Composition of Mappings Given by Embedded Dependencies

Alan Nash[*]
Microsoft Research, USA
anash@math.ucsd.edu

Philip A. Bernstein
Microsoft Research, USA
philbe@microsoft.com

Sergey Melnik
Microsoft Research, USA
melnik@microsoft.com

ABSTRACT

Composition of mappings between schemas is essential to support schema evolution, data exchange, data integration, and other data management tasks. In many applications, mappings are given by embedded dependencies. In this paper, we study the issues involved in composing such mappings.

Our algorithms and results extend those of Fagin et al. [8] who studied composition of mappings given by several kinds of constraints. In particular, they proved that full source-to-target tuple-generating dependencies (tgds) are closed under composition, but embedded source-to-target tgds are not. They introduced a class of second-order constraints, *SO tgds*, that is closed under composition and has desirable properties for data exchange.

We study constraints that need not be source-to-target and we concentrate on obtaining (first-order) embedded dependencies. As part of this study, we also consider full dependencies and second-order constraints that arise from Skolemizing embedded dependencies. For each of the three classes of mappings that we study, we provide (a) an algorithm that attempts to compute the composition and (b) sufficient conditions on the input mappings that guarantee that the algorithm will succeed.

In addition, we give several negative results. In particular, we show that full dependencies are not closed under composition, and that second-order dependencies that are not limited to be source-to-target are not closed under restricted composition. Furthermore, we show that determining whether the composition can be given by these kinds of dependencies is undecidable.

1. INTRODUCTION

Many data management tasks, such as data translation, information integration, and database design require manipulation of database schemas and mappings between schemas. A *schema mapping* describes the relationship between the data instances of two schemas. Examples of schema mappings include SQL views, XSL transformations, integration constraints on schemas [11], and GLAV assertions in peer-to-peer systems [9]. *Mapping composition* refers to combining two mappings into a single one. If m_{12} is a mapping between schemas σ_1 and σ_2, and m_{23} is a mapping between schemas σ_2 and σ_3, then the composition $m_{12} \circ m_{23}$ of m_{12} and m_{23} is a mapping that captures the same relationship between σ_1 and σ_3 as the two mappings m_{12} and m_{23}.

[*]Current affiliation: University of California, San Diego

Composition of mappings generalizes composition of queries, which is implemented in most commercial database systems. It is known that composition of two first-order queries (a.k.a. view unfolding) is a first-order query. That is, first-order queries are closed under composition. The same holds for CQ and UCQ queries.

Query composition corresponds to composition of functional mappings. In a more general setting, the mappings to be composed may be non-functional and this makes the problem of composing them harder. For example, answering queries using views involves the composition of the query and the inverse of the view. But inverting a functional mapping often yields a non-functional mapping.

Composition was recently studied by Madhavan and Halevy [12] and by Fagin, Kolaitis, Popa, and Tan [8] (see "Related Work" below). Our work is closely related to that of Fagin et al. But whereas part of their focus is on mappings given by second-order constraints that are restricted to be source-to-target, we study constraints that need not be source-to-target and we try to obtain embedded dependencies as the result of our composition. We have several motivations for pursuing these directions, including the following:

- Allow schema constraints. If they are present, composition yields mappings that may include functional dependencies or inclusion dependencies that are not source-to-target.
- Support mappings that express equality of views and, more generally, symmetric data exchange and peer-to-peer systems.
- Obtain closure under most basic mapping operators; in particular, composition and inverse.
- Ensure that checking whether a pair of instances is in a mapping has low complexity.
- Be able to deploy composition mappings in existing database system products.

In this new context, we extend the results of Fagin et al. [8] in several directions. We study the composition of three related kinds of mappings:

1.	$\forall\mathrm{CQ}_0^=$-mappings	(given by full dependencies)
2.	$\forall\mathrm{CQ}^=$-mappings	(given by embedded dependencies)
3.	$\mathrm{Sk}\forall\mathrm{CQ}^=$-mappings	(given by second-order constraints)

and the corresponding mappings without equality. $\forall\mathrm{CQ}^=$-mappings subsume source-to-target tuple-generating dependencies (st-tgds), functional dependencies and inclusion dependencies, and can express view definitions. $\mathrm{Sk}\forall\mathrm{CQ}^=$ constraints subsume the second order (SO) tgds of [8], which in our terminology are source-to-target $\mathrm{Sk}\forall\mathrm{CQ}$ constraints.

The case of most interest to us is that of $\forall\mathrm{CQ}^=$-mappings. We show that one way to compose them is to:

1. Skolemize the $\forall\mathrm{CQ}^=$-mappings to get $\mathrm{Sk}\forall\mathrm{CQ}^=$-mappings,
2. find a finite $\mathrm{Sk}\forall\mathrm{CQ}^=$ axiomatization of all $\mathrm{Sk}\forall\mathrm{CQ}^=$ constraints that hold for the composition, and
3. de-Skolemize the finite $\mathrm{Sk}\forall\mathrm{CQ}^=$ axiomatization to get a $\forall\mathrm{CQ}^=$-mapping.

The first step is easy; the difficulties arise in Steps 2 and 3. In the work [8] of Fagin et al., the source-to-target restriction simplifies Step 2. And since the composition they consider is given by second-order constraints, Step 3 does not apply.

In Step 3, our goal is to obtain embedded dependencies, which requires eliminating second-order quantifiers.

In the case of $\mathrm{Sk}\forall\mathrm{CQ}$ we consider both *restricted* composition, in which we are not allowed to introduce new function symbols, and *unrestricted* composition, in which we are free to introduce new function symbols.

Observe that our mapping languages are capable of expressing within-schema constraints, such as inclusion and functional dependencies. In this paper, we assume that schema constraints are part of each mapping that mentions the schema. So we do not need to refer to them explicitly.

Contributions To list our contributions, we need to refer to many classes of mappings. To simplify the presentation, we use the following convention. Whenever we refer to a class of constraints without equality (for example, $\forall\mathrm{CQ}$) we imply that the result also holds for the corresponding class of constraints with equality (for example, $\forall\mathrm{CQ}^=$), unless otherwise stated. In contrast, whenever we refer to a class of constraints with equality, we do not imply the result holds for the corresponding class of constraints without equality. Furthermore our negative results do not require the use of constants and our positive results allow constants. Our contributions include the following.

Negative results:

1. We show that $\forall\mathrm{CQ}_0$-mappings are not closed under composition and that $\mathrm{Sk}\forall\mathrm{CQ}$-mappings are not closed under restricted composition (Theorem 1).
2. We show that the problem of determining whether the composition of two $\forall\mathrm{CQ}_0$-mappings is a $\forall\mathrm{CQ}_0$-mapping is undecidable (Theorem 2). This result carries over to $\forall\mathrm{CQ}$-mappings and to restricted composition of $\mathrm{Sk}\forall\mathrm{CQ}$-mappings.
3. Expressing the composition of two $\forall\mathrm{CQ}_0$-mappings may require $\forall\mathrm{CQ}_0$ constraints that are exponentially larger in size than the input mappings, even over fixed schemas (Example 4). This result carries over to $\forall\mathrm{CQ}$-mappings and $\mathrm{Sk}\forall\mathrm{CQ}$-mappings. We show that there are $\forall\mathrm{CQ}$-mappings that require exponentially larger expressions in $\mathrm{Sk}\forall\mathrm{CQ}$ than in $\forall\mathrm{CQ}$ (Theorem 8). That is, an exponential increase in size may occur independently at each of the Steps 2 and 3 of the algorithm outlined above.

Positive results:

4. We present necessary and sufficient (but uncomputable) syntactic conditions for composition of $\forall\mathrm{CQ}_0$ and $\mathrm{Sk}\forall\mathrm{CQ}$-mappings (Theorem 3 and Theorem 6).
5. We present algorithms that compute the composition of $\forall\mathrm{CQ}_0$ and $\mathrm{Sk}\forall\mathrm{CQ}$-mappings whenever they terminate (Corollary 1). These algorithms are very similar to each other and can be seen as an extension of the algorithm in [8] to handle mappings that are not restricted to being source-to-target.
6. We introduce exponential-time sufficient conditions for the algorithms above to terminate (Theorem 4).
7. We present an algorithm to compute the composition of $\forall\mathrm{CQ}$-mappings, which consists of three steps as outlined above: (1) Skolemize, (2) invoke the composition algorithm for $\mathrm{Sk}\forall\mathrm{CQ}$-mappings, and (3) de-Skolemize.
8. The de-Skolemization step may fail[1]. We show how to check in polynomial time whether it will succeed and whether its output will be exponentially larger than its input (Proposition 3).
9. We identify exponential-time recognizable subsets of $\forall\mathrm{CQ}_0$ and $\mathrm{Sk}\forall\mathrm{CQ}$ that are closed under composition and inverse and that include source-to-target constraints and constraints that express view definitions (Theorem 5). We do not provide such a subset of $\forall\mathrm{CQ}$ since the conditions on it would be very restrictive (to ensure that de-Skolemization succeeds).

Additional (minor) results for second-order constraints:

10. We show that $\mathrm{Sk}\forall\mathrm{CQ}$ requires special semantics (Example 8) and point out that under these semantics the safety condition is not needed to ensure domain-independence. Furthermore, unsafe source-to-target $\mathrm{Sk}\forall\mathrm{CQ}$ have $\mathbf{NP}$ data complexity, just as safe source-to-target $\mathrm{Sk}\forall\mathrm{CQ}$.
11. We identify two different kinds of composition of $\mathrm{Sk}\forall\mathrm{CQ}$-mappings: restricted and unrestricted.
12. We show that unsafe $\mathrm{Sk}\forall\mathrm{CQ}^=$-mappings are closed under unrestricted composition and present a linear-time composition algorithm (Theorem 9). This algorithm "cheats" by encoding the intermediate instance in the composition in an uninformative way, exploiting the special semantics.
13. We introduce another fragment of second-order logic, $\exists\mathrm{SO}\forall\mathrm{CQ}^=$. We show that every finite set of source-to-target constraints $\mathrm{Sk}\forall\mathrm{CQ}^=$ is equivalent, under the special semantics, to a finite set of source-to-target $\exists\mathrm{SO}\forall\mathrm{CQ}^=$ constraints under the usual semantics for $\exists\mathrm{SO}$ (Theorem 10).

Composition is only one of many useful operations on mappings. Bernstein et al. [2, 3] introduced a general framework, called *model management*, in which operators on schemas and mappings are used to simplify the development of metadata-intensive applications. The basic operators include domain, range, composition, and inverse.

14. We show that domain, range, and composition are closely related and can be reduced to each other (Proposition 5).

This is one reason why, in this context, composition and inverse are fundamental. The latter is easy if we use symmetric restrictions on the languages that define our mappings. Thus, $\forall\mathrm{CQ}_0^=$, $\forall\mathrm{CQ}^=$, and $\mathrm{Sk}\forall\mathrm{CQ}^=$ mappings have trivial inverse mappings. On the other hand, composition turns out to be very hard and is the primary subject of this paper.

Related Work A first semantics for composition of mappings was proposed in the pioneering work [12] by Madhavan and Halevy. Under their definition, m_{13} is a composition of m_{12} and m_{23} if the certain answers obtained by way of m_{13} for any query in a class of queries $\mathcal{L}$ against schema σ_3 are precisely those that can be obtained by using m_{23} and m_{12} in sequence. Notice that in their definition, composition depends on the query class $\mathcal{L}$. They focused on the relational case and considered mappings given by a certain class of tgds, which they call GLAV formulas.

[1]After all, $\mathrm{Sk}\forall\mathrm{CQ}$ has more expressive power than $\forall\mathrm{CQ}$ since, as shown in [8], it can encode $\mathbf{NP}$-complete problems.

