Modulo Constraints and the Complexity of Typechecking XML Views [⋆] [⋆⋆]

Jerzy Marcinkowski and Piotr Wieczorek

Institute of Computer Science,
University of Wrocław, Poland

Abstract. The typechecking problem for transformations of relational data into tree data is the following: given a relational-to-XML transformation P, and an XML type d, decide whether for every database instance $\mathcal{D}$ the result of the transformation P on $\mathcal{D}$ satisfies d. TreeQL programs with projection-free conjunctive queries (see [2]) are considered as transformations and DTDs with arbitrary regular expressions as XML types.

A non-elementary upper bound for the typechecking problem was already given by Alon et al. [2] (although in a more general setting, where equality and negation in projection-free conjunctive queries and additional universal integrity constraints are allowed).

In this paper we show that the typechecking problem is coNEXPTIME-complete.

As an intermediate step we consider the following problem, which can be formulated independently of XML notions. Given a set of triples of the form (φ, k, j), where φ is a projection-free conjunctive query and k, j are natural numbers, decide whether there exists a database $\mathcal{D}$ such that, for each triple (φ, k, j) in the set, there exists a natural number α, such that there are exactly $k + j * \alpha$ tuples satisfying the query φ in $\mathcal{D}$. Our main technical contribution consists of a NEXPTIME algorithm for the last problem.

1 Introduction

During the last few years, XML [19] has become the standard in data exchange on the web. Since the data to be exchanged is most commonly stored in relational databases, one uses transformation programs to automatically transform the relational data into actual XML. In order to ensure that this data is correctly interpreted by all receivers, the generated XML is required to be of a specific agreed-upon *type*. Here, a type is essentially just a set of trees (i.e., a *tree language*). Many proposals for defining such types exist, amongst them DTDs [19], XML Schemas [20], and RELAX NG [5] definitions. Here, RELAX NG is the

[⋆] Partially supported by Polish Ministry of Science and Higher Education research project N206022 31/3660, 2006/2009.

[⋆⋆] This paper is an extended version of [21], where the coNEXPTIME upper bound was shown.

most powerful, as it can define the full class of *regular tree languages* [17], while XML Schemas and DTDs can only define fragments of this class [15]. In general, the *relational-to-XML typechecking problem* consists of deciding, given a transformation program P, an output type, and a set of integrity constraints, whether every database that satisfies the integrity constraints is transformed into a tree in the output type. Thus, the problem is parameterized by:

- the class of transformations,
- the class of output tree languages,
- the class of integrity constraints.

Alon et al. [2] present a study of decidability and complexity of many versions of the problem. As a formalism to define transformations the authors introduce TreeQL programs. TreeQL is an abstraction of practical languages such as RXL [8]. A TreeQL program is a tree in which each node is labeled with a symbol from a finite alphabet and with a logical formula.

The result of a TreeQL program P on a database $\mathcal{D}$ is a tree reflecting the structure of P such that each node n of P is substituted by as many nodes as there are tuples in the database $\mathcal{D}$ satisfying the formula at n. The nodes of the output tree inherit, as their labels, the symbols that label nodes of the program tree. The output type is specified by a DTD — a formalism which puts local restrictions on trees, that is, it restricts how the sequence of child labels of a node looks like.

Decidability results in [2] include a coNEXPTIME upper bound on typechecking TreeQL programs with conjunctive queries (with negation and equality), DTDs with star-free regular languages as the output types and the integrity constraints in FO($\exists^*\forall^*$). When arbitrary regular expressions are allowed in DTDs the authors show decidability of typechecking TreeQL programs with projection-free conjunctive queries [1] (with negation and equality) and integrity constraints in FO($\forall^*$). In the latter case, however, the complexity is prohibitively high—the proof uses a combinatorial argument based on Ramsey's Theorem and yields a non-elementary upper bound. It was left as an open problem in [2] whether the bound can be improved. We show that such an improvement is possible, at least for a restricted case: the typechecking problem for DTDs with arbitrary regular expressions as the output types and projection-free conjunctive queries in TreeQL programs, but without integrity constraints is coNEXPTIME-complete.

Our approach is as follows. Inspired by the notion of *the modulo property* [2], we perform the reduction of the complement of the typechecking problem to the following problem. Given a set of triples of the form (φ, k, j), where φ is a projection-free conjunctive query and k, j are natural numbers, decide whether there exists a database $\mathcal{D}$ such that for each triple (φ, k, j) in the set, there exists a natural number α, such that there are exactly $k + j * \alpha$ tuples satisfying the query φ in $\mathcal{D}$. Such triples put constraints on a relational database that

[1] If conjunctive queries with projections are allowed, the problem (even in our simple setting) is not known to be decidable.

we call *modulo constraints*. The main technical part of the proof consists of a NEXPTIME algorithm for deciding satisfiability of a set of modulo constraints.

Alon et al. [2, Theorem 4.1] show that the typechecking problem for TreeQL programs with projection-free CQs with negation and DTDs with very simple regular expressions is coNEXPTIME-hard. The authors of [2] note that it is open whether this lower bound holds when negation is not allowed in the queries. In Section 8, we give a positive answer, showing that the typechecking problem is still coNEXPTIME-hard for TreeQL programs with projection-free CQs without negation, provided that arbitrary regular expressions are allowed in DTDs. Consequently, our coNEXPTIME upper bound for the typechecking problem is tight. Similarly as before, first we consider the problem of satisfiability of a set of modulo constraints and then we show the lower bound for the typechecking problem.

Related work. The typechecking problem recently gained a lot of attention in the literature, especially in the context of typechecking XML-to-XML transformations. As relational structures can easily be encoded as trees, this problem is closely related to ours. In the XML-to-XML context we are given input and output tree languages and a transformation, and we are asked whether every tree in the input tree language is transformed to a tree in the output language. The problem was studied in [16], where the input and output types were regular tree languages and transformations were expressed by k-pebble transducers. As long as the data values in trees are not considered, the problem is decidable. The complexity, however, is non-elementary. If the nodes in trees can be equipped with data values from an infinite domain, in addition to the tags from a finite alphabet, then, as it was shown in [3], the problem quickly gets undecidable and in the decidable cases the complexity is rather high.

In [12] and [13] Martens and Neven considered transformations in a form of a single top-down traversal of the input tree, during which every node can be replaced by a new tree, deleted or copied. Such transformations can be used for restructuring and filtering rather than for advanced querying, but on the other hand, the obtained complexity results range from EXPTIME to PTIME.

Macro tree transducers (mtts) [7] are a general formalism that covers many useful XML transformations. In particular, all standard features of most XML transformation languages can be modeled by compositions of three mtts [10]. The typechecking problem for mtts was studied in [10, 11], where, similarly as in [16], the technique of *inverse type inference* was used. Inverse type inference computes the pre-image of all ill-formed outputs. Then, the typechecking reduces to checking whether the intersection of a given input type and the computed pre-image is empty. The typechecking problem for mtts is decidable and there are practical subclasses for which it is tractable [11, 9].

The all above results assume that the input and output tree languages as well as the transformation are part of the input. If fixed input or/and output languages are considered then the computational complexity of the typechecking problem can be lowered in a number of cases [14].

Outline of the paper. The rest of the paper is organized as follows. In Section 2 we give the necessary preliminaries. In a short Section 3 we state Theorem 5, which is our main theorem, and formulate an intermediate result – the main lemma needed for the proof of Theorem 5. In Sections 4-6, which are the main technical part, we prove the intermediate result. In Section 7 we use some of the ideas from [2] and show how the intermediate result implies the main result. Finally, in Section 8 we prove the matching lower bounds, first for the intermediate problem and then, as a consequence, the lower bound for the main problem.

2 Preliminaries

XML and XML Types. We abstract XML documents as ordered, unranked, finite trees whose nodes are labeled with symbols from some finite alphabet Σ (see Figure 1). We denote the label of a node v by $\mathrm{lab}(v)$ and the root node of a tree t by $\mathrm{root}(t)$. *A Document Type Definition (DTD)* is a way of defining a tree language. A DTD d specifies for each symbol $\sigma \in \Sigma$ a regular expression that defines a regular language $d(\sigma)$.[2] We say that a tree t satisfies d if for every node v of t with children $v_1, \dots, v_n$ the word $lab(v_1) \dots lab(v_n)$ is in the regular language $d(lab(v))$. If v is a leaf, then the empty word ϵ has to be in $d(lab(v))$. The language of trees satisfying a DTD d is denoted by $L(d)$.

Databases and queries. *A vocabulary S* is a set of *relation symbols* and a set of *constant symbols. A database over S* or an *S-structure* $\mathcal{A}$ [1, 6] consists of a domain $\mathrm{dom}(\mathcal{A})$, a relation $R^{\mathcal{A}}$ for each relation symbol $R \in S$ and an element $c^{\mathcal{A}}$ in $\mathrm{dom}(\mathcal{A})$ for each constant symbol c in S. A relation $R^{\mathcal{A}}$ is a finite set of tuples of elements of $\mathrm{dom}(\mathcal{A})$. Thus, we do not allow duplicates in our databases. We assume that each element in $\mathrm{dom}(\mathcal{A})$ is in some tuple in some relation of $\mathcal{A}$. *An atomic formula over S (an atom over S)* is a formula of the form $R(\boldsymbol{a})$, where R is a relational symbol in S and a tuple $\boldsymbol{a}$ consists of constant symbols from S and variables. *A projection-free conjunctive query (projection-free CQ)* $\varphi(x_1, \dots, x_n)$ is a conjunction of atomic formulas. By $\mathrm{Vars}(\varphi)$ we denote the set of variables of φ (note that all variables in projection-free CQs are free).

