The Complexity of Data Exchange

Phokion G. Kolaitis [*]
IBM Almaden
kolaitis@almaden.ibm.com

Jonathan Panttaja [†]
UC Santa Cruz
jpanttaj@cs.ucsc.edu

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

ABSTRACT

Data exchange is the problem of transforming data structured under a source schema into data structured under a target schema in such a way that all constraints of a schema mapping are satisfied. At the heart of data exchange, lies a basic decision problem, called the existence-of-solutions problem: given a source instance, is there a target instance that satisfies the constraints of the schema mapping at hand? Earlier work showed that for schema mappings specified by embedded implicational dependencies, this problem is solvable in polynomial time, assuming that (1) the schema mapping is kept fixed and (2) the constraints of the schema mapping satisfy a certain structural condition, called weak acyclicity.

We investigate the effect of these assumptions on the complexity of the existence-of-solutions problem, and show that each one is indispensable in deriving polynomial-time algorithms for this problem. Specifically, using machinery from universal algebra, we show that if the weak acyclicity assumption is relaxed even in a minimal way, then the existence-of-solutions problem becomes undecidable. We also show that if, in addition to the source instance, the schema mapping is part of the input, then the existence-of-solutions problem becomes EXPTIME-complete. Thus, there is a provable exponential gap between the data complexity and the combined complexity of data exchange. Finally, we study restricted classes of schema mappings and develop a comprehensive picture for the combined complexity of the existence-of-solutions problem for these restrictions. In particular, depending on the restriction considered, the combined complexity of this problem turns out to be either EXPTIME-complete or coNP-complete.

Categories and Subject Descriptors

F.2 [**Analysis of Algorithms and Problem Complexity**]: General; H.2 [**Database Management**]: Heterogenous Databases—*Data Translation*

General Terms

Algorithms, Languages, Theory

Keywords

Data exchange, schema mappings, complexity, undecidability, database dependencies

[*]On leave from UC Santa Cruz.

[†]Supported by NSF grant IIS-0430994 and NSF CAREER Award IIS-0347065.

1. INTRODUCTION

Data exchange is the problem of transforming data structured under a schema, called the source schema, to data structured under a different schema, called the target schema. Even though data exchange is regarded as the "oldest database problem" [4], it is only in the past few years that a systematic study of data exchange in its own right has been carried out. The semantics and fundamental algorithmic issues in data exchange between relational schemas were first investigated in [12] and explored further in [13, 14, 15, 23]; data exchange between XML schemas was studied in [25, 1].

Data exchange has been formalized using the concept of a *schema mapping*, a concept that has also been widely used in data integration (see the survey [22]). By definition, a *schema mapping* (also called a *data exchange setting*) is a quadruple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$, where $\mathbf{S}$ is a source schema, $\mathbf{T}$ is a target schema, Σ_{st} is a set of constraints between $\mathbf{S}$ and $\mathbf{T}$, and Σ_t is a set of constraints on $\mathbf{T}$. Every schema mapping $\mathcal{M}$ gives rise to the following *data exchange problem*: given a source instance I, construct a *solution for* I, that is, a target instance J such that the pair (I, J) satisfies every constraint in $\Sigma_{st} \cup \Sigma_t$. In general, no solution may exist for a given source instance I. Thus, the first component of the data exchange problem for $\mathcal{M}$ is the following decision problem, called the *existence-of-solutions problem for* $\mathcal{M}$: given a source instance I, does a solution for I exist? Even if a solution for I exists, however, it need not be unique. Thus, the second component of the data exchange problem is to "sort out" the solutions (when they exist) and return a "good" solution that reflects the source data as accurately as possible.

In [12], the data exchange problem was studied for schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ in which Σ_{st} is a finite set of *source-to-target tuple generating dependencies* (in short, *s-t tgds*) and Σ_t is a finite set of *target tuple-generating dependencies* (in short, *target tgds*) and *target equality-generating dependencies* (in short, *target egds*). By definition, a s-t tgd is a first-order formula of the form $\forall \mathbf{x}(\varphi(\mathbf{x}) \to (\exists \mathbf{y})\psi(\mathbf{x}, \mathbf{y}))$, where $\varphi(\mathbf{x})$ is a conjunction of atoms over $\mathbf{S}$ and $\psi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms over $\mathbf{T}$. Similarly, a target tgd is a formula of the same form in which both $\varphi(\mathbf{x})$ and $\psi(\mathbf{x}, \mathbf{y})$ are conjunctions of atoms over $\mathbf{T}$. Finally, a target egd is a formula of the form $\forall \mathbf{x}(\chi(\mathbf{x}) \to x_i = x_j)$, where $\chi(\mathbf{x})$ is a conjunction of atoms over $\mathbf{T}$ and x_i, x_j are variables in $\mathbf{x}$. Note that, taken together, tgds and egds have the same expressive power as *embedded implicational dependencies* [11].

The concept of a *universal solution* was introduced in [12] and a case was made that universal solutions are the preferred "good" solutions in data exchange. After this, it was shown that if the constraints of a schema mapping satisfy a certain acyclicity condition, then the existence-of-solutions problem is tractable and universal solutions are efficiently computable. More precisely, if $\mathcal{M}$

= $(\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ is a fixed schema mapping such that Σ_{st} is a set of s-t tgds and Σ_t is the union of a set of target egds with a *weakly acyclic* set of target tgds, then there is a polynomial-time algorithm with the following property: given a source instance I, the algorithm determines whether or not a solution for I exists and, if so, it constructs a particular universal solution for I. Note that weakly acyclic sets of tgds contain as special cases both acyclic sets of inclusion dependencies [6] and sets of *full* tuple-generating dependencies (full tgds) [3], where a *full* tgd is a tgd of the form $\forall \mathbf{x}(\varphi(\mathbf{x}) \rightarrow \psi(\mathbf{x}))$ (that is, no existential quantifiers occur in the right-hand side).

In this paper, we explore the complexity of data exchange by investigating two different issues. The first is: How critical is the weak acyclicity assumption in obtaining the above polynomial-time results for data exchange? Specifically, do these results hold for schema mappings that satisfy conditions less stringent than weak acyclicity? The second issue is the *combined complexity* of data exchange. The polynomial-time results assume a fixed schema mapping, and so they are about the *data complexity* of data exchange. Do these results still hold when the schema mapping itself is part of the input? If not, how do they change? In particular, what is the complexity of the existence-of-solutions problem when the input consists of both a schema mapping $\mathcal{M}$ and a source instance I?

Our first main result asserts that there is a fixed schema mapping $\mathcal{M}^* = (\mathbf{S}^*, \mathbf{T}^*, \Sigma^*_{st}, \Sigma^*_t)$ such that the existence-of-solutions problem for $\mathcal{M}^*$ is undecidable. Moreover, $\mathcal{M}^*$ is such that Σ^*_{st} consists of a single full s-t tgd and Σ^*_t consists of one target egd, one full target tgd, and one non-weakly acyclic target tgd. This result shows that weak acyclicity is indispensable for the tractability of the existence-of-solution problem; in fact, since every set of full tgds is weakly acyclic, our result implies that adding just one non-weakly acyclic target tgd to a set of target egds and full target tgds may cause a dramatic jump from polynomial-time solvability to algorithmic unsolvability. The undecidability of the existence-of-solutions problem for $\mathcal{M}^*$ is obtained via a reduction from the *embedding problem* for finite semigroups: given a finite partial semigroup, is it embeddable in a finite semigroup? In turn, the undecidability of the embedding problem for finite semigroups follows from the undecidability of the *word problem* for finite semigroups, due to Gurevich [19], and an intimate connection between the solvability of the word problem and the solvability of the embedding problem, discovered by Evans [9]. It is perhaps worth pointing out that the undecidability of various word problems has been used in the past to establish the undecidability of the implication problem for various classes of database dependencies (see, for instance, [2, 20]). To the best of our knowledge, this is the first time that the embedding problem is being used to obtain undecidability results in database theory.

Vardi [27] developed a useful taxonomy for categorizing the complexity of problems that involve formulas and databases, such as the query evaluation problem. If the input consists of both a formula and a database, then we talk about the *combined complexity* of the problem. By fixing the formula, however, we obtain a family of algorithmic problems (one for each fixed formula) in which the input is just a database; in this case, we talk about the *data complexity* of each of these problems. As a general rule, the combined complexity is higher than the data complexity. For instance, the combined complexity of conjunctive query evaluation is NP-complete, while the data complexity of evaluating a fixed conjunctive query is in LOGSPACE. Actually, the gap may be even higher: the combined complexity of first-order query evaluation is PSPACE-complete, while the data complexity of evaluating a fixed first-order query is in LOGSPACE.

Vardi's taxonomy makes sense in the context of data exchange, since this problem involves two parameters: schema mappings and source instances. As stated earlier, the complexity-theoretic analysis of data exchange in [12] focused exclusively on data complexity, since it was carried out under the assumption that the schema mapping is kept fixed. This is perfectly meaningful in applications in which the same schema mapping is repeatedly used to move data from a fixed source schema to a fixed target schema. In many applications, however, the schema mappings change frequently, because the constraints between the two schemas tend to change or because the schemas themselves tend to evolve over time. In turn, this calls for an investigation of the combined complexity of data exchange.