Madhavan and Halevy showed that the result of composition may be an infinite set of formulas when the query language $\mathcal{L}$ is that of conjunctive queries, and proposed algorithms for the cases when composition can be done. Their definition has some disadvantages. In particular, the result of composition varies depending on the choice of the query language $\mathcal{L}$. Also, the definition is asymmetric. That is, it is based on queries over σ_3 and does not consider queries over σ_1.

An alternative, language-invariant semantics for mapping composition was proposed independently by Fagin et al. [8] and Melnik [13, Chapter 4]. They considered mappings as binary relations on instances of schemas and defined mapping composition as a set-theoretic composition of such binary relations. This semantics makes the result of mapping composition unique and does not depend on a specific logical formalism chosen for representing mappings and queries.

Fagin et al. [8] were the first to embark on a systematic investigation of mapping composition under these natural semantics. They presented many fundamental results; we survey only some of them here. First, they showed that *full* st-tgds are closed under composition, but that *embedded* st-tgds are not. To obtain closure in a more general setting, they introduced *SO tgds*, a second-order extension of st-tgds. These arise from Skolemizing embedded st-tgds. They showed that SO tgds are strictly more expressive than st-tgds and are closed under composition. This makes them a suitable mapping language for data exchange and query-rewriting scenarios [7, 16]. Further results in [8] are that composition of mappings given by st-tgds may give mappings undefinable in $L^{\omega}_{\infty\omega}$ and that composition of FO-mappings may give uncomputable mappings.

We extend the seminal work of Fagin et al. [8] in two principal directions: (1) we study constraints that need not be source-to-target and (2) we concentrate on obtaining embedded dependencies (which are first-order). Very roughly speaking, the main two challenges that we face involve recursion and de-Skolemization.

Outline This paper is structured as follows. In Section 2, we give a formal definition of mapping composition and specify the mapping languages that we consider. In Section 3 we present the deductive system used in our proofs. In Sections 4, 5, and 6 we study the composition of mappings given by, respectively, full, second-order, and embedded dependencies. In Section 7 we examine the semantics of $\mathrm{Sk}\forall\mathrm{CQ}^=$ and study unrestricted composition. In Section 8 we briefly consider how some other basic operators on mappings such as domain and range relate to composition. Section 9 is the conclusion. Due to space limitations, some proofs are only outlined and others are entirely omitted.

2. PRELIMINARIES

Queries. CQ is the set of conjunctive queries and CQ_0 is the set of queries with conjunction but no existential quantifiers.[2] $\mathrm{CQ}^=$ and $\mathrm{CQ}_0^=$ are the corresponding query classes that also include equality.

Constraints. A *constraint* is a boolean query. We denote sets of constraints with capital Greek letters and individual constraints with lowercase Greek letters. We will need to refer to the following kinds of constraints:

Name	Form
$\forall\mathrm{CQ}_0$	$\forall\bar{x}(\phi(\bar{x}) \to \psi(\bar{x}))$
$\forall\mathrm{CQ}$	$\forall\bar{x}(\phi(\bar{x}) \to \exists\bar{y}\,\xi(\bar{x},\bar{y}))$
$\mathrm{Sk}\forall\mathrm{CQ}$	$\exists\bar{f}\forall\bar{x}(\phi(\bar{x}),\chi(\bar{x}) \to \psi(\bar{x}))$
$\exists\mathrm{SO}\forall\mathrm{CQ}$	$\exists\bar{R}\forall\bar{x}(\phi(\bar{x}) \to \exists\bar{y}\,\xi(\bar{x},\bar{y}))$

where $\phi(\bar{x})$ is a conjunction of relational atoms with variables in $\bar{x}$, $\chi(\bar{x})$ is a set of equations between variables or between a variable and a term, $\psi(\bar{x})$ is a conjunction of relational atoms with variables in $\bar{x}$, and $\xi(\bar{x},\bar{y})$ is a conjunction of relational atoms with variables in $\bar{x}$ and $\bar{y}$. Terms are built from variables and functions in $\bar{f}$.

[2]We are not aware of any standard notation for this class.

We further require that every variable in $\bar{x}$ be *safe*. A variable is safe if it appears in a relational atom in $\phi(\bar{x})$ or alone on one side of an equation in χ where the other side is a term constructed from safe variables.

We define $\forall\mathrm{CQ}_0^=$, $\forall\mathrm{CQ}^=$, $\mathrm{Sk}\forall\mathrm{CQ}^=$, and $\exists\mathrm{SO}\forall\mathrm{CQ}^=$ the same way, except that we also allow equations on the right hand side of $\to$. The intuition for these names is that $\forall\mathrm{CQ}_0$ and $\forall\mathrm{CQ}$ are constraints given by inclusion of CQ_0 and CQ queries respectively. Skolemizing $\forall\mathrm{CQ}$ constraints gives $\mathrm{Sk}\forall\mathrm{CQ}$ constraints (but not all $\mathrm{Sk}\forall\mathrm{CQ}$ constraints correspond to Skolemized $\forall\mathrm{CQ}$ constraints). $\exists\mathrm{SO}\forall\mathrm{CQ}$ constraints are obtained by adding existential second-order quantification over relations to $\forall\mathrm{CQ}$.

The *Skolemization* of a formula is the result of applying the following replacement: for every ocurrence of a first-order existential quantifier v, remove $\exists v$ and replace the quantified variable wherever it appears in the scope of the quantifier with a new term of the form $f(\bar{x})$ where f is a new function symbol. In addition, introduce a second-order quantifier f just outside the scope of $\bar{x}$. For example, the Skolemization of $\forall xy(R(xy) \to \exists z\, S(xz))$ is $\forall xy(R(xy) \to S(y, f(xy)))$. The classes of constraints in $\mathrm{Sk}\forall\mathrm{CQ}$ we define here are in *unnested* form, so we would rewrite this as $\forall xy(R(xy), z = f(xy) \to S(yz))$. We overload notation and assume that $\mathrm{Sk}\forall\mathrm{CQ}$ also contains nested formulas of the kind above to simplify the presentation. The existential second-order quantification in $\mathrm{Sk}\forall\mathrm{CQ}$ and $\exists\mathrm{SO}\forall\mathrm{CQ}$ apply to a finite set of constraints, not necessarily just one (this is not easy to illustrate in the table above). Formally, we achieve this through a single sentence, which is the conjunction of the constraints in this finite set.

Given a source signature σ_S and a target signature σ_T disjoint from σ_S, a constraint is *source-to-target (ST)* if all the relational atoms in ϕ are over σ_S or $\bar{R}$ and all relational atoms in ψ or ξ are over σ_R or $\bar{R}$. The following table compares our names to some of the existing names for classes of constraints ("tgd" stands for *tuple-generating dependency*).

$\forall\mathrm{CQ}_0$	Full tgds [1]
$\forall\mathrm{CQ}_0^=$	Full dependencies [1]
$\forall\mathrm{CQ}$	Embedded tgds [1]
$\forall\mathrm{CQ}^=$	Embedded dependencies [1]
ST $\mathrm{Sk}\forall\mathrm{CQ}$	SO tgds [8]

A *signature* is a function from a set of relation symbols to positive integers which give their arities. In this paper, we use the terms signature and schema synonymously. Given a set of constraints Σ over the signature $\sigma_1 \cup \sigma_2$, $\Sigma|_{\sigma_1}$ is the set of constraints in Σ which contain only relation symbols in σ_1.

Mappings. A schema *mapping* is a binary relation on instances of database schemas. (An *instance* of a database schema is a database that conforms to that schema.) Given a class of constraints $\mathcal{L}$, we associate to every expression of the form $(\sigma_1, \sigma_2, \Sigma_{12})$ the mapping

$$\{\langle A, B\rangle : (A, B) \models \Sigma_{12}\}.$$

Here Σ_{12} is a finite subset of $\mathcal{L}$ over the signature $\sigma_1 \cup \sigma_2$, σ_1 is the *input (or source) signature*, σ_2 is the *output (or target) signature*, A is a database with signature σ_1, and B is a database with signature σ_2. To simplify the presentation, we require that σ_1 and σ_2 be disjoint (otherwise, we do some renaming). (A, B) is the database with signature $\sigma_1 \cup \sigma_2$ obtained from taking all the relations in A and B together. Its domain is the union of the domains of A and B.

We say that m is *given* by expression E if the mapping that corresponds to E is m. Furthermore, we say that m is an *$\mathcal{L}$-mapping* if m is given by an expression $(\sigma_1, \sigma_2, \Sigma_{12})$ where Σ_{12} is a finite subset of $\mathcal{L}$.

Composition. Given two mappings m_{12} and m_{23}, the composition $m_{12} \circ m_{23}$ is the unique mapping

$$\{\langle A, C\rangle : \exists B(\langle A, B\rangle \in m_{12} \wedge \langle B, C\rangle \in m_{23})\}.$$

We are concerned here with the following problem: given two expressions of the form specified above, find an expression for the composition. That is, we are concerned with the syntactic counterpart to the semantic operation defined above. We say that two $\mathcal{L}$-mappings given by the expressions $(\sigma_1, \sigma_2, \Sigma_{12})$ and $(\sigma_3, \sigma_4, \Sigma_{12})$ are *compatible* if $\sigma_2 = \sigma_3$ and $\sigma_1, \sigma_2, \sigma_4$ are pairwise disjoint. We only consider composition of compatible $\mathcal{L}$-mappings and therefore we have only a partial composition operation on expressions. We say that $\mathcal{L}$ is *closed under composition* if the composition of any two compatible $\mathcal{L}$-mappings is an $\mathcal{L}$-mapping. Closure under inverse is defined similarly.

Notice that under this definition, it is possible to get a language that is closed under both inverse and composition by considering the set of Sk$\forall$CQ-mappings which are either source-to-target (ST), target-to-source (TS), or both (B) (the mapping given by the empty set of constraints is the only one that is both). This is because the composition of ST with ST gives ST, of TS with TS gives TS, and any other combination gives B. Furthermore, the inverse of ST is TS, of TS is ST, and of B is B.

3. DEDUCTIONS

In some of the following results and algorithms, we will need to refer to some specific deductive system. Here we outline its basics; the details are not essential.

We write $\forall\mathrm{CQ}_0$ or Sk$\forall$CQ constraints augmented with constants as *rules* of the form $\phi(\bar{x}), \chi(\bar{x}) \rightarrow \psi(\bar{x}))$ leaving the second-order quantifiers over functions $\exists\bar{f}$ and the first-order universal quantifiers $\forall\bar{x}$ implicit. We call $\phi(\bar{x}), \chi(\bar{x})$ the *premise* and $\psi(\bar{x})$ the *conclusion*. (Similarly for $\forall\mathrm{CQ}_0^=$ or Sk$\forall\mathrm{CQ}^=$ constraints.) If the premise is empty, we write only the conclusion. We call rules of the form $\psi(\bar{c})$, where $\bar{c}$ is a tuple of constants, *facts*. In most cases we will assume, without loss of generality, that our rules have a single atom in the conclusion since every rule with k atoms in the conclusion can always be rewritten as k rules each with a single atom in the conclusion.

DEFINITION 1. A *deduction* from rules Σ is a sequence of *rules*, each obtained in one of three ways:

1. by copying a rule from Σ,
2. by adding atoms to the premise of a rule and/or renaming variables in a rule appearing earlier in the sequence, or
3. by applying resolution on two rules appearing earlier in the sequence.

We call such rules *axiom* rules, *expand/rename* rules, and *resolution* rules respectively. We say that a deduction *has length* n if it consists of n lines.