For a formula φ and a variable substitution θ, by $\varphi[\theta]$ we denote the formula φ after substituting each variable x of φ by $\theta(x)$. Let $\varphi(\boldsymbol{x})$ be a projection-free CQ with variables $\boldsymbol{x}$, let $\mathcal{A}$ be a database and let $\boldsymbol{e}$ be a tuple of elements in $\mathrm{dom}(\mathcal{A})$. We say that $\mathcal{A} \models \varphi(\boldsymbol{e})$ or $\boldsymbol{e}$ *satisfies* φ *in* $\mathcal{A}$ if there is a mapping θ that substitutes the elements $\boldsymbol{e}$ for the variables $\boldsymbol{x}$, such that for each atom $R(\boldsymbol{a})$ of $\varphi[\theta]$ the tuple $\boldsymbol{a}$ is in the relation $R^{\mathcal{A}}$. The basic projection-free CQs cannot contain inequalities. However, we also consider projection-free CQs with inequalities of the form $x \neq y$ and $x \neq c$, where x and y are variables and c is

[2] We allow the full class of regular expressions (with concatenation, union and Kleene star), in particular we do not restrict regular expressions to be deterministic. We use regular expressions which are not deterministic in the proof of the lower bound in Section 8.

a constant symbol. Let $\varphi(\boldsymbol{x})$ be a projection-free CQ with inequalities, let $\mathcal{A}$ be a database and let $\boldsymbol{e}$ be a tuple of elements in $\mathrm{dom}(\mathcal{A})$. We say that $\mathcal{A} \models \varphi(\boldsymbol{e})$ if there is a mapping θ that substitutes the elements $\boldsymbol{e}$ for the variables $\boldsymbol{x}$, such that for each atom $R(\boldsymbol{a})$ of $\varphi[\theta]$ the tuple $\boldsymbol{a}$ is in $R^{\mathcal{A}}$, for each inequality of the form $x \neq y$ in φ it holds $\theta(x) \neq \theta(y)$ and for each inequality of the form $x \neq c$ in φ it holds $\theta(x) \neq c^{\mathcal{A}}$.

We say that $\mathcal{A}$ is *a substructure* of $\mathcal{B}$ if $\mathrm{dom}(\mathcal{A}) \subseteq \mathrm{dom}(\mathcal{B})$, for each relation symbol R, $R^{\mathcal{A}}$ is the subset of $R^{\mathcal{B}}$ induced by $\mathrm{dom}(\mathcal{A})$, and for each constant symbol c it holds that $c^{\mathcal{A}} = c^{\mathcal{B}}$.

A conjunctive query (CQ) $\varphi(\boldsymbol{x})$ is a formula of the form: $\exists y_1 \ldots \exists y_n \psi(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{y} = y_1, \ldots, y_n$ and $\psi(\boldsymbol{x}, \boldsymbol{y})$ is a projection-free CQ.

By $\#(\varphi, \mathcal{A})$ we denote the number of tuples satisfying a query φ in a database $\mathcal{A}$. Also, by $|A|$ we denote the number of elements in a set A and by $\mathrm{size}(\mathcal{A})$ we denote the size of $\mathcal{A}$.

A canonical structure $\mathcal{C}_\varphi$ for a projection-free conjunctive query φ is the structure over the vocabulary given by φ defined in the following way. The domain of $\mathcal{C}_\varphi$ is the set of the variables and the constants of φ. Each constant symbol is interpreted as itself. Each relation $R^{\mathcal{C}_\varphi}$ is defined as the set of all tuples $\boldsymbol{a}$ of variables and constants from atomic formulas $R(\boldsymbol{a})$ of φ.

The canonical structure for a query of the form $\varphi \wedge \psi$, where φ is a projection-free CQ and ψ is a conjunction of inequalities of the form $x \neq y$ and $x \neq c$ is defined as $\mathcal{C}_\varphi$.

TreeQL and typechecking. The following definitions come from [2], but are tailored for our setting.

Definition 1. *1. A TreeQL program is an ordered, unranked tree P with labels. The following properties hold for P:*
- *The root is labeled with an element from alphabet Σ.*
- *Every non-root element node is labeled with a pair (σ, φ), where $\sigma \in \Sigma$ and φ is a projection-free CQ. The formula in a node v is denoted by* $\mathrm{formula}(v)$.
- $\mathrm{Vars}(\mathrm{formula}(v)) \subseteq \mathrm{Vars}(\mathrm{formula}(v'))$*, for all non-root nodes v and v', where v' is a descendant of v.*

2. Let $\mathcal{A}$ be a database and P a TreeQL program. A tree $P(\mathcal{A})$ generated from $\mathcal{A}$ is defined as follows:
- *The root is (root(P), $\emptyset$).*
- *The non-root nodes are the pairs (v, θ), where v is a non-root node of P and θ is a substitution for variables* $\mathrm{Vars}(\mathrm{formula}(v))$*, such that $\mathcal{A} \models \varphi[\theta]$, for every formula φ labeling v or labeling an ancestor of v in P.*
- *The edges in $P(\mathcal{A})$ are $((v, \theta), (v', \theta'))$ such that v' is a child of v in P and θ' is an extension of θ (i.e. θ' agrees with θ on* $\mathrm{Vars}(\mathrm{formula}(v))$*).*
- *Sibling nodes in $P(\mathcal{A})$ are ordered as follows: if v and v' are siblings in P and v occurs before v', then all nodes (v, θ) occur before all nodes (v', θ') in $P(\mathcal{A})$. For a given v in P, the ordering of nodes (v, θ) and (v, θ') is irrelevant in our setting, so it is not considered here (see Remark 2 below).*

– *Finally, the label of a node (v, θ) is the Σ-label of v in P.*

Remark 2. We use the following observation from [2]. If d is a DTD then d does not distinguish among trees $P(\mathcal{A})$ for distinct orderings of the nodes (v, θ) and (v, θ'), for each v in P. As we consider DTDs as XML types, we can safely ignore the ordering of such nodes.

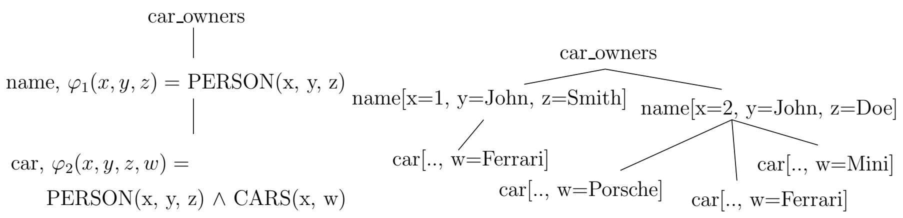
Definition 3. *A TreeQL program P* typechecks *with respect to an output type d if and only if $P(\mathcal{A}) \subseteq L(d)$, for every database $\mathcal{A}$.*

Example 4. Consider a database $\mathcal{A}$ containing information about car owners, with two relations `PERSON(Id, FirstName, LastName)` and `CARS(Id, Car)`:

PERSON	Id	FirstName	LastName
	1	`John`	`Smith`
	2	`John`	`Doe`

CARS	Id	Car
	1	`Ferrari`
	2	`Porsche`
	2	`Ferrari`
	2	`Mini`

In Figure 1 we present a program R and a tree $R(\mathcal{A})$ resulting from the transformation of the database $\mathcal{A}$ by the program R. The tree $R(\mathcal{A})$ satisfies the following DTD d: $d(\texttt{car\_owners}) = \texttt{name}^*$, $d(\texttt{name}) = \texttt{car} \cdot \texttt{car}^*$, $d(\texttt{car}) = \epsilon$. However, the program R does not typecheck w.r.t. the DTD d because for any database $\mathcal{B}$ containing information about a person without a car, the tree $R(\mathcal{B})$ does not satisfy d. In particular, it violates the requirement $d(\texttt{name}) = \texttt{car} \cdot \texttt{car}^*$.


car_owners
name, $\varphi_1(x, y, z)$ = PERSON(x, y, z)
car, $\varphi_2(x, y, z, w)$ = PERSON(x, y, z) $\wedge$ CARS(x, w)
car_owners
name[x=1, y=John, z=Smith]
name[x=2, y=John, z=Doe]
car[.., w=Ferrari]
car[.., w=Porsche]
car[.., w=Ferrari]
car[.., w=Mini]

(a) The program R.

(b) The XML tree $R(\mathcal{A})$ — the annotations of substitutions (in brackets) are not part of the tree.

Fig. 1. A TreeQL query and its result (Example 4).

3 The Upper Bound

Now we are able to formulate our main theorem.

Theorem 5 (Main Theorem). *The problem of typechecking a TreeQL program with projection-free conjunctive queries w.r.t. a DTD with arbitrary regular expressions is in coNEXPTIME.*

Sections 4-7 are devoted to proving this theorem. In Section 8 we prove coNEXPTIME-hardness of the typechecking problem; thus the upper bound from Theorem 5 is tight.

