1, \ldots, x_m) \rightarrow \\ \psi_T(x_1, \ldots, x_m, f_1(x_1, \ldots, x_m), \ldots, f_n(x_1, \ldots, x_m))).$$

4.1 Necessity of Equalities

Our definition of SO tgds allows for equalities between terms in the formulas ϕ_i, even though we just saw that SO tgds that represent first-order tgds do not require equalities. We now show that such equalities are necessary, since it may not be possible to define the composition of two schema mappings otherwise.

THEOREM 4.4. *There exist schema mappings $\mathcal{M}_{12}$=($\mathbf{S}_1$, $\mathbf{S}_2$, Σ_{12}) and $\mathcal{M}_{23}$=($\mathbf{S}_2$, $\mathbf{S}_3$, Σ_{23}) where Σ_{12} is a single tgd and Σ_{23} is a set of full tgds, such that the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is given by an SO tgd that is not equivalent to any SO tgd without equalities.*

PROOF. Let $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$, Σ_{12}, Σ_{23}, and Σ_{13} be as in Example 4.2. We need only show that there is no SO tgd without equalities that is logically equivalent to Σ_{13}.

Define I_1 by letting $\texttt{Emp}^{I_1} = \{\text{Bob}\}$. Define I_3 by letting $\texttt{Mgr}^{I_3} = \{(\text{Bob}, \text{Susan})\}$ and $\texttt{SelfMgr}^{I_3} = \emptyset$. Define I_3' by letting $\texttt{Mgr}^{I_3'} = \{(\text{Bob}, \text{Bob})\}$ and $\texttt{SelfMgr}^{I_3'} = \emptyset$. It is easy to see that $\langle I_1, I_3 \rangle \models \Sigma_{13}$; intuitively, we let $f(\text{Bob}) = \text{Susan}$. It is also easy to see that $\langle I_1, I_3' \rangle \not\models \Sigma_{13}$, since $\texttt{SelfMgr}^{I_3'}$ does not contain Bob.

We shall show that every SO tgd without equalities that is satisfied by $\langle I_1, I_3 \rangle$ is also satisfied by $\langle I_1, I_3' \rangle$. Since also $\langle I_1, I_3 \rangle \models \Sigma_{13}$ but $\langle I_1, I_3' \rangle \not\models \Sigma_{13}$, it follows easily that Σ_{13} is not equivalent to any SO tgd without equalities, which proves the theorem. In fact, we note for later use that this shows that Σ_{13} is not equivalent to any set (even infinite) of SO tgds without equalities.

Let σ be an SO tgd without equalities that is satisfied by $\langle I_1, I_3 \rangle$. The proof is complete if we show that σ is satisfied by $\langle I_1, I_3' \rangle$. Assume that σ is

$$\exists f_1 ... \exists f_m ((\forall \mathbf{x_1}(\phi_1 \rightarrow \psi_1)) \wedge ... \wedge (\forall \mathbf{x_n}(\phi_n \rightarrow \psi_n))).$$

We begin by showing that $\texttt{SelfMgr}$ does not appear in σ. Assume that $\texttt{SelfMgr}$ appears in σ; we shall derive a contradiction. By the definition of an SO tgd, we know that there is i and some term t such that $\texttt{SelfMgr}(t)$ appears in ψ_i. Since by assumption ϕ_i does not contain any equalities, it follows that ϕ_i contains only formulas of the form $\texttt{Emp}(y)$, with y a member of $\mathbf{x_i}$. So ϕ_i can be satisfied in I_1, by letting Bob play the role of all of the variables in $\mathbf{x_i}$. Since $\texttt{SelfMgr}^{I_3}$ is empty, it follows that ψ_i is not satisfied under this (or any) assignment. Therefore, σ is not satisfied in $\langle I_1, I_3 \rangle$, which is the desired contradiction.