A rule r obtained by expand/rename from rule r' may have additional atoms in the premise, may have variables replaced (consistently) by arbitrary terms, may have equations of the form $v = t$ between a variable v and a term t removed whenever v does not appear elsewhere in the rule, and may have replacements in the conclusion consistent with equations in the premise. A rule ξ is a *variant* of ξ' if ξ can be deduced from ξ' without using resolution and conversely. Since each rule has a single atom in the conclusion, a rule r obtained by resolution from rules p, q consists of the conclusion of q and the premises in p and q that do not appear in the conclusion of p.

To illustrate the deductions introduced in Definition 1, consider the following examples: The rule $R(xy), z = f(xy) \rightarrow S(xz)$ is a valid result of applying expand/rename to $R(uv) \rightarrow S(uf(uv))$. The rule $R(xy), S(yz) \rightarrow S(xz)$ is the result of applying resolution to rules $R(xy) \rightarrow S(xy)$ and $S(xy), S(yz) \rightarrow S(xz)$.

We call a resolution step a σ_2-resolution if it involves the elimination of an atom with a relation symbol from σ_2. In the example above, if σ_2 contains S, then we have a σ_2-resolution.

We annotate our deductions by numbering the rules in them in ascending order and by adding annotations to each line indicating how that line was obtained. It is enough to annotate a resolution rule with just two numbers and an expand/rename rule with a single number and a variable assignment. Axiom rules are indicated through a lack of any other annotation. A variable assignment is a list of items of the form $x := y$ where x is a variable and y is a term.

EXAMPLE 1. Given

- $\Delta := \{R(1, 1)\}$ and
- $\Sigma := \{R(xy) \rightarrow S(xy),\ S(zz) \rightarrow T(zz)\}$,

the following is a valid deduction from $\Sigma \cup \Delta$:

1.	$R(1, 1)$	
2.	$R(xy) \rightarrow S(xy)$	
3.	$R(1, 1) \rightarrow S(1, 1)$	[2] $x := 1, y := 1$
4.	$S(1, 1)$	[1,3]
5.	$S(zz) \rightarrow T(zz)$	
6.	$S(1, 1) \rightarrow T(1, 1)$	[5] $z := 1$
7.	$T(1, 1)$	[4,6]

Here rules 1, 2, and 5 are axioms, 3 and 6 are expand/rename, and 4 and 7 are resolution. □

We call a sequence of at most two rename-only steps followed by a resolution step on the results of these steps a *rename-resolution*. In the example above, 4 is obtained by rename-resolution from 1 and 2 and 7 is obtain by rename-resolution from 4 and 5. A σ_2-rename-resolution is a rename-resolution where the resolution step is a σ_2-resolution.

If there is a deduction from a set of constraints Σ where the last line contains a constraint ξ, we say that ξ is deduced from Σ which we write $\Sigma \vdash \xi$. We write $\Sigma \vdash \Sigma'$ in case $\Sigma \vdash \xi$ for every $\xi \in \Sigma'$. The *$\mathcal{L}$-deductive closure* of Σ is

$$\mathrm{DC}(\mathcal{L}, \Sigma) := \{\xi \in \mathcal{L} : \Sigma \vdash \xi\}.$$

We write $\mathrm{DC}(\Sigma)$ when $\mathcal{L}$ is clear from the context.

We write $D \models \Sigma$ if all constraints in Σ are true in D. We write $\Sigma \models \Sigma'$ if, for all instances D, $D \models \Sigma$ implies $D \models \Sigma'$. It is easy to check that if $\Sigma \vdash \Sigma'$ then also $\Sigma \models \Sigma'$; i.e., the deductive system is *sound*.

Chase. Here we define a modified chase procedure which is needed in the proof of several results below.

DEFINITION 2. Given an instance D, the result of *chasing* D with constraints $\Sigma \subseteq \mathrm{Sk}\forall\mathrm{CQ}^=$ and the set of Skolem functions F, which we denote $\mathrm{chase}(D, \Sigma, F)$ is the database D'' obtained from

$$D' := \{R_i(\bar{c}) : \Sigma \cup \Delta \cup \Phi \vdash R_i(\bar{c})\}$$

where

- $\bar{c}$ is a tuple of constants,
- Δ is the set of facts given by D:

$$\Delta := \{R_i(\bar{c}) : D \models R_i(\bar{c})\}$$

- and Φ is the set of facts given by F:

$$\Phi := \{f(\bar{c}) = a : f \in F, f(\bar{c}) = a\}$$

as follows. Define

$$c_0 \equiv c_1 \text{ iff } \Sigma \cup \Delta \cup \Phi \vdash c_0 = c_1.$$

Now to obtain D'' from D', pick one constant c_0 from every equivalence class and replace every constant in that equivalence class with c_0. That is, $D'' := D'/\equiv$. All functions in F are required to have the same domain which includes D. If they have finite domain, then $\mathrm{chase}(D, \Sigma, F)$ is finite.

This definition is a variation on the usual definition, where the functions F are constructed during the chase process.

The important property we need of this modified chase is:

PROPOSITION 1. $\mathrm{chase}(D, \Sigma, F) \models \Sigma$.

4. FULL DEPENDENCIES

We start by studying composition of $\forall\mathrm{CQ}_0^=$-mappings; that is, mappings given by full dependencies. All results in this section apply to both $\forall\mathrm{CQ}_0^=$ and $\forall\mathrm{CQ}_0$-mappings (mappings given by full tgds). We will see that the techniques introduced to handle these cases can be extended to handle $\mathrm{Sk}\forall\mathrm{CQ}^=$ and $\forall\mathrm{CQ}^=$-mappings. We first show that $\forall\mathrm{CQ}_0$ is not closed under composition (Theorem 1) and, furthermore, that determining whether the composition of two $\forall\mathrm{CQ}_0^=$-mappings is a $\forall\mathrm{CQ}_0^=$-mapping is undecidable (Theorem 2). Then we give necessary and sufficient, non-computable conditions for the composition of two $\forall\mathrm{CQ}_0$-mappings to be a $\forall\mathrm{CQ}_0$-mapping (Theorem 3). Theorem 3 suggests the following algorithm to compute the composition of two $\forall\mathrm{CQ}_0^=$-mappings given by $(\sigma_1, \sigma_2, \Sigma_{12})$ and $(\sigma_2, \sigma_3, \Sigma_{23})$:

Procedure $\forall\mathrm{CQ}_0^=$COMPOSE$(\Sigma_{12}, \Sigma_{23})$

```
Set $\Sigma := \Sigma_{12} \cup \Sigma_{23}$
Repeat
  Set $\Sigma' := \emptyset$
  For every pair $\phi, \psi \in \Sigma$
      For every way in which $\phi, \psi$ can be $\sigma_2$-rename-resolved
      to yield $\xi$ and if there is no variant of $\xi$ in $\Sigma$
          set $\Sigma' := \Sigma' \cup \{\xi\}$
  Set $\Sigma := \Sigma \cup \Sigma'$
Until $\Sigma' = \emptyset$
Return $\Sigma_{13} := \Sigma|_{\sigma_{13}}$
```

$\forall\mathrm{CQ}_0^=$COMPOSE works correctly on $\forall\mathrm{CQ}_0^=$-mappings satisfying the conditions of Theorem 4, which can be checked in exponential time (see also Corollary 1). It is clear that the obstacle to composition is recursion, yet recursion is not always a problem (Example 2). We also define good-$\forall\mathrm{CQ}_0^=$, a subset of $\forall\mathrm{CQ}_0^=$ recognizable in exponential time, which is closed under composition (Theorem 5). It is tempting to conjecture that total and surjective $\forall\mathrm{CQ}_0^=$-mappings (or $\forall\mathrm{CQ}_0$-mappings) are closed under composition. Such mappings do not imply any within-schema constraints and have desirable properties for view integration (they are called conflict-free in [4]). We show that this conjecture does not hold (Example 5).

THEOREM 1. *There are $\forall\mathrm{CQ}_0$-mappings whose composition is not an FO-mapping. In particular, $\forall\mathrm{CQ}_0$ is not closed under composition.*

PROOF. Consider the $\forall\mathrm{CQ}_0$-mappings m_{12} and m_{23} given by $(\sigma_1, \sigma_2, \Sigma_{12})$ and $(\sigma_2, \sigma_3, \Sigma_{23})$ where

Σ_{12} is	$R(xy)$	$\to$	$S(xy)$
	$S(xy), S(yz)$	$\to$	$S(xz)$
Σ_{23} is	$S(xy)$	$\to$	$T(xy)$

and where $\sigma_1 = \{R\}$, $\sigma_2 = \{S\}$, and $\sigma_3 = \{T\}$. Together, these constraints say that $R \subseteq S \subseteq T$ and that S is transitively closed. The composition $m_{12} \circ m_{23}$ is the set of all pairs (R, T) such that $\mathrm{tc}(R) \subseteq T$. Intuitively, the $\forall\mathrm{CQ}_0^=$-constraints which express the composition are constraints of the form

$$R(x, v_1), R(v_1, v_2), \ldots, R(v_{i-1}, v_i), R(v_i, y) \to T(x, y)$$

but no finite set of them expresses $\mathrm{tc}(R) \subseteq T$.

In fact, the composition $m_{12} \circ m_{23}$ is not even expressible in FO, since if we had an FO sentence ϕ such that

$$\langle R, T\rangle \in m_{12} \circ m_{23} \text{ iff } (R, T) \models \phi$$

we could create an FO formula $\psi(x, y)$ obtained by replacing every occurrence of $T(u, v)$ in ϕ with $x \neq u \vee y \neq v$. Then given a domain D with $R \subseteq D^2$ we would have $R \models \psi[a, b]$ iff $(R, D^2 - \langle a, b\rangle) \models \phi$ iff $\mathrm{tc}(R) \subseteq D^2 - \langle a, b\rangle$ iff $\langle a, b\rangle \notin \mathrm{tc}(R)$ and therefore $\forall x \forall y \neg\psi(x, y)$ would say that R is a connected graph, contradicting the fact that this can not be expressed in FO (see, e.g., Example 2.3.8 in [6]). ☐

THEOREM 2. *Checking whether the composition of two $\forall\mathrm{CQ}_0$-mapping is a $\forall\mathrm{CQ}_0$-mappings is undecidable (in fact, coRE-hard).*

PROOF. (Outline) We reduce Post's correspondence problem (PCP)—known to be undecidable (see, e.g., [15])—to the problem of deciding whether $m_{12} \circ m_{23}$ is a $\forall\mathrm{CQ}_0$-mapping where m_{12} and m_{23} are $\forall\mathrm{CQ}_0$ mappings. This reduction is partially inspired by an undecidability proof by Christoph Koch (Theorem 3.1 in [10]). Given a PCP problem, we define m_{23} so that there is a solution to the PCP problem iff $m_{12} \circ m_{23}$ is not a $\forall\mathrm{CQ}_0$-mapping (m_{12} does not depend on the PCP problem). In fact, we will have that the composition is the mapping given by the empty set of constraints in case the PCP problem has no solution and a mapping that requires infinitely many $\forall\mathrm{CQ}_0$constraints in case the PCP problem has a solution.

The $\forall\mathrm{CQ}_0$-mappings m_{12}, m_{23} are given by $(\sigma_1, \sigma_2, \Sigma_{12})$ and $(\sigma_2, \sigma_3, \Sigma_{23})$ where

- $\sigma_1 = \{A, B, O, Z\}$,
- $\sigma_2 = \{\hat{A}, \hat{B}, \hat{O}, \hat{Z}\}$, and
- $\sigma_3 = \{T\}$

with A, B unary and all other relations binary and with Σ_{12} and Σ_{23} as described below.