Definition 6. *A set of modulo constraints $\mathit{\Gamma}$ is a finite set of triples of the form (φ, k, j), where φ is a projection-free conjunctive query and k, j are natural numbers (represented in binary).*

We say that a database $\mathcal{A}$ satisfies a set of modulo constraints $\mathit{\Gamma}$ (we write $\mathcal{A} \models \mathit{\Gamma}$) if and only if for each $(\varphi, k, j) \in \mathit{\Gamma}$ there exists $\alpha \in \mathbb{N}$ such that:

$$\#(\varphi, \mathcal{A}) = k + (\alpha * j)$$

where the meaning of $\#(\varphi, \mathcal{A})$ is as defined in Section 2.

We say that $\mathit{\Gamma}$ is satisfiable if there is a database $\mathcal{A}$ such that $\mathcal{A} \models \mathit{\Gamma}$.

Of course, we assume that $0 \in \mathbb{N}$, so in particular $\mathit{\Gamma}$ can contain some triples of the form $(\varphi, k, 0)$. For parsimony, we will often omit the word *modulo* when referring to modulo constraints.

Now, we formulate the intermediate result, which is the main technical part of the proof of Theorem 5.

Theorem 7 (Intermediate Result). *Let $\mathit{\Gamma}$ be a set of constraints with projection-free conjunctive queries. The problem whether $\mathit{\Gamma}$ is satisfiable is in NEXPTIME.*

In Sections 4-6 we prove the intermediate result, and in Section 7 we show how it implies the main result.

4 Outline of the Proof of the Intermediate Result

We use the following notation. Let $\mathit{\Gamma}$ be a set of modulo constraints, then: $\mathit{\Gamma}_{\mathrm{CONST}}$ is the set of *constant* constraints: $\mathit{\Gamma}_{\mathrm{CONST}} = \{(\varphi, k, j) \in \mathit{\Gamma} \mid j = 0\}$, and $\mathit{\Gamma}_{\mathrm{PROP}}$ is the set of *proper* constraints: $\mathit{\Gamma}_{\mathrm{PROP}} = \{(\varphi, k, j) \in \mathit{\Gamma} \mid j > 0\}$. Of course, we have: $\mathit{\Gamma} = \mathit{\Gamma}_{\mathrm{CONST}} \cup \mathit{\Gamma}_{\mathrm{PROP}}$. The maximal number appearing in a set of modulo constraints $\mathit{\Gamma}$ is denoted by $\max_{k,j}(\mathit{\Gamma})$ and the least common multiple of all numbers j in $\mathit{\Gamma}$ is denoted by $\mathrm{lcm}_j(\mathit{\Gamma})$. By $\max_{\varphi}(\mathit{\Gamma})$ we denote the maximal size of a formula in a constraint from $\mathit{\Gamma}$ and by $\max_{C_\varphi}(\mathit{\Gamma})$ we denote the maximal size of a canonical structure for a formula in a constraint in $\mathit{\Gamma}$.

The crux to Theorem 7 lies in the following small model property, which we prove in Sections 5 and 6.

Proposition 8 (Small model property for modulo constraints). *If a set $\mathit{\Gamma}$ of modulo constraints is satisfiable then there exists a database $\mathcal{B}$ of size at most exponential in $\mathit{\Gamma}$ such that $\mathcal{B} \models \mathit{\Gamma}$.*

Note that we do not fix the vocabulary used in the constraints in $\mathit{\Gamma}$. That is, the vocabulary is implicitly given by the constants and relations used in the formulas in the constraints in $\mathit{\Gamma}$.

Observe that Theorem 7 indeed follows from Proposition 8 since now it suffices to

1. guess a database $\mathcal{B}$ of size at most exponential in Γ and
2. check that this database satisfies the constraints.

The latter boils down to evaluating all of the projection-free conjunctive queries in Γ on $\mathcal{B}$, which can certainly be done in time exponential in Γ and polynomial in $\mathcal{B}$ (resulting in time exponential in Γ) [1].

In the rest of the proof we fix a satisfiable set of modulo constraints $\Gamma = \Gamma_{\mathrm{CONST}} \cup \Gamma_{\mathrm{PROP}}$ and a database $\mathcal{A}$ satisfying Γ. To prove Proposition 8, we show that given Γ and $\mathcal{A}$ there exists a set of constraints Γ' of size exponential in Γ such that

1. $\mathcal{A} \models \Gamma'$;
2. every $\mathcal{B}$ that satisfies Γ' also satisfies Γ;
3. from Γ' we can construct a $\mathcal{B}$ of size exponential in Γ that satisfies Γ'.

Actually, Γ' is not a set of constraints as defined, but a set of *extended constraints* – constraints (φ, k, j) such that φ can also mention inequalities $x \neq c$ and $x \neq y$.

We will define Γ' as $\Gamma_{\mathrm{CONST}} \cup \Gamma'_{\mathrm{PROP}}$, where Γ'_{PROP} is going to be a new set of proper constraints in some normal form. In order to understand why we need this normal form, it is helpful to have in mind the general idea of the construction of a database that satisfies Γ'. First, we are going to fix some small structure $\mathcal{A}_{\mathrm{CONST}}$. All constraints from Γ_{CONST} will be satisfied in $\mathcal{A}_{\mathrm{CONST}}$, and $\mathcal{A}_{\mathrm{CONST}}$ will interpret all constants appearing in the formulas in Γ [3]. Then, we will satisfy the constraints in Γ'_{PROP} one by one, by extending $\mathcal{A}_{\mathrm{CONST}}$ with some number of isomorphic copies of the canonical structures for the formulas in the constraints. Namely, we take the union of the structures in which the constants from the canonical structures are identified with the respective elements of $\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$ and all elements corresponding to the variables in each copy of a canonical structure are fresh. (See Definition 10 and Example 12). However, if one does this naively, it can happen that satisfying a constraint causes other constraints which already had been satisfied to fail (see examples 12 and 30). Fortunately, if the constraints in Γ'_{PROP} are in the normal form, we can process them in an order that prevents the above problem.

The proof is organized as follows. In Section 5 we show that for a set Γ_{PROP} of proper constraints, there exists a set Γ'_{PROP} which is of at most exponential size w.r.t. the size of Γ_{PROP}, and such that $\Gamma'_{\mathrm{PROP}} \cup \Gamma_{\mathrm{CONST}}$ is satisfiable if and only if $\Gamma_{\mathrm{PROP}} \cup \Gamma_{\mathrm{CONST}}$ is satisfiable. Then, in Section 6 we show that for Γ'_{PROP} in the normal form, if $\Gamma_{\mathrm{CONST}} \cup \Gamma'_{\mathrm{PROP}}$ is satisfiable then it has a model of at most exponential size w.r.t. the size of the original input instance $\Gamma_{\mathrm{CONST}} \cup \Gamma_{\mathrm{PROP}}$.

[3] That is, for each constant symbol c that is mentioned in a formula in a constraint in Γ, there is an element $c^{\mathcal{A}_{\mathrm{CONST}}}$ in $\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$.

5 From $\mathit{\Gamma}_{\mathrm{PROP}}$ to its Normal Form

Step 1.

In the following lemma we fix an exponential substructure $\mathcal{A}_{\mathrm{CONST}}$ of $\mathcal{A}$ that satisfies $\mathit{\Gamma}_{\mathrm{CONST}}$ and such that all constants appearing in $\mathit{\Gamma}$ are interpreted in $\mathcal{A}_{\mathrm{CONST}}$.

Lemma 9. *Let $\mathit{\Gamma} = \mathit{\Gamma}_{\mathrm{CONST}} \cup \mathit{\Gamma}_{\mathrm{PROP}}$ be a set of modulo constraints and let a database $\mathcal{A}$ satisfy $\mathit{\Gamma}$. Then there exists a substructure $\mathcal{A}_{\mathrm{CONST}}$ of $\mathcal{A}$ that satisfies $\mathit{\Gamma}_{\mathrm{CONST}}$ and such that all constant symbols from the formulas in the constraints in $\mathit{\Gamma}$ are interpreted in $\mathcal{A}_{\mathrm{CONST}}$. The size of $\mathcal{A}_{\mathrm{CONST}}$ is at most exponential w.r.t. the size of $\mathit{\Gamma}$.*

Proof. Consider the database $\mathcal{A}$ and the set $\mathit{\Gamma} = \mathit{\Gamma}_{\mathrm{CONST}} \cup \mathit{\Gamma}_{\mathrm{PROP}}$. Since $\mathcal{A} \models \mathit{\Gamma}$ then, obviously, we have $\mathcal{A} \models \mathit{\Gamma}_{\mathrm{CONST}}$. Let $\mathcal{A}_{\mathrm{CONST}}$ be the substructure of $\mathcal{A}$ induced by the elements from $\mathrm{dom}(\mathcal{A})$ that are a constant in $\mathit{\Gamma}$ or are in some tuple satisfying a formula in a constraint from $\mathit{\Gamma}_{\mathrm{CONST}}$. Clearly, $\mathcal{A}_{\mathrm{CONST}} \models \mathit{\Gamma}_{\mathrm{CONST}}$.