Clearly, if we allow arbitrary schema mappings as part of the input, then the existence-of-solutions problem is undecidable, since, as described above, this problem may be undecidable even for a fixed schema mapping in which the set of target tgds is not weakly acyclic. For this reason, we focus on the combined complexity of the existence-of-solutions problem for the class $\mathcal{C}$ of all schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ in which Σ_{st} is a finite set of s-t tgds and Σ_t is the union of a finite set of target egds with a finite weakly acyclic set of target tgds. By inspecting the proofs in [12], it is easy to see that the combined complexity of the existence-of-solutions problem for $\mathcal{C}$ is in EXPTIME. Here, we establish a matching lower bound by showing that this problem is EXPTIME-complete. Since $\text{PTIME} \neq \text{EXPTIME}$, this result shows that there is a provable gap between the data complexity and the combined complexity of data exchange. As a matter of fact, the combined complexity of the existence-of-solutions problem remains EXPTIME-complete even for the class of all schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ in which Σ_{st} is a set of full s-t tgds and Σ_t is the union of a set of target egds with a set of full target tgds. These hardness results are proved via rather delicate reductions from the combined complexity of *Datalog sirups* (single-rule Datalog programs), which was shown to be EXPTIME-complete by Gottlob and Papadimitriou [16].

After this, we study the combined complexity of the existence-of-solutions problem for restricted classes of schema mappings and develop a comprehensive picture that is summarized in Tables 1 and 2. In particular, the combined complexity of this problem can be EXPTIME-complete even if the source schema and the target schema are fixed, Σ_{st} is fixed and consists of full s-t tgds, while Σ_t varies and consists of one target egd and one non-full target tgd. In contrast, if the source and the target schemas are fixed, Σ_{st} varies and consists of full s-t tgds, and Σ_t varies and consists of target egds and full target tgds, then the existence-of-solutions problem is always in coNP; moreover, it can be coNP-complete for fixed source and target schemas, and a fixed set Σ_{st} of full s-t tgds.

2. BACKGROUND

A *schema* is a finite collection $\mathbf{R} = (R_1, \ldots, R_k)$ of relation symbols, each of a fixed arity. An *instance* I over $\mathbf{R}$ is a sequence $(R^I_1, \ldots, R^I_k)$ such that each R^I_i is a finite relation of the same arity as R_i. We shall often use R_i to denote both the relation symbol and the relation R^I_i that interprets it. Given a tuple $\mathbf{t}$, we denote by $R(\mathbf{t})$ the association between $\mathbf{t}$ and the relation R where it occurs. Let $\mathbf{S} = (S_1, \ldots, S_n)$ and $\mathbf{T} = (T_1, \ldots, T_m)$ be two disjoint schemas. We refer to $\mathbf{S}$ as the *source* schema and to $\mathbf{T}$ as the *target* schema. Instances over $\mathbf{S}$ are called *source* instances, and instances over $\mathbf{T}$ are called *target* instances.

In this paper, we consider schema mappings of the form $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$, where $\mathbf{S}$ is a source schema, $\mathbf{T}$ is a target schema, Σ_{st} is a finite set of s-t tgds, and Σ_t is the union of a finite set target tgds with a finite set of target egds. As described in Section 1, if

	Schema Mapping	Existence-of-Solutions Problem
Data	Fixed, set of target tgds is arbitrary	Can be undecidable
Complexity	Fixed, set of target tgds is weakly acyclic	In PTIME; can be PTIME-complete
Combined	Varies, set of target tgds is weakly acyclic	In EXPTIME; can be EXPTIME-complete
Complexity	Schemas are fixed, constraints vary, all tgds are full	In coNP; can be coNP-complete

Table 1: Complexity of Data Exchange

I is a source instance, then a *solution for* I is a target instance J such that the pair (I, J) satisfies every constraint in $\Sigma_{st} \cup \Sigma_t$. The *data exchange problem associated with* $\mathcal{M}$ asks, given a source instance I, to construct a solution J for I. The *existence-of-solutions problem for* $\mathcal{M}$ asks: given a source instance I, does a solution for I exist? Clearly, this decision problem underlies the data exchange problem itself, and any algorithm for the data exchange problem will also solve the existence-of-solutions problem. Observe that if $\Sigma_t = \emptyset$, then every source instance has a solution. The situation, however, changes if target constraints are present.

EXAMPLE 2.1. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be the schema mapping in which $\mathbf{S} = \{E\}$, $\mathbf{T} = \{F\}$, and

$$\begin{array}{rcl} \Sigma_{st} & = & \{E(x,z) \rightarrow \exists y (F(x,y) \wedge F(y,z))\} \\ \Sigma_t & = & \{F(x,y) \wedge F(y,z) \rightarrow F(x,z), \\ & & F(x,u) \wedge F(x,v) \rightarrow u = v\} \end{array}$$

Note that the universal quantifiers were omitted in the above constraints; in the sequel, we will often do this, and implicitly assume such quantification. The source instance $I = \{E(1,2)\}$ has a solution $J = \{F(1,2), F(2,2)\}$. In contrast, no solution exists for the source instance $I' = \{E(1,2), E(2,1)\}$.

We now give the definition of a *weakly acyclic* set of target tgds. This crucial concept was formulated by A. Deutsch and L. Popa in 2001, and independently used in [12] and [7] (in the latter paper, under the term *constraints with stratified witness*).

DEFINITION 2.2. Let Σ be a set of tgds over a schema $\mathbf{T}$. Construct a directed graph, called the *dependency graph*, as follows:

(i) Nodes: For every pair (R, A) with R a relation symbol of the schema and A an attribute of R, there is a distinct node; call such a pair (R, A) a *position* (here, the attributes of R are identified with natural numbers between 1 and m, where m is the arity of R).

(ii) Edges: For every tgd $\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y})$ in Σ and for every x in $\mathbf{x}$ that occurs in ψ, and for every occurrence of x in ϕ in position (R, A_i):

1. For every occurrence of x in ψ in position (S, B_j), add an edge $(R, A_i) \rightarrow (S, B_j)$ (if it does not already exist).

2. In addition, for every existentially quantified variable y and for every occurrence of y in ψ in position (T, C_k), add a *special edge* $(R, A_i) \rightarrow (T, C_k)$ (if it does not already exists). Note that there may be two edges in the same direction between two nodes but exactly one of the two edges is special.

• We say that Σ is *weakly acyclic* if the dependency graph has no cycle going through a special edge.

• We say that a tgd θ is *weakly acyclic* if the singleton set $\{\theta\}$ is weakly acyclic. □

EXAMPLE 2.3. The tgd $E(x, y) \rightarrow \exists z\, E(x, z)$ is weakly acyclic; in contrast, the tgd $E(x, y) \rightarrow \exists z\, E(y, z)$ is not, because the dependency graph contains a special self-loop (see Figure 1).

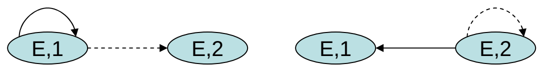

Figure 1: The dependency graphs of $E(x,y) \rightarrow \exists z E(x,z)$ and $E(x,y) \rightarrow \exists z E(y,z)$. Special edges are shown in dotted lines.

Since no existentially quantified variables occur in full tgds, every set of full tgds is weakly acyclic. Note also that if W_1 and W_2 are two weakly acyclic sets of tgds, then their union $W_1 \cup W_2$ need not be weakly acyclic.

In [12], it was shown that weak acyclicity is a sufficient condition for the tractability of the data exchange problem; in particular, it is a sufficient condition for the tractability of the existence-of-solutions problem. The precise result is stated next.

THEOREM 2.4. *([12]) Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a schema mapping such that Σ_{st} is a set of s-t tgds and Σ_t is the union of a set of target egds with a weakly acyclic set of target tgds. Then, there is an algorithm that is based on the chase procedure and has the following properties:*

(1) Given a source instance I, the algorithm determines whether a solution for I exists and, if so, it constructs a solution for I (in fact, it constructs a universal solution).

(2) The running time of the algorithm is bounded by a polynomial in the size of the source instance I.

Note that if $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ is a schema mapping such that Σ_t is a weakly acyclic set of target tgds (in particular, Σ_t contains no target egds), then the existence-of-solutions problem for $\mathcal{M}$ is trivial, because it is easy to see that in this case every source instance has a solution. It follows that if Σ_t is a set of full target tgds, then the existence-of-solutions problem for $\mathcal{M}$ is trivial.

3. DATA COMPLEXITY

The preceding Theorem 2.4 is a result about the *data complexity* of data exchange, because the schema mapping $\mathcal{M}$ is assumed to be fixed. In this section, we investigate the data complexity of data exchange in more depth. We begin with a simple, yet illuminating, result.

3.1 PTIME-Completeness

Computational complexity theory has unveiled fine distinctions between polynomial-time solvable problems. In particular, it has been shown that there exist PTIME-*complete* problems, that is to say, decision problems that are polynomial-time solvable and have the property that every polynomial-time solvable problem is reducible to them via a logarithmic-space reduction. The significance of this is that PTIME-complete problems are regarded as inherently sequential, thus they can not be solved dramatically faster using parallel algorithms (see [24, 18]). Here, we observe that the existence-of-solutions problem may be PTIME-complete, even for schema mappings that satisfy the conditions of Theorem 2.4.