We write $x0011y$ for the set of atoms

$$A(x), Z(xa), Z(ab), O(bc), O(cy), B(y)$$

over σ_1 which corresponds to a path from A to B through Z, O (Z stands for "zero" and O stands for "one"). Similarly, we write ${}^{x}_{x'}0011^{y}_{y'}$ for the set of atoms

$$\hat{A}(xx'), \hat{Z}(xa), \hat{Z}(ab), \hat{O}(bc), \hat{O}(cy), \hat{Z}(x'a'), \hat{Z}(a'b'), \hat{O}(b'c'), \hat{O}(c'y'), \hat{B}(yy') \quad (1)$$

over σ_2. Σ_{12} contains the constraint

$$A(x), Z(xy), A(x'), Z(x'y') \to \hat{A}(xx'), \hat{Z}(xy), \hat{Z}(x'y')$$

(let us call this an AZ pattern) and all corresponding constraints for the patterns $AO, ZZ, ZO, OZ, OO, ZB, OB$. These constraints allow us to deduce $xSy, x'Sy' \to {}^{x}_{x'}S^{y}_{y'}$ for any string S.

We encode each PCP "domino" by a constraint in Σ_{23} as follows. For example, we encode the domino 0011/011 as the constraint

$$\hat{A}(xx'), \hat{Z}(xa), \hat{Z}(ab), \hat{O}(bc), \hat{O}(cy), \hat{Z}(x'a'), \hat{O}(a'b'), \hat{O}(b'y') \to \hat{A}(xx'), \hat{A}(yy') \quad (2)$$

Finally, we add to Σ_{23} the constraint

$$\hat{A}(xx'), \hat{A}(yy'), \hat{B}(yy') \to T(xy).$$

If the PCP problem has a solution for the string S over $\{0,1\}^*$. Then we can only deduce infinitely many constraints of the form

$$xS^ky, x'S^ky' \to T(xy)$$

and their variants, where S^k is the string S repeated k times. Notice that these constraints are over $\sigma_1 \cup \sigma_3$ and that none of them can be obtained from any other.

Conversely, any constraint over $\sigma_1 \cup \sigma_2$ that can be deduced from $\Sigma_{12} \cup \Sigma_{23}$ must be of the form $xSy, x'Sy' \to T(xy)$ (or a variant) and it encodes a solution to the PCP problem. $\square$

THEOREM 3. *If the $\forall\mathrm{CQ}_0^=$-mappings m_{12}, m_{23} are given by $(\sigma_1, \sigma_2, \Sigma_{12})$ and $(\sigma_2, \sigma_3, \Sigma_{23})$ with $\Sigma_{123} := \Sigma_{12} \cup \Sigma_{23}$ and $\sigma_{13} = \sigma_1 \cup \sigma_3$, then the following are equivalent*

1. *There is a finite set of constraints $\Sigma_{13} \subseteq \forall\mathrm{CQ}_0^=$ over the signature σ_{13} such that $m := m_{12} \circ m_{23}$ is given by $(\sigma_1, \sigma_3, \Sigma_{13})$.*

2. *There is a finite set of constraints $\Sigma_{13} \subseteq \forall\mathrm{CQ}_0^=$ over the signature σ_{13} such that*
$$\mathrm{DC}(\forall\mathrm{CQ}_0^=, \Sigma_{123})|_{\sigma_{13}} = \mathrm{DC}(\forall\mathrm{CQ}_0^=, \Sigma_{13}).$$

3. *There is k such that for every ξ over σ_{13} satisfying $\Sigma_{123} \vdash \xi$ there is a deduction of ξ from Σ_{123} using at most k σ_2-resolutions.*

PROOF. The proof uses Lemmas 1 and 2 below. First we show the equivalence of (1) and (2) then we show the equivalence of (2) and (3).

Assume (2) holds. Then $\langle A, C\rangle \in m_{12} \cdot m_{23}$

iff	$\exists B\,(A,B,C) \models \Sigma_{123}$	(by definition of $\cdot$)
iff	$(A,C) \models \mathrm{DC}(\forall\mathrm{CQ}_0^=, \Sigma_{123})|_{\sigma_{13}}$	(by Lemma 1)
iff	$(A,C) \models \mathrm{DC}(\forall\mathrm{CQ}_0^=, \Sigma_{13})$	(since (2) holds)
iff	$(A,C) \models \Sigma_{13}$.	(since DC is sound)

This shows that (1) holds.

Conversely, assume (1) holds. Then
$(A,C) \models \mathrm{DC}(\forall\mathrm{CQ}_0^=, \Sigma_{123})|_{\sigma_{13}}$

iff	$\exists B\,(A,B,C) \models \Sigma_{123}$	(by Lemma 1)
iff	$\langle A,C\rangle \in m_{12} \cdot m_{23}$	(by definition of $\cdot$)
iff	$(A,C) \models \Sigma_{13}$	(since (1) holds)
iff	$(A,C) \models \mathrm{DC}(\forall\mathrm{CQ}_0^=, \Sigma_{13})$.	(since DC is sound)

This shows that (2) holds.

Now assume (3) holds. Set Σ to the set of all constraints in $\mathrm{DC}(\forall\mathrm{CQ}_0^=, \Sigma_{123})|_{\sigma_{13}}$ which can be deduced using at most k σ_2-resolutions and no other resolutions. Clearly, every constraint in Σ can be obtained by expand/rename from a finite subset $\Sigma_{13} \subseteq \Sigma$. We show that (2) holds.

Assume that there is a deduction γ witnessing $\Sigma_{123} \vdash \xi$. Since (3) holds, we can assume that γ has $m' \leq m$ σ_2-resolutions. By Lemma 2 there is a deduction γ' witnessing $\Sigma_{123} \vdash \xi$ also with m' σ_2-resolutions and with all of them occurring before any other resolutions.

Since the last line of γ' does not contain any symbols from σ_2 we can assume that γ' does not contain any lines containing symbols from σ_2 after the last σ_2-resolution.

Break γ' into two parts γ'_1 the initial segment of γ up to and including the last σ_2-resolution and γ'_2 the remainder of γ'. Every constraint ψ in γ'_1 must be in Σ, by definition of Σ and therefore we must have $\Sigma_{13} \vdash \psi$. Since every constraint ψ in γ'_2 does not contain any symbols from σ_2 and since $\Sigma_{123}|_{\sigma_{13}} \subseteq \Sigma_{13}$, we also have $\Sigma_{13} \vdash \psi$. Therefore, $\Sigma_{13} \vdash \xi$ as desired.

Conversely, assume (2) holds. Take k to be the total number of σ_2-resolutions needed to deduce every $\psi \in \Sigma_{13}$ from Σ_{123}. Assume $\Sigma_{123} \vdash \xi$. Then there is a deduction γ witnessing $\Sigma_{13} \vdash \xi$. Clearly, γ has no σ_2-resolutions. From γ, we obtain γ' witnessing $\Sigma_{123} \vdash \xi$ by appending to γ a deduction of every constraint in Σ_{13} and by replacing every line where an axiom from Σ_{13} is used by a vacuous expand/rename of the line where the deduction of that axiom ends. Clearly, γ' has exactly k σ_2-resolutions as desired. This shows that (3) holds. $\square$

LEMMA 1. *Under the hypotheses of Theorem 3, the following are equivalent:*

1. $(A,C) \models \mathrm{DC}(\forall\mathrm{CQ}_0^=, \Sigma_{123})|_{\sigma_{13}}$.
2. $\exists B\,(A,B,C) \models \Sigma_{123}$.

PROOF. Assume $(A,B,C) \models \Sigma_{123}$ for some B. Then $(A,B,C) \models \mathrm{DC}(\forall\mathrm{CQ}_0^=, \Sigma_{123})$ (by soundness) and therefore $(A,C) \models \mathrm{DC}(\forall\mathrm{CQ}_0^=, \Sigma_{123})|_{\sigma_{13}}$ since B is not mentioned in $\mathrm{DC}(\forall\mathrm{CQ}_0^=, \Sigma_{123})|_{\sigma_{13}}$.

Conversely, assume $(A,C) \models \mathrm{DC}(\forall\mathrm{CQ}_0^=, \Sigma_{123})|_{\sigma_{13}}$. We set $(A', B, C') := \mathrm{chase}((A, \emptyset, C), \Sigma_{123})$.

If the chase terminates and $A = A'$ and $C = C'$, then we have $(A,B,C) \models \Sigma_{123}$ by Proposition 1 which implies $(A,B) \models \Sigma_{12}$ and $(B,C) \models \Sigma_{23}$, as desired.

It is clear that the chase terminates since no new constants are introduced. Now assume, to get a contradiction, that $A \neq A'$ or $C \neq C'$. Set Δ_{AC} to the set of facts given by A and C. Then we must have

$$\Sigma_{123} \cup \Delta_{AC} \vdash R(\bar{c})$$

where R is a relation in A or C not containing $\bar{c}$ or

$$\Sigma_{123} \cup \Delta_{AC} \vdash c_0 = c_1$$

where c_0, c_1 are distinct constants in A or C.

We consider the former case; the latter is similar. If $\Sigma_{123} \cup \Delta_{AC} \vdash R(\bar{c})$ then by Proposition 2 below there exists ξ such that $\Sigma_{123} \vdash \xi$ and $\Delta_{AC}, \xi \vdash R(\bar{c})$. Since $(A,C) \models \Delta_{AC}$, it follows that $(A,C) \not\models \xi$, contradicting $(A,C) \models \mathrm{DC}(\forall\mathrm{CQ}_0^=, \Sigma_{123})|_{\sigma_{13}}$. $\square$

PROPOSITION 2. *If Δ is a set of facts in $\mathcal{L}$, and $\Sigma \cup \{\phi\} \subseteq \mathcal{L}$ for $\mathcal{L} \in \{\forall\mathrm{CQ}_0^=, \mathrm{Sk}\forall\mathrm{CQ}^=\}$, then the following are equivalent:*

1. *$\Sigma \cup \Delta \vdash \phi$.*
2. *There is ξ such that $\Sigma \vdash \xi$ and $\Delta, \xi \vdash \phi$.*

LEMMA 2. *Under the hypotheses of Theorem 3, if there is a deduction γ witnessing $\Sigma_{123} \vdash \xi$ with at most k σ_2-resolutions, then there is γ' witnessing $\Sigma_{123} \vdash \xi$ with at most k σ_2-resolutions and where furthermore all σ_2-resolutions occur before all other resolutions.*

COROLLARY 1. *Under the hypotheses of Theorem 3, $\forall CQ_0^=$* COMPOSE*(Σ_{12}, Σ_{23}), whenever it terminates, yields Σ_{13} such that $m_{12} \circ m_{23}$ is given by $(\sigma_1, \sigma_3, \Sigma_{13})$.*

Notice that after the k-th iteration of the main loop, Σ will contain a variant of every constraint that can be deduced using at most k σ_2-resolution steps. The constraints in the proof of Theorem 1 fail to satisfy (3) of Theorem 3 and therefore $\forall CQ_0^=$COMPOSE(Σ_{12}, Σ_{23}) will not terminate when Σ_{12} and Σ_{23} are as in the proof of Theorem 1 for input. In contrast, $\forall CQ_0^=$COMPOSE(Σ_{12}, Σ_{23}) will terminate on Σ_{12}, Σ_{23} from the example below, which does satisfy (3) of Theorem 3, so recursion is not always bad.