It remains to calculate the upper bound on the size of $\mathcal{A}_{\mathrm{CONST}}$. Let s be the size of $\mathit{\Gamma}$. For each constraint $(\varphi, k, 0) \in \mathit{\Gamma}_{\mathrm{CONST}}$ there are exactly k tuples satisfying φ. Hence, in the worst case, the number of elements of $\mathrm{dom}(\mathcal{A})$ for these tuples is k times the arity of φ. Thus, the size of $\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$ is bounded from above by the sum of the number of constants in $\mathit{\Gamma}$ (which is $\mathcal{O}(s)$) and the product of:

- the number of constraints in $\mathit{\Gamma}_{\mathrm{CONST}}$: $\mathcal{O}(s)$;
- the maximal arity of a formula in $\mathit{\Gamma}_{\mathrm{CONST}}$: $\mathcal{O}(s)$;
- the value of the maximal number k from $\mathit{\Gamma}_{\mathrm{CONST}}$: $\mathcal{O}(2^s)$ – recall that the numbers in $\mathit{\Gamma}$ are in binary.

Hence, the size of $\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$ is $\mathcal{O}(s^2\, 2^s)$.

The size of each relation in $\mathcal{A}_{\mathrm{CONST}}$ is bounded from above by the number of n-tuples of elements in $\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$, where $n = \mathcal{O}(s)$ is the arity of the relation.[4] Thus it is $\mathcal{O}((s^2\, 2^s)^s))$. Finally, since the number of relations is $\mathcal{O}(s)$, we obtain the upper bound on the size of $\mathcal{A}_{\mathrm{CONST}}$: $\mathcal{O}(s\; s^{2s}\; 2^{(s^2)})$. □

The database $\mathcal{A}_{\mathrm{CONST}}$ will be the starting point in the construction of a small database $\mathcal{B}$ that satisfies $\mathit{\Gamma}$. Namely, $\mathcal{A}_{\mathrm{CONST}}$ will be a substructure of all databases which we are going to consider in the rest of the proof. For simplicity we treat all the elements of $\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$ as constants (i.e. we extend the vocabulary provided by $\mathit{\Gamma}$ with a set of constants $\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$, interpreted as themselves). A normal form of a set of constraints, which we define next, will also depend on $\mathcal{A}_{\mathrm{CONST}}$.

Before we move to Step 2, we need to introduce some notations and definitions.

[4] Recall that we assume set semantics.

Definition 10. *Let $\mathcal{A}$ be a structure over a vocabulary S and let $\mathcal{A}'$ be a structure over a vocabulary S' such that*

1. *$S = S' \cup S_C$, where S_C may contain constant symbols only;*
2. *for every constant symbol c in S', $c^{\mathcal{A}'}$ is an element of $\mathrm{dom}(\mathcal{A})$;*
3. *apart from the set $\{c_1^{\mathcal{A}'}, \ldots, c_s^{\mathcal{A}'}\}$ consisting of all interpretations of constant symbols in S', $\mathrm{dom}(\mathcal{A}')$ is disjoint with $\mathrm{dom}(\mathcal{A})$.*

We define the union $\mathcal{A} \uplus \mathcal{A}'$ as the following structure over the vocabulary S. The domain of $\mathcal{A} \uplus \mathcal{A}'$ is defined as $\mathrm{dom}(\mathcal{A}) \cup \mathrm{dom}(\mathcal{A}')$. For each constant symbol c in S we define $c^{\mathcal{A} \uplus \mathcal{A}'}$ as $c^{\mathcal{A}}$. For each relational symbol R in S we define $R^{\mathcal{A} \uplus \mathcal{A}'}$ as $R^{\mathcal{A}} \cup R^{\mathcal{A}'}$.

If we say that we extend a database $\mathcal{A}$ with a copy of $\mathcal{A}'$ *then we mean that we construct a new database $\mathcal{A} \uplus \mathcal{A}''$, where $\mathcal{A}''$ is an isomorphic copy of $\mathcal{A}'$.*

Recall that by $\mathcal{C}_\psi$ we mean the canonical structure for ψ.

Definition 11. *We say that a total order $<$ on a set of proper constraints Φ is* correct w.r.t. $\mathcal{A}_{\mathrm{CONST}}$ *if for each $(\varphi, k, j), (\varphi', k', j') \in \Phi$ such that $(\varphi, k, j) < (\varphi', k', j')$ and for each tuple $\boldsymbol{e}$ that contains at least one element*[5] *from* $\mathrm{dom}(\mathcal{C}_{\varphi'}) \setminus \mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$*, it does not hold $\mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi'} \models \varphi(\boldsymbol{e})$.*

Example 12. Consider the set of constraints $\Gamma = \{\ t_0 = (R(c,c) \wedge P(c), 1, 0), t_1 = (\varphi_1, 2, 2), t_2 = (\varphi_2, 1, 2)\ \}$, where $\varphi_1 = R(c,x) \wedge P(x)$ and $\varphi_2 = R(c,x)$. Obviously, $\Gamma_{\mathrm{CONST}} = \{t_0\}$ and $\Gamma_{\mathrm{PROP}} = \{t_1, t_2\}$. Notice that the database $\mathcal{A}_{\mathrm{CONST}} = \{R(c,c), P(c)\}$ satisfies Γ_{CONST}. First, we would like to satisfy the constraint t_1. At the moment, there is a single tuple (c) in $\mathcal{A}_{\mathrm{CONST}}$ that satisfies φ_1. Thus, we extend $\mathcal{A}_{\mathrm{CONST}}$ with a copy of $\mathcal{C}_{\varphi_1}$. This way we obtain the structure $\mathcal{A}_1 = \mathcal{A}_{\mathrm{CONST}} \cup \{R(c, e_1), P(e_1)\}$. Clearly, $\mathcal{A}_1 \models t_1$ since there are exactly 2 tuples satisfying φ_1: (c) and (e_1). Notice that these 2 tuples satisfy φ_2 as well. Hence, t_2 is not satisfied in $\mathcal{A}_1$. Next, we extend $\mathcal{A}_1$ with a copy of $\mathcal{C}_{\varphi_2}$, obtaining the database $\mathcal{A}_2 = \mathcal{A}_1 \cup \{R(c, e_2)\}$. Now, there are 3 tuples satisfying φ_2: $(c), (e_1), (e_2)$. Clearly, $\mathcal{A}_2$ satisfies Γ. Finally notice that the ordering $t_1 < t_2$ that we used is correct w.r.t. $\mathcal{A}_{\mathrm{CONST}}$. This is since for all tuples $\boldsymbol{e}$ that contain at least one element not from $\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$ we have $\mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi_2} \not\models \varphi_1(\boldsymbol{e})$.

Now, notice that if we chose to satisfy t_2 first and only then to satisfy t_1, we would not succeed —this would be since each time we extend a database with a copy of $\mathcal{C}_{\varphi_1}$, we also cause the increase of the number of tuples satisfying φ_2. This is since the ordering $t_2 < t_1$ is not correct w.r.t. $\mathcal{A}_{\mathrm{CONST}}$.

Reduced sets of constraints.

Throughout the construction of Γ'_{PROP} we will use the following observation.

Definition 13. *We call a set of proper (extended) constraints* reduced *if the set does not contain two distinct constraints (φ, k, j) and (φ', k', j') with φ and φ' equivalent.*

[5] We would like to ignore tuples $\boldsymbol{e}$ such that $\mathcal{A}_{\mathrm{CONST}} \models \varphi(\boldsymbol{e})$.

Lemma 14. *For every satisfiable set of proper (extended) constraints Φ there exists an equivalent reduced set Δ of proper constraints (in the sense that for every database $\mathcal{A}'$ it holds $\mathcal{A} \models \Phi$ if and only if $\mathcal{A}' \models \Delta$). Moreover:*

1. $|\Delta| \leq |\Phi|$;
2. $\max_{\varphi}(\Delta) \leq \max_{\varphi}(\Phi)$;
3. $\max_{k,j}(\Delta) \leq \max_{k,j}(\Phi) + \mathrm{lcm}_j(\Phi)$; *and*
4. $\mathrm{lcm}_j(\Delta) = \mathrm{lcm}_j(\Phi)$.

Proof. If Φ is already reduced then we put $\Delta := \Phi$. Otherwise we define Δ in the following way.

We call a subset Φ_{EQ} of Φ *reducible* if all formulas in the constraints in Φ_{EQ} are equivalent and $|\Phi_{\mathrm{EQ}}| \geq 2$.

Consider some reducible set Φ_{EQ}. We will construct a single constraint $t(\Phi_{\mathrm{EQ}})$ such that for every database $\mathcal{A}'$ it holds $\mathcal{A}' \models \Phi_{\mathrm{EQ}}$ if and only if $\mathcal{A}' \models \{t(\Phi_{\mathrm{EQ}})\}$.

Let $\Phi_{\mathrm{EQ}} = \{(\varphi_i, k_i, j_i) \mid i = 1, \ldots, n\}$. Since the formulas φ_i are equivalent we can use $\varphi := \varphi_1$ as the representant of the equivalence class.

Notice that the constraints in Φ_{EQ} form the following set of congruences, where x is the number of tuples satisfying φ in a satisfying database:

$$\begin{cases} x \equiv k_1 \pmod{j_1} \\ \ldots \\ x \equiv k_n \pmod{j_n} \end{cases}$$

with the additional requirement that $x \geq \max(k_1, \ldots, k_n)$ – this requirement is necessary because in Definition 6 we do not assume that $k_i \leq j_i$.