We conclude the proof by showing that $\langle I_1, I_3' \rangle$ satisfies σ. Let the role of every function symbol f_j be played by constant functions (of the appropriate arity) that always take on the value Bob. Consider a clause $\forall \mathbf{x_i}(\phi_i \rightarrow \psi_i)$ of σ. We must show that if ϕ_i holds in I_1 for some assignment to the variables in $\mathbf{x_i}$, then ψ_i holds in I_3' for the same assignment to the variables in $\mathbf{x_i}$. It follows from the fourth condition in the definition of SO tgds that the conjuncts of ϕ_i are precisely all formulas of the form $\texttt{Emp}(x)$ for x in $\mathbf{x_i}$. Since ϕ_i holds in I_1, every variable x in $\mathbf{x_i}$ is assigned the value Bob. Therefore, every term in ψ_i is assigned the value Bob. Since by assumption ψ_i does not contain $\texttt{SelfMgr}$, it follows that every conjunct in ψ_i is of the form $\texttt{Mgr}(t_1, t_2)$. Since, as we just showed, t_1 and t_2 are both assigned the value Bob, it follows that ψ_i holds in I_3'. This was to be shown. □

As we noted in the proof, the proof actually shows that in the example considered, even an infinite set of SO tgds without equalities cannot define the composition. Since every source-to-target tgd is equivalent to an SO tgd without equalities, this proves Proposition 3.5.

We consider it quite interesting that allowing equalities in SO tgds is necessary to make them sufficiently expressive. This is particularly true because the "obvious" way to define SO tgds does not allow equalities. Indeed, as we saw, when we Skolemize a source-to-target tgd to obtain an SO tgd, no equalities are introduced.

4.2 Composability of Second-Order TGDs

As we have seen earlier, sets of source-to-target tgds are not closed under composition. By contrast, we show in this section that SO tgds are closed under composition. Given two schema mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ where Σ_{12} and Σ_{23} are SO tgds, the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ is always definable by an SO tgd. We show this by exhibiting a composition algorithm and then giving a theorem that says that the composition algorithm is correct.

Algorithm Compose($\mathcal{M}_{12}$, $\mathcal{M}_{23}$)

Input: Two schema mappings $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$, where Σ_{12} and Σ_{23} are SO tgds.
Output: A schema mapping $\mathcal{M}_{13} = (\mathbf{S}_1, \mathbf{S}_3, \Sigma_{13})$, which is the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$ and where Σ_{13} is an SO tgd.

1. (*Split up the SO tgds in Σ_{12} and Σ_{23}.*)
   Initialize $\mathcal{S}_{12}$ and $\mathcal{S}_{23}$ to each be the empty set. Assume that the SO tgd in Σ_{12} is

   $$\exists f_1 ... \exists f_m ((\forall \mathbf{x_1}(\phi_1 \rightarrow \psi_1)) \wedge ... \wedge (\forall \mathbf{x_n}(\phi_n \rightarrow \psi_n))).$$

   Put each of the n implications $\phi_i \rightarrow \psi_i$, for $1 \leq i \leq n$, into $\mathcal{S}_{12}$. We do likewise for Σ_{23} and $\mathcal{S}_{23}$. Each implication χ in $\mathcal{S}_{12}$ has the form $\phi(\mathbf{x}) \rightarrow \bigwedge_{j=1}^{k} R_j(\mathbf{t_j})$ where $\mathbf{x}$ contains the universally quantified variables and each $\mathbf{t_j}$, for $1 \leq j \leq k$, is a vector of terms over $\mathbf{x}$. We then replace each such implication χ in $\mathcal{S}_{12}$ with k implications:

$$\phi(\mathbf{x}) \rightarrow R_1(\mathbf{t_1}),\ ...,\ \phi(\mathbf{x}) \rightarrow R_k(\mathbf{t_k})$$

2. (*Compose $\mathcal{S}_{12}$ with $\mathcal{S}_{23}$.*)
   Repeat the following until every relation symbol in the left-hand side of every formula in $\mathcal{S}_{23}$ is from $\mathbf{S}_1$.

