Data Exchange with Arithmetic Operations*

Balder ten Cate
UC Santa Cruz
btencate@ucsc.edu

Phokion G. Kolaitis
UC Santa Cruz &
IBM Research - Almaden
kolaitis@cs.ucsc.edu

Walied Othman
University of Zurich
walied.othman@geo.uzh.ch

ABSTRACT

Data exchange is the problem of transforming data structured under a source schema into data structured under a target schema, taking into account structural relationships between the two schemas, which are described by a schema mapping. Existing schema-mapping languages lack the ability to express arithmetic operations, such as addition and multiplication, which naturally arise in data warehousing, ETL applications, and applications involving scientific data. We initiate the study of data exchange for arithmetic schema mappings, that is, schema mappings specified by source-to-target dependencies and target dependencies that may include arithmetic formulas interpreted over the algebraic real numbers (we restrict attention to algebraic real numbers to maintain finite presentability).

We show that, for arithmetic schema mappings without target dependencies, the existence-of-solutions problem can be solved in polynomial time, and, if a solution exists, then a universal solution (suitably defined) exists and can be computed in polynomial time. In the case of arithmetic schema mappings with a weakly acyclic set of target dependencies, a universal solution may not exist, but a finite universal basis exists (if a solution exists) and can be computed in polynomial space. The existence-of-solutions problem turns out to be NP-hard, and solvable in PSPACE. In fact, we show it is $\exists\mathbb{R}$-complete, which means that it has the same complexity as the decision problem for the existential theory of the real numbers, or, equivalently, the problem of deciding whether or not a quantifier-free arithmetic formula has a solution over the real numbers. If we allow only linear arithmetic formulas in the schema mapping and in the query, interpreted over the rational numbers, then the existence-of-solutions problem is NP-complete. We obtain analogous complexity results for the data complexity of computing the certain answers of arithmetic conjunctive queries and linear arithmetic conjunctive queries.

*Research on this paper was partially supported by NSF Grants IIS-0905276 and IIS-1217869.

EDBT/ICDT '13 March 18 - 22 2013, Genoa, Italy

Categories and Subject Descriptors: H.2.5 [Database Management]: Heterogeneous Databases; H.2.4 [Database Management]: Systems - *relational databases*

General Terms: Algorithms, Design, Languages

Keywords: schema mappings, arithmetic, data exchange

1. Introduction & Summary of Results

Data exchange, also known as data translation, is typically described as the problem of transforming data structured under one schema, called the source schema, into data structured under a different schema, called the target schema. The formal study of relational data exchange (i.e., data exchange between relational schemas) was initiated in [14] and was followed by numerous investigations (see [5, 20]).

The transformation of source data into target data must satisfy certain constraints that specify the relationship between the source and the target schemas. Thus, a central direction of research in data exchange has focused on the schema-mapping languages used to express this relationship. In the case of relational data exchange, the most widely used and studied schema-mapping language is the language consisting of source-to-target tuple-generating dependencies (s-t tgds), target tuple-generating dependencies (target tgds), and target equality-generating dependencies (target egds). An s-t tgd is an expression of the form $\forall\mathbf{x}(\varphi(\mathbf{x}) \rightarrow \exists\mathbf{y}\psi(\mathbf{x},\mathbf{y}))$, where $\varphi(\mathbf{x})$ is a conjunction of source atoms and $\psi(\mathbf{x},\mathbf{y})$ is a conjunction of target atoms. A target tgd is an expression of the same form, except that both $\varphi(\mathbf{x})$ and $\psi(\mathbf{x},\mathbf{y})$ are conjunctions of target atoms. Finally, a target egd is an expression of the form $\forall\mathbf{x}(\varphi(\mathbf{x}) \rightarrow x_i = x_j)$, where $\varphi(\mathbf{x})$ is a conjunction of target atoms, and the variables x_i and x_j are among those in $\mathbf{x}$. From now on, we will refer to this framework as the standard relational data exchange.

The language of s-t tgds, target tgds, and target egds strikes a good balance between expressive power and overall tame algorithmic behavior as regards data exchange purposes. There are, however, important applications in which this language cannot express the required constraints. In particular, s-t tgds cannot express constraints involving operations on data values, and relationships between data values, beyond equality. Such constraints arise naturally in several different contexts, including data warehousing, ETL (extract-transform-load) applications, and applications involving scientific data.

In this paper, we initiate the study of data exchange with built-in arithmetic operations. This framework extends

standard relational data exchange by allowing *arithmetic formulas*, using *comparisons (order)*, *addition*, and *multiplication*, in the specification of dependencies. In particular, it makes it possible to express the following constraints (we have omitted universal quantifiers in the front):

- $\text{TEMPF}(date, x) \rightarrow \exists y\ \text{TEMPC}(date, y) \wedge y = \frac{5}{9}(x - 32)$
- $\text{TRANSACTION}(date, x) \wedge \text{EXCHRATE}(date, rate) \rightarrow \exists y\ \text{RECORD}(date, y) \wedge y = x \times rate$
- $\text{LOCATION}(a, x, y) \wedge \text{LOCATION}(b, x', y') \wedge (x - x')^2 + (y - y')^2 \leq 5^2 \rightarrow \text{ADJACENT}(a, b)$
- $\text{LOCATION}(a, x, y) \wedge \text{LOCATION}(b, x', y') \rightarrow \exists d\ (\text{DISTANCE}(a, b, d) \wedge d \geq 0 \wedge (x-x')^2+(y-y')^2 = d^2)$.

When formalizing data exchange with arithmetic operations, the first task is to make precise the domain of values on which the arithmetic operations are defined. In [2], a study of data exchange with arithmetic comparisons was carried out, that is, a linear order $<$ was allowed to be used in dependencies and this linear order was interpreted over the rational numbers (equivalently, over a countable dense linear order without endpoints). When addition $+$ and multiplication $\times$ are also allowed in dependencies, it is no longer possible to work with rational numbers only, since, for example, the last of the above four dependencies may force the generation of irrational numbers. In view of this, it is natural to consider interpreting the arithmetic operations $<$, $+$, and $\times$ over the set of real numbers. However, the real numbers form an uncountable set and so they do not admit a finitary representation, which is an important requirement in analyzing algorithmic problems and taking into account the size of inputs. To overcome this, we restrict attention to instances containing only *algebraic* real numbers. Recall that a real number is *algebraic* if it is the root of a polynomial with integer coefficients. Thus, $\sqrt{2}$ is an algebraic real number, while e and π are known not to be. Algebraic real numbers possess several different finitary representations with good computational properties. Furthermore, the set $\mathbb{A}$ of algebraic real numbers, equipped with $<$, $+$, and $\times$, forms a countable *elementary substructure* of the real numbers; this implies that the algebraic real numbers satisfy precisely the same first-order sentences that the real numbers do.

By Tarski's theorem [28], the real numbers with $<$, $+$, and $\times$ (hence, also the algebraic real numbers) admit *quantifier-elimination*, which means that every first-order formula is equivalent to a quantifier-free one. Thus, in formalizing data exchange with arithmetic operations, we may (and we will) restrict attention to quantifier-free arithmetic formulas.

In what follows, we will study data exchange with source-to-target arithmetic dependencies and target arithmetic dependencies. A *source-to-target arithmetic dependency* is an expression $\forall \mathbf{x}(\varphi(\mathbf{x}) \wedge \beta(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}) \wedge \gamma(\mathbf{x}, \mathbf{y}))$, such that φ is a conjunction of source atoms, ψ is a (possibly empty) conjunction of target atoms, and β and γ are quantifier-free arithmetic formulas. A *target arithmetic dependency* is a syntactically similar expression in which both φ and ψ are conjunctions of target atoms; note that target egds are a special case of target arithmetic dependencies. Therefore, this framework generalizes both the framework of standard relational data exchange in [14] and the framework of data exchange with arithmetic comparisons in [2].

In standard relational data exchange, *universal solutions* play a crucial role. Intuitively, given a source instance I, a target instance J is a universal solution for I if J is the "most general" target instance that can be materialized so that the dependencies at hand are satisfied. An important property of universal solutions is that the *certain answers* of target conjunctive queries can be obtained by simply evaluating the conjunctive queries on a universal solution; in fact, this property characterizes universal solutions [14]. In general, universal solutions in standard data exchange are target instances that contain *null values*. In data exchange with arithmetic operations, it becomes necessary to consider target instances in which the interpretation of these null values is further constrained by an arithmetic formula, because, otherwise, even very simple source instances do not have a universal solution (Proposition 3.16). Intuitively, this is so because the arithmetic dependencies may impose arithmetic constraints on the possible interpretations of null values in target instances. For this reason, we define suitable notions of *solution* and *universal solution* using instances of the form $\mathcal{J} = (J, \alpha)$, where J is a target instance (possibly with null values) and α is a quantifier-free arithmetic formula.

Summary of Results We show that in data exchange settings with source-to-target arithmetic dependencies, but without target constraints of any kind, every source instance that has a solution has a universal solution; moreover, a universal solution can be constructed in polynomial time via a chase procedure tailored to source-to-target arithmetic dependencies. In data exchange settings with source-to-target arithmetic dependencies and with target arithmetic dependencies, universal solutions need not exist in general, even when solutions exist. However, we show that if the target arithmetic dependencies form a weakly acyclic set, then a finite *universal basis* of solutions exists and can be constructed in PSPACE via a more sophisticated *disjunctive* chase procedure tailored to target arithmetic dependencies.

A universal basis can be used to compute the certain answers of *arithmetic conjunctive queries*, that is, queries that are conjunctions of target atoms and a quantifier-free arithmetic formula. It turns out that the exact complexity of the certain answers of arithmetic conjunctive queries is intimately connected to a well-known decision problem about first-order logic on the real numbers, namely, the problem of deciding whether or not a given existential arithmetic sentence is true on the structure $\mathbb{R}$ of the real numbers. In terms of the standard computational complexity classes, the best result known about this problem is that it is NP-hard and that it belongs to PSPACE (NP-hardness is an easy observation, while membership in PSPACE is a highly non-trivial theorem due to Canny [10]). For this reason, Schaefer [27] introduced $\exists\mathbb{R}$, a new complexity class consisting of all decision problems that can be reduced in polynomial time to the problem of deciding whether or not an existential arithmetic sentence is true on $\mathbb{R}$. In [27], several geometric and topological problems were shown to be $\exists\mathbb{R}$-complete, such as the RECTILINEAR CROSSING NUMBER PROBLEM, which asks for the minimal number of crossings in a straight-line drawing of a graph (see also [9]). We show that, in the presence of source-to-target arithmetic dependencies and weakly acyclic target arithmetic dependencies, the problem of computing the certain answers of target arithmetic conjunctive queries is co$\exists\mathbb{R}$-complete in data complexity.