Since Φ is satisfiable such a number x exists. Further, using an easy generalization of Chinese Remainder Theorem it is possible to show that x is unique modulo the least common multiple of the numbers $j_1, \ldots, j_n$ (recall that $j_i > 0$ in proper constraints). Hence, we define the constraint $t(\Phi_{\mathrm{EQ}})$ as (φ, k, j) where

– j is $\mathrm{lcm}(j_1, \ldots, j_n)$;
– k is the smallest number which is at least $\max(k_1, \ldots, k_n)$, such that $k \equiv x \pmod{j}$.

Clearly, $k \leq \max(k_1, \ldots, k_n) + j \leq \max_{k,j}(\Phi) + \mathrm{lcm}_j(\Phi)$. Moreover for every $\mathcal{A}'$ it holds $\mathcal{A}' \models \Phi_{\mathrm{EQ}}$ if and only if $\mathcal{A}' \models \{t(\Phi_{\mathrm{EQ}})\}$. Hence, for every $\mathcal{A}'$ it holds $\mathcal{A}' \models \Phi$ if and only if $\mathcal{A}' \models \Phi \setminus \Phi_{\mathrm{EQ}} \cup \{t(\Phi_{\mathrm{EQ}})\}$.

Now, consider the class $\{\Phi^1_{\mathrm{EQ}}, \Phi^2_{\mathrm{EQ}}, \ldots, \Phi^h_{\mathrm{EQ}}\}$ of all maximal, reducible subsets of Φ. Then we define Δ as

$$\left(\Phi \setminus \bigcup_{i=1,2,\ldots,h} \Phi^i_{\mathrm{EQ}}\right) \cup \bigcup_{i=1,2,\ldots,h} \{t(\Phi^i_{\mathrm{EQ}})\}.$$

It is easy to see that conditions (1)-(4) from Lemma 14 hold. □

Step 2.

We start with the following definition.

Definition 15. *We say that a set of constraints Φ is in* the 1st normal form w.r.t. $\mathcal{A}_{\text{CONST}}$ *if all formulas in the proper constraints of Φ are of the form $\varphi(\boldsymbol{x}) \wedge$ NotConstants$(\boldsymbol{x})$, where $\varphi(\boldsymbol{x})$ is a projection-free conjunctive query with free variables $\boldsymbol{x}$ and the formula NotConstants$(\boldsymbol{x})$ is the conjunction of inequalities of the form $x \neq c$ for each variable x of φ and each $c \in \operatorname{dom}(\mathcal{A}_{\text{CONST}})$.*

In Step 2 we transform the set of constraints Γ_{PROP} into a set Γ_1 of constraints, which is small enough, is in the 1st normal form w.r.t. $\mathcal{A}_{\text{CONST}}$, and is satisfiable if and only if Γ_{PROP} is. In order to do so we produce separate constraints for each (possibly partial) substitution of variables of φ with elements of $\operatorname{dom}(\mathcal{A}_{\text{CONST}})$. This way we fix which variables are substituted with elements from $\operatorname{dom}(\mathcal{A}_{\text{CONST}})$. Additionally, a formula NotConstants ensures that each such substitution is final i.e. it forbids substituting constants for the remaining variables in the resulting constraints. Recall the basic intuition of the construction – we want to satisfy each proper constraint by extending $\mathcal{A}_{\text{CONST}}$ with copies of the canonical structure for the formula in the constraint. After Step 2 it will be clear how to do it – the constants (including the variables substituted in this step with elements from $\operatorname{dom}(\mathcal{A}_{\text{CONST}})$) will be identified with respective elements of $\operatorname{dom}(\mathcal{A}_{\text{CONST}})$ and for all variables we will substitute fresh elements.

Lemma 16. *Let $\Gamma = \Gamma_{\text{CONST}} \cup \Gamma_{\text{PROP}}$ be a set of constraints, let $\mathcal{A}$ satisfy Γ and let $\mathcal{A}_{\text{CONST}}$, a substructure of $\mathcal{A}$, satisfy Γ_{CONST}. There exists a reduced set of extended constraints Γ_1 in the 1st normal form w.r.t. $\mathcal{A}_{\text{CONST}}$ such that*

1. *$\mathcal{A} \models \Gamma_1$;*
2. *for every $\mathcal{A}'$ which is a superstructure of $\mathcal{A}_{\text{CONST}}$ such that $\mathcal{A}' \models \Gamma_{\text{CONST}}$ and $\mathcal{A}' \models \Gamma_1$ we also have $\mathcal{A}' \models \Gamma$;*
3. *(a) $|\Gamma_1| \leq |\Gamma| * |\operatorname{dom}(\mathcal{A}_{\text{CONST}})|^{2\max_{C_\varphi}(\Gamma)}$;*
   (b) $\max_\varphi(\Gamma_1) \leq \max_\varphi(\Gamma) + \mathcal{O}(\max_{C_\varphi}(\Gamma) * |\operatorname{dom}(\mathcal{A}_{\text{CONST}})|)$;
   (c) $\max_{C_\varphi}(\Gamma_1) \leq \max_{C_\varphi}(\Gamma)$;
   (d) $\max_{k,j}(\Gamma_1) \leq 2\max_{k,j}(\Gamma) + \operatorname{lcm}_j(\Gamma)$;
   (e) $\operatorname{lcm}_j(\Gamma_1) = \operatorname{lcm}_j(\Gamma)$.

Notice that Lemma 16 shares its structure with Lemma 26 and Lemma 32: claims 1 and 2 assert that the new set of constraints is equisatisfiable with the old one, at least when we restrict ourselves to superstructures of $\mathcal{A}_{\text{CONST}}$. Claim 3 asserts that the new set of constraints is small: not only it is of exponential size but also its coefficients are small enough to make the next step of the construction possible.

Let us also mention that the proof of Lemma 16 (and also of lemmas 26 and 32) is *existential* not *constructive* – we need to know $\mathcal{A}$ to be able to perform it.

Proof. Consider a constraint $t = (\varphi, k, j)$ in Γ_{PROP}. We define Γ_t as a set of new constraints of the form $(\psi_\theta, k_\theta, j)$, for each $V \subseteq \operatorname{Vars}(\varphi)$ and for each $\theta \colon V \to \operatorname{dom}(\mathcal{A}_{\text{CONST}})$, where:

$$\psi_\theta = \varphi[\theta] \wedge \operatorname{NotConstants}(\operatorname{Vars}(\varphi[\theta])).$$

The inequalities are introduced to ensure that for any database $\mathcal{A}$ such that $\mathcal{A} \models \Gamma_t$, in any tuple satisfying $\varphi[\theta]$ no variable from $\mathrm{Vars}(\varphi[\theta])$ is substituted with an element from $\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$.

Then we define for each θ the number k_θ as

$$\max\{x \in \mathbb{N} \mid x \leq k+j \ \wedge \ x \leq \#(\psi_\theta, \mathcal{A}) \ \wedge \ x \equiv \#(\psi_\theta, \mathcal{A}) \pmod{j}\}.$$

Notice that we cannot simply define k_θ as $\#(\psi_\theta, \mathcal{A}) \bmod j$ since in the proof of claim 2 of Lemma 16 we need that the total sum $\sum_\theta k_\theta$ is at least k.

Let Γ_1 be the reduced set of constraints equivalent to $\bigcup_{t \in \Gamma_{\mathrm{PROP}}} \Gamma_t$. Clearly, we can use Lemma 14 to prove that Γ_1 exists.

Example 17. Let $t = (\varphi, 1, 2)$ with $\varphi = R(c,x) \wedge R(x,y)$, let $\mathcal{A}_{\mathrm{CONST}} = \{R(c,d), R(c,c)\}$ and let $\mathcal{A} = \mathcal{A}_{\mathrm{CONST}} \cup \{R(d,e)\}$. Notice that $\#(\varphi, \mathcal{A}) = 3$ (the satisfying tuples are $(d,e), (c,d), (c,c)$). In the following table we show for each substitution θ the formula ψ_θ and the number k_θ.

θ	ψ_θ	k_θ
$\emptyset$	$R(c,x) \wedge R(x,y) \wedge x \neq c \wedge x \neq d \wedge y \neq c \wedge y \neq d$	0
$\{x \to c\}$	$R(c,c) \wedge R(c,y) \wedge y \neq c \wedge y \neq d$	0
$\{x \to d\}$	$R(c,d) \wedge R(d,y) \wedge y \neq c \wedge y \neq d$	1
$\{y \to c\}$	$R(c,x) \wedge R(x,c) \wedge x \neq c \wedge x \neq d$	0
$\{y \to d\}$	$R(c,x) \wedge R(x,d) \wedge x \neq c \wedge x \neq d$	0
$\{x \to c, y \to c\}$	$R(c,c)$	1
$\{x \to c, y \to d\}$	$R(c,c) \wedge R(c,d)$	1
$\{x \to d, y \to c\}$	$R(c,d) \wedge R(d,c)$	0
$\{x \to d, y \to d\}$	$R(c,d) \wedge R(d,d)$	0

Notice that some of the formulas contain ground atoms only (i.e. atoms without variables). Therefore, we know whether they are true or false in any superstructure of $\mathcal{A}_{\mathrm{CONST}}$. Notice also that for true sentences we have $k_\theta = 1$ and for false sentences we have $k_\theta = 0$. In general, it is also possible that some formulas in Γ_1 are unsatisfiable because of the ground atoms. Then k_θ would be 0 for them.