PROPOSITION 3.1. *There exists a schema mapping $\mathcal{M}'$=($\mathbf{S}'$, $\mathbf{T}'$, Σ'_{st}, Σ'_t) with the following properties: (1) Σ'_{st} is a set of full s-t tgds; (2) Σ'_t is a set of full target tgds and a single target egd; and (3) the existence-of-solutions problem for $\mathcal{M}'$ is* PTIME-*complete.*

PROOF. *(Sketch)* We will exhibit a logarithmic-space reduction from HORN 3SAT, the satisfiability problem for Horn formulas with at most three literals per clause; this problem is well known to be PTIME-complete (see [18]).

Let $\mathcal{M}'$=($\mathbf{S}'$, $\mathbf{T}'$, Σ'_{st}, Σ'_t) be the following schema mapping. The source schema $\mathbf{S}'$ consists of two ternary relational symbols P, N, and a unary relational symbol V. The target schema $\mathbf{T}'$ consists of two ternary relational symbols P', N', and three unary relational symbols V', M', W'. The constraints are:

$$\begin{aligned}
\Sigma_{st} :\ & P(x,y,z) \rightarrow P'(x,y,z)\\
& N(x,y,z) \rightarrow N'(x,y,z)\\
& V(x) \rightarrow V'(x)\\
\Sigma_{t} :\ & W'(u) \wedge W'(v) \rightarrow u = v\\
& P'(x,x,x) \rightarrow M'(x)\\
& P'(x,y,z) \wedge M'(y) \wedge M'(z) \rightarrow M'(x)\\
& N'(x,y,z) \wedge M'(x) \wedge M'(y) \wedge M'(z) \wedge V'(u) \rightarrow W'(u)
\end{aligned}$$

Given a Horn 3CNF-formula φ, construct a source instance $I(\varphi)$ in which P and N encode the clauses of φ as follows: $P(x,y,z)$ iff $(x \vee \neg y \vee \neg z)$; similarly, $N(x,y,z)$ iff $(\neg x \vee \neg y \vee \neg z)$. Unit positive clauses $(x \vee x \vee x)$ are encoded by $P(x,x,x)$. Finally, let $V = \{0,1\}$. It is not hard to show that φ is satisfiable if and only there is a solution for the source instance $I(\varphi)$. Intuitively, the constraints in Σ'_t simulate the unit propagation algorithm, which is correct and complete for Horn satisfiability. $\square$

3.2 Undecidability

In this section, we establish one of the main results of this paper. Specifically, we show that if the set of target tgds of a schema mapping $\mathcal{M}$ is not weakly acyclic, then the existence-of-solutions problem for $\mathcal{M}$ may very well be undecidable. Thus, that the weak acyclicity assumption is of the essence in obtaining a polynomial-time algorithm for the existence-of-solutions problem.

Before stating and proving our undecidability result, we need a fair amount of background from universal algebra.

An *algebra* is a structure $\mathbf{A}$ of the form $\mathbf{A} = (A, f_1, \ldots, f_m)$ such that A is a non-empty set, called the *domain* of $\mathbf{A}$, and each f_i is a function from some power A^{k_i} of A to A. The *signature* of an algebra $\mathbf{A} = (A, f_1, \ldots, f_m)$ is the tuple $(k_1, \ldots, k_m)$ where k_i is the arity of the function f_i, $1 \le i \le m$. A *finite algebra* is an algebra whose domain is a finite set. For example, a *semigroup* is an algebra $\mathbf{A} = (A, f)$ such that f is an associative binary function on A; this means that, for every $a, b, c \in A$, we have that $f(f(a,b),c) = f(a, f(b,c))$.

An *identity* (or *equation*) is a formula of the form $\forall \mathbf{x}(s = t)$, where s and t are terms built from variables in $\mathbf{x}$ and function symbols. For example, $\forall x \forall y \forall z (f(f(x,y),z) = f(x, f(y,z)))$ is an identity that axiomatizes associativity. A *quasi-identity* (or *conditional equation*) is a formula of the form

$$\forall \mathbf{x}(s_1 = t_1 \wedge \cdots \wedge s_k = t_k \rightarrow u = v),$$

where $s_1, \ldots, s_k, t_1, \ldots, t_k, u, v$ are terms built from the variables in $\mathbf{x}$ and functions from the algebra. For example, the formula $\forall x \forall y \forall z (f(x,y) = f(z,y) \rightarrow x = z)$ is a quasi-identity that axiomatizes the *right-cancellation* property.

If $\mathcal{K}$ is a class of algebras of the same signature and θ is a quasi-identity, then we write $\mathcal{K} \models \theta$ to denote that every algebra in $\mathcal{K}$ satisfies θ. This concept gives rise to the following fundamental decision problem.

DEFINITION 3.2. *Let $\mathcal{K}$ be a class of algebras. The* uniform word problem for $\mathcal{K}$ *or, simply,* the word problem for $\mathcal{K}$ *asks: given a quasi-identity θ, does $\mathcal{K} \models \theta$?*

The word problem is one of the oldest and most extensively studied problems in algebra and logic. In particular, back in 1947, Post [26] showed that word problem for the class of all semigroups is undecidable. In 1966, Gurevich [19] showed that the word problem for the class of all finite semigroups is undecidable; this follows also from a stronger recursive inseparability result obtained by Gurevich and Lewis [21] much later. As we mentioned in Section 1, the undecidability of the word problem for particular classes of algebras has been used to prove the undecidability of the implication problem for various classes of database dependencies [2, 20].

We now bring into the picture another decision problem, called the *embedding problem*, which is intimately connected to the word problem, but less well known. A *partial algebra* is a structure $\mathbf{B}$ of the form $\mathbf{B} = (B, h_1, \ldots, h_m)$, where B is a non-empty set and each h_i is a partial function from a power B^{k_i} to B (that is, h_i need not be defined on every k_i-tuple from B). Let $\mathbf{B} = (B, h_1, \ldots, h_m)$ be a partial algebra and $\mathbf{A} = (A, f_1, \ldots, f_m)$ an algebra of the same signature. We say that $\mathbf{B}$ is *embeddable in* $\mathbf{A}$ if $B \subseteq A$ and each f_i is an extension of h_i, that is, whenever $h_i(b_1, \ldots, b_{k_i})$ is defined, we have that $f(b_1, \ldots, b_{k_i}) = h(b_1, \ldots, b_{k_i})$.

DEFINITION 3.3. *Let $\mathcal{K}$ be a class of algebras. The* embedding problem for $\mathcal{K}$ *asks: given a finite partial algebra $\mathbf{B}$, is $\mathbf{B}$ embeddable in some algebra $\mathbf{A}$ in $\mathcal{K}$?*

In a series of papers, including [8, 9, 10], Evans investigated the embedding problem for *varieties*, that is, classes of algebras axiomatized by identities. One of his main findings is that the word problem for a variety $\mathcal{K}$ is decidable if and only if the embedding problem for $\mathcal{K}$ is decidable [9]. A somewhat different proof of this result can be found in Grätzer's [17] book. In fact, Grätzer's proof holds for essentially arbitrary classes of algebras.

THEOREM 3.4. *([9, 17]) Let $\mathcal{K}$ be a class of algebras that is closed under isomorphisms. Then the word problem for $\mathcal{K}$ is decidable if and only if the embedding problem for $\mathcal{K}$ is decidable.*

The following result is an immediate consequence of Gurevich's undecidability of the word problem for finite semigroups and Evans's connection between the word problem and the embedding problem.

COROLLARY 3.5. *The embedding problem for the class $\mathcal{S}$ of all finite semigroups is undecidable.*

We now state and prove the key result of this section.

THEOREM 3.6. *There exists a schema mapping $\mathcal{M}^*$ = ($\mathbf{S}^*$, $\mathbf{T}^*$, Σ^*_{st}, Σ^*_t) having the following properties:*

1. *Σ^*_{st} consists of one full s-t tgd.*
2. *Σ^*_t consists of one target egd, one target full tgd, and one non-weakly acyclic target tgd.*
3. *The existence-of-solutions problem for $\mathcal{M}^*$ is undecidable.*

PROOF. We give a schema mapping $\mathcal{M}^*$=($\mathbf{S}^*$, $\mathbf{T}^*$, Σ^*_{st}, Σ^*_t) such that the embedding problem for the class of all finite semigroups is reducible to the existence-of-solutions problem for $\mathcal{M}^*$.

1. $\mathbf{S}^* = \{R\}$ and $\mathbf{T}^* = \{R'\}$, where R and R' are ternary relation symbols. The intuition is that R encodes the *graph* of a partial binary function, and R' encodes the *graph* of a (total) binary function.

2. $\Sigma^*_{st} = \{R(x,y,z) \rightarrow R'(x,y,z)\}$.

3. Σ_t has the following members:

(i) A target egd asserting that R' is a partial function:

$$R'(x,y,z) \wedge R'(x,y,w) \rightarrow z = w.$$

(ii) A target full tgd asserting that the partial function encoded by R' is associative:

$$R'(x,y,u) \wedge R'(y,z,v) \wedge R'(u,z,w) \rightarrow R'(x,v,w).$$

(iii) A target non-full tgd asserting that the partial function encoded by R' is actually total; this last property is expressed by asserting that if two elements u and v occur in two triples in R', then there is an element w such that $R'(u,v,w)$:

$$\begin{aligned} R'(x,y,z) \wedge R'(x',y',z') \rightarrow \exists w_1 \cdots \exists w_9 \\ (R'(x,x',w_1) \wedge R'(x,y',w_2) \wedge R'(x,z',w_3) \wedge \\ R'(y,x',w_4) \wedge R'(y,y',w_5) \wedge R'(y,z',w_6) \wedge \\ R'(z,x',w_7) \wedge R'(z,y',w_8) \wedge R'(z,z',w_9)). \end{aligned}$$

The target non-full tgd in Σ_t is *not* weakly acyclic, since its dependency graph contains a special edge from $(R',1)$ to $(R',3)$, and an edge from $(R',3)$ to $(R',1)$ (where $1,2,3$ are the attributes of R').