EXAMPLE 2. Consider the $\forall CQ_0$-mappings m_{12} and m_{23} given by $(\sigma_1, \sigma_2, \Sigma_{12})$ and $(\sigma_2, \sigma_3, \Sigma_{23})$ where

Σ_{12} is	$R(xy)$	$\rightarrow$	$S(xy)$
	$S(xy), S(yz)$	$\rightarrow$	$R(xz)$
Σ_{23} is	$S(xy)$	$\rightarrow$	$T(xy)$

and where $\sigma_1 = \{R\}$, $\sigma_2 = \{S\}$, and $\sigma_3 = \{T\}$. Together, these constraints say that $R \subseteq S \subseteq T$, and that R and S are transitively closed (because the constraints

$$\begin{array}{rcl} S(xy), S(yz) & \rightarrow & S(xz) \\ R(xy), R(yz) & \rightarrow & R(xz) \end{array}$$

can be deduced from Σ_{12}). The constraints

$$\begin{array}{rcl} R(xy), R(yz) & \rightarrow & R(xz) \\ R(xy) & \rightarrow & T(xy) \end{array}$$

express exactly the composition $m_{12} \circ m_{23}$, and are exactly those found by $\forall CQ_0^=$COMPOSE(Σ_{12}, Σ_{23}). $\square$

The coRE-hardness from Theorem 2 implies that algorithm $\forall CQ_0^=$COMPOSE may not terminate even when the composition is a $\forall CQ_0^=$-mapping. This happens, for example, in the following case.

EXAMPLE 3. Consider the $\forall CQ_0$-mappings m_{12} and m_{23} given by $(\sigma_1, \sigma_2, \Sigma_{12})$ and $(\sigma_2, \sigma_3, \Sigma_{23})$ where

Σ_{12} is	$R(xy)$	$\rightarrow$	$S(xy)$
	$R(xy), R(yz)$	$\rightarrow$	$R(xz)$
	$S(xy), S(yz)$	$\rightarrow$	$S(xz)$
Σ_{23} is	$S(xy)$	$\rightarrow$	$T(xy)$

and where $\sigma_1 = \{R\}$, $\sigma_2 = \{S\}$, and $\sigma_3 = \{T\}$.
The constraints

$$\begin{array}{rcl} R(xy), R(yz) & \rightarrow & R(xz) \\ R(xy) & \rightarrow & T(xy) \end{array}$$

express exactly the composition $m_{12} \circ m_{23}$, but algorithm $\forall CQ_0^=$COMPOSE will never terminate since it will deduce at least the infinitely many constraints it would deduce in the proof of Theorem 1. This is because Σ_{12} here includes all the constraints in Σ_{12} there. $\square$

Even if the algorithm terminates, it may produce a result which is exponential in the size of the input mappings. This is unavoidable, as the following example shows.[3]

[3]This is essentially a result on query unfolding [14]. Lucian Popa first brought this to our attention through an example that required a varying schema.

EXAMPLE 4. There is a $\forall CQ_0$-mapping m_{12} and a sequence of $\forall CQ_0$-mappings m_{23}^k given by Σ_{12} and Σ_{23}^k over fixed signatures $\sigma_1 = \{R\}$, $\sigma_2 = \{S\}$, and $\sigma_3 = \{T\}$ where R, S, and T are binary relations such that the composition $m_{12} \circ m_{23}^k$ grows exponentially in the size of Σ_{23}^k.

The mapping m_{12} given by:

$R(xy), R(yx)$	$\rightarrow$	$S(xy)$
$R(xy), R(xx)$	$\rightarrow$	$S(xy)$

and the family of mappings m_{23}^k given by Σ_{23}^k which contains the single constraint

$S(xu_1), S(u_1u_2), ..., S(u_{k-1}y)$	$\rightarrow$	$T(xy)$

saying that if there is a path of length k in S then there is an edge in T. $\square$

The following conditions are sufficient for algorithm $\forall CQ_0^=$COMPOSE to terminate. On the other hand, Example 5 below illustrates a case where this condition is violated.

THEOREM 4. *Under the hypotheses of Theorem 3, if no constraint of the form $\phi(\bar{z}), S(\bar{y}) \rightarrow S(\bar{x})$ can be deduced from Σ_{123} using only σ_2-rename-resolutions, such that*

1. *$\phi(\bar{z})$ is a conjunction of atoms over σ_{123},*
2. *there is no atom $S(\bar{w})$ in $\phi(\bar{z})$ with $\bar{x} \subseteq \bar{w}$,*
3. *$\{\bar{x}\} \nsubseteq \{\bar{y}\}$, and*
4. *S is a relation symbol in σ_2*

*then $\forall CQ_0^=$*COMPOSE*(Σ_{12}, Σ_{23}) terminates and therefore $m_{12} \circ m_{23}$ is a $\forall CQ_0^=$-mapping. Furthermore, these conditions can be verified in exponential time in the size of $\Sigma_{12} \cup \Sigma_{23}$.*

PROOF. We omit the proof of the main part due to space constraints. The conditions can be checked in exponential time as follows. Run $k_{\sigma_2} - 1$ iterations of the main loop of $\forall CQ_0^=$COMPOSE(Σ_{12}, Σ_{23}) where k_{σ_2} is the number of relation symbols in σ_2 and check whether a constraint of the form given below appears in Σ. $\square$

DEFINITION 3. A $\forall CQ_0^=$-mapping is a *good-$\forall CQ_0^=$-mapping* if it is given by $(\sigma_1, \sigma_2, \Sigma_{12})$ and no constraint of the form $\phi(\bar{z}), S(\bar{y}) \rightarrow S'(\bar{x})$ where

1. $\phi(\bar{z})$ is a conjunction of atoms over $\sigma_1 \cup \sigma_2$,
2. there is no atom $S(\bar{w})$ in $\phi(\bar{z})$ with $\bar{x} \subseteq \bar{w}$,
3. $\{\bar{x}\} \nsubseteq \{\bar{y}\}$, and
4. S and S' are both relation symbols in σ_1 or both in σ_2

can be deduced from Σ_{12} using only σ_1-rename-resolutions or only σ_2-rename-resolutions We define good-$\forall CQ_0$ similarly.

THEOREM 5. *good-$\forall CQ_0^=$ and good-$\forall CQ_0$ are closed under composition and inverse.*

We examined many other subsets of $\forall CQ_0^=$ for closure under composition and inverse, but were unable to find more natural conditions of similarly wide applicability. Since source-to-target $\forall CQ_0^=$-constraints are total and surjective, it is natural to wonder whether the set of all total and surjective $\forall CQ_0^=$-mappings is closed under composition. The following example shows it is not.

EXAMPLE 5. Consider the $\forall CQ_0$-mappings m_{12} and m_{23} given by $(\sigma_1, \sigma_2, \Sigma_{12})$ and $(\sigma_2, \sigma_3, \Sigma_{23})$ where

Σ_{12} is	$R(xy)$	$\to$	$S(xy)$
	$R(xy), S(yz)$	$\to$	$S(xz)$
Σ_{23} is	$S(xy)$	$\to$	$T(xy)$

and where $\sigma_1 = \{R\}$, $\sigma_2 = \{S\}$, and $\sigma_3 = \{T\}$. Here m_{12} and m_{23} are total and surjective and their composition says that $\mathrm{tc}(R) \subseteq T$, which we have seen in the proof of Theorem 1 is not expressible even in FO. □

5. SECOND-ORDER DEPENDENCIES

In order to handle existential quantifiers in a $\forall\mathrm{CQ}^=$-mapping, we will first convert the $\forall\mathrm{CQ}^=$ constraints which specify the mapping into $\mathrm{Sk}\forall\mathrm{CQ}^=$ constraints (by Skolemizing) and this will give us $\mathrm{Sk}\forall\mathrm{CQ}^=$-mappings. Therefore, in this section we focus on the composition of $\mathrm{Sk}\forall\mathrm{CQ}^=$-mappings; in the next section we consider how to convert $\mathrm{Sk}\forall\mathrm{CQ}^=$-mappings back to $\forall\mathrm{CQ}^=$-mappings. There are two cases of composition to consider. *Unrestricted* composition, in which we are allowed to introduce additional existentially-quantified functions in order to express the composition and *restricted* composition in which we are only allowed to use function symbols from the input mappings. In this section we concentrate on restricted composition. In Section 7 we examine unrestricted composition. $\mathrm{Sk}\forall\mathrm{CQ}^=$ constraints require special semantics, which we also examine in Section 7. All results in this section apply to both $\mathrm{Sk}\forall\mathrm{CQ}^=$ and $\mathrm{Sk}\forall\mathrm{CQ}$-mappings (which correspond to SO tgds which are not restricted to being source-to-target).

Theorems 1 and 2 from the previous section show that $\mathrm{Sk}\forall\mathrm{CQ}$ is not closed under restricted composition and that determining whether the restricted composition of two $\mathrm{Sk}\forall\mathrm{CQ}$-mappings is a $\mathrm{Sk}\forall\mathrm{CQ}$-mapping is undecidable. As in the case of $\forall\mathrm{CQ}_0^=$-mappings, we give necessary and sufficient, non-computable conditions for two $\mathrm{Sk}\forall\mathrm{CQ}^=$-mappings to have restricted composition (Theorem 6), and we give sufficient conditions for restricted composition that can be checked efficiently.

Theorem 6 suggests essentially the same algorithm for composition of $\mathrm{Sk}\forall\mathrm{CQ}^=$-mappings as $\forall\mathrm{CQ}_0^=$COMPOSE; we call it $\mathrm{Sk}\forall\mathrm{CQ}^=$COMPOSE. The only difference between them is that $\mathrm{Sk}\forall\mathrm{CQ}^=$COMPOSE operates on $\mathrm{Sk}\forall\mathrm{CQ}^=$ constraints while $\forall\mathrm{CQ}_0^=$COMPOSE operates on $\forall\mathrm{CQ}_0^=$ constraints. Correctness of $\mathrm{Sk}\forall\mathrm{CQ}^=$COMPOSE, sufficient conditions for its termination, and good-$\mathrm{Sk}\forall\mathrm{CQ}^=$-mappings are defined for $\mathrm{Sk}\forall\mathrm{CQ}^=$ just like for $\forall\mathrm{CQ}_0^=$.

THEOREM 6. *If the* $\mathrm{Sk}\forall\mathrm{CQ}^=$*-mappings* m_{12}, m_{23} *are given by* $(\sigma_1, \sigma_2, \Sigma_{12})$ *and* $(\sigma_2, \sigma_3, \Sigma_{23})$ *with* $\Sigma_{123} := \Sigma_{12} \cup \Sigma_{23}$ *and* $\sigma_{13} = \sigma_1 \cup \sigma_3$*, then the following are equivalent*

1. *There is a finite set of constraints* $\Sigma_{13} \subseteq \mathrm{Sk}\forall\mathrm{CQ}^=$ *over the signature* σ_{13} *such that* $m := m_{12} \circ m_{23}$ *is given by* $(\sigma_1, \sigma_3, \Sigma_{13})$ *where* Σ_{13} *has no function symbols or constants other than those appearing in* Σ_{123}.

2. *There is a finite set of constraints* $\Sigma_{13} \subseteq \mathrm{Sk}\forall\mathrm{CQ}^=$ *over the signature* σ_{13} *such that*
$$\mathrm{DC}(\mathrm{Sk}\forall\mathrm{CQ}^=, \Sigma_{123})|_{\sigma_{13}} = \mathrm{DC}(\mathrm{Sk}\forall\mathrm{CQ}^=, \Sigma_{13})$$
where Σ_{13} *has no function symbols or constants other than those appearing in* Σ_{123}.