Now, we will prove the following observations.

Observation 18. Consider a database $\mathcal{B}$ which is a superstructure of $\mathcal{A}_{\mathrm{CONST}}$ and a constraint $(\varphi, k, j) \in \Gamma_{\mathrm{PROP}}$. Then $\sum_\theta \#(\psi_\theta, \mathcal{B}) = \#(\varphi, \mathcal{B})$.

Proof. Let $V = \mathrm{Vars}(\varphi)$. Let Φ be the set of all tuples satisfying φ in $\mathcal{B}$ and let Ψ be the set of all pairs of the form $(\theta, \boldsymbol{u})$. where θ is a substitution from a set $X \subseteq V$ to $\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$, and $\boldsymbol{u}$ is a tuple of elements of $\mathrm{dom}(\mathcal{B}) \setminus \mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$ that satisfies ψ_θ in $\mathcal{B}$. We define a function $f : \Psi \to \Phi$. Consider a pair $(\theta, \boldsymbol{u}) \in \Psi$. Let $h : V \setminus X \to \boldsymbol{u}$ be the substitution witnessing that $\mathcal{B} \models \psi_\theta(\boldsymbol{u})$. Then the union $\theta \cup h : V \to \mathrm{dom}(\mathcal{B})$ is a substitution witnessing that for some tuple $\boldsymbol{e}$ we have $\mathcal{B} \models \varphi(\boldsymbol{e})$. We define f on $(\theta, \boldsymbol{u})$ as the tuple $\boldsymbol{e}$. Now, it is easy to prove that f is a bijection between Ψ and Φ. $\square$

Observation 19. Consider a constraint $(\varphi, k, j) \in \Gamma_{\mathrm{PROP}}$. There exists a number $\alpha \in \mathbb{N}$ such that $\sum_\theta k_\theta = k + \alpha * j$.

Proof. Since $\mathcal{A} \models (\varphi, k, j)$ we have $\#(\varphi, \mathcal{A}) = k + \alpha' * j$, for some $\alpha' \in \mathbb{N}$. Hence from Observation 18 we have $\sum_\theta \#(\psi_\theta, \mathcal{A}) = k + \alpha' * j$. Notice that the definition of k_θ guarantees that (1) $k_\theta \equiv \#(\psi_\theta, \mathcal{A}) \pmod{j}$ and (2) if $k_\theta < k$ then $k_\theta = \#(\psi_\theta, \mathcal{A})$ (this is since the numbers between k and $k + j$ contain all possible remainders modulo j). Since (1) $\sum_\theta k_\theta \equiv \sum_\theta \#(\psi_\theta, \mathcal{A}) \pmod{j}$ and thus $\sum_\theta k_\theta \equiv k + \alpha' * j \pmod{j}$. Moreover since (2) $\sum_\theta k_\theta \geq k$. Finally, we conclude that there exists some $\alpha \in \mathbb{N}$ such that $\sum_\theta k_\theta = k + \alpha * j$. □

Now we are ready to prove the claims 1—3 of Lemma 16.

1. Consider a constraint $(\psi_\theta, k_\theta, j) \in \bigcup_{t \in \Gamma_{\mathrm{PROP}}} \Gamma_t$. Notice that $\#(\psi_\theta, \mathcal{A}) = k_\theta + \alpha * j$, for some $\alpha \in \mathbb{N}$. This is since from the definition of k_θ we have that $k_\theta \leq \#(\psi_\theta, \mathcal{A})$ and that $k_\theta \equiv \#(\psi_\theta, \mathcal{A}) \pmod{j}$. Hence $\mathcal{A} \models \{(\psi_\theta, k_\theta, j)\}$. Thus we have $\mathcal{A} \models \bigcup_{t \in \Gamma_{\mathrm{PROP}}} \Gamma_t$ and then $\mathcal{A} \models \Gamma_1$ because $\bigcup_{t \in \Gamma_{\mathrm{PROP}}} \Gamma_t$ and Γ_1 are equivalent.
2. It suffices to show that $\mathcal{A}' \models \Gamma_{\mathrm{PROP}}$. Let $t = (\varphi, k, j)$ be some constraint in Γ_{PROP}. Since $\mathcal{A}' \models \Gamma_1$ and Γ_1 and $\bigcup_{t \in \Gamma_{\mathrm{PROP}}} \Gamma_t$ are equivalent, we have that $\mathcal{A}' \models \Gamma_t$. Thus $\forall \theta\ \mathcal{A}' \models (\psi_\theta, k_\theta, j)$. Hence, $\forall \theta\ \#(\psi_\theta, \mathcal{A}') = k_\theta + \alpha_\theta * j$, for some $\alpha_\theta \in \mathbb{N}$. From Observation 18, $\#(\varphi, \mathcal{A}') = \sum_\theta \#(\psi_\theta, \mathcal{A}')$. Hence $\#(\varphi, \mathcal{A}') = \sum_\theta (k_\theta + \alpha_\theta * j)$ and thus $\#(\varphi, \mathcal{A}') = \sum_\theta k_\theta + \alpha' * j$, for some $\alpha' \in \mathbb{N}$. Now, using Observation 19, we obtain that $\#(\varphi, \mathcal{A}') = k + \alpha * j + \alpha' * j$, for some $\alpha, \alpha' \in \mathbb{N}$ and thus $\mathcal{A}' \models (\varphi, k, j)$.
3. (a) This is since for each constraint $(\varphi, k, j) \in \Gamma_{\mathrm{PROP}}$ the number of partial substitutions $\theta : \mathrm{Vars}(\varphi) \to \mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$ is bounded from above by $2^{|\mathrm{Vars}(\varphi)|} * |\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})|^{|\mathrm{Vars}(\varphi)|}$ and thus by $|\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})|^{2|\mathrm{Vars}(\varphi)|}$ and $|\mathrm{Vars}(\varphi)| \leq \max_{C_\varphi}(\Gamma)$.
   (b) Each formula in a proper constraint in Γ_1 has at most $|\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})| * \max_{C_\varphi}(\Gamma)$ inequalities in the NotConstants subformula.
   (c) The NotConstants subformula is ignored in the construction of a canonical structure.
   (d) Let $t = (\varphi, k, j)$ be a constraint in Γ_{PROP} and let $(\psi_\theta, k_\theta, j)$ be a constraint in Γ_t. From the definition of k_θ we have that $k_\theta \leq k + j$ and thus $\max_{k,j}(\bigcup_{t \in \Gamma_{\mathrm{PROP}}} \Gamma_t) \leq 2\max_{k,j}(\Gamma)$. From Lemma 14 we have that $\max_{k,j}(\Gamma_1) \leq 2\max_{k,j}(\Gamma) + \mathrm{lcm}_j(\Gamma)$.
   (e) This follows directly from the definition of Γ_1 and from Lemma 14.

This concludes the proof of Lemma 16. □

Step 3.

In this step we define the 2nd normal form w.r.t. $\mathcal{A}_{\mathrm{CONST}}$ of a set of constraints. Then we show the existence of a reduced set of constraints Γ_2 which is equivalent to Γ_1, is in the 2nd normal form w.r.t. $\mathcal{A}_{\mathrm{CONST}}$ and whose size is affordable.

We will prove later that each formula in a constraint in the 2nd normal form has the following property.

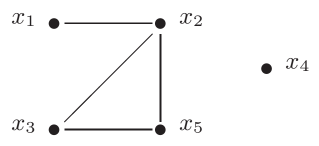


Fig. 2. The graph for the canonical structure from Example 24.

Definition 20. *We say that a formula* φ *is* controllable *if for all databases* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *and for all tuples* $\boldsymbol{e}$ *such that* $\mathcal{A}_1 \uplus \mathcal{A}_2 \models \varphi(\boldsymbol{e})$*,* $\boldsymbol{e}$ *is contained either in* $\mathrm{dom}(A_1)$ *or in* $\mathrm{dom}(\mathcal{A}_2)$*.*

Recall that the general idea of the construction is to satisfy in each step some constraint of the form (ψ, k, j), by extending some candidate database $\mathcal{A}$ with isomorphic copies of $\mathcal{C}_\psi$. Notice that if a formula φ is controllable then $\#(\varphi, \mathcal{A} \uplus \mathcal{C}_\psi) = \#(\varphi, \mathcal{A}) + \#(\varphi, \mathcal{C}_\psi)$. Thus the number of new tuples satisfying φ in $\mathcal{A} \uplus \mathcal{C}_\psi$ does not depend on $\mathcal{A}$.

The motivation why we need Step 3 can be best explained using the following example.