Given a partial algebra $\mathbf{B} = (B,g)$ with g a partial binary function, construct the source instance $R(\mathbf{B}) = \{(a,b,c) \in B^3 : g(a,b) = c\}$. It is easy to see that $\mathbf{B}$ is embeddable in a finite semigroup if and only if there is a solution for $R(\mathbf{B})$ under $\mathcal{M}^*$. □

Note that, by definition, a *solution* in data exchange is a set of finite relations, since a database instance is a set of finite relations. Even if the definitions in data exchange are relaxed to allow for sets of finite or infinite relations as solutions, the above schema mapping $\mathcal{M}^*$ can also be used to show that the (relaxed) existence-of-solutions problem is undecidable. For this, instead of the undecidability of the embedding problem for finite semigroups, we use the undecidability of the embedding problem for semigroups, which follows from Post's [26] undecidability of the word problem for semigroups and Evans' Theorem 3.4 in this section. In contrast, it follows from results by Calì et al. [5] that the existence of finite or infinite solutions is decidable for schema mappings $\mathcal{M}$ in which Σ_t consists of key constraints and foreign-key constraints (the latter need not form a weakly acyclic set).

4. COMBINED COMPLEXITY

In what follows, we investigate the combined complexity of the existence-of-solutions problem, that is to say, we study this problem when the input consists of both a schema mapping and a source instance. We begin by formalizing the *existence-of-solutions problem for a class $\mathcal{C}$ of schema mappings* and then proceed to investigate the complexity of this problem.

DEFINITION 4.1. Let $\mathcal{C}$ be a class of schema mappings. The *existence-of-solutions problem for $\mathcal{C}$* is the following decision problem: given a schema mapping $\mathcal{M}$ in $\mathcal{C}$, and a source instance I, does a solution for I under $\mathcal{M}$ exists?

In view of our undecidability Theorem 3.6, we will only consider schema mappings in which the target tgds form a weakly acyclic set. We begin by deriving an EXPTIME upper bound for the existence-of-solutions problem for the class of all schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ in which Σ_{st} is a set of s-t tgds and Σ_t is the union of a set of target egds with a weakly acyclic set of target tgds. Moreover, we identify a family of restricted classes of schema mappings for which this problem is in coNP. We then establish a matching EXPTIME-hard lower bound for the general case and a matching coNP-hard lower bound for the restricted one; furthermore, we prove that slight relaxations of the restrictions cause the complexity to jump from coNP to EXPTIME-hard.

4.1 Combined Complexity: Upper Bounds

An inspection of the proof of Theorem 3.9 of [12] reveals that the following result holds.

THEOREM 4.2. *(Implicit in [12]) Let $\mathcal{E}$ be the class of all schema mappings $(\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ such that Σ_{st} is a set of s-t tgds, and Σ_t is the union of a set of egds with a weakly acyclic set of tgds. The existence-of-solutions problem for $\mathcal{E}$ is in* EXPTIME.

PROOF. *(Hint)* This follows from the proof of Theorem 3.9 of [12]. It was shown in [12] that the length of every chase sequence of I with $\Sigma_{st} \cup \Sigma_t$ is bounded by a polynomial in the size of the instance I. This expression is exponential in the size of $\Sigma_{st} \cup \Sigma_t$ and the size of $\mathbf{T}$. □

In contrast, for the classes of schema mappings such that the source and target schemas are fixed, and the s-t tgds and the target tgds vary but are full, the existence-of-solutions problem is in coNP.

THEOREM 4.3. *Let $\mathbf{S}^*$ be a fixed source schema and $\mathbf{T}^*$ a fixed target schema. Let $\mathcal{C}(\mathbf{S}^*, \mathbf{T}^*)$ be the class of all schema mappings $(\mathbf{S}^*, \mathbf{T}^*, \Sigma_{st}, \Sigma_t)$ such that Σ_{st} is a set of full s-t tgds and Σ_t is the union of a set of egds with a set of full tgds. The existence-of-solutions problem for $\mathcal{C}(\mathbf{S}^*, \mathbf{T}^*)$ is in* coNP.

PROOF. Since $\Sigma_{st} \cup \Sigma_t$ consists of only full tgds, any solution will have at most $|I|^{|\mathbf{T}|}$ number of tuples, where $|I|$ denotes the number distinct values in I and $|\mathbf{T}|$ denotes the sum of arities of each relation symbol in $\mathbf{T}$. Since $\mathbf{T}$ is fixed, the size of any solution is polynomial in the size of I.

If there is no solution, it must be that some egd in Σ_t is violated. We can therefore guess a failing chase sequence, where the last chase step is an application of the egd that causes the violation. Since the tgds in Σ_{st} and Σ_t are full, only the last chase step in the chase sequence is an egd chase step; every other chase step in this sequence is a tgd chase step. Because there are polynomially many tuples that can be generated in the target and each tgd chase step generates at least one new tuple in the target instance, we conclude that the length of this failing chase sequence is polynomial in the size of I. Clearly, verifying that this is a failing chase sequence can be done in polynomial time: (1) for every chase step that generates K_2 from K_1 with tgd σ and homomorphism h, let σ be of the form $\phi(\mathbf{x}) \rightarrow \psi(\mathbf{x})$. We verify that every tuple in $\phi(h(\mathbf{x}))$ is from K_1, some tuple in $\psi(h(\mathbf{x}))$ does not exist in K_1, and that K_2 is the union of K_1 with the tuples $\psi(h(\mathbf{x}))$. This verification takes polynomial time. For the last step where an egd of the form $\phi'(\mathbf{x}) \rightarrow x_1 = x_2$ is applied on the instance K_1, we verify that every tuple in $\phi'(h(\mathbf{x}))$ is from K_1 and that $h(x_1) \neq h(x_2)$. Clearly, this verification step also takes polynomial time. □

It is worth noting that if Σ_t consists of only a weakly acyclic set of tgds (and no egds), then every source instance has a solution and so the existence-of-solutions problem is trivial. Indeed, in this case, the algorithm of [12] for computing a solution will always terminate and produce a solution, because no finite chase is failing. This observation and the preceding Theorems 4.2 and 4.3 are summarized in the first three rows of Table 2.

We now proceed to derive EXPTIME-hard and coNP-hard lower bounds; the presence of target egds will play a crucial role in proving these results.

	$\mathbf{S}, \mathbf{T}$	Σ_{st}: tgds	Σ_t: egds and a weakly acyclic set of tgds	Combined Complexity	Theorem
Upper Bounds	vary	varies	varies	in EXPTIME	4.2
	fixed	varies; full tgds only	varies; egds and full tgds only	in coNP	4.3
	vary	varies	varies; no egds	trivial	-
Lower Bounds	vary	varies; full tgds only	varies; egds and full tgds only	can be EXPTIME-complete	4.5
	fixed	varies	varies; egds only	can be EXPTIME-complete	4.8
	fixed	fixed; full tgds only	varies	can be EXPTIME-complete	4.9
	fixed	fixed; full tgds only	varies; egds only	can be coNP-complete	4.10
	fixed	varies; full tgds only	fixed; egds only	can be coNP-complete	4.12

Table 2: Combined Complexity of Data Exchange

4.2 Combined Complexity: Lower Bounds

In this section, we show that the existence-of-solutions problem for the class $\mathcal{E}$ of schema mappings in Theorem 4.2 is EXPTIME-complete. We also show that there are fixed schema mappings $\mathbf{S}^*$ and $\mathbf{T}^*$ such that the existence-of-solutions problem for the class $\mathcal{C}(\mathbf{S}^*, \mathbf{T}^*)$ in Theorem 4.3 is coNP-complete. In addition, we establish that if we consider slight extensions of the classes $\mathcal{C}(\mathbf{S}^*, \mathbf{T}^*)$, then the existence-of-solutions problem may become EXPTIME-complete. Specifically, EXPTIME-completeness holds for the following extensions of the class $C(\mathbf{S}^*, \mathbf{T}^*)$:

1. the schemas are allowed to vary (Theorem 4.5);
2. the s-t tgds are non-full (Theorem 4.8);
3. the target tgds are non-full (Theorem 4.9).

Actually, Theorem 4.8 holds even when all target dependencies are egds, and Theorem 4.9 holds even when the s-t tgds are fixed and full. These results are summarized in the first three rows of the lower part of Table 2.

We also show that the existence-of-solutions problem may be coNP-complete even for a class of schema mappings in which the schemas are fixed, the s-t tgds are full and held fixed, and the target dependencies are allowed to vary, but all are egds. (Theorem 4.10). Recall that the existence-of-solutions problem is solvable in polynomial time for a fixed schema mapping with target egds as its only target constraints. Hence, the slight relaxation that allows the target egds to vary may cause the complexity to become coNP-complete. Finally, we show that the existence-of-solutions problem may be coNP-complete for another slight relaxation in which the target egds are fixed, but the full s-t tgds are allowed to vary (Theorem 4.12). These results are summarised in the last two rows of Table 2.

4.2.1 EXPTIME-Completeness

Our EXPTIME-hardness reductions make use of a combined complexity result of *Datalog sirups* from Gottlob and Papadimitriou [16], which we describe next.