We also investigate *linear* arithmetic dependencies and *linear* arithmetic conjunctive queries, that is, arithmetic dependencies and arithmetic conjunctive queries in which the quantifier-free arithmetic formulas involve only order $<$ and

addition +. In this context, it makes perfectly good sense to go back to the rational numbers, since the rational numbers equipped with $<$ and $+$ form an elementary substructure of the real numbers equipped with $<$ and $+$, and also admit elimination of quantifiers [15]. We show that, in the presence of source-to-target linear arithmetic dependencies and weakly acyclic target liner arithmetic dependencies, computing the certain answers of target linear arithmetic conjunctive queries is a coNP-complete problem in data complexity. It should be noted that, as shown in [2], the same complexity result holds in the presence of dependencies and conjunctive queries that involve only order. Thus, extending the data exchange framework from order to order and addition does not increase the complexity of query answering, while extending the framework to order, addition, and multiplication does, unless NP $= \exists\mathbb{R}$, which is considered an unlikely possibility.

Due to space limitations, the proofs of several results are only sketched or omitted altogether. Detailed proofs will be provided in the full version of the paper.

2. Preliminaries

In this section, we introduce the arithmetic language used and present the necessary background material.

Definition 2.1 (Arithmetic formulas)**.** A *polynomial* is an expression built up from variables and rational numbers, using the binary functions $+$ and $\times$. As usual, t^k is a shorthand for the k-fold multiplication $t \times \cdots \times t$, while $t - t'$ is a shorthand for $t + (-1 \times t')$. A *polynomial comparison* is an expression of the form $t_1 \ \mathcal{O} \ t_2$, where t_1 and t_2 are polynomials and $\mathcal{O} \in \{<, \leq, =, \geq, >, \neq\}$.

- A *quantifier-free arithmetic formula* $\alpha(\mathbf{x})$ is a Boolean combination of polynomial comparisons.
- An *arithmetic formula* $\alpha(\mathbf{x})$ is a first-order formula built from polynomial comparisons.
- An arithmetic formula is *linear* if it uses no multiplication, other than multiplication by a constant.

The *length* of an arithmetic formula is simply its length as a syntactic expression, where the coefficients are written in binary notation.

The variables (free and quantified ones) in arithmetic formulas range over real numbers. More precisely, we consider the mathematical structure $(\mathbb{R}, +, \times, <, (c)_{c\in\mathbb{Q}})$, where $\mathbb{R}$ is the set of real numbers and $\mathbb{Q}$ is the set of rational numbers. For simplicity, we denote this structure by $\mathbb{R}$ as well.

- A *solution* of an arithmetic formula $\alpha(\mathbf{x})$ is a tuple of real numbers $\mathbf{r}$ such that $\mathbb{R} \models \alpha(\mathbf{r})$.
- An arithmetic formula is *solvable* if it has a solution.
- Two arithmetic formulas are *equivalent* if they have the same solutions.

Examples of polynomial comparisons are $y = \frac{5}{9}(x - 32)$ and $x < y^2$. An example of a quantifier-free arithmetic formula is $(y = \frac{5}{9}(x-32)) \wedge (y < 300)$; in fact, this is a linear one. An example of an arithmetic formula with quantifiers is $\exists y(y^2 = x)$, which happens to be equivalent to the quantifier-free arithmetic formula $x \geq 0$. The following classical result by Tarski asserts that this equivalence is no accident

Theorem 2.2 (Elimination of Quantifiers over the Reals)**.** *Every arithmetic formula is equivalent to a quantifier-free arithmetic formula* [28]*. Moreover, given an arithmetic formula, an equivalent quantifier-free one can be constructed in* 2ExpTime [18, 25].

This result is tight: there are arithmetic formulas for which the smallest equivalent quantifier-free arithmetic formula has doubly exponential length [31, 11].

The following decision problem, called the *solvability problem for quantifier-free arithmetic formulas*, will be of great interest to us: given a quantifier-free arithmetic formula, does it have a solution? Equivalently, this problem asks: given an existential arithmetic sentence (i.e., an arithmetic sentence in prenex normal form in which all the quantifiers are existential), is it true on $\mathbb{R}$? It is easy to see that this problem is NP-hard; moreover, Canny [10] has shown that it is in PSpace. However, the exact complexity of this problem has not been pinpointed in terms of the standard computational complexity classes (e.g., it is not known to be in NP or to be PSPACE-complete). For this reason, Schaefer [27] introduced a new complexity class, called $\exists\mathbb{R}$, that is built around this problem. By definition, $\exists\mathbb{R}$ is the collection of all decision problems that have a polynomial-time reduction to the solvability problem for quantifier-free arithmetic formulas. Thus, the solvability problem for quantifier-free arithmetic formulas is $\exists\mathbb{R}$-complete. Several other decision problems of geometric character have been shown to be $\exists\mathbb{R}$-complete, a fact that justifies the introduction and study of $\exists\mathbb{R}$ as a complexity class in its own right. In terms of standard complexity classes, $\exists\mathbb{R}$ is sandwiched between NP and PSpace, that is to say, NP $\subseteq \exists\mathbb{R} \subseteq$ PSpace.

The next result summarizes the preceding discussion.

Theorem 2.3. *The solvability problem for quantifier-free arithmetic formulas is $\exists\mathbb{R}$-complete; in particular, it is* NP*-hard and decidable in* PSpace.

Since the real numbers form an uncountable set, they do not admit a finitary representation system. In contrast, the *algebraic real numbers* do admit finitary representations. Recall that a real number is *algebraic* if it is a root of a polynomial with integer coefficients. There are several different well known representations of algebraic real numbers that are known to have good computational properties. Here, we will use the *isolating-interval representation* of algebraic real numbers (see, e.g., [13]). Specifically, if an algebraic real number a is the root of a polynomial $p(x)$ with integer coefficients, then an *isolating-interval representation* $\langle a \rangle$ of a is an expression $\langle a \rangle = (p(x), r, s)$, where r and s are rational numbers such that a is the only root of the polynomial $p(x)$ in the interval $[r, s]$. For example, $(x^2 - 2, 0, 1.5)$ is an isolating-interval representation of $\sqrt{2}$. Clearly, every algebraic real number has more than one isolating-interval representations. However, for the results presented here, any such representation will suffice.

The algebraic real numbers form a countable set, which we will denote by $\mathbb{A}$; we will use the same symbol to also denote the substructure $(\mathbb{A}, +, \times, <, (c)_{c\in\mathbb{Q}})$ of $\mathbb{R}$ induced by the algebraic real numbers. The following theorem tells that, as regards arithmetic constraints, we may restrict attention to algebraic real numbers.

Theorem 2.4 (Algebraic Numbers Suffice (from [28]))**.** *An arithmetic formula is solvable if and only if it has a solution consisting of algebraic real numbers. Moreover, if $\alpha(\mathbf{x})$ is an arithmetic formula and $\mathbf{r}$ a tuple of algebraic real numbers, then we have that $\mathbb{R} \models \alpha(\mathbf{r})$ if and only if $\mathbb{A} \models \alpha(\mathbf{r})$.*

In addition, the arithmetic calculations on algebraic real numbers, using the isolating-interval representation, can be carried out efficiently in parallel.

Theorem 2.5 ([8]). *Let $\alpha(\boldsymbol{x})$ be a fixed quantifier-free arithmetic formula. The following problem is in* NC*: given a sequence of algebraic real numbers $\boldsymbol{r}$ specified using the isolating-interval representation, is $\boldsymbol{r}$ a solution for $\alpha(\boldsymbol{x})$?*

As is well known, NC is a subclass of PTime that captures the notion of efficient parallel computation (see [17]).

We will also be interested in the solvability problem for linear quantifier-free arithmetic formulas. This problem is NP-complete. Indeed, membership in NP follows from linear programming together with propositional satisfiability: to decide solvability, guess a truth assignment for the comparisons occurring in the given formula, such that the formula evaluates to true, and check consistency of the chosen truth assignment in polynomial time via linear programming [19]. NP-hardness for this problem follows immediately from propositional satisfiability by using the comparisons $x > 0$ and $x \leq 0$ to encode the truth values "true" and "false".

Let $\mathbb{LR} = (\mathbb{R}, +, <, (c)_{c\in\mathbb{Q}})$ be the structure formed by the real numbers equipped with addition and order. Similarly, let $\mathbb{LQ} = (\mathbb{Q}, +, <, (c)_{c\in\mathbb{Q}})$ be the structure formed by the rational numbers equipped with addition and order. As shown in [15], the structure $\mathbb{LR}$ admits elimination of quantifiers; moreover, as regards linear arithmetic formulas, we may restrict attention to rational numbers.

The next result summarizes the preceding discussion about linear arithmetic formulas.

Theorem 2.6. *The following statements are true.*

- *The solvability problem for linear quantifier-free arithmetic formulas is* NP*-complete.*
- *A linear arithmetic formula is solvable if and only if it has a solution consisting of rational numbers. Moreover, if $\alpha(\mathbf{x})$ is a linear arithmetic formula and $\mathbf{r}$ a tuple of rational number, then we have that $\mathbb{LR} \models \alpha(\mathbf{r})$ if and only if $\mathbb{LQ} \models \alpha(\mathbf{r})$.*

3. Data Exchange over $\mathbb{A}$

To simplify presentation, we assume that all constant values occurring in source instances are algebraic real numbers. The results can be lifted to the setting in which attributes of relations in the schemas are typed, and arithmetic formulas are restricted to attributes that have a numerical type.

3.1 $\mathbb{A}$-Instances and Ground $\mathbb{A}$-Instances

Definition 3.1. Let $\mathbb{A}$ be the set of algebraic real numbers. Let Nulls be a countably infinite set of *labeled nulls* (or, *variables*) disjoint from $\mathbb{A}$, and let $\mathbf{R} = (R_1, \ldots, R_n)$ be a schema.

- An $\mathbb{A}$*-instance* over the schema $\mathbf{R}$ is a pair $\mathcal{I} = (I, \alpha)$ with $I = (R_1^I, \ldots, R_n^I)$, where each R_k^I is a finite relation over $\mathbb{A} \cup \mathsf{Nulls}$ of appropriate arity, and α is a quantifier-free arithmetic formula whose free variables are the nulls occurring in I.
- A *ground* $\mathbb{A}$*-instance* over a schema $\mathbf{R} = (R_1, \ldots, R_n)$ is a tuple $I = (R_1^I, \ldots, R_n^I)$, where each R_k^I is a finite relation over $\mathbb{A}$ of appropriate arity.
- If $\mathcal{I}$ is an $\mathbb{A}$-instance, then $\mathsf{Rep}(\mathcal{I})$ denotes the set of all ground $\mathbb{A}$-instances over $\mathbf{R}$ that can be obtained from $\mathcal{I} = (I, \alpha)$ by uniformly substituting algebraic real numbers for the labeled nulls occurring in $\mathcal{I}$, subject to the arithmetic formula α. Formally, we say that a function $f : \mathsf{Nulls} \to \mathbb{A}$ *satisfies* α if α evaluates to true when each variable is mapped to its f-image. Let $f(I)$ be the ground $\mathbb{A}$-instance over $\mathbf{R}$ obtained by replacing each null in I by its f-image. Then for $\mathcal{I} = (I, \alpha)$, we define $\mathsf{Rep}(\mathcal{I}) = \{f(I) \mid f \text{ satisfies } \alpha\}$.