   For each implication χ in $\mathcal{S}_{23}$ of the form $\psi \rightarrow \gamma$ where there is an atom $R(\mathbf{x})$ in ψ such that R is a relation symbol in $\mathbf{S}_2$, we perform the following steps to replace $R(\mathbf{x})$ with atoms over $\mathbf{S}_1$. (The equalities in ψ are left unchanged.) Let

   $$\phi_1 \rightarrow R(\mathbf{t_1}),\ ...,\ \phi_m \rightarrow R(\mathbf{t_m})$$

   be all the implications in $\mathcal{S}_{12}$ whose right-hand side has the relation symbol R in it. (The variables in these tgds had been renamed so that they do not overlap with the variables in χ.) If no such implications exist in $\mathcal{S}_{12}$, we remove χ from $\mathcal{S}_{23}$. Otherwise, for each such implication $\phi_i \rightarrow R(\mathbf{t_i})$, let θ_i be the conjunction of the equalities between the variables in $R(\mathbf{x})$ and the corresponding terms in $R(\mathbf{t_i})$, position by position. For example, the conjunction of equalities, position by position, between $R(x_1, x_2, x_3)$ and $R(z_1, f_2(z_2), f_1(z_3))$ is $(x_1 = z_1) \wedge (x_2 = f_2(z_2)) \wedge (x_3 = f_1(z_3))$. Observe that every equality that is generated has the form $x = t$ where x is a variable and t is a term. Remove χ from $\mathcal{S}_{23}$ and add m implications to $\mathcal{S}_{23}$ as follows: replace $R(\mathbf{x})$ in χ with $\phi_i \wedge \theta_i$ and add the resulting implication to $\mathcal{S}_{23}$, for $1 \leq i \leq m$.

3. (*Construct $\mathcal{M}_{13}$.*)
   Let $\mathcal{S}_{23} = \{\chi_1, ..., \chi_r\}$ where $\chi_1, \ldots, \chi_r$ are all the implications constructed in the previous step. Let Σ_{13} be the following SO tgd:

   $$\exists f_1 \ldots \exists f_s\ (\forall \mathbf{x_1} \chi_1 \wedge \ldots \wedge \forall \mathbf{x_r} \chi_r)$$

   where $f_1, \ldots, f_s$ are all of the function symbols that appear in any of the implications in $\mathcal{S}_{23}$ and the variables in $\mathbf{x_i}$ are all the variables found in the implication χ_i, for $1 \leq i \leq r$.

   Return $\mathcal{M}_{13} = (\mathbf{S}_1, \mathbf{S}_3, \Sigma_{13})$. □

The intuition behind why the algorithm is able to eliminate the relation symbols in $\mathbf{S}_2$ is that all that matters about the relations in the intermediate instance I_2 is the equality pattern among the tuples, rather than the exact values. Note that the number of formulas in the set $\mathcal{S}_{13}$, and hence the size of Σ_{13}, is exponential in the maximum number of relational atoms that can appear in the left-hand side of an implication in Σ_{23}.

It can be verified that the algorithm generates second-order tgds that are valid according to Definition 4.3. More concretely, in Step 2 of the algorithm, all the equalities that are generated are of the form $x = t$ where x is a variable and t is a safe term with respect to the left-hand side (ϕ_i) of some implication in $\mathcal{S}_{12}$. But then t continues to be a safe term with respect to the new left-hand side of the newly generated implication (recall that an atomic formula $R(\mathbf{x})$ is replaced by $\phi_i \wedge \theta$). Hence, x itself is safe.

We can make use of the algorithm to compose schema mappings where Σ_{12} and Σ_{23} are specified by finite sets of source-to-target tgds by first transforming each of Σ_{12} and Σ_{23} into an SO tgd (as described in the previous section) and then passing the resulting schema mappings to the composition algorithm.