Single rule Datalog programs A *single rule program (sirup)* is a Datalog program with one rule and a number of *initializations* consisting of *facts*. The rule is of the form: $A_0 \leftarrow A_1, ..., A_m$, where each A_i, $0 \leq i \leq m$, is an atom. An *atom* is a formula $p(t_1, ..., t_n)$, where p is a predicate symbol of arity n and t_i, $1 \leq i \leq n$, is a variable or a constant. The *head* of the rule is A_0, while $A_1, ..., A_m$ is the *body* of the rule. The predicate symbol that appears in A_0 is called an *intensional database predicate (idb)* symbol, while those that appear only in the body of the rule are called *extensional database predicate (edb)* symbols. A rule of the form $A_0 \leftarrow$, (with an empty body) is a *fact*. We say the fact is *ground* if every term that appears in A_0 is a constant. The edb symbols occur in a database $\mathbf{D}$ and if the idb symbol also occurs in the body of a sirup rule, it is initialized by facts. A *single ground fact (SGF) sirup* is a Datalog program with one rule and at most one ground fact. Gottlob and Papadimitriou [16] investigated the combined complexity of Datalog programs, which is the following decision problem: given a Datalog program P, a database $\mathbf{D}$, and a ground fact δ, is it the case that $P \cup \mathbf{D} \models \delta$? In other words, is δ derivable from $\mathbf{D}$ via P? They showed that even if Datalog programs are limited to SGF sirups, the combined complexity is EXPTIME-complete. We refer to this decision problem as the *SGF sirup problem.*

THEOREM 4.4. *(Combined complexity of SGF sirups [16]) The combined complexity of the SGF sirup problem is* EXPTIME-*complete.*

Using the preceding theorem, we show that the existence-of-solutions problem for $\mathcal{C}$ is EXPTIME-hard, where $\mathcal{C}$ is the class of schema mappings such that Σ_{st} consists of only full s-t tgds, and Σ_t is the union of a set of egds with a set of full tgds. Observe that this is only a slight extension of the class $\mathcal{C}(\mathbf{S}^*, \mathbf{T}^*)$ of schema mappings in Theorem 4.3; here, the source and target schemas are allowed to vary.

THEOREM 4.5. *Let $\mathcal{C}$ be the class of all schema mappings ($\mathbf{S}$, $\mathbf{T}$, Σ_{st}, Σ_t), where Σ_{st} is a set of full s-t tgds and Σ_t is the union of a set of egds with a set of full tgds. The existence-of-solutions problem for $\mathcal{C}$ is* EXPTIME-*hard.*

PROOF. We are given a SGF sirup P, a database $\mathbf{D}$, and a fact δ which we wish to determine whether $P \cup \mathbf{D} \models \delta$. Let the sirup rule be of the form: $A(\mathbf{x}) \leftarrow \mathbf{Q}_1(\mathbf{x_1}), \ldots, \mathbf{Q}_n(\mathbf{x_n})$, where each symbol $\mathbf{Q}_i$, $1 \leq i \leq n$, either represents an extensional database predicate or the intensional database predicate A. If A occurs among $\mathbf{Q}_1, ..., \mathbf{Q}_n$, then A is initialized by a ground fact. Let the arity of A be k and let δ denote the fact $A(c_1, ..., c_k)$.

Based on P, $\mathbf{D}$, δ, and the ground fact, we construct a schema mapping and a source instance I as follows: The source schema $\mathbf{S}$ consists of all the extensional database predicates $R_1, ..., R_m$ as well as two relational symbols A and W. Each R_i, $1 \leq i \leq m$, has the same arity as the corresponding relation in $\mathbf{D}$, A has arity k, and W has arity $k + 1$. The target schema is isomorphic to $\mathbf{S}$. It consists of the relational symbols $R'_1, \ldots, R'_m$, A', and W'. For Σ_{st}, we create $m + 2$ full s-t tgds to copy each source relation to its corresponding target relation:

$$\begin{aligned}\Sigma_{st} : & R_i(\mathbf{x}) \rightarrow R'_i(\mathbf{x}), 1 \leq i \leq m \\ & A(\mathbf{y}) \rightarrow A'(\mathbf{y}) \\ & W(\mathbf{z}) \rightarrow W'(\mathbf{z})\end{aligned}$$

Next, we construct two dependencies in Σ_t. The first is a full tgd that "implements" the sirup rule. The second is an egd that tests the existence of δ.

$$\begin{aligned}\Sigma_t : & \mathbf{Q}'_1(\mathbf{x}_1) \wedge \cdots \wedge \mathbf{Q}'_n(\mathbf{x}_n) \rightarrow A'(\mathbf{x}) \\ & A'(x_1, x_2, \ldots, x_k) \wedge W'(x_1, x_2, \ldots, x_k, y) \rightarrow x_1 = y\end{aligned}$$

In the first tgd, $\mathbf{Q}_i$ represents the ith predicate symbol, $1 \leq i \leq m$, in the body of the sirup rule P. For example, if the sirup rule is $A(x_1) \leftarrow R_2(x_1, x_2), R_1(x_2, x_3), A(x_3)$, then the first tgd in Σ_t is $R_2'(x_1, x_2) \wedge R_1'(x_2, x_3) \wedge A'(x_3) \rightarrow A'(x_1)$.

We create a source instance I as follows: First, we populate $R_1, \ldots, R_m$, and A with the tuples from the corresponding relations in $\mathbf{D}$ and the ground fact, respectively. Then, we populate W with a single tuple $(c_1, c_2, \ldots, c_k, d)$, where d is a fresh symbol that does not occur elsewhere in I.

Clearly, the schema mapping and source instance I can be constructed with one pass through the sirup input P, $\mathbf{D}$, δ, and the ground fact. We show that there is no solution for I under the constructed schema mapping if and only if $P \cup \mathbf{D} \models \delta$. If there is no solution, then there is a failing chase sequence that applies the egd in Σ_t. Since any application of the egd in a chase sequence must make use of the tuple $W'(c_1, ..., c_k, d)$, it must be that $A'(c_1, ..., c_k)$ also exists in the target instance. Thus, we conclude that δ is derivable from $\mathbf{D}$ via P. Conversely, if $P \cup \mathbf{D} \models \delta$, then chasing I with $\Sigma_{st} \cup \Sigma_t$ will produce $A'(c_1, ..., c_k)$. Since $W'(c_1, ..., c_k, d)$ exists in the target instance, the egd can be applied and therefore, the chase fails. This means that there is no solution. □

COROLLARY 4.6. *Let $\mathcal{C}$ be the class of all schema mappings $(\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$, where Σ_{st} is a set of s-t tgds and Σ_t is the union of a set of egds with a weakly acyclic set of tgds. The existence-of-solutions problem for $\mathcal{C}$ is* EXPTIME*-complete.*

The above corollary is a consequence of Theorem 4.2 and Theorem 4.5. Observe that in the reduction of Theorem 4.5, a target tgd was used to implement the sirup rule. In Theorem 4.8, we show that even when the schemas are kept fixed and there are no target tgds, but we allow non-full s-t tgds, then the EXPTIME-hard lower bound continues to hold. Towards Theorem 4.8, we first prove a lemma. The proof of this lemma shows how one can avoid using a target tgd to implement the sirup rule.

LEMMA 4.7. *Let $\mathcal{C}$ be the class of all schema mappings ($\mathbf{S}$, $\mathbf{T}$, Σ_{st}, Σ_t), where Σ_t consists of only egds. The existence-of-solutions problem for $\mathcal{C}$ is* EXPTIME*-hard.*

PROOF. To avoid using a target tgd to implement the sirup rule, we eagerly pre-compute all possible tuples that can be returned by the sirup rule with a s-t tgd. We do this by generating all possible k-tuples with a s-t tgd, where k is the arity of the head of the sirup rule, with values from the source instance. Along with every possible k-tuple, this s-t tgd will also generate a null that is used to indicate whether this k-tuple is in the result of the sirup rule. Hence, the s-t tgd is non-full. Subsequently, an egd in Σ_t is used to simulate the sirup rule and if a k-tuple is indeed in the result of the sirup rule, the egd will convert the associated null of that k-tuple into some constant. Thus, one can determine whether a k-tuple is part of the result of a sirup rule depending on whether the tuple is associated with a null or a constant. As in the proof of Theorem 4.5, another egd is used to detect when the given fact δ is produced and if so, equates one constant with another to obtain a violation. We describe the proof in detail next.

This proof is by reduction from the SGF sirup problem. We are given a SGF sirup rule P, a database $\mathbf{D}$, and a fact δ which we wish to determine whether $P \cup \mathbf{D} \models \delta$. Let the sirup rule be of the form: $A(\mathbf{x}) \leftarrow \mathbf{Q}_1(\mathbf{x_1}), \ldots, \mathbf{Q}_n(\mathbf{x_n})$, where each symbol $\mathbf{Q}_i$, $1 \leq i \leq n$, either represents an extensional database predicate or the intensional database predicate A. If A occurs among $\mathbf{Q}_1, ..., \mathbf{Q}_n$, then we assume that there is a ground fact that initializes A. Let the arity of A be k and let δ denote the fact $A(c_1, ..., c_k)$.