In the above definition, we use algebraic real numbers to ensure that instances are finitely presentable. Note that every ground $\mathbb{A}$-instance I can be viewed as the $\mathbb{A}$-instance $\mathcal{I} = (I, \top)$, where $\top$ stands for a trivial arithmetic formula that always evaluates to 'true'. As an example of a non-ground $\mathbb{A}$-instance, let $\mathcal{I} = (I, \alpha)$, where I consists of the single fact $\text{Location}(r, N_1, N_2, 1030)$ and α is the constraint $(N_1 + N_2 < 200) \wedge (N_1 < 150) \wedge (N_2 > 50)$. Then the ground $\mathbb{A}$-instance $\{\text{Location}(r, 100, 75, 1030)\}$ is a member of $\mathsf{Rep}(\mathcal{I})$, whereas the ground $\mathbb{A}$-instance $\{\text{Location}(r, 130, 75, 1030)\}$ is not.

It follows from the definition that for a given $\mathbb{A}$-instance $\mathcal{I} = (I, \alpha)$, we have that $\mathsf{Rep}(\mathcal{I}) = \emptyset$ if and only if the quantifier-free arithmetic formula α has no solution.

Proposition 3.2. *The following problem is co$\exists\mathbb{R}$-complete: given an $\mathbb{A}$-instance $\mathcal{I}$, is $\mathsf{Rep}(\mathcal{I}) = \emptyset$?*

A geometric view of Definition 3.1 is as follows. If n is the number of nulls that appear in $\mathcal{I} = (I, \alpha)$, then α defines an algebraic surface in $\mathbb{A}^n$. Every point on that surface represents a ground $\mathbb{A}$-instance and is an element of $\mathsf{Rep}(\mathcal{I})$. Note that this relationship is not one-to-one, since multiple points may represent the same ground $\mathbb{A}$-instance.

3.2 Arithmetic Conjunctive Queries

Definition 3.3 (Arithmetic conjunctive queries). Let $\mathbf{R}$ be a schema and k a positive integer.

- A k-ary *arithmetic conjunctive query* over $\mathbf{R}$ is an expression $q(\mathbf{x}) = \exists\mathbf{y}\ \phi(\mathbf{x}, \mathbf{y}) \wedge \beta(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ is a k-tuple of variables, $\phi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms over $\mathbf{R}$ containing all the variables in $\mathbf{x}, \mathbf{y}$, and $\beta(\mathbf{x}, \mathbf{y})$ is a quantifier-free arithmetic formula. Arithmetic terms are not allowed in the relational atoms of $\phi(\mathbf{x}, \mathbf{y})$, but can be simulated using the arithmetic formula $\beta(\mathbf{x}, \mathbf{y})$.
- If I is a ground $\mathbb{A}$-instance, then $q(I)$ denotes the answers of q in I, defined in the standard way.
- If $\mathcal{I} = (I, \alpha)$ is an $\mathbb{A}$-instance $\mathcal{I}$, then $q(\mathcal{I})$ is defined as $\bigcap\{q(I') \mid I' \in \mathsf{Rep}(\mathcal{I})\}$.

For example, consider a schema consisting of a single binary relation R. Let I be the ground $\mathbb{A}$-instance consisting of the facts $R(1, 2)$, $R(2, 2)$, and $R(3, 4)$, and let $q(x)$ be the query $\exists y\,(R(x, y) \wedge x + y \geq 4)$. Then $q(I) = \{2, 3\}$.

Proposition 3.4. *Let q be a fixed arithmetic conjunctive query q. The following problem is solvable in* PTime*: given a ground $\mathbb{A}$-instance I, compute $q(I)$.*

Proof. We can compute $q(I)$ by first considering only the relational part of q, enumerating all satisfying variable assignments (there are polynomially many, because q is fixed), and then, for each such assignment, testing whether it satisfies the arithmetic formula. The latter task can be done in NC, and, hence, in PTime, by Theorem 2.5. □

Theorem 3.5. *Let q be a fixed arithmetic conjunctive query. The following problem is in co$\exists\mathbb{R}$: given an $\mathbb{A}$-instance $\mathcal{I} = (I, \alpha)$ and a tuple $\boldsymbol{a}$, test whether $\boldsymbol{a} \in q(\mathcal{I})$. Moreover, there are conjunctive queries without arithmetic formulas for which the problem is co$\exists\mathbb{R}$-complete.*

Proof. For the upper bound, we give a polynomial-time reduction to the solvability problem for quantifier-free arithmetic formulas. Let $q(\mathbf{x})$ be the query $\exists\mathbf{y}(\phi(\mathbf{x},\mathbf{y}) \wedge \beta(\mathbf{x},\mathbf{y}))$, where all variables in $\mathbf{x}$ and $\mathbf{y}$ occur in $\phi(\mathbf{x},\mathbf{y})$. Let $\phi^I(\mathbf{x},\mathbf{y})$ be the formula obtained from $\phi(\mathbf{x},\mathbf{y})$ by replacing each relational atom $R(z_1,\ldots,z_n)$ by the disjunction $\bigvee\{(z_1 = a_1) \wedge \cdots \wedge (z_n = a_n) \mid (a_1,\ldots,a_n) \in R^I\}$. We have that $\mathbf{a} \in q(\mathcal{I})$ if and only if the quantifier-free formula

$$\alpha \wedge \neg \bigvee_{h:\{\mathbf{y}\}\to adom(I)} (\phi^I(\mathbf{a}, h(\mathbf{y})) \wedge \beta(\mathbf{a}, h(\mathbf{y})))$$

is unsolvable. On the face of it, this formula is not an arithmetic one, since algebraic real numbers occur in it as constants. However, the isolating-interval representation makes it possible to transform it to an equivalent quantifier-free arithmetic formula. Specifically, if the isolating-interval representation of an algebraic real number a in $\mathbf{a}$ is $(p(x), r, s)$, then we replace a by a new variable v and add as a conjunct the arithmetic formula $(p(v) = 0) \wedge (r \le v) \wedge (v \le s)$. The unsolvability of the resulting quantifier-free arithmetic formula can be tested in co$\exists\mathbb{R}$. In general, the above formula is exponentially long, but it is of polynomial length when the query is fixed.

The lower bound follows from Proposition 3.2: if P is a relation that is empty in $\mathcal{I}$, then $\exists\mathbf{x}P\mathbf{x}$ is true in $\mathcal{I}$ if and only if $\mathsf{Rep}(\mathcal{I}) = \emptyset$. □

3.3 Arithmetic Dependencies

Definition 3.6. An *arithmetic dependency* over a schema $\mathbf{R}$ is a first-order sentence of the form

$$\forall\mathbf{x}(\phi(\mathbf{x}) \wedge \beta(\mathbf{x}) \to \exists\mathbf{y}\ \psi(\mathbf{x},\mathbf{y}) \wedge \gamma(\mathbf{x},\mathbf{y})),$$

where $\phi(\mathbf{x})$ is a conjunction of atoms from $\mathbf{R}$ containing all variables in $\mathbf{x}$, $\psi(\mathbf{x},\mathbf{y})$ is a (possibly empty) conjunction of atoms from $\mathbf{R}$ containing all variables in $\mathbf{y}$, and $\beta(\mathbf{x})$ and $\gamma(\mathbf{x},\mathbf{y})$ are quantifier-free arithmetic formulas (possibly $\top$, in which case we will omit them in our notation).

A *constraint-generating dependency* (cgd) is an arithmetic dependency as above in which $\mathbf{y}$ and $\psi(\mathbf{x},\mathbf{y})$ are empty. An *arithmetic equality-generating dependency* is a constraint-generating dependency in which $\gamma(\mathbf{x})$ is a single equality.

For a ground $\mathbb{A}$-instance I, the semantics of arithmetic dependencies are the standard semantics of first-order logic on I. For an $\mathbb{A}$-instance $\mathcal{I} = (I, \alpha)$, the semantics of arithmetic dependencies a are given by considering $\mathsf{Rep}(\mathcal{I})$.

Definition 3.7. Let $\mathcal{I} = (I, \alpha)$ be an $\mathbb{A}$-instance and let σ be an arithmetic dependency over a schema $\mathbf{R}$. We say that $\mathcal{I} = (I, \alpha)$ *satisfies* σ if every ground $\mathbb{A}$-instance $I' \in \mathsf{Rep}(\mathcal{I})$ satisfies σ.

Theorem 3.8. *Let σ be a fixed arithmetic dependency. The problem of testing whether a given $\mathbb{A}$-instance satisfies σ is in co$\exists\mathbb{R}$. Depending on σ, this problem can be co$\exists\mathbb{R}$-complete.*

Proof. The proof is along the same general lines as that of Theorem 3.5. Let $\mathcal{I} = (I, \alpha)$ and let σ be of the form

$$\forall\mathbf{x}(\phi(\mathbf{x}) \wedge \beta(\mathbf{x}) \to \exists\mathbf{y}\ \psi(\mathbf{x},\mathbf{y}) \wedge \gamma(\mathbf{x},\mathbf{y})).$$

Let $\phi^I(\mathbf{x})$ be the formula obtained from $\phi(\mathbf{x})$ by replacing each relational atom $R(x_1,\ldots,x_n)$ by the disjunction $\bigvee\{(x_1 = a_1) \wedge \cdots \wedge (x_n = a_n) \mid (a_1,\ldots,a_n) \in R^I\}$; we define ψ^I analogously.

Then $\mathcal{I}$ satisfies σ if and only if the quantifier-free formula

$$\alpha \wedge \neg \bigwedge_{h:\mathbf{x}\to adom(I)} \Bigg(\phi^I\left(h(\mathbf{x}) \wedge \beta(h(\mathbf{x}))\right) \ \to \bigvee_{g:\mathbf{y}\to adom(I)} \psi^I\left(h(\mathbf{x}), g(\mathbf{y})\right) \wedge \gamma\left(h(\mathbf{x}), g(\mathbf{y})\right) \Bigg)$$

has no solution; this is decidable in co$\exists\mathbb{R}$. Note that, although the above formulas are in general exponentially long, they are only polynomially long when σ is fixed.

The co$\exists\mathbb{R}$-hardness result holds because $R(\mathbf{x}) \to Q(\mathbf{x})$ is satisfied in $\mathcal{I}$ if and only if $\mathsf{Rep}(\mathcal{I}) = \emptyset$, where R is a relation occurring in $\mathcal{I}$ and Q is a relation not occurring in $\mathcal{I}$. □

3.4 Data Exchange and Certain Answers

Definition 3.9. Let $\mathbf{S}$ and $\mathbf{T}$ be two disjoint schemas, called the *source schema* and the *target schema*.