Example 4.5. We illustrate the steps of the composition algorithm using the schema mappings of Example 4.2. We transform Σ_{12} and Σ_{23} into the following SO tgds, Σ'_{12} and Σ'_{23}:

$$\begin{aligned}\Sigma'_{12} &= \exists f (\forall e (\mathtt{Emp}(e) \rightarrow \mathtt{Mgr}_1(e, f(e))))\\ \Sigma'_{23} &= \forall e \forall m (\mathtt{Mgr}_1(e, m) \rightarrow \mathtt{Mgr}(e, m)) \wedge\\ &\quad\ \forall e (\mathtt{Mgr}_1(e, e) \rightarrow \mathtt{SelfMgr}(e))\end{aligned}$$

We run the composition algorithm with $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma'_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma'_{23})$. After step 1, the sets $\mathcal{S}_{12}$ and $\mathcal{S}_{23}$ consist of the following implications:

$$\begin{aligned}\mathcal{S}_{12}:&\quad \mathtt{Emp}(e) \rightarrow \mathtt{Mgr}_1(e, f(e))\\ \mathcal{S}_{23}:&\quad \mathtt{Mgr}_1(e, m) \rightarrow \mathtt{Mgr}(e, m)\\ &\quad \mathtt{Mgr}_1(e, e) \rightarrow \mathtt{SelfMgr}(e)\end{aligned}$$

After step 2, the set $\mathcal{S}_{23}$ contains two implications, χ_1 and χ_2:

$$\begin{aligned}\chi_1:&\quad \mathtt{Emp}(e_0) \wedge (e = e_0) \wedge (m = f(e_0)) \rightarrow \mathtt{Mgr}(e, m)\\ \chi_2:&\quad \mathtt{Emp}(e_0) \wedge (e = e_0) \wedge (e = f(e_0)) \rightarrow \mathtt{SelfMgr}(e)\end{aligned}$$

Therefore, after step 3, the algorithm returns $\mathcal{M}_{13} = (\mathbf{S}_1, \mathbf{S}_3, \Sigma_{13})$ where Σ_{13} is the following SO tgd:

$$\exists f (\forall e_0 \forall e \forall m\, \chi_1 \wedge \forall e_0 \forall e\, \chi_2)$$

It can be easily verified that the above SO tgd is equivalent to the SO tgd that was shown in Example 4.2. Essentially, whenever the left-hand side of an SO tgd contains an equality of the form $y = t$, where y is a variable and t is some term in which y does not occur, we can replace all the occurrences of y with t and then remove y, together with the equality $y = t$, to obtain a simplified SO tgd. □

Theorem 4.6. *Let $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$, where Σ_{12} and Σ_{23} are SO tgds. Then the algorithm Compose($\mathcal{M}_{12}$, $\mathcal{M}_{23}$) returns a schema mapping $\mathcal{M}_{13} = (\mathbf{S}_1, \mathbf{S}_3, \Sigma_{13})$ such that Σ_{13} is an SO tgd and $\mathcal{M}_{13} = \mathcal{M}_{12} \circ \mathcal{M}_{23}$.*

5. Data Exchange with Second-Order TGDs

Our main motivation for studying composition of schema mappings stems from data exchange [10, 11]. A specific case of data exchange is one in which we are given a source schema, a target schema, and a schema mapping specified by a set of source-to-target tgds. Given an instance over the source schema, we are interested in materializing a target instance that satisfies the specification. In the case of two or more successive data exchange scenarios and when only a final instance over the final target schema is of interest, we would like to avoid materializing intermediate instances, and hence use the schema mapping that is the composition of the sequence of schema mappings. However, as we have argued so far, the language of source-to-target tgds may no longer be appropriate in this case. Instead we need to use second-order tgds. In this section, we describe how to modify the classical chase technique [3] to handle SO tgds (rather than the usual first-order tgds) in order to produce *universal* solutions [10] for data exchange. In particular, we introduce chasing with SO tgds as a variation on logic program evaluation [16] that has polynomial-time computation, and give a proposition that says that its result is a universal solution.

Using an example, we illustrate next the chase with SO tgds. We will keep the discussion informal and omit the full details of the definition of this chase.