Example 21. Let $\varphi = R_1(c, x_1) \wedge R_2(c, x_2) \wedge \mathrm{NotConstants}(\{x_1, x_2\})$. Then φ turns out not to be controllable. Consider, for example, the structure $\mathcal{C} = \{R_1(c, e)\}$. Then $\mathcal{C} \uplus \mathcal{C}_\varphi = \{R_1(c, e), R_1(c, x_1), R_2(c, x_2)\}$ and there is a tuple (e, x_2) such that $\mathcal{C} \uplus \mathcal{C}_\varphi \models \varphi(e, x_2)$. Clearly, $e \in \mathrm{dom}(\mathcal{C})$ and $x_2 \notin \mathrm{dom}(\mathcal{C})$. Intuitively, the source of the problem is that x_1 and x_2 are in some sense *disconnected* in φ (even though they are connected by the constant c). We formalize this intuition below.

Definition 22. *Let* $\mathcal{B}$ *be a database such that all constants of* $\mathcal{B}$ *are interpreted in* $\mathcal{A}_{\mathrm{CONST}}$*. Define* $\mathrm{GRAPH}(\mathcal{B})$ *to be a graph whose vertices are the elements of* $\mathrm{dom}(\mathcal{B}) \setminus \mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$*. There is an edge between vertices* e_1*,* e_2 *of* $\mathrm{GRAPH}(\mathcal{B})$ *if there is a tuple* $\boldsymbol{e}$ *in some relation* R *in* $\mathcal{B}$ *such that* $\boldsymbol{e}$ *contains both* e_1 *and* e_2*.*

Clearly, the set of vertices of $\mathrm{GRAPH}(\mathcal{C}_\varphi)$, the graph for the canonical structure for φ, is exactly the set of variables of φ.

Definition 23. A connected subformula *of* φ *is a formula* $\varphi_D(\boldsymbol{x_D})$ *defined as* $\bigwedge_{R(\boldsymbol{x_R}) \in D} R(\boldsymbol{x_R}) \wedge \mathrm{NotConstants}(\boldsymbol{x_D})$*, where* D *is a maximal set of non-ground atoms (i.e. atoms with variables), such that* $\mathrm{GRAPH}(\mathcal{C}_{\varphi_D})$ *is a connected component of* $\mathrm{GRAPH}(\mathcal{C}_\varphi)$*.*

Notice that a formula φ is a conjunction of its connected subformulas and its ground atoms. (i.e. atoms without variables).

Example 24. Consider the following formula $\varphi(x_1, \ldots, x_5)$

$$R_1(x_1, x_2) \wedge R_2(x_2, x_3, c_1, x_5, c_2) \wedge R_1(c_1, c_3) \wedge R_1(c_3, c_3)$$

$$\wedge R_1(x_4, c_1) \wedge R_1(c_2, x_4) \wedge \mathrm{NotConstants}(\{x_1, \ldots, x_5\}),$$

where c_1, c_2, c_3 are in $\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$. Vertices of $\mathrm{GRAPH}(\mathcal{C}_\varphi)$ are $\{x_1, \ldots, x_5\}$ (see Figure 2), and edges of $\mathrm{GRAPH}(\mathcal{C}_\varphi)$ are $\{x_1, x_2\}$, $\{x_2, x_3\}$, $\{x_2, x_5\}$ and $\{x_3, x_5\}$. Clearly, $\mathrm{GRAPH}(\mathcal{C}_\varphi)$ has two connected components, namely $\{x_4\}$ and $\{x_1, x_2, x_3, x_5\}$.

There are two connected subformulas of φ:

$$R_1(x_1, x_2) \wedge R_2(x_2, x_3, c_1, x_5, c_2) \wedge \mathrm{NotConstants}(\{x_1, x_2, x_3, x_5\})$$

and

$$R_1(x_4, c_1) \wedge R_1(c_2, x_4) \wedge \mathrm{NotConstants}(\{x_4\}).$$

The ground atoms of φ are $R_1(c_1, c_3)$ and $R_1(c_3, c_3)$.

Definition 25. *We say that a set of constraints Φ is in* the 2nd normal form w.r.t. $\mathcal{A}_{\mathrm{CONST}}$ *if Φ is in the 1st normal form w.r.t. $\mathcal{A}_{\mathrm{CONST}}$ and all formulas in the constraints in Φ_{PROP}*

- *consist of exactly one connected subformula; and*
- *contain no ground atoms.*

Lemma 26. *Let Γ_1 be a set of constraints in the 1st normal form w.r.t. $\mathcal{A}_{\mathrm{CONST}}$ and let $\mathcal{A}$ satisfy Γ_1. There exists a reduced set of extended constraints Γ_2 in the 2nd normal form w.r.t. $\mathcal{A}_{\mathrm{CONST}}$ such that*

1. *$\mathcal{A} \models \Gamma_2$;*
2. *for every $\mathcal{A}'$ which is a superstructure of $\mathcal{A}_{\mathrm{CONST}}$ such that $\mathcal{A}' \models \Gamma_{\mathrm{CONST}}$ and $\mathcal{A}' \models \Gamma_2$ we also have $\mathcal{A}' \models \Gamma_1$;*
3. *(a) $|\Gamma_2| \leq |\Gamma_1| * \max_{C_\varphi}(\Gamma_1)$;*
   (b) $\max_\varphi(\Gamma_2) \leq \max_\varphi(\Gamma_1)$;
   (c) $\max_{C_\varphi}(\Gamma_2) \leq \max_{C_\varphi}(\Gamma_1)$;
   (d) $\max_{k,j}(\Gamma_2) \leq 2\max_{k,j}(\Gamma_1) + \mathrm{lcm}_j(\Gamma_1)$;
   (e) $\mathrm{lcm}_j(\Gamma_2) \leq \mathrm{lcm}_j(\Gamma_1)$.

Proof. We construct separate constraints for the connected subformulas of the formulas in the constraints in Γ_1. We also forget about the ground atoms in the formulas in the constraints. The reason is that the ground atoms are already determined to be either true or false in $\mathcal{A}_{\mathrm{CONST}}$ and we will not add any new tuples consisting exclusively of constants during the construction of $\mathcal{B} \supseteq \mathcal{A}_{\mathrm{CONST}}$. For the same reason we forget about the formulas with ground atoms only.

Consider a constraint $t = (\varphi, k, j)$ in Γ_1. Let $\varphi_1, \ldots, \varphi_n$ be the connected subformulas of φ and let $\psi_1, \ldots, \psi_l$ be the ground atoms of φ. Let $m_h \in \{0, 1\}$ be 1 if ψ_h holds in $\mathcal{A}_{\mathrm{CONST}}$ and 0 otherwise (for $h = 1, \ldots, l$). Notice that the sets of variables in formulas $\varphi_1, \ldots, \varphi_n$ partition the set of variables in φ. Thus the number of tuples satisfying φ is the product of the numbers of tuples satisfying φ_i, (for $i = 1 \ldots, n$), times the product of m_h, for $h = 1, \ldots, l$.

Since Γ_1 is satisfiable it is impossible that $k > 0$ and some $m_h = 0$. For the same reason it is also impossible that φ contains ground atoms only and $k > 1$. [6] If $k = 0$ and some $m_h = 0$ then t is satisfied in every superstructure of $\mathcal{A}_{\text{CONST}}$ so we put $\Gamma_t = \emptyset$. Similarly, if φ contains true ground atoms only and $k = 1$ then $\Gamma_t = \emptyset$. If $k \geq 0$ and for all $h = 1, \ldots, l$ the number $m_h = 1$, then we define Γ_t as $\{(\varphi_i, k_i, j) \mid i \in \{1, \ldots, n\}\}$, where for each $i = 1, \ldots, n$ the number k_i is defined as

$$\max\{x \in \mathbb{N} \mid x \leq k + j \ \wedge\ x \leq \#(\varphi_i, \mathcal{A}) \ \wedge\ x \equiv \#(\varphi_i, \mathcal{A}) \pmod{j}\}.$$

Then we define Γ_2 as the reduced set equivalent to $\bigcup_{t \in \Gamma_1} \Gamma_t$.

We can prove the following observation with similar arguments as in the proof of Observation 19.

Observation 27. Consider a constraint $(\varphi, k, j) \in \Gamma_1$. There exists a number $\alpha \in \mathbb{N}$ such that $\prod_i k_i = k + \alpha * j$.

We prove the claims 1—3 of Lemma 26.