- A *source-to-target arithmetic dependency* is an arithmetic dependency

  $$\forall\mathbf{x}(\phi(\mathbf{x}) \wedge \beta(\mathbf{x}) \to \exists\mathbf{y}\ \psi(\mathbf{x},\mathbf{y}) \wedge \gamma(\mathbf{x},\mathbf{y})),$$

  such that ϕ contains only relations from $\mathbf{S}$ and ψ contains only relations from $\mathbf{T}$.
- A *target arithmetic dependency* is an arithmetic dependency over $\mathbf{T}$.
- An *arithmetic schema mapping* is a tuple of the form $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$, where Σ_{st} is a finite set of source-to-target arithmetic dependencies and Σ_t is a finite set of target arithmetic dependencies.

We now introduce two notions of solutions in data exchange with arithmetic operations, as well as the notion of the certain answers.

Definition 3.10. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be an arithmetic schema mapping and let I be a ground $\mathbb{A}$-instance over the source schema $\mathbf{S}$.

- A ground $\mathbb{A}$-instance J over the target schema $\mathbf{T}$ is a *ground solution* for I w.r.t. $\mathcal{M}$ if the pair (I, J) satisfies Σ_{st}, and J satisfies Σ_t. The set of all ground solutions for I is denoted by $\mathsf{GSol}(I)$.
- An $\mathbb{A}$-instance $\mathcal{J} = (J, \alpha)$ over the target schema $\mathbf{T}$ is a *solution* for I w.r.t. $\mathcal{M}$ if the pair $(I, \mathcal{J})$ satisfies Σ_{st}, and $\mathcal{J}$ satisfies Σ_t. The set of all solutions for I is denoted by $\mathsf{Sol}(I)$.
- If q is an arithmetic conjunctive query over the target schema $\mathbf{T}$, then the *certain answers* of q on I w.r.t. $\mathcal{M}$, denoted by $\mathsf{Cert}_{\mathcal{M}}(q, I)$ or, simply, $\mathsf{Cert}(q, I)$, are defined as $\mathsf{Cert}_{\mathcal{M}}(q, I) = \bigcap\{q(J) \mid J \in \mathsf{GSol}(I)\}$.

Observe that the notion of the certain answers was defined using ground solutions. One could also consider an alternative definition in which the certain answers are defined using

solutions, i.e., by considering the set $\bigcap\{q(\mathcal{J}) \mid \mathcal{J} \in \mathsf{Sol}(I)\}$. It is easy to see, however, that these two notions coincide. This is so because, as seen earlier, every ground $\mathbb{A}$-instance can be viewed as an $\mathbb{A}$-instance and also, by Definition 3.3, if $\mathcal{J}$ is an $\mathbb{A}$-instance, then $q(\mathcal{J}) = \bigcap\{q(J') \mid J' \in \mathsf{Rep}(\mathcal{J})\}$. Thus, the following result holds.

Proposition 3.11. *Assume that $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ is an arithmetic schema mapping. For every arithmetic conjunctive query q over the target schema $\mathbf{T}$ and for every ground $\mathbb{A}$-instance I over the source schema $\mathbf{S}$, we have that $\mathsf{Cert}_{\mathcal{M}}(q, I) = \bigcap\{q(\mathcal{J}) \mid \mathcal{J} \in \mathsf{Sol}(I)\}$.*

Example 3.12. Let $\mathcal{M}$ be the arithmetic schema mapping specified by the source-to-target arithmetic dependency

$$\textsc{Detection}(\text{sensor}, \text{rfid}, t) \wedge \textsc{SensorLoc}(\text{sensor}, x, y) \rightarrow \textsc{Location}(\text{rfid}, x', y', t) \wedge (x' - x)^2 < 1 \wedge (y' - y)^2 < 1.$$

Let I be the ground source $\mathbb{A}$-instance consisting of the facts $\textsc{Detection}(s1, r1, 1800)$ and $\textsc{SensorLoc}(s1, 37, 122)$, where $s1$ and $r1$ are actual values encoding a sensor and an RFID tag. Then an example of a ground solution for I with respect to $\mathcal{M}$ is the target $\mathbb{A}$-instance consisting of the fact $\textsc{Location}(r1, 37, 122.5, 1800)$. Consider also the target arithmetic conjunctive query

$$q(r) := \exists x \exists y \exists t(\textsc{Location}(r, x, y, t) \ \wedge \ (30 < x < 40) \wedge (100 < y < 140) \wedge (1200 < t < 2300))$$

Then $r1$ belongs to $q(J)$ for *every* $J \in \mathsf{GSol}(I)$, and, hence, $r1 \in \mathsf{Cert}_{\mathcal{M}}(q, I)$.

In the by now classical setting of data exchange with source-to-target tuple generating dependencies and target tuple-generating dependencies, a universal solution for a source instance is a solution that can be homomorphically mapped into every solution. Universal solutions are also characterized by the property that the certain answers of every conjunctive query can be obtained by evaluation the query on the universal solution. We use this latter property to define the notion of a universal solution in the context of data exchange with arithmetic schema mappings.

Definition 3.13. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be an arithmetic schema mapping and I a ground source $\mathbb{A}$-instance. A target $\mathbb{A}$-instance $\mathcal{J}$ is a *universal solution* for I w.r.t. $\mathcal{M}$ if it is a solution for I w.r.t. $\mathcal{M}$ and for every arithmetic conjunctive query q, we have that $\mathsf{Cert}_{\mathcal{M}}(q, I) = q(\mathcal{J})$.

Example 3.14. Consider again the arithmetic schema mapping and ground source $\mathbb{A}$-instance from Example 3.12. Let $\mathcal{J} = (J, \alpha)$ where $J = \{\textsc{Location}(r1, N_1, N_2, 1800)\}$ and α is the arithmetic formula $(36 < N_1 < 38) \wedge (121 < N_2 < 123)$. It may be verified that $\mathcal{J}$ is a universal solution for I.

As another example, let $\mathcal{M}$ be the arithmetic schema mapping specified by the arithmetic dependency $\forall x\ (P(x) \rightarrow \exists y(Q(x, y) \wedge y > 0))$. Consider the ground source $\mathbb{A}$-instance $I = \{P(0)\}$. Then the target $\mathbb{A}$-instance $\mathcal{J} = (\{Q(0, N)\}, N > 0)$ is a universal solution for I with respect to $\mathcal{M}$. This example can also be used to show that, in the context of data exchange with arithmetic comparisons, it is crucial to allow arithmetic formulas in (non-ground) instances. Indeed, as will be shown in Proposition 3.16 below, I does not have any universal solution, or even a finite universal basis (to be defined next), consisting of $\mathbb{A}$-instances without arithmetic formulas.

In Section 4.1, it will be shown that if $\mathcal{M}$ is an arithmetic schema mapping with no target arithmetic dependencies, then if a ground source $\mathbb{A}$-instance has a solution, then it has a universal solution. In contrast, there are arithmetic schema mappings with target arithmetic dependencies for which universal solutions need not exist, even though solutions exist. To see this, consider the arithmetic schema mapping $\mathcal{M}$ with $\Sigma_{st} = \{\forall x(P(x) \rightarrow \exists yzR(y, z))\}$ and $\Sigma_t = \{\forall xy(R(x, y) \wedge x \neq y \rightarrow Q_1(x)),\ \forall x(R(x, x) \rightarrow Q_2(x))\}$. If $I = \{P(a)\}$, then there is no universal solution for I w.r.t. $\mathcal{M}$. The reason is that the universality condition implies that if $\mathcal{J}$ were a universal solution, then $\mathcal{J}$ must contain no Q_1-atom and also no Q_2-atom. This state of affairs motivates the introduction of the notion of a *universal basis*.

Definition 3.15. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be an arithmetic schema mapping and I a ground source $\mathbb{A}$-instance. A collection $\{\mathcal{J}_1, \ldots, \mathcal{J}_n\}$ of target $\mathbb{A}$-instances is a *universal basis* for I w.r.t. $\mathcal{M}$ if each instance $\mathcal{J}_i$, $1 \leq i \leq n$, is a solution for I, and for every arithmetic conjunctive query q, we have that $\mathsf{Cert}_{\mathcal{M}}(q, I) = \bigcap_{1 \leq i \leq n} q(\mathcal{J}_i)$.

Consider again the arithmetic schema mapping $\mathcal{M}$ with $\Sigma_{st} = \{\forall x(P(x) \rightarrow \exists yzR(y, z))\}$ and $\Sigma_t = \{\forall xy(R(x, y) \wedge x \neq y \rightarrow Q_1(x)),\ \forall x(R(x, x) \rightarrow Q_2(x))\}$. It is easy to verify that the $\mathbb{A}$-instances $\mathcal{J}_1 = (\{R(N_1, N_2), Q_1(a)\}, N_1 \neq N_2)$ and $\mathcal{J}_2 = (\{R(N_1, N_1), Q_2(a)\}, \top)$ form a universal basis for $I = \{P(a)\}$ with respect to $\mathcal{M}$. The next result shows that arithmetic formulas are of the essence in the definition of $\mathbb{A}$-instances.

Proposition 3.16. *Let $\mathcal{M}$ be the arithmetic schema mapping specified by the arithmetic dependency $\forall x\ (P(x) \rightarrow \exists y\ (Q(x, y) \wedge y > 0))$. Let $I = \{P(0)\}$ and let $\{\mathcal{J}_1, \ldots, \mathcal{J}_n\}$ be an arbitrary collection of target $\mathbb{A}$-instances whose arithmetic formula is $\top$. Then $\{\mathcal{J}_1, \ldots, \mathcal{J}_n\}$ cannot be a universal basis for I with respect to $\mathcal{M}$.*

Proof. Towards a contradiction, suppose that $\{\mathcal{J}_1, \ldots, \mathcal{J}_n\}$ is a universal basis for I such that the arithmetic formula in each $\mathbb{A}$-instance $\mathcal{J}_i$ is $\top$.

First, we argue that each $\mathcal{J}_i$ must contain a fact of the form $Q(0, a_i)$ where a_i is a positive number. Indeed, if some $\mathcal{J}_i$ did not contain such a fact, then, by mapping all nulls to negative numbers, one would have that $\mathsf{Rep}(\mathcal{J}_i)$ contains a ground $\mathbb{A}$-instance that falsifies the Boolean arithmetic conjunctive query $q = \exists y Q(x, y) \wedge y > 0$; however, the certain answer of q on I is 'true'.