3. *There is* k *such that for every* ξ *over* σ_{13} *satisfying* $\Sigma_{123} \vdash \xi$ *there is a deduction of* ξ *from* Σ_{123} *using at most* k σ_2*-resolutions.*

PROOF. Essentially the same as that of Theorem 3, using Lemma 3 below instead of Lemma 1. □

LEMMA 3. *Under the hypotheses of Theorem 6, the following are equivalent:*

1. $(A, C) \models \mathrm{DC}(\mathrm{Sk}\forall\mathrm{CQ}^=, \Sigma_{123})|_{\sigma_{13}}$.
2. $\exists B\, (A, B, C) \models \Sigma_{123}$.

6. EMBEDDED DEPENDENCIES

Now we consider composition of $\forall\mathrm{CQ}^=$-mappings; that is, mappings given by embedded dependencies. To compute the composition of two $\forall\mathrm{CQ}^=$-mappings m_{12}, m_{23} given by $(\sigma_1, \sigma_2, \Sigma_{12})$ and $(\sigma_2, \sigma_3, \Sigma_{23})$ we will proceed in three steps, as follows.

Procedure $\forall\mathrm{CQ}^=$COMPOSE$(\Sigma_{12}, \Sigma_{23})$

```
1. $\Sigma'_{12}$ := SKOLEMIZE($\Sigma_{12}$)
   $\Sigma'_{23}$ := SKOLEMIZE($\Sigma_{23}$)
2. $\Sigma'_{13}$ := $\mathrm{Sk}\forall\mathrm{CQ}^=$COMPOSE($\Sigma'_{12}, \Sigma'_{23}$)
3. Return DESKOLEMIZE($\Sigma'_{13}$)
```

The first step, SKOLEMIZE, is straightforward and the second step, $\mathrm{Sk}\forall\mathrm{CQ}^=$COMPOSE, has been discussed in the previous section, so here we concentrate on the third step, de-Skolemization. We provide a sound (but not complete) algorithm for de-Skolemization: DESKOLEMIZE (Theorem 7). Even if the second step succeeds, it may be impossible to find $\Sigma_{13} \subseteq \forall\mathrm{CQ}^=$ such that $\Sigma'_{13} \equiv \Sigma_{13}$ (Example 6) so we identify necessary and sufficient, polynomial-time checkable conditions for the third step to succeed (Proposition 3). DESKOLEMIZE may produce a result of size exponential in the size of its input; we show that in the general case this is unavoidable (Theorem 8), but we also provide polynomial-time checkable conditions for DESKOLEMIZE to run in polynomial time in the size of its input (Proposition 3).

The following example from [8] shows that de-Skolemization is not always possible.

EXAMPLE 6. Consider the $\forall\mathrm{CQ}^=$-mappings m_{12} and m_{23} given by $(\sigma_1, \sigma_2, \Sigma_{12})$ and $(\sigma_2, \sigma_3, \Sigma_{23})$ where

Σ_{12} is	$E(x,y)$	$\to$	$F(x,y)$
	$E(x,y)$	$\to$	$\exists u\, C(x,u)$
	$E(x,y)$	$\to$	$\exists v\, C(y,v)$
Σ_{23} is	$F(x,y), C(x,u), C(y,v)$	$\to$	$D(u,v)$

and where $\sigma_1 = \{E\}$, $\sigma_2 = \{F, C\}$, and $\sigma_3 = \{D\}$. Here, Steps 1 and 2 of $\forall\mathrm{CQ}^=$COMPOSE succeed, but Step 3 fails no matter what algorithm is used for it, since $m_{12} \circ m_{23}$ is not a $\forall\mathrm{CQ}^=$-mapping as shown in [8]. □

The algorithm DESKOLEMIZE which we present below, depends on $\vdash^*$, which is *some* sound polynomial-time approximation of $\models$. That is, if $\Sigma \vdash^* \phi$, then $\Sigma \models \phi$. Of course, the converse may not hold. There are known cases in which there exists such $\vdash^*$ which is also complete (i.e., the converse *does* hold). For example, this is the case when Σ has *stratified witnesses* (see [5] and [7]). Lack of space prevents us from discussing options for the implementation of $\vdash^*$. DESKOLEMIZE works for any $\vdash^*$ satisfying the condition above; the more complete $\vdash^*$ is, the larger the set of inputs on which DESKOLEMIZE succeeds.

Procedure DESKOLEMIZE(Σ)

```
1. Unnest:
   Set $\Lambda_1 := \{\phi' : \phi \in \Sigma\}$ where $\phi'$ is equivalent to and
   obtained from $\phi$ by "unnesting" terms and eliminating terms
   from relational atoms and from the conclusion so that in $\phi'$:
     (a) Function symbols occur only in equations in the
         premise.
```

(b) Every term $f(\bar{u})$ occurs in only one atom.
(c) Every equation is of the form $v = f(\bar{u})$ for some variable v (a *term* variable for f), function f, and sequence of variables $\bar{u}$ (a *defining* equation) or of the form $v = u$ for two *term* variables u and v (a *restricting* equation).
(d) The conclusion contains at most one atom, which is not a defining equation.

We call relational atoms in which a term variable occurs *restricting* atoms. If A is a restricting relational atom or a restricting equation in which v occurs, where v is a term variable for f, we call A an f-*restricting* atom. We call a constraint *restricted* if it has restricting atoms in the premise.

2. **Check for cycles:**
For every $\phi \in \Lambda_1$, construct the graph G_ϕ where the edges are variables in ϕ and where there is an edge (v, u) iff there is an equation of the form $v = f(\ldots u \ldots)$. If G_ϕ has a cycle, abort. Otherwise, set $\Lambda_2 := \Lambda_1$.

3. **Check for argument compatibility:**
For every $\phi \in \Lambda_2$ check that ϕ does not contain two atoms with the same function symbol. If it does, abort. Otherwise, set $\Lambda_3 := \Lambda_2$.

4. **Align variables:**
Rename the variables in Λ_3 to obtain Λ_4 satisfying:

(a) For every function symbol f and any two equations of the form $v = f(\bar{u})$ and $y = f(\bar{x})$ in Λ_4, v is the same variable as y and $\bar{u}$ is the same sequence of variables as $\bar{x}$.
(b) For every two different function symbols f and g and any two equations of the form $v = f(\bar{u})$ and $y = g(\bar{x})$ in Λ_4, v and y are different variables.

If this is not possible, abort.

5. **Eliminate restricting atoms:**
Pick some ordering of the function symbols in Λ_4: $f_1, \ldots, f_k$. Set $\Delta_0 = \Lambda_4$. For $n = 1, \ldots, k-1$, set $\Delta_{n+1} := \{\phi' : \phi \in \Delta_n\}$ where ϕ' is obtained from ϕ as follows. Set ψ to be ϕ with the f_{n+1}-restricting atoms removed from the premise. If $\Delta_n \vdash^* \psi$, set $\phi' := \psi$; otherwise set $\phi' := \phi$. In any case, we have $\Delta_{n+1} \equiv \Delta_n$. Set $\Lambda_5 := \Delta_k$.

6. **Eliminate constraints with restricting atoms:**
Set Λ_6 to be the set of constraints $\phi \in \Lambda_5$ which can not be eliminated according to the following test: ϕ can be eliminated if

(a) ϕ contains an f-restricting atom in the premise, and
(b) there is no constraint $\psi \in \Lambda_5$ which has an f-restricting atom in the conclusion and no f-restricting atoms in the premise.

7. **Check for restricted constraints:**
Set Λ to the set of unrestricted constraints in Λ_6. If there is any $\phi \in \Lambda_6$ such that $\Lambda \not\vdash^* \phi$, abort. Otherwise, set $\Lambda_7 := \Lambda$.

8. **Check for dependencies:**
For every $\phi \in \Lambda_7$ and every variable v in ϕ, define $D_{\phi,v}$ as follows. If v is not a term variable, set $D_{\phi,v} = \{v\}$. If v is a term variable and $v = f(\bar{u})$ its defining equation in ϕ, then set $D_{\phi,v} := \bigcup_{u \in \{\bar{u}\}} D_{\phi,u}$.[4] Intuitively, $D_{\phi,v}$ is the set of variables on which v depends. Set $V_\phi :=$ the set of variables which appear in the conclusion of ϕ. For every term variable v in V_ϕ, check that $D_{\phi,v} = \bigcup_{u \in V_\phi} D_{\phi,u}$. If this fails, abort. Otherwise, set $\Lambda_8 := \Lambda_7$.

9. **Combine dependencies:**
Set $\Lambda_9 := \{\psi_\Phi : \emptyset \neq \Phi \subseteq \Lambda_8\}$ where ψ_Φ is defined as follows. If there is a function f which appears in every $\phi \in \Phi$, then the premise of ψ_Φ consists of the atoms in all the premises in Φ and the conclusion of ψ_Φ consists of the atoms in all the conclusions of Φ (remove duplicate atoms). Otherwise, ψ_Φ is some constraint in Φ. Notice that $\Lambda_9 \supseteq \Lambda_8$ since $\psi_{\{\phi\}} = \phi$.

10. **Remove redundant constraints:**
Pick some set $\Lambda_{10} \subseteq \Lambda_9$ such that $\Lambda_{10} \vdash \phi$ for every $\phi \in \Lambda_9$, and such that this does not hold for any proper subset of Λ_{10}.

11. **Replace functions with $\exists$ variables:**
Set $\Lambda_{11} := \{\phi' : \phi \in \Lambda_{10}\}$ where the premise of ϕ' is the premise of ϕ with all equations removed and where the conclusion of ϕ' is the conclusion of ϕ, with all variables appearing on the left of equations in ϕ existentially quantified.

12. **Eliminate unnecessary $\exists$ variables:**
Set $\Lambda_{12} := \{\phi' : \phi \in \Lambda_{11}\}$ and return Λ_{12} where ϕ' is like ϕ, but where existentially quantified variables which do not appear in the conclusion atom have been removed (with their corresponding existential quantifier).

EXAMPLE 7. Consider three runs of the algorithm DESKOLEMIZE(Σ_{13}^i), for $i \in \{1, 2, 3\}$. Let $\Sigma_{13}^i = \{\gamma_1, \ldots, \gamma_i\}$ be a set of the following (unnested) Sk$\forall$CQ constraints:

γ_1	$R_1(y), R_2(x), y = f(x)$	$\rightarrow$	$T_1(x)$
γ_2	$R_2(x), y = f(x)$	$\rightarrow$	$T_2(y)$
γ_3	$R_2(x), y = f(x)$	$\rightarrow$	$R_1(y)$

For completeness, we note that each Σ_{13}^i is obtained by first de-Skolemizing $\forall$CQ$^=$-mappings given by Σ_{12}^i and Σ_{23}^i, which are shown below, and then invoking Sk$\forall$CQ$^=$COMPOSE:

i	Σ_{12}^i	Σ_{23}^i	Σ_{13}^i
1.	$\{\alpha_1, \alpha_2\}$	$\{\beta_2\}$	$\{\gamma_1\}$
2.	$\{\alpha_1, \alpha_2\}$	$\{\beta_1, \beta_2\}$	$\{\gamma_1, \gamma_2\}$
3.	$\{\alpha_1, \alpha_2, \alpha_3\}$	$\{\beta_1, \beta_2\}$	$\{\gamma_1, \gamma_2, \gamma_3\}$

Dependencies $\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2$ are specified as:

α_1	$R_1(x)$	$\rightarrow$	$S_1(x)$
α_2	$R_2(y)$	$\rightarrow$	$\exists z(S_2(zy))$
α_3	$S_2(zy)$	$\rightarrow$	$R_1(z)$
β_1	$S_2(zy)$	$\rightarrow$	$T_2(z)$
β_2	$S_1(x), S_2(xy)$	$\rightarrow$	$T_1(y)$

In all three runs of DESKOLEMIZE(Σ_{13}^i), Steps 2 and 3 pass since each of γ_1, γ_2, γ_3 is cycle-free and has no multiple atoms with the same function symbol. Step 4 has no effect, since the variable names of the dependencies are already aligned. The remaining steps are explained below.