Based on P, $\mathbf{D}$, δ, and the ground fact, we construct a schema mapping and a source instance I as follows: The source schema $\mathbf{S}$ consists of all the extensional database predicates $R_1, ..., R_m$ as well as four relational symbols A, W, U, and V. Note that the relational symbol A in $\mathbf{S}$ is the same as the intensional database predicate of the sirup P. Each R_i, $1 \leq i \leq m$, has the same arity as the corresponding relation in $\mathbf{D}$, and A has arity k, W has arity $k+1$ and the relations U and V are unary. The target schema consists of the relational symbols $R_1', \ldots, R_m'$, A', W', U', and V'. The arities of each relation in $\mathbf{T}$ is the same as the corresponding relation in $\mathbf{S}$ except that A' has arity $k+1$. For Σ_{st}, we create the following s-t tgds:

$$\begin{aligned}
\Sigma_{st} : \; & R_1(\mathbf{x}) \rightarrow R_1'(\mathbf{x}) \\
& \ldots \\
& R_m(\mathbf{x}) \rightarrow R_m'(\mathbf{x}) \\
& U(x) \rightarrow U'(x) \\
& W(x_1, ..., x_k, x) \rightarrow W'(x_1, ..., x_k, x) \\
& U(x) \wedge A(x_1, ..., x_k) \rightarrow A'(x_1, ..., x_k, x) \\
& V(x_1) \wedge ... \wedge V(x_k) \rightarrow \exists N \, A'(x_1, ..., x_k, N)
\end{aligned}$$

The dependencies in Σ_t consists of the following egds:

$$\begin{aligned}
\Sigma_t : \; & U'(y) \wedge A'(x_1, ..., x_k, x) \wedge \mathbf{Q}_1'(\mathbf{y}_1) \wedge ... \wedge \mathbf{Q}_n'(\mathbf{y}_n) \\
& \rightarrow x = y \\
& U'(y) \wedge W'(x_1, ..., x_k, z) \wedge A'(x_1, ..., x_k, y) \rightarrow y = z
\end{aligned}$$

The symbol $\mathbf{Q}_i'$, $1 \leq i \leq n$, represents the underlying predicate symbols. We create the source instance I as follows. As in the proof of Theorem 4.5, we populate $R_1, \ldots, R_n$, and A with tuples from the corresponding relations in $\mathbf{D}$ and the ground fact. For W, we create a single tuple $(c_1, c_2, \ldots, c_k, d)$ where d is a fresh value that does not occur elsewhere in I. Lastly, for U, we create single tuple with the value "1", and for V, we insert all distinct values of P, $\mathbf{D}$, δ, and the ground fact, each as a unary tuple in V.

The first m tgds of Σ_{st} copies the R_i to R_i', where $1 \leq i \leq m$. The next two s-t tgds copy tuples in U and W to U' and W', respectively. The last two s-t tgds in Σ_{st} build the relation A': one tgd copies every tuple from A to A' along with an additional column which contains the value "1". The last tgd generates all possible tuples composed of values from the source instance I in the target relation A', along with a labeled null. Intuitively, a labeled null indicates that the tuple is not in the result of applying the sirup rule, and a value "1" indicates otherwise. The first egd in Σ_t simulates the sirup rule. It sets the last column v of a tuple $(a_1, ..., a_k, v)$ in A' to "1" if and only if $P \cup \mathbf{D} \models A(a_1, ..., a_k)$. For example, if the sirup rule is $A(x_1) \leftarrow R_2(x_1, x_2), R_1(x_2, x_3), A(x_3)$, then the first egd in Σ_t is $U'(y) \wedge A'(x_1, x) \wedge R_2'(x_1, x_2) \wedge R_1'(x_2, x_3) \wedge A'(x_3, y) \rightarrow x = y$. The second egd has a similar effect to that of the egd in the proof of Theorem 4.5. It detects whether the fact δ is in the result of the sirup rule. If so, this egd will be violated as it equates "d" with "1".

It is easy to see that the schema mapping and the source instance I can be constructed in polynomial time. Moreover, there is no solution to the schema mapping with source instance I if and only if $P \cup \mathbf{D} \models \delta$. If there is no solution, we know that the chase must fail on one of the egds in Σ_t. Since the chase of I with Σ_{st} only adds the tuple "1" to U' and no other tuples are added to U', we know that y is always bound to the value "1" with the first egd in Σ_t. Since the last column of every tuple in A' is either a labeled null or the value "1", this egd cannot cause a violation. Hence, we conclude that the violation is due to the second egd in Σ_t. Since the chase only adds the tuple $(c_1, ..., c_k, d)$ to W' and y always binds to the value "1", the violation must have been due to the existence of a tuple $A'(c_1, ..., c_k, 1)$. Therefore, we conclude that $P \cup \mathbf{D} \models \delta$. It is easy to see that if $P \cup \mathbf{D} \models \delta$, then there will be no solution since the last egd of Σ_t can never be satisfied. □

Using Lemma 4.7, we are now ready to show that the EXPTIME-hard lower bound continues to hold even when we fix the source and target schemas.

THEOREM 4.8. *There exists a fixed source schema $\mathbf{S}^*$ and a fixed target schema $\mathbf{T}^*$ such that the existence-of-solutions problem for $\mathcal{C}^*$ is* EXPTIME*-hard, where $\mathcal{C}^*$ is the class of all schema mappings $(\mathbf{S}^*, \mathbf{T}^*, \Sigma_{st}, \Sigma_t)$ such that Σ_{st} consists of s-t tgds with at most two existentially-quantified variables, and Σ_t consists of only two egds.*

PROOF. We describe the idea behind how we fix the source and target schema before we present the proof. The fixed source schema has a 5-ary relation symbol G and we use a trick to encode every tuple of every relation of $\mathbf{D}$ in G. There is also a binary relation O that keeps a linear ordering of the extensional and intensional predicate symbols of $P \cup \mathbf{D}$. For example, suppose P is the sirup rule $A(x_1) \leftarrow R_2(x_1, x_2), R_1(x_2, x_3), A(x_3)$, and $R_1 = \{(a, a)\}$, $R_2 = \{ (c, d), (e, f) \}$, and the ground fact is $A(b) \leftarrow$. Then, G and O are shown below:

G:

R_1	1	2	id1	a
R_1	2	3	id1	a
R_2	1	2	id2	c
R_2	2	3	id2	d
R_2	1	2	id3	e
R_2	2	3	id3	f
A	1	2	id5	b

O:

R_1	R_2
R_2	A

Intuitively, the first two tuples in G represent $R_1(a, a)$. The next two tuples in G represent $R_2(c, d)$ and so on. The last tuple in G represents the ground fact $A(b)$. The first column of a tuple in G stores the relation name that the tuple belongs to. The fourth column of a tuple in G contains the identifier of the original tuple. It is used to identify the set of tuples in G that describe the same original tuple. The second and third columns of a tuple in G encode the position of the original tuple that the value in the last column of this tuple belongs to. For the two tuples with identifier id2, the second and third column values are (1,2) and (2,3), respectively. Hence, we conclude that the values c and d belong to the first column and respectively, second column of the original tuple in R_2. Note that a pair of numbers (1,2) is used to store the position of c, instead of a single number, in order to retrieve the values of the original tuple in the correct order. It is easy to see that one can reconstruct the database $\mathbf{D}$ from G. The relation O stores a linear order of the relations in $\mathbf{D}$ which we will need in order to retrieve tuples from the desired relations. Using the same idea behind the construction of G, we store the fact δ in a separate fixed arity relation, say, W. Hence, we can fix the source schema in the proof of Lemma 4.7 with this encoding. The fixed target schema is isomorphic to the source schema but consists of disjoint relation symbols. The s-t tgds essentially copies each source relations to its corresponding target relation. There is a non-full s-t tgd that generates, in advance, all possible output tuples as described in the proof sketch of Lemma 4.7. To store every possible output tuple of a sirup rule as several tuples in a fixed arity relation, we now need an additional existentially-quantified variable to generate tuple identifiers.

We prove our result by a reduction from the SGF sirup problem. As before, we are given a SGF sirup P, a database $\mathbf{D}$, and a fact δ which we wish to determine whether $P \cup \mathbf{D} \models \delta$. Let the sirup rule be of the form: $A(\mathbf{x}) \leftarrow \mathbf{Q}_1(\mathbf{x_1}), \ldots, \mathbf{Q}_n(\mathbf{x_n})$, where each symbol $\mathbf{Q}_i$, $1 \leq i \leq n$, either represents an extensional database predicate or the intensional database predicate A. If A occurs among $\mathbf{Q}_1, ..., \mathbf{Q}_n$, then there is a ground fact that initializes A. Let the arity of A be k and let δ denote the fact $A(c_1, ..., c_k)$.

The source schema $\mathbf{S}$ contains the relational symbols O, G, W, A, U, and V, where O is a binary relation that stores a linear order of the relations involved, G is a 5-ary relation that stores all tuples of each relation, W is a ternary relation that stores the fact δ, and A is a ternary relation that stores the initialization tuple, if any. Note that the relational symbol A in $\mathbf{S}$ is the same as the intensional database predicate of the sirup P. The relations U and V are unary, where U stores the single value "1" and V stores all possible values that occur in P, $\mathbf{D}$ and δ. The target schema $\mathbf{T}$ is an isomorphic copy of $\mathbf{S}$ and consists of the relational symbols O', G', W', A', U', and V'.