Next, let k be a number such that if one of the $\mathbb{A}$-instances $\mathcal{J}_i$ contains a fact of the form $Q(0, a_i)$, then we have that $a_i < k$. Consider the Boolean arithmetic conjunctive query $q' = \exists x\ Q(0, x) \wedge 0 < x < k$. By construction, $q'(\mathcal{J}_i)$ is true for all $i \leq n$, while the certain answer of q' on I is 'false'. □

4. The Chase and Certain Answers

4.1 The case without target dependencies

Theorem 4.1. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be an arithmetic schema mapping without any target dependencies. There is a* PTime *algorithm that, given a ground source $\mathbb{A}$-instance I, tests for the existence of solutions for I w.r.t. $\mathcal{M}$, and if the answer is positive, constructs a target $\mathbb{A}$-instance that is a universal solution for I w.r.t. $\mathcal{M}$.*

Proof. For each source-to-target arithmetic dependency σ of the form $\forall \mathbf{x}(\phi(\mathbf{x}) \wedge \beta(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}) \wedge \gamma(\mathbf{x}, \mathbf{y}))$ belonging to Σ_{st}, let $\gamma_\sigma(\mathbf{x})$ denote a quantifier-free arithmetic formula that is equivalent to the arithmetic formula $\exists \mathbf{y} \gamma(\mathbf{x}, \mathbf{y})$. Note that such a quantifier-free arithmetic formula exists by Theorem 2.2. The construction of the quantifier-free arithmetic formulas γ_σ can be viewed as a pre-processing step whose complexity depends only on the arithmetic schema mapping, not on the source $\mathbb{A}$-instance.

For a sequence of algebraic real numbers $\mathbf{r}$, we have that $\gamma_\sigma(\mathbf{r})$ holds if and only if the quantifier-free arithmetic formula $\gamma(\mathbf{r}, \mathbf{N})$ has a solution, where $\mathbf{N}$ is a sequence of distinct fresh null values. Intuitively, we will make use of γ_σ in the following way: whenever the left-hand side of σ is satisfied in I under some assignment of values to the universally quantified variables, we will evaluate γ_σ to test whether the right-hand side of σ can be satisfied in the target $\mathbb{A}$-instance. If the answer is negative, then, clearly, I has no solutions. If the answer is positive, then a universal solution is constructed by choosing fresh null values for the existentially quantified variables, and adding the facts and arithmetic formula of the right-hand side of σ to our target $\mathbb{A}$-instance.

We now describe the algorithm for testing the existence of solutions and for constructing a universal solution, if a solution exists, in more precise terms. The algorithm maintains a target $\mathbb{A}$-instance $\mathcal{J} = (J, \alpha)$. Initially, $J = \emptyset$ and $\alpha = \top$. For each source-to-target arithmetic dependency

$$\forall \mathbf{x}(\phi(\mathbf{x}) \wedge \beta(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}) \wedge \gamma(\mathbf{x}, \mathbf{y}))$$

belonging to Σ_{st}, and for each tuple of numbers $\mathbf{r}$ from $adom(I)$ such that I contains all facts in $\phi(\mathbf{r})$, and $\beta(\mathbf{r})$ holds, the algorithm tests whether $\gamma_\sigma(\mathbf{r})$ holds.

- If the answer is negative, the algorithm terminates and reports that no solution exists.
- If the answer is positive, the algorithm proceeds as follows: Consider first the simpler case where $\mathbf{r}$ consists entirely of rational numbers. In this case, we may choose fresh null values $\mathbf{N}$ for the variables $\mathbf{y}$, add $\psi(\mathbf{r}, \mathbf{N})$ to J and add $\gamma(\mathbf{r}, \mathbf{N})$ as a conjunct to α. In the general case, $\mathbf{r}$ consists of algebraic numbers, and doing the same would not yield a valid $\mathbb{A}$-instance, because only rational numbers are allowed in the arithmetic formulas defining $\mathbb{A}$-instances. Instead, in this case, we uniformly replace the occurrences of each algebraic number a with representation $\langle a \rangle = (p(x), r, s)$ by a fresh null value N_a in K_a and in $\gamma(\mathbf{r}, \mathbf{N})$, and we extend the arithmetic formula with a further conjunct $p(N_a) = 0 \wedge r \leq N_a \leq s$, which, intuitively, forces N_a to take the value a (cf. also the proof of Theorem 3.5). Note that the newly introduced null value N_a is guaranteed to occur in at least one relational atom (as required by the definition of $\mathbb{A}$-instances).

To complete the proof, first note that the above procedure runs in polynomial time (the tests of $\beta(\mathbf{r})$ and $\gamma_\sigma(\mathbf{r})$ can be performed in NC by Theorem 2.5). Second, note that, if the algorithm succeeds, the result $\mathcal{J}$ is a target $\mathbb{A}$-instance that, by construction, satisfies the source-to-target arithmetic dependencies in Σ_{st}. Moreover, if the algorithm succeeds, then $\mathsf{Rep}(\mathcal{J}) \neq \emptyset$, which implies that I has a ground solution. To see this, note that, whenever during the execution of the algorithm a conjunct $\gamma(\mathbf{r}, \mathbf{N})$ was added to α, then $\gamma_\sigma(\mathbf{r})$ was true, which means that $\gamma(\mathbf{r}, \mathbf{N})$ has a solution. Since each conjunct of α uses a disjoint set of nulls, this implies that the entire formula α has a solution; therefore, $\mathsf{Rep}(\mathcal{J}) \neq \emptyset$.

Now, it only remains to show that, if the algorithm succeeds, then $\mathcal{J}$ is a universal solution for I w.r.t. $\mathcal{M}$. This follows from the fact that each ground solution K of I has a subinstance $\widetilde{K} \subseteq K$ such that $\widetilde{K} \in \mathsf{Rep}(\mathcal{J})$. Indeed, such a $\widetilde{K}$ may be obtained using a modified version of the above algorithm: for each source-to-target arithmetic dependency

$$\forall \mathbf{x}(\phi(\mathbf{x}) \wedge \beta(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}) \wedge \gamma(\mathbf{x}, \mathbf{y}))$$

belonging to Σ_{st}, and for each tuple of numbers $\mathbf{r}$ from $adom(I)$ such that I contains all facts in $\phi(\mathbf{r})$ and $\beta(\mathbf{r})$ holds, the algorithm, instead of choosing fresh nulls, uses values $\mathbf{r}'$ from $adom(K)$ such that the facts in $\psi(\mathbf{r}, \mathbf{r}')$ belong to K and such that $\gamma(\mathbf{r}, \mathbf{r}')$ is satisfied. □

4.2 The case with target dependencies

As we saw earlier, in the presence of target dependencies, a single universal solution may not exist. Here, we show that, if the set of target dependencies is *weakly acyclic*, then a finite universal basis always exists (if a solution exists) and can be constructed by a polynomial-space algorithm.

Definition 4.2 (Weak acyclicity)**.** Let Σ_t be a set of target arithmetic dependencies. The dependency graph of Σ_t is constructed as follows. The nodes are the *positions* $R.A$, where R is a target relation and A is one of its attributes. Each arithmetic dependency

$$\forall \mathbf{x}\ (\varphi(\mathbf{x}) \wedge \beta(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}) \wedge \gamma(\mathbf{x}, \mathbf{y}))$$

in Σ_t gives rise to zero or more edges, as follows. A variable x that occurs both in the antecedent and in the consequent of the dependency is said to be a *propagated variable* (note that x must appear in ϕ but may not appear in ψ). For each propagated variable x, we do the following:

- Draw an edge from every position in which x occurs in ϕ to every position in which x occurs in ψ
- Draw a *special* edge from every position in which x occurs in ϕ to all positions in which one or more existentially quantified variables occur in ψ.

Σ_t is *weakly acyclic* if its dependency graph does not contain a cycle through a special edge.

The above definition of weak acyclicity coincides with the original definition in [14] for dependencies without arithmetic formulas. Indeed, for a set Σ of arithmetic dependencies, let $\widehat{\Sigma}$ be the set that contains, for every $\sigma \in \Sigma$ of the form $\forall \mathbf{x}\ (\varphi(\mathbf{x}) \wedge \beta(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}) \wedge \gamma(\mathbf{x}, \mathbf{y}))$, with ψ a non-empty conjunction of atoms, the dependency obtained by removing β and γ from σ (note that we ignore all constraint-generating dependencies, i.e., arithmetic dependencies where ψ is the empty conjunction). Then Σ is weakly acyclic as defined above if and only if $\widehat{\Sigma}$ is weakly acyclic as defined in [14]. In particular, equality-generating dependencies, being a special case of constraint-generating dependencies, are allowed to occur in Σ.

Definition 4.3 (Chase step)**.** Let $\mathcal{K} = (K, \alpha_K)$ be a target $\mathbb{A}$-instance and let

$$\sigma = \forall \mathbf{x}, \mathbf{y}(\phi(\mathbf{x}, \mathbf{y}) \wedge \beta(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x}, \mathbf{z}) \wedge \gamma(\mathbf{x}, \mathbf{z}))$$

be a target arithmetic dependency, where we use the notation $\mathbf{x}$ and $\mathbf{y}$ to distinguish between the universally quantified variables that occur in ψ, and the ones that do not.

Consider a tuple of values (constants or nulls) $\mathbf{w}, \mathbf{u}$ from $adom(K)$ such that all facts in $\phi(\mathbf{w}, \mathbf{u})$ belong to K.

Intuitively, there are two ways in which the arithmetic dependency σ may be satisfied when algebraic numbers are substituted for the nulls in $\mathbf{w}$ and in $\mathbf{u}$: either $\beta(\mathbf{w}, \mathbf{u})$ is false (in which case the left-hand side of the implication is false, and therefore the implication is satisfied), or the right-hand side of σ must be satisfied under some assignment that maps the variables $\mathbf{x}$ to $\mathbf{w}$. Correspondingly, we write

- $\mathcal{K} \xrightarrow{L\ \sigma,\mathbf{w},\mathbf{u}} \mathcal{K}'$ if $\mathcal{K}' = (K, \alpha_K \wedge \neg\beta(\mathbf{w}, \mathbf{u}))$, and
- $\mathcal{K} \xrightarrow{R\ \sigma,\mathbf{w}} \mathcal{K}'$ if $\mathcal{K}' = (K', \alpha_K \wedge \beta(\mathbf{w}, \mathbf{u}) \wedge \gamma(\mathbf{w}, \mathbf{N}))$, where $\mathbf{N}$ are fresh nulls, and K' extends K with the facts in $\psi(\mathbf{w}, \mathbf{N})$.

As in the proof of Theorem 4.1, since $\mathbf{w}$ may contain irrational algebraic numbers, the above definition of $\mathcal{K}'$ may not yield a valid $\mathbb{A}$-instance. We solve this problem by uniformly replacing all occurrences of algebraic numbers a by a fresh null value N_a and then extending the arithmetic formula with an extra conjunct that, intuitively, forces N_a to take the value a, exactly in the same way as in the proof of Theorem 4.1. If $\widehat{\mathcal{K}'}$ is the $\mathbb{A}$-instance obtained from $\mathcal{K}'$ in this way, then we use $\mathcal{K}'$ and $\widehat{\mathcal{K}'}$ interchangeably. In particular, we also write $\mathcal{K} \xrightarrow{L\ \sigma,\mathbf{w},\mathbf{u}} \widehat{\mathcal{K}'}$ and, respectively, $\mathcal{K} \xrightarrow{R\ \sigma,\mathbf{w}} \widehat{\mathcal{K}'}$.