Example 5.1. Consider the schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ where Σ_{st} is the following SO tgd:

$$\begin{aligned}&\exists f(\ \forall x \forall y\,(R(x,y) \rightarrow U(x,y,f(x))) \wedge \\ &\quad \forall x \forall x' \forall y \forall y' \\ &\qquad (R(x,y) \wedge R(x',y') \wedge (f(x)=f(x')) \rightarrow T(y,y')))\end{aligned}$$

Each of the formulas of the form $\forall \mathbf{x}(\phi \rightarrow \psi)$ that appear under the scope of the existentially quantified functions can be thought of a source-to-target "tgd". The difference from a normal source-to-target tgd is that now we can have function terms in the relational atoms of ψ, as well as equality atoms in ϕ . Suppose now that we are given a source instance I where R consists of the following three tuples: (a,b), (a,c), and (d,e). The chase starts with an instance of the form $\langle I, \emptyset \rangle$ and constructs an instance of the form $\langle I, J \rangle$ by applying all the "tgds" until these "tgds" are all satisfied. A "tgd" is applied when the left-hand side ϕ of the "tgd" can be mapped to I but the corresponding right-hand side ψ does not yet exist in J, in which case we add it to J. By applying the first "tgd" in Σ_{st}, for the first tuple (a,b) of R we would generate a tuple $(a,b,f(a))$ in U. Note that *ground* function terms such as $f(a)$ may now appear in tuples of J. A *ground term* g is defined recursively as either a value v from the source or a function term of the form $f(g_1,\ldots,g_k)$ where $g_1,\ldots,g_k$ are ground terms. In applying the same "tgd", this time for the tuple (a,c) of R, we generate $(a,c,f(a))$ in U (the same ground function term $f(a)$ appears again). Finally, for the last tuple (d,e) and the same "tgd" we generate $(d,e,f(d))$ in U.

To apply the second "tgd" in Σ_{st}, we see that only the combinations $f(a)=f(a)$ and $f(d)=f(d)$ can satisfy the equality $f(x)=f(x')$. (Two ground terms are treated as equal precisely if they are syntactically identical.) Hence the chase will generate the tuples (b,b), (b,c), (c,b), (c,c) and (e,e) in T.

At the end of the chase, we replace all the ground function terms by special values that we call *nulls*. Here we make sure that two ground terms are replaced by the same null precisely if they are syntactically identical. For our example, we obtain an instance J where U contains the tuples $(a,b,X),(a,c,X),(d,e,Y)$, where X and Y are distinct nulls replacing $f(a)$ and $f(d)$ respectively. The relation T remains the same, as it contains only source values. $\square$

The concept of a *universal solution* for the data exchange problem was introduced in [10]. Given a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ and an instance I over S, a *universal solution* of I under $\mathcal{M}$ is a solution J of I under $\mathcal{M}$ such that for every solution J' of I under $\mathcal{M}$, there exists a homomorphism (as defined in Section 3.2) $h : J \rightarrow J'$ with the property that $h(c) = c$ for every source value c that occurs in I. In the case where Σ_{st} contains only source-to-target tgds, it is known that chasing I with Σ_{st} produces a universal solution[2]. The next proposition asserts that a similar result holds when we chase a source instance I with SO tgds.

Proposition 5.2. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a schema mapping where Σ_{st} is an SO tgd. Then for every source instance I over $\mathbf{S}$, chasing $\langle I, \emptyset \rangle$ with Σ_{st} terminates in polynomial time (in the size of I) with a result $\langle I, J \rangle$. Moreover, J is a universal solution of I under $\mathcal{M}$.*

The above proposition has an immediate but important consequence in terms of query answering over the target schema. Concretely, let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a schema mapping and let q be a k-ary query over the target schema $\mathbf{T}$. Given a source instance I over $\mathbf{S}$, the set of the certain answers for q with respect to $\mathcal{M}$ and I, denoted by $\mathit{certain}_{\mathcal{M}}(q,I)$, is defined as the set of all k-tuples t of values from I such that, for every solution J of I under $\mathcal{M}$, we have that $t \in q(J)$. It was shown in [10] that if J is a universal solution of I under $\mathcal{M}$ and q is a union of conjunctive queries, then $\mathit{certain}_{\mathcal{M}}(q,I)$ equals $q(J)_{\downarrow}$, which is the result of evaluating q on J and then keeping only those tuples formed entirely of values from I. The equality $\mathit{certain}_{\mathcal{M}}(q,I) = q(J)_{\downarrow}$ holds for arbitrarily specified schema mappings $\mathcal{M}$. In particular, it holds for schema mappings specified by SO tgds. This fact, taken together with Proposition 5.2, implies the following result.