[4]This is an inductive definition; it requires the check in Step 2.

In the run DESKOLEMIZE($\{\gamma_1\}$), Step 5 has no effect, because $\{\gamma_1\}$ is a singleton set. Its only member γ_1 gets eliminated in Step 6, since there are no rules in $\{\gamma_1\}$ with f-restricting atoms in conclusions. Intuitively, γ_1 is a tautology because we can always construct f whose range is disjoint with R_1. Hence, $\{\gamma_1\}$ is equivalent to the empty set of constraints, which is trivially in $\forall$CQ.

In the run DESKOLEMIZE($\{\gamma_1, \gamma_2\}$), γ_2 contains an f-restricting atom T_2 in its conclusion. Hence, we cannot eliminate the restricted constraint γ_1 in Step 6, and so de-Skolemization aborts in Step 7.

In the run DESKOLEMIZE($\{\gamma_1, \gamma_2, \gamma_3\}$), we are able to de-Skolemize despite γ_2. In Step 5, $\Delta_0 = \{\gamma_1, \gamma_2, \gamma_3\}$. By considering the only function symbol f, we get $\Delta_1 = \{\psi, \gamma_2, \gamma_3\} \equiv \Delta_0$ where ψ is obtained by eliminating the restricting atom $R_1(y)$ from the premise of γ_1 as

$$\psi \quad := \quad R_2(x), y = f(x) \to T_1(x)$$

Clearly, $\Delta_0 \vdash^* \psi$, since $\Delta_0 \supset \{\gamma_1, \gamma_3\} \vdash^* \psi$. Δ_1 has no restricting constraints, so Step 6 has no effect and Step 7 passes. Step 8 succeeds with $\Lambda_8 = \Lambda_7 = \{\psi, \gamma_2, \gamma_3\}$, since every dependency in Δ_1 has at most one term variable y in its conclusion. Taking γ_3 as an example, we get $V_{\gamma_3} = \{x, y\}$, $D_{\gamma_3,x} = D_{\gamma_3,y} = \bigcup_{u \in V_{\gamma_3}} D_{\gamma_3,u} = \{x\}$.
In Step 9, combining the dependencies for $\Phi = \Lambda_8$ yields

$$\gamma_4 \quad := \quad R_1(y), R_2(x), y = f(x) \to T_1(x), T_2(y), R_1(y)$$

(Combinations resulting from proper subsets of Λ_8 are not shown for brevity). In Step 10, we remove the redundant constraints which include ψ, γ_2, γ_3, because they share the premise with γ_4 and their conclusion is subsumed by that of γ_4; we obtain $\Lambda_{10} = \{\gamma_4\}$. Finally, replacing function f by an existential variable in γ_4 yields

$$\Lambda_{12} = \{R_2(x) \to \exists y(T_1(x) \wedge T_2(y) \wedge R_1(y))\}$$

Thus, DESKOLEMIZE($\{\gamma_1, \gamma_2, \gamma_3\}$) $\subseteq \forall$CQ. ☐

THEOREM 7. *If* DESKOLEMIZE(Σ) *succeeds on input* $\Sigma \subseteq$ Sk$\forall$CQ$^=$, *then* $\Sigma' :=$ DESKOLEMIZE(Σ) *satisfies*

$$\Sigma' \subseteq \forall\text{CQ}^= \text{ and } \Sigma' \equiv \Sigma.$$

The following is clear from the description of the algorithm.

PROPOSITION 3.

1. DESKOLEMIZE(Σ) *succeeds on input* $\Sigma \subseteq$ Sk$\forall$CQ$^=$ *iff it reaches Step 9, which can be checked in polynomial time in the size of* Σ.

2. *Furthermore, if for some constant* ℓ *independent of* Σ, *for every function symbol in* Λ_8 *there are no more than* ℓ *rules in* Λ_8 *in which* f *occurs, then* DESKOLEMIZE(Σ) *runs in polynomial time in the size of* Σ. *This can be checked in polynomial time in the size of* Σ.

Since DESKOLEMIZE(Σ) may produce a result of size exponential in the size of Σ due to Step 9, $\forall$CQ$^=$COMPOSE(Σ_{12}, Σ_{23}) may produce a result of size exponential in the size of $\Sigma_{12} \cup \Sigma_{23}$ due to step 3, even when step to gives a polynomial-size result. The following example shows that in the general case this is unavoidable.

THEOREM 8. *There are two sequences of* $\forall$CQ*-mappings* m^k_{12} *and* m^k_{23} *given by* Σ^k_{12} *and* Σ^k_{23} *such that the* $\forall$CQ*-composition* $m^k_{12} \circ m^k_{23}$ *grows exponentially in the size of* $\Sigma^k_{12} \cup \Sigma^k_{23}$, *but the* Sk$\forall$CQ*-composition* $m^k_{12} \circ m^k_{23}$ *grows linearly in the size of* $\Sigma^k_{12} \cup \Sigma^k_{23}$,

PROOF. (Outline) Set $[k] := \{1, \ldots, k\}$. Consider the $\forall$CQ-mappings m^k_{12} and m^k_{23} given by $(\sigma^k_1, \sigma^k_2, \Sigma^k_{12})$ and $(\sigma^k_2, \sigma^k_3, \Sigma^k_{23})$ where

Σ^k_{12} is	$R_0(x)$	$\to$	$\exists y\, S_0(xy)$
	$R_i(x)$	$\to$	$S_i(x)$
Σ^k_{23} is	$S_0(xy), S_i(x)$	$\to$	$T_i(y)$

for $i \in [k]$ and where $\sigma^k_1 = \{R_i : i \in \{0, \ldots, k\}\}$, $\sigma^k_2 = \{S_i : i \in \{0, \ldots, k\}\}$, and $\sigma^k_3 = \{T_i : i \in [k]\}$. The $\forall$CQ-composition $m^k_{13} := m^k_{12} \circ m^k_{13}$ is given by the set Σ^k_{13} of constraints

$$R_0(x), R_Z(x) \to \exists y\, T_Z(y)$$

such that $Z \subseteq [k]$ where $R_Z(x) := \bigwedge_{i \in Z} R_i(x)$ and similarly for T_Z. On the other hand, m^k_{13} is not given by any $(\sigma_1, \sigma_3, \Sigma)$ where Σ has fewer than $2^k - 1$ constraints (we omit the inexpressibility proof due to space constraints.) The Sk$\forall$CQ-composition $m^k_{13} := m^k_{12} \circ m^k_{13}$ is given by the set Σ'^k_{13} of constraints

$$R_0(x), y = f(x), R_i(x) \to T_i(y)$$

for $i \in [k]$, which grows linearly in the size of $\Sigma^k_{12} \cup \Sigma^k_{23}$ ☐

7. UNRESTRICTED COMPOSITION

In this section we examine the semantics of Sk$\forall$CQ$^=$ constraints and study unrestricted composition (where we allow the introduction of new existentially-quantified symbols to express the composition). The semantics of Sk$\forall$CQ$^=$ are somewhat special, but seem to be needed to obtain domain independence (Example 8). Under these semantics, the safety condition in Sk$\forall$CQ$^=$ is no longer necessary to provide domain independence (intuitively, the domain is large enough to allow all existential witnesses to be distinct). Furthermore, unsafe source-to-target Sk$\forall$CQ$^=$ have **NP** data complexity, just as their safe counterparts. We show that unsafe Sk$\forall$CQ$^=$-mappings *are* closed under unrestricted composition and that, furthermore, composition can be computed in linear time (Theorem 9). Intuitively, this is because we can assert the existence of the relations in the intermediate signature σ_2 using existentially-quantified function symbols. In contrast to results in previous sections, here we use essentially the fact that we have equations in the conclusions.

We introduce another fragment of SO, $\exists$SO$\forall$CQ$^=$ which has the standard second-order semantics. We show that source-to-target Sk$\forall$CQ$^=$-mappings are also source-to-target $\exists$SO$\forall$CQ$^=$-mappings (Theorem 10).[5] However, this translation may incur and exponential increase in size.

We first discuss the semantics of Sk$\forall$CQ$^=$ constraints. The main question is, what is the universe from which the functions can take values? That is, what is their allowed range? Intuitively, the problem is with the universe of the existentially quantified intermediate database.

EXAMPLE 8. Consider the $\forall$CQ$_0$-mappings m^k_{12} and m^k_{23} given by $(\sigma_1, \sigma^k_2, \Sigma^k_{12})$ and $(\sigma^k_2, \sigma_3, \Sigma^k_{23})$ where

Σ^k_{12} is	$R(x)$	$\to$	$\exists y S_i(y)$
Σ^k_{23} is	$S_i(x), S_j(x)$	$\to$	$T(x)$

for $1 \le i, j \le k$, $\sigma_1 = \{R\}$, $\sigma^k_2 = \{S_1, \ldots, S_k\}$, and $\sigma_3 = \{T\}$.

Consider the case where $R = \{1, \ldots, k-1\}$ in A and T is empty in C. Firstly, notice that $(A, C) \in m^k_{12} \circ m^k_{23}$ as witnessed by the database B where $S_i = \{i\}$.

[5]when restricted to structures with at least two elements in the active domain

Skolemizing and composing the constraints above we obtain Σ_{13}^k given by the set of constraints

$$\{R(x), f_i(x) = y, f_j(x) = y \rightarrow T(y) : 1 \leq i, j \leq k\}$$

where f_i is the Skolem function corresponding to $\exists y S_i(y)$. If we restrict the range of every f_i to fall within the domain of R and T, then one of the constraints above must fail if the domain of R and T is limited to $\{1, \ldots, k-1\}$ (the active domain) since in this case, for every x, we have only $k-1$ values to choose from for $f_i(x)$ and therefore we have $(A, C) \not\models \Sigma_{13}^k$. On the other hand, if we keep the same relations R and T, but allow the domain to have at least k values, then we have $(A, C) \models \Sigma_{13}^k$ witnessed by setting $f_i(x) = i$ for all i, x.

This shows that if we restrict the range of the Skolem functions to be domain of the input structures, then we may have domain-dependent formulas, even though they satisfy the safety conditions. Certainly, no such domain-dependent formulas can express the composition, since whether (R, T) belong to the composition or not does not depend on their domains. $\square$

Therefore we require all databases to be finite (i.e., all relations are finite), but to have an implicit countably infinite universe. Notice that no finite domain would work for all constraints since the example above gives a family of sets of constraints for which the meaning changes depending on whether the domain has size less than k.

We allow the functions to take any values from this implicit universe, as long as their range is finite. This last restriction is not necessary when we restrict our mappings to be source-to-target, but is needed in the general case we consider here since we need to ensure the intermediate instance encoded by the functions is finite.

These assumptions ensure that the deductive system that we introduced in Section 2 is sound for $\text{Sk}\forall\text{CQ}^=$, which is needed for Theorem 6 below.

An interesting side-effect of these semantics is that the safety condition on $\text{Sk}\forall\text{CQ}^=$ is no longer necessary to ensure domain-independence. Furthermore, we have the following.