Note that it may be the case that $\mathsf{Rep}(\mathcal{K}') = \emptyset$, even when $\mathsf{Rep}(\mathcal{K}) \neq \emptyset$. This may happen, for example, when $\beta(\mathbf{w}, \mathbf{u})$ or its negation is implied by the arithmetic formula of $\mathcal{K}$.

From a geometric perspective, the chase step can be understood as follows. Let $\mathcal{K} = (K, \alpha)$ be a target $\mathbb{A}$-instance, let σ be a dependency as in Definition 4.3, and suppose that all the facts in $\phi(\mathbf{w}, \mathbf{u})$ belong to K. If $\mathbf{N} = N_1, \ldots, N_n$ is an enumeration of all null values that occur in K, then the arithmetic formulas α and $\beta(\mathbf{w}, \mathbf{u})$ naturally define (possibly empty) algebraic surfaces in $\mathbb{A}^n$. Let A be the algebraic surface defined by α and let B be the algebraic surface defined by $\beta(\mathbf{w}, \mathbf{u})$. Each point in A corresponds to some $K \in \mathsf{Rep}(\mathcal{K})$. The chase step, intuitively, involves splitting the surface A into $A \setminus B$ and $A \cap B$, by refining the arithmetic formula α into $\alpha \wedge \neg\beta(\mathbf{w}, \mathbf{u})$ and $\alpha \wedge \beta(\mathbf{w}, \mathbf{u})$. This partitions $\mathsf{Rep}(\mathcal{K})$ into those ground $\mathbb{A}$-instances where $\beta(\mathbf{w}, \mathbf{u})$ holds, and those where it does not. Furthermore, in the latter case, the facts and arithmetical formula in the right-hand side of the dependency σ are added to the instance as well.

Definition 4.4 (Chase tree)**.** Let $\mathcal{K}_0$ be a target $\mathbb{A}$-instance, and Σ_t a set of target arithmetic dependencies. A *chase tree* for $\mathcal{K}_0$ with respect to Σ_t is a binary tree in which every node is labeled by a target $\mathbb{A}$-instance, and the edges are labeled as well, such that the following conditions hold:

- The root is labeled $\mathcal{K}_0$.
- Every non-leaf node labeled $\mathcal{K}$ has exactly two children, $\mathcal{K}_1$ and $\mathcal{K}_2$ with the following property: for some $\sigma \in \Sigma_t$ and for some tuples $\mathbf{w}, \mathbf{u}$, we have that $\mathcal{K} \xrightarrow{L\ \sigma,\mathbf{w},\mathbf{u}} \mathcal{K}_1$ and $\mathcal{K} \xrightarrow{R\ \sigma,\mathbf{w}} \mathcal{K}_2$ are valid chase steps, and, moreover, the edge from $\mathcal{K}$ to $\mathcal{K}_1$ is labeled $L\ \sigma, \mathbf{w}, \mathbf{u}$, and the edge from $\mathcal{K}$ to $\mathcal{K}_2$ is labeled $R\ \sigma, \mathbf{w}$.
- Each leaf node is labeled by a target $\mathbb{A}$-instance $\mathcal{K}$ satisfying Σ_t (we allow for the trivial case where $\mathsf{Rep}(\mathcal{K}) = \emptyset$).
- No edge label appears more than once on the same branch of the tree.

The *result* of a finite chase tree is the set of all labels $\mathcal{K}$ of leaf nodes, for which $\mathsf{Rep}(\mathcal{K}) \neq \emptyset$.

To simplify the exposition of the results below, it is convenient to assume that the arithmetic dependencies are in a special normal form.

Definition 4.5. We say that an arithmetic dependency

$$\forall \mathbf{x}(\phi(\mathbf{x}) \wedge \beta(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}) \wedge \gamma(\mathbf{x}, \mathbf{y}))$$

is *decoupled* if every variable occurs at most once in ϕ.

Every non-decoupled arithmetic dependency is logically equivalent to a decoupled arithmetic dependency, which can be obtained from it in linear time by renaming variables and adding one or more equalities as conjuncts to β. For example, $\forall x(P(x, x) \rightarrow Q(x))$ is logically equivalent to $\forall xy(P(x, y) \wedge x = y \rightarrow Q(x))$. This transformation does not affect weak acyclicity.

The following example explains why it is helpful to assume that the arithmetic dependencies are decoupled. Consider the target arithmetic dependency $\sigma = \forall x(R(x, x) \rightarrow P(x))$, and let $\mathcal{J}$ be the target $\mathbb{A}$-instance $(\{R(a, N)\}, N = a)$. Clearly, σ is *not* satisfied by $\mathcal{J}$. In fact, σ is false in every $J \in \mathsf{Rep}(\mathcal{J})$. Nevertheless, there is no chase step that can be applied here. Intuitively, this is because, in order to chase $\mathcal{J}$ with σ, we need to map the variable x to a and to N simultaneously, which is not possible. It follows that $\mathcal{J}$ does not have any chase tree with respect to σ. On the other hand, if we consider the decoupled arithmetic dependency $\sigma' = \forall xy(R(x, y) \wedge x = y \rightarrow P(x))$, which is equivalent to σ, then this problem does not arise. Indeed, we have:

Proposition 4.6. *If Σ_t is a set of decoupled target arithmetic dependencies, then every target $\mathbb{A}$-instance has a, possibly infinite, chase tree with respect to Σ_t.*

Proposition 4.7. *Let $\mathcal{M} = (\boldsymbol{S}, \boldsymbol{T}, \Sigma_{st}, \Sigma_t)$ be an arithmetic schema mapping. Let I be a ground source $\mathbb{A}$-instance, and let $\mathcal{J}_0$ be a target $\mathbb{A}$-instance that is a universal solution for I with respect to Σ_{st}. If t is a finite chase tree for $\mathcal{J}_0$ w.r.t. to Σ_t, then the result of t is a universal basis for I w.r.t. $\mathcal{M}$.*

Proof. (Sketch) Let $\mathcal{B}$ be the result of t. By construction, $\mathcal{B}$ consists of target $\mathbb{A}$-instances that are solutions of I (note that the source-to-target arithmetic dependencies in Σ_{st} remain true as the target $\mathbb{A}$-instance is extended with new facts and arithmetic formulas during the chase). To see that $\mathcal{B}$ is a universal basis for I, we use the same strategy as in the proof of Theorem 4.1, i.e., we show that for every ground solution K, there is a subset $\widetilde{K} \subseteq K$ such that $\widetilde{K} \in \mathsf{Rep}(\mathcal{J})$ for some $\mathcal{J} \in \mathcal{B}$ (such a $\mathcal{J}$ can be found as the leaf of a branch of t that is by obtained by letting the given ground $\mathbb{A}$-instance K guide our choices in the chase tree). □

Proposition 4.8. *For every weakly acyclic set of target arithmetic dependencies Σ_t, there is a polynomial $p(\cdot)$ such that for every target $\mathbb{A}$-instance $\mathcal{K}$, the depth of every chase tree of $\mathcal{K}$ with respect to Σ_t is bounded by $p(|\mathcal{K}|)$.*

We now establish the main result of this section.

Theorem 4.9. *Let $\mathcal{M} = (\boldsymbol{S}, \boldsymbol{T}, \Sigma_{st}, \Sigma_t)$ be an arithmetic schema mapping, such that Σ_t is weakly acyclic. There is*

a polynomial-space algorithm that, given a ground source $\mathbb{A}$-instance I, tests whether a solution exists and, if so, produces a finite universal basis for I, consisting of target $\mathbb{A}$-instances whose size is bounded by $p(|I|)$, for some polynomial $p(\cdot)$ that depends only on $\mathcal{M}$.

Proof. As discussed earlier, we may assume without loss of generality that Σ_t consists of decoupled target arithmetic dependencies. The algorithm first tests for the existence of a solution of I with respect to Σ_{st}. If there is no solution, then also there is no solution with respect to Σ_{st} and Σ_t. Otherwise, the algorithm continues to construct in polynomial time a target $\mathbb{A}$-instance $\mathcal{J}_0$ that is a universal solution for I with respect to Σ_{st}. Next, the algorithm constructs a finite chase tree for $\mathcal{J}_0$ with respect to Σ_t, in a depth-first manner, keeping in memory only one branch at a time. It enumerates all leafs $\mathcal{J}$ and outputs those for which $\mathsf{Rep}(\mathcal{J}) \neq \emptyset$. The resulting set $\mathcal{B}$ of target $\mathbb{A}$-instances is guaranteed to be a, possibly empty, universal basis for I. Moreover, a solution for I exists if and only if $\mathcal{B}$ is non-empty. The algorithm requires only polynomial space (using Proposition 3.2 and Theorem 3.8, and the fact that co$\exists\mathbb{R} \subseteq$ PSpace). □

Note that, in Theorem 4.9, the size of the universal basis may be exponential, but each member of the universal basis is of polynomial size; moreover, the algorithm can enumerate the members of the universal basis (in exponential time) using only a polynomial amount of memory.

4.3 Query answering

Theorem 4.1, together with Theorem 3.5, immediately implies that, for arithmetic schema mappings without target dependencies, the certain answers of every fixed arithmetic query can be computed in co$\exists\mathbb{R}$. We will show that the same holds true in the more general case of arithmetic schema mappings with a weakly acyclic set of target dependencies; moreover, computing the certain answers can be a co$\exists\mathbb{R}$-complete problem.

Theorem 4.10. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be an arithmetic schema mapping, where Σ_t is a weakly acyclic set of target arithmetic dependencies.*

1. *The problem of testing whether a given ground source $\mathbb{A}$-instance has a solution with respect to $\mathcal{M}$ is in $\exists\mathbb{R}$. Depending on $\mathcal{M}$, it can be $\exists\mathbb{R}$-complete.*
2. *For every fixed arithmetic conjunctive query q, the problem of computing $\mathsf{Cert}_{\mathcal{M}}(q, I)$, given a ground source $\mathbb{A}$-instance I, is in co$\exists\mathbb{R}$. Depending on $\mathcal{M}$ and q, it can be co$\exists\mathbb{R}$-complete.*

In what follows, we describe the basic approach and main ingredients of the proof. First, it is easy to see that it suffices to establish the upper bound (membership in co$\exists\mathbb{R}$) for computing the certain answers, and the lower bound ($\exists\mathbb{R}$-hardness) for the existence of solutions problem.

The argument for the upper bound relies on the following non-trivial closure property of the complexity class $\exists\mathbb{R}$, which we establish here: $\exists\mathbb{R}$ is closed under NP-reductions. An NP-*reduction* from a decision problem A to a decision problem B is a non-deterministic polynomial-time Turing machine T with output, such that the (possibly exponentially large) set of possible outputs $\{b_1, \ldots, b_n\}$ of T on an input a satisfies the condition that a is in A if and only if at least one b_i is in B. Ordinary polynomial-time many-one reductions are a special case of NP-reductions, in which the Turing machine is deterministic. We say that a complexity class C is *closed under* NP-*reductions* if the following holds: whenever a decision problem A admits an NP-reduction to a problem B in C, then A is also in C. For example, it is easy to see that the complexity classes NP and PSpace are closed under NP-reductions. The complexity class PTime is not closed under NP-reductions, unless P=NP.