Next, we show how the source instance I is constructed based on P, $\mathbf{D}$, δ, and the ground fact before we describe the construction of Σ_{st} and Σ_t. Suppose $R_1, \ldots R_m$ are the extensional database predicates, and A is the only intensional database predicate. We now construct O to contain a linear order on the relational symbols that occur in $\mathbf{D}$ and A. That is, we define O=$\{(R_1, R_2)$, ..., (R_{m-1}, R_m), $(R_m, A)\}$. Next, we construct the relation W to hold δ. That is, $W = \{(1, 2, c_1)$, $(2, 3, c_2)$,...,$(k, k+1, c_k)$, $(k+1, k+2, d)\}$. Intuitively, the ordering of the values in δ is maintained by the pair of numbers in the first two columns of each tuple in W. The value d is not used anywhere else and stored as an extra column to the tuple δ. We will subsequently use the existence of d to test if a solution was found. Finally, we construct the relation G. For every tuple $(a_1, \ldots, a_{k_i})$ of a relation R_i, where R_i has arity k_i, we create k_i tuples in G using the trick we described earlier: $(R_i, 1, 2, n, a_1)$,$(R_i, 2, 3, n, a_2)$, ..., $(R_i, k_i, k_i + 1, n, a_{k_i})$. The fourth column holds a value n, which is a unique identifier for the tuple and every tuple has a different identifier. We repeat this process for the ground fact, if any. We create a single tuple "1" in U and for V, we insert all distinct values of P, $\mathbf{D}$, δ, and the ground fact, each as a unary tuple in V.

We describe next the construction of Σ_{st} and Σ_t with an example first. Suppose the sirup rule is $A(x_1) \leftarrow R_2(x_1, x_2)$, $R_1(x_2, x_3)$, $A(x_3)$. There are tgds in Σ_{st} that copy each relation G, U, W, and O in the source to G', U', W', and O' in the target. In addition, there are two tgds that build the relation A' in G in the target, similar to what was described in Lemma 4.7.

$$U(y) \wedge O(u_1, u_2) \wedge O(u_2, u_0) \wedge W(s_1, s_2, z_1) \wedge W(s_2, s_3, z_2) \wedge$$
$$G(u_0, p_1, p_2, y_0, x) \rightarrow G'(u_0, s_1, s_2, y_0, x) \wedge G'(u_0, s_2, s_3, y_0, y)$$

$$V(x) \wedge O(u_1, u_2) \wedge O(u_2, u_0) \wedge W(s_1, s_2, z_1) \wedge W(s_2, s_3, z_2)$$
$$\rightarrow \exists N \exists M\, G'(u_0, s_1, s_2, N, x) \wedge G'(u_0, s_2, s_3, N, M)$$

The first s-t tgd copies every tuple in G that belongs to A over to G' and extends the tuple with a value "1" at the second column. The second s-t tgd generates every possible output tuple of the sirup rule along with a labeled null N that is the identifier of the output tuple. An additional labeled null M is also generated at the second column which will be used to indicate whether this tuple exists in the result of the sirup rule. The first egd in Σ_t is

$$U'(y) \wedge O'(u_1, u_2) \wedge O'(u_2, u_0) \wedge$$
$$G'(u_0, s_1, s_2, y_0, x_1) \wedge G'(u_0, s_2, s_3, y_0, z) \wedge$$
$$G'(u_2, p_1, p_2, y_1, x_1) \wedge G'(u_2, p_2, p_3, y_1, x_2) \wedge$$
$$G'(u_1, q_1, q_2, y_2, x_2) \wedge G'(u_1, q_2, q_3, y_2, x_3) \wedge$$
$$G'(u_0, r_1, r_2, y_3, x_3) \wedge G'(u_0, r_2, r_3, y_3, y) \rightarrow y = z$$

and the second egd is

$$U'(y) \wedge O'(u_1, u_2) \wedge O'(u_2, u_0) \wedge W'(v_1, v_2, x) \wedge W'(v_2, v_3, z)$$
$$G'(u_0, r_1, r_2, y_0, x) \wedge G'(u_0, r_2, r_3, y_0, y) \rightarrow y = z$$

These egds are a "rewriting" of the egds in Σ_t of the proof of Lemma 4.7 based on the fixed schema $\mathbf{T}$. The first egd simulates the sirup rule and sets the last column of a tuple in G' to "1" whenever the tuple is generated by the sirup rule. The second egd detects if δ has been produced.

More precisely, the set Σ_{st} consists of the following s-t tgds:

1) $G(u,v,x,y,z) \rightarrow G'(u,v,x,y,z)$
2) $U(x) \rightarrow U'(x)$
3) $W(x_1,x_2,y) \rightarrow W'(x_1,x_2,y)$
4) $O(x,y) \rightarrow O'(x,y)$
5) $U(y) \wedge O(u_1,u_2) \wedge ... \wedge O(u_{m-1},u_m) \wedge O(u_m,u_0) \wedge$
$W(s_1,s_2,z_1) \wedge ... \wedge W(s_{k+1},s_{k+2},z_{k+1})$
$G(u_0,s_1,s_2,y_0,x_1) \wedge ... \wedge G(u_0,s_k,s_{k+1},y_0,x_k)$
$\rightarrow G'(u_0,s_1,s_2,y_0,x_1) \wedge ... \wedge G'(u_0,s_k,s_{k+1},y_0,x_k) \wedge$
$G'(u_0,s_{k+1},s_{k+2},y_0,y)$
6) $V(x_1) \wedge ... \wedge V(x_k) \wedge W(s_1,s_2,z_1) \wedge ... \wedge W(s_{k+1},s_{k+2},z_{k+1})$
$\wedge O(u_1,u_2) \wedge ... \wedge O(u_{m-1},u_m) \wedge O(u_m,u_0)$
$\rightarrow \exists N \exists M(G'(u_0,s_1,s_2,N,x_1) \wedge ... \wedge G'(u_0,s_k,s_{k+1},N,x_k) \wedge$
$G'(u_0,s_{k+1},s_{k+2},N,M))$

and Σ_t consists of the following egds:

1) $U'(y) \wedge O'(u_1,u_2) \wedge ... \wedge O'(u_{m-1},u_m) \wedge O'(u_m,u_0) \wedge$
$G'(u_0,s_1,s_2,y_0,z_1) \wedge ... \wedge G'(u_0,s_k,s_{k+1},y_0,z_k) \wedge$
$G'(u_0,s_{k+1},s_{k+2},y_0,z) \wedge$
$G'(\mathbf{q_1},s_1,s_2,y_1,x_1^1) \wedge ... \wedge G'(\mathbf{q_1},s_{k_1},s_{k_1+1},y_1,x_{k_1}^1)$
...
$G'(\mathbf{q_n},s_1,s_2,y_n,x_1^n) \wedge ... \wedge G'(\mathbf{q_n},s_{k_n},s_{k_n+1},y_n,x_{k_n}^n)$
$\rightarrow z = y$
2) $U'(y) \wedge O'(u_1,u_2) \wedge ... \wedge O'(u_{m-1},u_m) \wedge O'(u_m,u_0) \wedge$
$W'(s_1,s_2,x_1) \wedge ... \wedge W'(s_k,s_{k+1},x_k) \wedge W'(s_{k+1},s_{k+2},z) \wedge$
$G'(u_0,p_1,p_2,y_0,x_1) \wedge ... \wedge G'(u_0,p_k,p_{k+1},y_0,x_k) \wedge$
$G'(u_0,p_{k+1},p_{k+2},y_0,y) \wedge$
$\rightarrow y = z$

In the first egd above, the symbol $\mathbf{q}_i$ represents variable u_j if the predicate symbol of the ith subgoal in the body of the datalog rule is the relation symbol R_j. It represents u_0 if predicate symbol is A. The symbol x_j^i, where $1 \leq i \leq n$ and $1 \leq j \leq k_i$, represents the jth variable in the ith subgoal in the body of the datalog rule. Similarly, the symbol z_i, where $1 \leq i \leq k$, represents the ith variable in the head of the datalog rule.

A similar argument as in the proof of Lemma 4.7 shows that there is no solution to the schema mapping with source instance I if and only if $P \cup \mathbf{D} \models \delta$. ☐

A slight extension of the classes $\mathcal{C}(\mathbf{S}^*, \mathbf{T}^*)$ of schema mappings with a non-full but weakly acyclic target tgd may also cause the complexity of the existence-of-solutions problem to become EXPTIME-hard. In fact, this EXPTIME-hard lower bound holds even when the s-t tgds are held fixed.

THEOREM 4.9. *There exist a fixed source schema $\mathbf{S}^*$, a fixed target schema $\mathbf{T}^*$, and a fixed set Σ_{st}^* such that*

1. *Σ_{st}^* consists of full s-t tgds.*
2. *The existence-of-solutions problem for $\mathcal{C}^*$ is* EXPTIME-*hard, where $\mathcal{C}^*$ is the class of all schema mappings of the form $(\mathbf{S}^*, \mathbf{T}^*, \Sigma_{st}^*, \Sigma_t)$ and such that Σ_t consists of one egd and one weakly acyclic tgd with one existentially-quantified variable.*

PROOF. *(Sketch)* The basic idea behind the proof of this result uses the same encoding to fix the schemas as described in the proof sketch of Theorem 4.8. The source schema consists of three fixed arity relations symbols G, O and W. The fixed target schema is isomorphic to the source schema, with relation symbols G', O' and W' respectively. There are three fixed full s-t tgds that copy G, O and W to G', O' and W', respectively. In Σ_t, there is one target tgd that simulates the sirup rule and uses one existentially-quantified variable to generate tuple identifiers, in order to store an output of the sirup rule as several fixed arity tuples in G'. Also, there is one egd that detects the presence of δ in the target instance. ☐

4.2.2 coNP-Completeness

We show that the existence-of-solutions problem may be coNP-hard even when the schemas are held fixed and consist of a single relation, and Σ_{st} is fixed and consists of only a single full s-t tgd. Furthermore, our reduction requires only a single egd in Σ_t to derive this lower bound. Thus, a slight extension from a fixed schema mapping immediately gives us the coNP-hard lower bound.