THEOREM 9. *Unsafe* $\text{Sk}\forall\text{CQ}^=$*-mappings are closed under composition. Furthermore, their composition can be computed in linear time.*

PROOF. If $\text{Sk}\forall\text{CQ}^=$-mappings m_{12} and m_{23} are given by $(\sigma_1, \sigma_2, \Sigma_{12})$ and $(\sigma_2, \sigma_3, \Sigma_{23})$, we obtain Σ_{13} such that the composition $m_{12} \circ m_{23}$ is given by $(\sigma_1, \sigma_2, \Sigma_{23})$ as follows. We set $\Sigma_{13} := \Sigma'_{12} \cup \Sigma'_{23}$ where Σ'_{12} and Σ'_{23} are obtained from Σ_{12} and Σ_{23} by adding a new function symbol f_D and by replacing every occurrence of an atom $R(\bar{x})$ where $R \in \sigma_2$ is of arity r with $f_D(x_1) = x_1, \ldots, f_D(x_r) = x_r, g_R(\bar{x}) = h_R(\bar{x})$, where g_R, h_R are new function symbols.

Then if $\langle A, C \rangle \in m_{12} \circ m_{23}$, we have B such that $(A, B) \models \Sigma_{12}$ and $(B, C) \models \Sigma_{23}$. For every relation R in σ_2, we set $g_R(\bar{c}) = h_R(\bar{c})$ iff $R(\bar{c})$ holds in B by using any two different values from the domain. Furthermore, we set $f_D(c) := c$ if c is in the active domain of B and $f_D(c) := c'$ otherwise for some c' not in the domain of any relation in (A, C) Then $(A, C) \models \Sigma_{13}$.

Conversely, if $(A, C) \models \Sigma_{13}$, then we set $R := \{\bar{c} : f_D(c_1) = c_1, \ldots, f_D(c_r) = c_r, g_R(\bar{c}) = h_R(\bar{c})\}$ for every relation R in σ_2 to obtain B such that $(A, B) \models m_{12}$ and $(B, C) \models m_{23}$. Since the range of f_D is finite, R is finite.

The linear-time algorithm for composition is implicit in the proof above. $\square$

In certain sense, this algorithm "cheats" by encoding the intermediate instance in the composition in an uninformative way. It exploits the special semantics of $\text{Sk}\forall\text{CQ}$, including the requirement that all functions have finite range.

Since the semantics of $\text{Sk}\forall\text{CQ}^=$ are special, it is natural to ask whether the constraints in $\text{Sk}\forall\text{CQ}^=$ can be expressed in some fragment of $\exists\text{SO}$ under the usual second-order semantics. We show that this is possible for source-to-target $\text{Sk}\forall\text{CQ}^=$.

THEOREM 10. *Every finite set of source-to-target* $\text{Sk}\forall\text{CQ}^=$ *constraints (under the semantics described above) is equivalent to a finite set of source-to-target* $\exists\text{SO}\forall\text{CQ}^=$ *constraints (under the usual second-order semantics) when restricted to instances with at least two elements.*

PROOF. (Outline) In the case of source-to-target constraints, we know that we do not have "recursive" Skolem terms. That is, there are no Skolem terms of the form $f(\ldots f \ldots)$ (directly, or indirectly through equations). Therefore, there is a finite number of values we can refer to by building Skolem terms on top of the elements of the domain. Intuitively, these are all the elements that the intermediate database needs to have and the worst case is when they are all different. If the domain has n elements and we have p Skolem functions of arity q, then an easy upper bound on the number of elements we can refer to is $\leq n^{(p+q)^p}$ whenever $n \geq 2$. (This is shown by induction depth of the Skolem terms; at each step we go from $m \geq n$ possible values to

$$m + pm^q \leq (p+1)m^q \leq 2^p m^q \leq n^p m^q \leq m^{(p+q)}$$

possible values.) Therefore, we can encode all these values with tuples of arity $r = (p+q)^p$. We encode every value c from the original domain as the tuple $(c, \ldots, c)$; that is, c repeated r times.

Given a finite set Σ of source-to-target $\text{Sk}\forall\text{CQ}^=$ (which we assume w.l.o.g. to be in unnested form), we first compute r, then transform each constraint $\phi \in \Sigma$ by replacing every occurrence of an equation of the form $f(\bar{x}) = y$ with $F(\bar{x}, \bar{y})$ where $\bar{y}$ is a tuple of arity r. We also replace y with $\bar{y}$ everywhere except in relational atoms. To every relational atom, we add a set of equations of the form $y = y_1, \ldots, y = y_k$ which we abbreviate $y = \bar{y}$. Finally, we add constraints of the form

$$\ldots \rightarrow \exists \bar{y} F(\bar{x}, \bar{y})$$

$$F(\bar{x}, \bar{y}), F(\bar{x}, \bar{y}') \rightarrow \bar{y} = \bar{y}'$$

where $\ldots$ are obtained from any premises which mention f. Notice that we need both equations and FO existential quantifiers in the conclusions and that we may incur an exponential increase in size since we need r to be exponential in p. $\square$

Notice that the proof only requires that we do not have "recursive" Skolem terms. Source-to-target is a strong condition that ensures this, but weaker conditions on the set of constraints $\Sigma_{12} \cup \Sigma_{23}$ suffice. For example, it is enough to require that $\Sigma_{12} \cup \Sigma_{23}$ have *stratified witnesses* (see [5] and [7]). When such conditions hold, we can compose $\exists\text{SO}\forall\text{CQ}^=$-mappings using a technique similar to that of the proof of Theorem 10. In this case, we don't Skolemize, but replace every relation S of arity s from σ_2 with an existentially quantifed relation R of arity rs (where r is as in the proof of Theorem 10) Then we replace every occurence of a universally quantified variable v in S with $\bar{v}$ and add the equation $v = \bar{v}$ and replace every occurence of an existentially quantified variable u with bar u. This gives an algorithm for composition of source-to-target $\exists\text{SO}\forall\text{CQ}^=$-mappings.

8. OTHER BASIC OPERATORS

In addition to composition, we are interested in several other basic operators including domain, range, intersection, cross-product, and inverse. These operators take for input mappings and models and give as output mappings or models (a *model* is a set of instances). As in the case of mappings, we say that a model $\mathcal{A}$ is given by (σ_1, Σ_1) if it consists exactly of those databases over the signature σ_1 which satisfy the constraints Σ_1. If, furthermore, Σ_1 is finite subset of $\mathcal{L}$ we say that $\mathcal{A}$ is an $\mathcal{L}$-model. As in the case of composition, we say that $\mathcal{L}$ is *closed* under one of these operators if it produces an $\mathcal{L}$-model or $\mathcal{L}$-mapping whenever the inputs are compatible $\mathcal{L}$-models or $\mathcal{L}$-mappings.

PROPOSITION 4. *Every $\mathcal{L} \supseteq \forall\mathrm{CQ}_0$ is closed under identity, cross product and intersection.*

PROPOSITION 5. *Each one of the operators composition, range, and domain can be reduced to any one of the others.*

PROOF. If

- m_{12} is given by $(\sigma_1, \sigma_2, \Sigma_{12})$,
- m_{23} is given by $(\sigma_2, \sigma_3, \Sigma_{23})$,
- m_1 is given by $(\sigma_1 \cup \sigma_3, \sigma_2, \Sigma_{12} \cup \Sigma_{23})$,
- m_2 is given by $(\sigma_2, \sigma_1 \cup \sigma_3, \Sigma_{12} \cup \Sigma_{23})$,
- $\mathrm{dom}(m_1)$ is given $(\sigma_1 \cup \sigma_3, \Sigma_1)$, and
- $\mathrm{rng}(m_2)$ is given $(\sigma_1 \cup \sigma_3, \Sigma_2)$,

then $m_{12} \circ m_{23}$ is given by $(\sigma_1, \sigma_3, \Sigma_1)$ and $(\sigma_1, \sigma_3, \Sigma_2)$. Conversely, if

- m_{12} is given by $(\sigma_1, \sigma_2, \Sigma_{12})$,
- m_{21} is given by $(\sigma_2, \sigma_1, \emptyset)$,
- $m_{12} \circ m_{21}$ is given by $(\sigma_1, \sigma_1, \Sigma_1)$, and
- $m_{21} \circ m_{12}$ is given by $(\sigma_2, \sigma_2, \Sigma_2)$,

then $\mathrm{dom}(m_{12})$ and $\mathrm{rng}(m_{12})$ are given respectively by (σ_1, Σ_1) and (σ_2, Σ_2). $\square$

Proposition 5 and Theorem 2 give the following.

COROLLARY 2. *Checking whether the domain or range of a $\forall\mathrm{CQ}_0$-mapping is a $\forall\mathrm{CQ}_0$-model is undecidable.*

All the languages we consider satisfy the premises of Proposition 4. Therefore, Proposition 5 indicates that we can concentrate our attention on closure under composition and inverse. Notice that if an $\mathcal{L}$-mapping m is given by $(\sigma_1, \sigma_2, \Sigma_{12})$, then its inverse is given by $(\sigma_2, \sigma_1, \Sigma_{12})$, which is, of course, easy to compute. However, the restrictions on $\mathcal{L}$ may be such that the second expression no longer gives an $\mathcal{L}$-mapping. For example, this happens with source-to-target constraints. This is why we seek restrictions on $\mathcal{L}$ which are symmetric with respect to the input and output signatures and which guarantee closure under composition.

9. CONCLUSIONS

Mapping composition is one of the key operators that are used for manipulating schemas and mappings between schemas. We studied composition of mappings given by embedded dependencies, which are expressive enough for many data management applications. We addressed challenges that were not considered in prior work, in particular the ones due to recursion and de-Skolemization.

Acknowledgements. We are grateful to Ron Fagin, Phokion Kolaitis, Lucian Popa, and Wang-Chiew Tan for fruitful discussions on mapping composition and for their feedback on a preliminary version of this paper. We thank Andreas Blass who provided the formula ψ in the proof of Theorem 1 and contributed to Proposition 5.

10. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.
[2] P. A. Bernstein. Applying Model Management to Classical Meta-Data Problems. In *Conference on Innovative Data Systems Research (CIDR)*, pages 209–220, 2003.
[3] P. A. Bernstein, A. Y. Halevy, and R. Pottinger. A Vision of Management of Complex Models. *SIGMOD Record*, 29(4):55–63, 2000.
[4] M. A. Casanova and V. M. P. Vidal. Towards a Sound View Integration Methodology. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 36–47, 1983.
[5] A. Deutsch and V. Tannen. Reformulation of XML Queries and Constraints. In *Intl. Conf. on Database Theory (ICDT)*, 2003.
[6] H.-D. Ebbinhaus and J. Flum. *Finite Model Theory*. Springer, 1999.
[7] 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.
[8] R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Composing Schema Mappings: Second-Order Dependencies to the Rescue. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 83–94, 2004.
[9] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema Mediation in Peer Data Management Systems. In *Intl. Conf. on Data Engineering (ICDE)*, 2003.
[10] C. Koch. Query rewriting with symmetric constraints. In *Foundations of Information and Knowledge Systems (FoIKS)*, 2002.
[11] M. Lenzerini. Data Integration: A Theoretical Perspective. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 233–246, 2002.
[12] J. Madhavan and A. Y. Halevy. Composing Mappings Among Data Sources. In *International Conference on Very Large Data Bases (VLDB)*, pages 572–583, 2003.
[13] S. Melnik. *Generic Model Management: Concepts and Algorithms*. Ph.D. Thesis, University of Leipzig, Springer LNCS 2967, 2004.
[14] Y. Sagiv and M. Yannakakis. Equivalences among relational expressions with the union and difference operators. *Journal of the ACM*, 27(4):633–655, 1980.
[15] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 2nd edition, 1997.
[16] C. Yu and L. Popa. Constraint-Based XML Query Rewriting For Data Integration. In *ACM SIGMOD International Conference on Management of Data*, pages 371–382, 2004.