Corollary 5.3. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a schema mapping where Σ_{st} is an SO tgd. Let q be a union of conjunctive queries over the target schema $\mathbf{T}$. Then for every source instance I over $\mathbf{S}$, the set $\mathit{certain}_{\mathcal{M}}(q,I)$ can be computed in polynomial time (in the size of I).*

We point out an interesting contrast between the above result and one of the results on query answering given in [1]. There it was shown that when the source schema is described in terms of the target schema by means of arbitrary first-order views, computing the certain answers of conjunctive queries becomes undecidable. In contrast, our result shows that although the schema mappings that we consider go beyond first-order, computing the certain answers of unions of conjunctive queries remains in polynomial time, as it is with schema mappings specified by source-to-target tgds [10]. Thus, second-order tgds form a well-behaved fragment of second-order logic, since for the purposes of data exchange and query answering, second-order tgds behave similarly to source-to-target tgds.

6. Conclusions

We have introduced what we believe to be the right notion of the composition of two schema mappings. We have also introduced second-order tgds, which are a generalization of finite sets of source-to-target tgds. We show that second-order tgds are robust, in that the composition of mappings, each given by a second-order tgd, is also given by a second-order tgd. By contrast, when the mappings are each given by a

[2]This was proven in [10] for a more general setting that also includes target constraints.

finite set of source-to-target tgds, their composition may not be definable by even an infinite set of source-to-target tgds. We also show that second-order tgds possess good properties for data exchange.

Acknowledgments. The authors thank Moshe Y. Vardi for helpful suggestions.

7. REFERENCES

[1] S. Abiteboul and O. M. Duschka. Complexity of Answering Queries Using Materialized Views. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 254–263, 1998.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] C. Beeri and M. Y. Vardi. A Proof Procedure for Data Dependencies. *Journal of the Association for Computing Machinery (JACM)*, 31(4):718–741, 1984.

[4] C. Beeri and M. Y. Vardi. Formal Systems for Tuple and Equality Generating Dependencies. *SIAM J. on Computing*, 13(1):76–98, 1984.

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

[6] A. K. Chandra and D. Harel. Structure and Complexity of Relational Queries. *Journal of Computer and System Sciences*, 25(1):99–128, 1982.

[7] A. Dawar. A Restricted Second Order Logic for Finite Structures. *Information and Computation*, 143(2):154–174, 1998.

[8] R. Fagin. Generalized First-Order Spectra and Polynomial-Time Recognizable Sets. In R. M. Karp, editor, *Complexity of Computation, SIAM-AMS Proceedings, Vol. 7*, pages 43–73, 1974.

[9] R. Fagin. Horn Clauses and Database Dependencies. *Journal of the Association for Computing Machinery (JACM)*, 29(4):952–985, Oct. 1982.

[10] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. In *International Conference on Database Theory (ICDT)*, pages 207–224, 2003.

[11] R. Fagin, P. G. Kolaitis, and L. Popa. Data Exchange: Getting to the Core. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 90–101, 2003.

[12] T. Feder and M. Vardi. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory. *SIAM J. on Computing*, 28:57–104, 1998. Preliminary version in *Proc. 25th ACM Symp. on Theory of Computing*, May 1993, pp. 612–622.

[13] A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: Data Management Infrastructure for Semantic Web Applications. In *International World Wide Web Conference*, pages 556–567, 2003.

[14] N. Immerman. *Descriptive Complexity*. Springer, 1999.

[15] M. Lenzerini. Data Integration: A Theoretical Perspective. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 233–246, 2002.

[16] J. W. Lloyd. *Foundations of Logic Programming*. Springer, 1987.

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

[18] R. J. Miller, L. M. Haas, and M. Hernández. Schema Mapping as Query Discovery. In *International Conference on Very Large Data Bases (VLDB)*, pages 77–88, 2000.

[19] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernandez, and R. Fagin. Translating Web Data. In *International Conference on Very Large Data Bases (VLDB)*, pages 598–609, 2002.

[20] 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.