THEOREM 4.10. *There exist a fixed source schema $\mathbf{S}^*$, a fixed target schema $\mathbf{T}^*$, and a fixed set Σ_{st}^* with the following properties:*

1. *$\mathbf{S}^*$ and $\mathbf{T}^*$ consist of a single relational symbol, and Σ_{st}^* consists of a single full s-t tgd.*
2. *The existence-of-solutions problem for $\mathcal{C}^*$ is* coNP-*hard, where $\mathcal{C}^*$ is the class of all schema mappings $(\mathbf{S}^*, \mathbf{T}^*, \Sigma_{st}^*, \Sigma_t)$ such that Σ_t consists of a single egd.*

PROOF. We give a reduction from 3SAT. The source schema $\mathbf{S}$ and target schema $\mathbf{T}$ each consist of a single 5-ary relation D and D', respectively. There is only a single full s-t tgd $D(u,v,x,y,z) \rightarrow D'(u,v,x,y,z)$ in Σ_{st} that copies D to D'. Given a 3SAT formula φ with n clauses, we let $C_1, C_2, \ldots, C_n$ denote the n clauses. We create the instance $I(\varphi)$ as follows. For each clause C_i, $1 \leq i \leq n$, we create seven tuples in D. For example, if C_i is the clause $(x \vee y \vee \neg z)$, then the following seven tuples are created in D: (C_i,C_{i+1},1,1,1), (C_i,C_{i+1},1,1,0), (C_i,C_{i+1},1,0,1), (C_i,C_{i+1},0,1,1), (C_i,C_{i+1},1,0,0), (C_i,C_{i+1},0,1,0), (C_i,C_{i+1},0,0,0). There is one tuple for every satisfying assignment for C_i. The last three columns contain the satisfying assignment and the first two columns of every tuple contain the values C_i and C_{i+1}, to denote that this satisfying assignment belongs to the clause C_i. Hence, the tuple $(C_i, C_{i+1}, 0, 0, 1)$ does not occur in D since $(0,0,1)$ is not a satisfying assignment for C_i. After generating seven tuples for each clause, we add a "dummy" tuple to D: $(C_{n+1}, C_{n+1}, d, d, d)$. The value "$d$" is different any other value in the relation D.

Finally, we construct an egd of the form $\phi(\mathbf{x}) \rightarrow x_1 = x_2$ for Σ_t based on φ. The left-hand-side $\phi(\mathbf{x})$ is constructed as follows: There is a relational atom $D'(u_{n+1}, u_{n+1}, w, w, w)$ in $\phi(\mathbf{x})$, where w is a fresh variable not used anywhere else in $\phi(\mathbf{x})$. Then, for each clause C_i there is a relational atom $D'(u_i, u_{i+1}, x_i, y_i, z_i)$ where x_i, y_i and z_i are variables that represent the literals in C_i. The right hand side of the egd is $x = w$, where x is one of the variables representing a literal that occur in $\phi(\mathbf{x})$. As an example, suppose φ is the 3SAT formula $(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee \neg x_4)$. The following egd $D'(u_1,u_2,x_1,x_2,x_3) \wedge D'(u_2,u_3,x_2,x_3,x_4) \wedge D'(u_3,u_3,w,w,w) \rightarrow x_1 = w$ will be constructed.

Clearly, the schema mapping $\mathcal{D}(\varphi)$ and the source instance $I(\varphi)$ can be constructed in polynomial time in the size of φ. It is also easy to see that there is no solution for $I(\varphi)$ under $\mathcal{D}(\varphi)$ if and only if φ is satisfiable. ☐

Theorem 4.3 and Theorem 4.10 yield the following corollary.

COROLLARY 4.11. *Let $\mathbf{S}^*$ be a fixed source schema and $\mathbf{T}^*$ a fixed target schema. Let $\mathcal{C}(\mathbf{S}^*, \mathbf{T}^*)$ be the class of all schema mappings $(\mathbf{S}^*, \mathbf{T}^*, \Sigma_{st}, \Sigma_t)$, where Σ_{st} is a set of full s-t tgds, and Σ_t is the union of a set of egds with a set of full tgds. The existence-of-solutions problem for $\mathcal{C}(\mathbf{S}^*, \mathbf{T}^*)$ is* coNP-*complete.*

A slight variation of the proof of Theorem 4.10 yields the same lower bound, where Σ_t is now kept fixed, while Σ_{st} is allowed to vary but consists entirely of full s-t tgds.

THEOREM 4.12. *There exist a fixed source schema $\mathbf{S}^*$, a fixed target schema $\mathbf{T}^*$, and a fixed set Σ_t^* with the following properties:*

1. *$\mathbf{S}^*$ and $\mathbf{T}^*$ consist of a single relation symbol, and Σ_t^* consists of a single egd;*

2. *The existence-of-solutions problem for $\mathcal{C}^*$ is coNP-hard, where $\mathcal{C}^*$ is the class of all schema mappings $(\mathbf{S}^*, \mathbf{T}^*, \Sigma_{st}, \Sigma_t^*)$ such that Σ_{st} consists of a single full s-t tgd.*

5. CONCLUDING REMARKS

The results presented here shed light on both the data complexity and the combined complexity of data exchange for schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \mathbf{\Sigma_{st}}, \mathbf{\Sigma_t})$ in which Σ_{st} is a set of s-t tgds and Σ_t is a set of target egds and target tgds. Earlier work on the data complexity of data exchange showed that the existence-of-solutions problem is polynomial-time solvable, assuming that the target tgds form a weakly acyclic set. Our first main finding is that if the weak acyclicity assumption is relaxed, then the existence-of-solutions problem may become undecidable. To establish this, we brought into center stage the embedding problem from universal algebra, a problem intimately connected to the word problem, but not previously used in database theory. Regarding the combined complexity of data exchange, we demonstrated a provable exponential gap between the data complexity and the combined complexity of the existence-of-solutions problem for schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \mathbf{\Sigma_{st}}, \mathbf{\Sigma_t})$, where Σ_{st} is a set of s-t tgds and Σ_t is the union of a set of target egds with a weakly acyclic set of target tgds.

Acknowledgments: We are very grateful to Stanley Burris, Marcel Jackson, and Ross Willard for illuminating email exchanges about the embedding problem and pointers to the literature. We also wish to thank Moshe Y. Vardi for reminding us periodically that the combined complexity of data exchange had not been investigated.

6. REFERENCES

[1] M. Arenas and L. Libkin. XML Data Exchange: Consistency and Query Answering. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, pages 13–24, 2005.

[2] C. Beeri and M. Y. Vardi. The Implication Problem for Data Dependencies. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 73–85, 1981.

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

[5] A. Calì, D. Calvanese, G. D. Giacomo, and M. Lenzerini. Data integration under integrity constraints. *Information Systems*, 29:147–163, 2004.

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

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

[8] T. Evans. The word problem for abstract algebras. *Journal of the London Mathematical Society*, 26:64–71, 1951.

[9] T. Evans. Embeddability and the word problem. *Journal of the London Mathematical Society*, 28:76–80, 1953.

[10] T. Evans. Word Problems. *Bulletin of the American Mathematical Society*, 84(5):789–802, 1978.

[11] R. Fagin. Horn clauses and database dependencies. *Journal of the Association for Computing Machinery (JACM)*, 29(4):952–985, 1982.

[12] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. *Theoretical Computer Science (TCS)*, 336(1):89–124, 2005.

[13] R. Fagin, P. G. Kolaitis, and L. Popa. Data Exchange: Getting to the Core. *ACM Transactions on Database Systems (TODS)*, 30(1):174–210, 2005.

[14] R. Fagin, P. G. Kolaitis, L. Popa, and W. Tan. Composing Schema Mappings: Second-Order Dependencies to the Rescue. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, pages 83–94, 2004.

[15] G. Gottlob. Computing cores for data exchange: Hard cases and practical solutions. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, pages 148–159, 2005.

[16] G. Gottlob and C. Papadimitriou. On the complexity of single-rule datalog queries. *Information and Computation*, 183(1):104–122, 2003.

[17] G. Grätzer. *Universal Algebra, Second Edition*. Springer-Verlag, 1979.

[18] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory*. Oxford University Press, 1995.

[19] Y. Gurevich. The word problem for certain classes of semigroups. *Algebra and Logic*, 5:25–35, 1966.

[20] Y. Gurevich and H. R. Lewis. The Inference Problem for Template Dependencies. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, pages 221–229, 1982.

[21] Y. Gurevich and H. R. Lewis. The word problem for cancellation semigroups with zero. *Journal of Symbolic Logic*, 49(1):184–191, 1984.

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

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

[24] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

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

[26] E. L. Post. Recursive Unsolvability of a Problem of Thue. *Journal of Symbolic Logic*, 12(1):1–11, 1947.

[27] M. Y. Vardi. The complexity of relational query languages (extended abstract). In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 137–146, 1982.