Theorem 4.11. *$\exists\mathbb{R}$ is closed under* NP-*reductions.*

Theorem 4.11 is used to prove the upper bound for the second part of Theorem 4.10. Specifically, we give an NP-reduction from the problem of computing the certain answers of a query to the problem of evaluating the same query on an $\mathbb{A}$-instance; the latter problem was already shown to belong to $\exists\mathbb{R}$ in Theorem 3.5. Our NP-reduction, roughly, consists of non-deterministically choosing path in the chase tree, resulting either in a member of the universal basis or in a target $\mathbb{A}$-instance $\mathcal{J}$ such that $\mathsf{Rep}(\mathcal{J}) = \emptyset$.

For the co$\exists\mathbb{R}$-hardness in the first part of Theorem 4.10, observe first that the input to the existence-of-solutions problem consists of a ground source $\mathbb{A}$-instance only, i.e., an instance without nulls and without an arithmetic formula. At first sight, it is not clear how one would proceed to reduce the decision problem of the existential theory of the reals (where the input is an arithmetic formula) to the problem at hand, and one may even think that this is not possible. Instead, our proof is based on a reduction from the Rectilinear Crossing Number Problem, a classical problem from graph theory that we describe next.

The *rectilinear crossing number* of a graph G, lin-cr(G), is the smallest number of crossings in a straight-line drawing of G, that is, a drawing in which every edge is represented by a straight-line segment and at most two edges intersect in a point. Let $\mathbf{K}_n$ be the n-element clique. It is easy to see that lin-cr($\mathbf{K}_4$) = 0. Furthermore, lin-cr($\mathbf{K}_5$) = 1, because, in every straight-line drawing of this graph, at least one pair of edges cross, and it is possible to draw this graph with straight lines, such that only one pair of edges cross (see [32] for a picture). Computing the rectilinear crossing number has turned out to be a challenging problem, even for rather simple graphs, such as cliques. Specifically, over the past few decades, the exact value of lin-cr($\mathbf{K}_n$) has been determined for $n \leq 30$, but is currently open for $n > 30$ (see [24]). The Rectilinear Crossing Number Problem is the following decision problem: given a graph G and a natural number k, is lin-cr(G) $\leq k$? It has been recently shown that the Rectilinear Crossing Number Problem is $\exists\mathbb{R}$-complete ([27], building on [9]).

We show that there is an arithmetic schema mapping $\mathcal{M}$ with a weakly acyclic set of arithmetic target constraints, such that, given a graph G and a natural number k, we can construct in polynomial time a ground source $\mathbb{A}$-instance $I_{G,k}$, whose ground solutions, intuitively, represent the straight-line drawings of G with at most k crossings. Interestingly, multiplication is used in the reduction only for expressing that three points are colinear.

5. Linear Arithmetic and the Rationals

We used the language of arithmetic formulas to specify operations and constraints on real numbers. In this section, we

consider *linear arithmetic formulas* over the *rationals*. By definition, a *linear arithmetic formula* is an arithmetic formula that only uses multiplications with constants. In other words, we disallow polynomials of degree two or higher.

The restriction to rational numbers and the restriction to linear arithmetic formulas go hand-in-hand in a natural way, for the following reasons. To begin with, the first-order theory of the rationals with addition and multiplication is undecidable [26] (hence, in particular, it does not admit quantifier elimination). Moreover, even the decidability of solvability of quantifier-free arithmetic formulas over the rationals (known as "Hilbert's 10th problem over the rationals") is an open problem. Also, while systems of linear equations with rational coefficients can be solved in polynomial time, it is not known whether the same holds true when algebraic real numbers are allowed as coefficients. Indeed, this problem subsumes the "sums-of-square-roots" problem, which asks for given integers $k_1, \ldots, k_n, k_{n+1}$ (in binary) whether $\sqrt{k_1} + \cdots + \sqrt{k_n} < k_{n+1}$, and which is not known to be solvable in polynomial time (see [4]).

An arithmetic dependency is *linear* if it uses only linear arithmetic formulas. Likewise for *linear* arithmetic conjunctive queries. A *linear arithmetic schema mapping* is a schema mapping specified by linear arithmetic dependencies.

Definition 5.1. Let $\mathbf{R}$ be a schema. An $\mathbb{LQ}$-instance over $\mathbf{R}$ is a pair $\mathcal{I} = (I, \alpha)$ with $I = (R_1^I, \ldots, R_n^I)$, where each R_k^I is a finite relation on $\mathbb{Q} \cup \mathsf{Nulls}$ of appropriate arity, and α is a linear quantifier-free arithmetic formula whose free variables are the nulls occurring in I.

Here, all rational numbers are assumed to be represented as pairs of integers (written in binary notation).

Let $\mathcal{I}$ be an $\mathbb{LQ}$-instance and q a linear arithmetic conjunctive query. In principle, $q(\mathcal{I})$ could be defined in two ways: in terms of $\mathsf{Rep}(\mathcal{I})$, or in terms of just the $\mathbb{LQ}$-instances in $\mathsf{Rep}(\mathcal{I})$. It turns out that this does not make a difference.

Proposition 5.2. *Let $\mathcal{I}$ be a $\mathbb{LQ}$-instance and q a linear arithmetic conjunctive query. Then*

$$\begin{aligned} q(\mathcal{I}) &= \bigcap\{q(I) \mid I \in \mathsf{Rep}(\mathcal{I})\} \\ &= \bigcap\{q(I) \mid I \in \mathsf{Rep}(\mathcal{I}) \text{ and } I \text{ is an } \mathbb{LQ}\text{-instance}\} \end{aligned}$$

In particular, if $\mathsf{Rep}(\mathcal{I}) \neq \emptyset$, then $\mathsf{Rep}(\mathcal{I})$ contains an $\mathbb{LQ}$-instance.

Proof. The first part of the equation holds by definition, while the second part follows from Theorem 2.4. □

Straightforward adaptations of the proofs of Proposition 3.2, Theorem 3.5, and Theorem 3.8 yield:

Proposition 5.3. *Let q be a linear arithmetic conjunctive query, and let t be a linear arithmetic dependency.*

1. *The following problem is co*NP*-complete: given an $\mathbb{LQ}$-instance $\mathcal{I}$, is $\mathsf{Rep}(\mathcal{I}) = \emptyset$?*
2. *The following problem is in co*NP*: given an $\mathbb{LQ}$-instance $\mathcal{I} = (I, \alpha)$ and a tuple of rational numbers $\boldsymbol{a}$, test whether $\boldsymbol{a} \in q(\mathcal{I})$. Moreover, there are conjunctive queries without arithmetic formulas for which the problem is co*NP*-complete.*
3. *The problem of testing whether a given $\mathbb{LQ}$-instance weakly satisfies t is in* NP*. Depending on t, this problem can be* NP*-complete.*
4. *The problem of testing whether a given $\mathbb{LQ}$-instance satisfies t is in co*NP*. Depending on t, this problem can be co*NP*-complete.*

A close analysis of the proof of Theorem 4.1 and Theorem 4.9 shows that, when the input consists of a ground source $\mathbb{LQ}$-instance and a linear arithmetic schema mapping, then the output consists of $\mathbb{LQ}$-instances as well. In other words, no irrational numbers and no multiplications are introduced during the chase. Based on this, and on the fact that the complexity class NP is (trivially) closed under NP-reductions, we can adapt the proof of Theorem 4.10 in a straightforward manner to establish the upper bounds stated in the following result.

Theorem 5.4. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a linear arithmetic schema mapping, where Σ_t is a weakly acyclic set of linear target arithmetic dependencies. Then*

1. *The problem of testing that a given ground source $\mathbb{LQ}$-instance has a solution with respect to $\mathcal{M}$ is in* NP*. Depending on $\mathcal{M}$, it can be* NP*-complete.*
2. *For every linear arithmetic conjunctive query q, the problem of computing $\mathsf{Cert}_M(q, I)$, given a ground source $\mathbb{LQ}$-instance I, is in co*NP*. Depending on $\mathcal{M}$ and q, it can be co*NP*-complete.*

The lower bounds in Theorem 5.4 follow immediately from the fact that there is a schema mapping $\mathcal{M}$ without target constraints and without arithmetic formulas, and a conjunctive query with inequalities q, for which computing certain answers is coNP-complete [1] (see also [14, 21]).

6. Related Work

In [23], *constrained tuple-generating dependencies* (ctgds) were first introduced as a generalization of tuple-generating dependencies, constrained functional dependencies [22], and constraint-generating dependencies [6]. These ctgds are defined in [23] relative to a given *constraint domain*, which is a pair $(\mathcal{D}, \mathcal{L})$, where $\mathcal{D}$ is a structure and $\mathcal{L}$ is a fragment of first-order logic. Arithmetic dependencies are the special case of ctgds where $\mathcal{D} = (\mathbb{A}, <, +, \times, (c)_{c\in\mathbb{Q}})$ and $\mathcal{L}$ is the set of all quantifier-free formulas. In [23], two chase procedures were proposed for testing implication between ctgds. These chase procedures, however, do not always terminate; this is so because ctgds generalize tuple-generating dependencies, and the implication problem for the latter is undecidable [7]. In [30], a third (incomplete) chase procedure is introduced, which improves on both previous chase procedures, and which, moreover, can be used to test implication for *disjunctive ctgds*, that is, the generalization of ctgds that allows for arbitrary use of disjunction in conclusion of dependencies. In [12], yet another chase procedure for constrained tuple-generating dependencies is proposed, which is shown to constitute a sound and complete proof system, when interpreted over possibly infinite instances. Each of these papers considers arbitrary constraint domains $(\mathcal{D}, \mathcal{L})$, and relies on an oracle for testing satisfiability and entailment of $\mathcal{L}$-formulas over $\mathcal{D}$. In particular, no decision procedures or computational complexity results are obtained.

In [2], Afrati, Li, and Pavlaki studied *data exchange settings with arithmetic comparisons* (DEAC). Syntactically, DEAC settings are the special case of arithmetic schema mappings in which addition, multiplication, negation, and

disjunction are not allowed in the arithmetic formulas. In particular, DEAC settings are linear arithmetic schema mappings. The arithmetic comparisons in DEAC settings considered in [2] are interpreted over an arbitrary countable dense linear order without endpoints, which we may assume, without loss of generality, to be the linear order of the rational numbers (since all countable dense linear orders without endpoints are isomorphic to the rational numbers). Similarly, the *conjunctive queries with arithmetic comparisons* (CQACs) studied in [2] are a special case of linear arithmetic conjunctive queries. Consequently, our work on linear arithmetic schema mappings and on linear arithmetic conjunctive queries, can be seen as extending the work on DEAC settings and CQACs in [2] to a richer schema-mapping language. Note, however, that some of terminology used [2] is different from the one used here; in particular, the term *solution* is used in [2] for what we called here a *universal basis*; we used the latter term because it had been introduced earlier in the context of peer data exchange [16]. Our results in Section 5 show that the data complexity of the certain answers of linear arithmetic conjunctive queries w.r.t. linear arithmetic schema mappings coincides with the data complexity of CQAs w.r.t. DEAC settings. Thus, our results imply that the extension of the DEAC setting in [2] to the richer setting of linear arithmetic schema mappings does not come at the expense of an increase in worst-case complexity.

It is worth noting that the main chase procedure proposed in [2] produces a universal basis (in our terminology) whose number of elements is at least as big as the number of elements in the universal basis produced by our chase algorithm, when applied to DEAC settings. Furthermore, there are DEAC settings for which the universal basis produced by our algorithm has exponentially fewer members than the universal basis produced by the main chase procedure in [2]. The following example illustrates this phenomenon.

Example 6.1. Let $\mathcal{M}$ be the schema mapping given by the source-to-target dependencies

$$\forall x(R(x) \rightarrow \exists y\ T(x,y)) \quad \text{and} \quad \forall x(P(x) \rightarrow \exists y\ Q(x,y))\ ,$$

and the target arithmetic dependency

$$\forall xy(T(x,y) \wedge x \neq y \rightarrow Q(x,y))\ .$$

Let I be a source $\mathbb{LQ}$-instance consisting of the facts $R(0)$ and $P(1), P(2), \ldots, P(n)$, for some natural number n. Our chase algorithm, on input I, produces a universal basis consisting of $(\{T(0,N_0), Q(0,N_0), Q(1,N_1), \ldots, Q(n,N_n)\}, 0 \neq N_0)$ and $(\{T(0,N_0), Q(1,N_1), \ldots, Q(n,N_n)\}, \neg(0 \neq N_0))$. This is optimal, because it can be shown that there is no single target $\mathbb{LQ}$-instance that is a universal solution for I. On the other hand, the chase method proposed in [2] yields a universal basis consisting of exponentially many target $\mathbb{LQ}$-instances (one for each way of ordering the null values $N_0, N_1, \ldots, N_n$ relative to each other and relative to the constants $0, 1, \ldots, n$). Intuitively, this difference in behavior stems from the fact that the chase from [2] maintains, at each step, a complete linear order on the nulls and the constants, and therefore, must branch whenever a new null value is introduced. Our chase, on the other hand, is "lazy" in the sense that it only branches when needed for evaluating a specific arithmetic formula in the left-hand side of a target arithmetic dependency.

Finally, we note that the need for arithmetic comparisons arises naturally in computing *core* universal solutions. Specifically, as shown in [29], for every schema mapping $\mathcal{M}$ specified by source-to-target tgds, there is a schema mapping $\mathcal{M}'$ specified by a generalization of source-to-target tgds that involves arithmetic comparisons, such that for every source instance I, the core of the universal solutions of I with respect to $\mathcal{M}$ can be obtained by chasing I with the dependencies of $\mathcal{M}'$. Moreover, the use of arithmetic comparisons cannot be avoided in general.

Data Exchange Setting	Existence of Sols.	Certain Answers
Arithmetic schema mappings without target dependencies over $(\mathbb{A}, <, +, \times)$	in PTIME	in co$\exists\mathbb{R}$
Arithmetic schema mappings with weakly acyclic target dependencies over $(\mathbb{A}, <, +, \times)$	$\exists\mathbb{R}$-complete	co$\exists\mathbb{R}$-complete
Linear arithmetic schema mappings without target dependencies over $(\mathbb{Q}, <, +)$	in PTIME	coNP-complete
Linear arithmetic schema mappings with weakly acyclic target dependencies over $(\mathbb{Q}, <, +)$	NP-complete	coNP-complete

Table 1: Summary of complexity results

7. Concluding Remarks

We have initiated the study of data exchange for arithmetic schema mappings and arithmetic queries. For arithmetic schema mappings without target dependencies, we have presented a polynomial-time algorithm that tests for the existence of solution, and, if there is a solution, computes a universal solution. In the case of arithmetic schema mappings with a weakly acyclic set of target dependencies, a universal solution may not exist, but we show that a finite universal basis can be computed in polynomial space (if a solution exists). Based on these results, we have classified the complexity of computing the certain answers of arithmetic conjunctive queries over the target schema. Our main complexity results are summarized in Table 1. One question that remains open is whether computing the certain answers of arithmetic conjunctive queries w.r.t. arithmetic schema mappings without target dependencies is a co$\exists\mathbb{R}$-complete problem. Our results also contribute to the understanding of the complexity class $\exists\mathbb{R}$, because we show that $\exists\mathbb{R}$ is closed under NP-reductions and also because our results identify several new $\exists\mathbb{R}$-complete problems.

The work reported here opens up several interesting directions for future research. We conclude by mentioning three such concrete directions.

The first direction is to identify natural cases of data exchange with arithmetic schema mapping for which the problems of testing for the existence of solutions and computing the certain answers have complexity lower than $\exists\mathbb{R}$-complete and co$\exists\mathbb{R}$-complete. Going back to the examples of arithmetic dependencies given in Section 1, we can see that the third example has no existential quantified variables (i.e., it is an example of a *full* arithmetic dependency), while the

other three are of a restricted form, namely, the existentially quantified variables are determined via polynomials by the values of the universally quantified variables, hence no genuine nulls are created. We believe that, generalizing from these examples, it will be possible to identify fragments of arithmetic dependencies that on the one hand are of practical relevance and, on the other, have the property that the basic computational tasks we considered are tractable.

The second direction concerns metadata management. How do we manage arithmetic schema mappings? In particular, what are the semantics of the composition and the inverse operator on arithmetic schema mappings, and what is the "right" language for expressing these operators?

The third direction is concerned with schema-mapping design. In [3], a methodology for schema-mapping design was proposed, based on the use of data examples. A fundamental problem in that context is the problem of finding a schema mapping that "fits" a collection of data examples. In general, for a given collection of data examples, a fitting schema mapping consisting of source-to-target dependencies and target dependencies may not exist. In such cases, the question arises whether we can find a schema mapping in a richer language and, in particular, whether we can find an *arithmetic* schema mapping that fits these data examples.

Acknowledgements. We thank Nimrod Megiddo and Marcus Schaefer for helpful discussions and pointers.

8. References

[1] S. Abiteboul and O. M. Duschka. Complexity of Answering Queries Using Materialized Views. In *PODS*, pages 254–263, 1998.

[2] F. N. Afrati, C. Li, and V. Pavlaki. Data exchange in the presence of arithmetic comparisons. In *EDBT*, pages 487–498, 2008.

[3] B. Alexe, B. ten Cate, P. G. Kolaitis, and W.-C. Tan. Designing and refining schema mappings via data examples. In *SIGMOD*, pages 133–144. ACM, 2011.

[4] E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen, and P. B. Miltersen. On the complexity of numerical analysis. In *IEEE Conference on Computational Complexity*, pages 331–339, 2006.

[5] M. Arenas, P. Barceló, L. Libkin, and F. Murlak. *Relational and XML Data Exchange*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

[6] M. Baudinet, J. Chomicki, and P. Wolper. Constraint-generating dependencies. *Journal of Computer and System Sciences*, 59(1):94 – 115, 1999.

[7] C. Beeri and M. Y. Vardi. The implication problem for data dependencies. In *ICALP*, pages 73–85, 1981.

[8] M. Ben-Or, D. Kozen, and J. Reif. The complexity of elementary algebra and geometry. In *STOC*, pages 457–464. ACM, 1984.

[9] D. Bienstock. Some provably hard crossing number problems. *Discrete and Computational Geometry*, 6:443–459, 1991.

[10] J. Canny. Some algebraic and geometric computations in PSPACE. In *STOC*, pages 460–467. ACM, 1988.

[11] J. H. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *J. Symb. Comput.*, 5:29–35, February 1988.

[12] D. Dou and S. Coulondre. A sound and complete chase procedure for constrained tuple-generating dependencies. *Journal of Intelligent Information Systems*, Online First:1–22, 2012.

[13] I. Z. Emiris, B. Mourrain, and E. P. Tsigaridas. Real algebraic numbers: Complexity analysis and experimentation. In *Reliable Implementation of Real Number Algorithms*, pages 57–82, 2008.

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

[15] J. Ferrante and C. Rackoff. A decision procedure for the first order theory of real addition with order. *SIAM J. Comput.*, 4(1):69–76, 1975.

[16] A. Fuxman, P. G. Kolaitis, R. J. Miller, and W. C. Tan. Peer data exchange. *ACM Trans. Database Syst.*, 31(4):1454–1498, 2006.

[17] R. Greenlaw, H. Hoover, and W. Ruzzo. *Limits to parallel computation: P-completeness theory*. Oxford University Press, 1995.

[18] J. Heintz, M.-F. Roy, and P. Solernó. Sur la complexité du principe de Tarski-Seidenberg. *Bulletin de la Société Mathématique de France*, tome 118(1):101–126, 1990.

[19] L. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[20] P. G. Kolaitis. Schema mappings, data exchange, and metadata management. In *PODS*, pages 61–75, 2005.

[21] A. Madry. Data exchange: On the complexity of answering queries with inequalities. *Inf. Process. Lett.*, 94(6):253–257, 2005.

[22] M. J. Maher. Constrained dependencies. *Theor. Comput. Sci.*, 173(1):113–149, 1997.

[23] M. J. Maher and D. Srivastava. Chasing constrained tuple-generating dependencies. In R. Hull, editor, *PODS*, pages 128–138. ACM Press, 1996.

[24] The rectilinear crossing number project. `http://www.ist.tugraz.at/staff/aichholzer/research/rp/triangulations/crossing/`. Accessed 10/17/2012.

[25] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. Part I. *J. Symb. Comput.*, 13(3):255–299, 1992.

[26] J. Robinson. Definability and decision problems in arithmetic. *J. Symb. Logic*, 14(2):pp. 98–114, 1949.

[27] M. Schaefer. Complexity of some geometric and topological problems. In *Graph Drawing*, volume 5849 of *LNCS*, pages 334–344. 2010.

[28] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1951.

[29] B. ten Cate, L. Chiticariu, P. G. Kolaitis, and W. C. Tan. Laconic schema mappings: Computing the core with sql queries. *PVLDB*, 2(1):1006–1017, 2009.

[30] J. Wang, R. W. Topor, and M. J. Maher. Reasoning with disjunctive constrained tuple-generating dependencies. In *DEXA*, pages 963–973, 2001.

[31] V. Weispfenning. The complexity of linear problems in fields. *J. Symb. Comput.*, 5:3–27, February 1988.

[32] Wolfram MathWorld. Rectilinear crossing problem. `http://mathworld.wolfram.com/RectilinearCrossingNumber.html`. Accessed 10/17/2012.