1. Consider a constraint $(\varphi_i, k_i, j) \in \bigcup_{t \in \Gamma_1} \Gamma_t$. Notice that $\#(\varphi_i, \mathcal{A}) = k_i + \alpha * j$, for some $\alpha \in \mathbb{N}$. This is since, from the definition of k_i, we have $k_i \leq \#(\varphi_i, \mathcal{A})$ and $k_i \equiv \#(\varphi_i, \mathcal{A}) \pmod{j}$. Hence $\mathcal{A} \models \{(\varphi_i, k_i, j)\}$.
   Thus we have $\mathcal{A} \models \bigcup_{t \in \Gamma_1} \Gamma_t$ and then $\mathcal{A} \models \Gamma_2$ because $\bigcup_{t \in \Gamma_1} \Gamma_t$ and Γ_2 are equivalent.
2. We have to show that $\mathcal{A}' \models \Gamma_1$. Let $t = (\varphi, k, j)$ be some constraint in Γ_1. Since $\mathcal{A}' \models \Gamma_2$ and Γ_2 and $\bigcup_{t \in \Gamma_1} \Gamma_t$ are equivalent, we have that $\mathcal{A}' \models \Gamma_t$. Thus, two cases are possible: either (1) $\Gamma_t = \emptyset$ or (2) for each constraint $(\varphi_i, k_i, j) \in \Gamma_t$, $\mathcal{A}' \models (\varphi_i, k_i, j)$. In the case (1) $\mathcal{A}' \models \{t\}$, since we put $\Gamma_t = \emptyset$ only if φ was a false sentence in $\mathcal{A}_{\text{CONST}}$ and $k = 0$ or φ was a true sentence in $\mathcal{A}_{\text{CONST}}$ and $k = 1$. In the case (2) we have, for all $i = 1, \ldots, n$, that $\#(\varphi_i, \mathcal{A}') = k_i + \alpha_i * j$, for some $\alpha_i \in \mathbb{N}$. During the construction we noted that $\#(\varphi, \mathcal{A}') = \prod_i \#(\varphi_i, \mathcal{A}')$. Hence, $\#(\varphi, \mathcal{A}') = \prod_i (k_i + \alpha_i * j)$ and thus $\#(\varphi, \mathcal{A}') = \prod_i k_i + \alpha' * j$, for some $\alpha' \in \mathbb{N}$. Now, using Observation 27, we obtain that $\#(\varphi, \mathcal{A}') = k + \alpha * j + \alpha' * j$, for some $\alpha, \alpha' \in \mathbb{N}$ and thus $\mathcal{A}' \models (\varphi, k, j)$.
3. (a) The number of connected subformulas of a formula φ is bounded from above by the size of the canonical structure for φ.
   (b) We have in the constraints in Γ_2 the subformulas of the formulas in the constraints in Γ_1 only.
   (c) Same argument as above.
   (d) Let $t = (\varphi, k, j)$ be a constraint in Γ_1 and let (φ_i, k_i, j) be a constraint in Γ_t. From the definition of k_i we have that $k_i \leq k + j$ and thus $\max_{k,j}(\bigcup_{t \in \Gamma_1} \Gamma_t) \leq 2\max_{k,j}(\Gamma_1)$. From Lemma 14 we have $\max_{k,j}(\Gamma_2) \leq 2\max_{k,j}(\Gamma_1) + \text{lcm}_j(\Gamma_1)$.
   (e) This follows directly from the definition of Γ_2 and from Lemma 14.

[6] A formula consisting exclusively of ground atoms can be satisfied by the empty tuple only.

$\square$

We prove now that all formulas in constraints from a set in the 2nd normal form are controllable. In the following example we show the intuition behind the proof.

Example 28. Let $\varphi = R(x,y) \wedge R_0(x,c,z) \wedge R(z,w) \wedge \text{NotConstants}(\{x,y,z,w\})$ and let $\mathcal{B}$ and $\mathcal{C}$ be some databases such that all constants in them are interpreted in $\text{dom}(\mathcal{A}_{\text{CONST}})$. Thus, speaking informally, $\mathcal{B}$ and $\mathcal{C}$ may be connected in $\mathcal{B} \uplus \mathcal{C}$ only through some elements of $\text{dom}(\mathcal{A}_{\text{CONST}})$. Clearly, φ consists of a single connected subformula. Let (e_x, e_y, e_z, e_w) be some tuple satisfying φ in $\mathcal{B} \uplus \mathcal{C}$. Because of the NotConstants subformula, none of the elements from the tuple is in $\text{dom}(\mathcal{A}_{\text{CONST}})$ and thus each of them is either in $\text{dom}(\mathcal{B})$ or in $\text{dom}(\mathcal{C})$. Notice also that $R^{\mathcal{B} \uplus \mathcal{C}}$ must contain (e_x, e_y) and (e_z, e_w), and $R_0^{\mathcal{B} \uplus \mathcal{C}}$ must contain (e_x, c, e_z). We show that if $e_x \in \text{dom}(\mathcal{B})$ then also each of e_y, e_z, e_w is in $\text{dom}(\mathcal{B})$. Indeed, since $e_x \in \text{dom}(\mathcal{B})$ we have that $(e_x, e_y) \in R^{\mathcal{B}}$ and $(e_x, c, e_z) \in R_0^{\mathcal{B}}$. This is since from the definition of $\uplus$ no relation of $\mathcal{B} \uplus \mathcal{C}$ can contain a tuple with an element e_1 from $\text{dom}(\mathcal{B})$ and an element e_2 from $\mathcal{C}$ such that e_1 and e_2 are not constants. Next, since $(e_x, c, e_z) \in R_0^{\mathcal{B}}$ then $(e_z, e_w) \in R^{\mathcal{B}}$. Analogously, if $e_x \in \text{dom}(\mathcal{C})$ then also each of e_y, e_z, e_w is in $\text{dom}(\mathcal{C})$.

Lemma 29. *If (φ, k, j) is in a set of constraints in the 2nd normal form then φ is controllable.*

Proof. Let $\mathcal{B}$ and $\mathcal{C}$ be structures such that for some tuple $\boldsymbol{e}$ we have $\mathcal{B} \uplus \mathcal{C} \models \varphi(\boldsymbol{e})$. Thus there is a homomorphism $h : \mathcal{C}_\varphi \to \mathcal{B} \uplus \mathcal{C}$ witnessing it. Clearly, h is an identity on the constants and maps the variables of φ to $\boldsymbol{e}$. Let us assume towards the contradiction that $\boldsymbol{e}$ contains $a \in \text{dom}(\mathcal{B})$ and $b \notin \text{dom}(\mathcal{B})$. Obviously, neither a nor b are constants since φ contains the NotConstants subformula. The 2nd normal form guarantees that $\text{GRAPH}(\mathcal{C}_\varphi)$ consists of a single connected component. Now, consider some elements x and y in $\text{dom}(\mathcal{C}_\varphi)$ such that $a = h(x)$ and $b = h(y)$. Clearly, x and y are connected in $\text{GRAPH}(\mathcal{C}_\varphi)$ by some path σ. Notice that each edge in σ corresponds to a tuple in a relation R of $\mathcal{C}_\varphi$. Each such tuple is mapped by the homomorphism h to a tuple in R in $\mathcal{B} \uplus \mathcal{C}$. Moreover, since φ contains the NotConstants subformula, h does not map the elements corresponding to the variables of φ to constants. Hence, there is a corresponding path σ' in $\text{GRAPH}(\mathcal{B} \uplus \mathcal{C})$ connecting a and b. In particular, there must be an edge in σ' connecting an element $a' \in \text{dom}(\mathcal{B})$ and an element $b' \notin \text{dom}(\mathcal{B})$ such that a', b' are not constants. Hence, there is a tuple in some relation of $\mathcal{B} \uplus \mathcal{C}$ containing a' and b'. However, it follows directly from the definition of $\uplus$ (Definition 10) that no relation of $\mathcal{B} \uplus \mathcal{C}$ can contain such a tuple. $\square$

Step 4.

Finally, in this step we introduce the normal form w.r.t. $\mathcal{A}_{\text{CONST}}$ of a set of a constraints and then we construct a set of constraints Γ_3 in this normal form. The following example explains the motivation for Step 4.

Example 30. Let $\varphi_1 = R(x_1, x_2) \wedge R(x_1, x_3)$ and $\varphi_2 = R(x_1, x_2)$. There are no constants in this example and hence φ_1 and φ_2 do not have the NotConstants subformulas. Let t_1 be a constraint containing the formula φ_1 and t_2 a constraint containing the formula φ_2. There is no ordering of t_1 and t_2 that is correct w.r.t $\mathcal{A}_{\text{CONST}}$. This is since $\mathcal{C}_{\varphi_1} \models \varphi_2(x_1, x_2)$ and $\mathcal{C}_{\varphi_1} \models \varphi_2(x_1, x_3)$ and at the same time $\mathcal{C}_{\varphi_2} \models \varphi_1(x_1, x_2, x_2)$.

Definition 31. *We say that a set of constraints Φ is in* the normal form w.r.t. $\mathcal{A}_{\text{CONST}}$ *if Φ is in the 2nd normal form w.r.t. $\mathcal{A}_{\text{CONST}}$ and all formulas in the constraints in Φ_{PROP} are of the form $\varphi(\boldsymbol{x}) \wedge NotConstants(\boldsymbol{x}) \wedge INEQ(\boldsymbol{x})$, where the formula $INEQ(\boldsymbol{x})$ is the conjunction of inequalities of the form $x \neq y$ for each pair of distinct variables x and y of $\boldsymbol{x}$.*

Lemma 32. *Let Γ_2 be a set of constraints in the 2nd normal form w.r.t. $\mathcal{A}_{\text{CONST}}$ and let $\mathcal{A}$ satisfy Γ_2. There exists a reduced set of extended constraints Γ_3 in the normal form w.r.t. $\mathcal{A}_{\text{CONST}}$ such that*

1. $\mathcal{A} \models \Gamma_3$;
2. *for every $\mathcal{A}'$ which is a superstructure of $\mathcal{A}_{\text{CONST}}$ such that $\mathcal{A}' \models \Gamma_{\text{CONST}}$ and $\mathcal{A}' \models \Gamma_3$ we also have $\mathcal{A}' \models \Gamma_2$;*
3. *(a)* $|\Gamma_3| \leq |\Gamma_2| * 2^{(\max_{C_\varphi}(\Gamma_2)^2)}$;
   (b) $\max_\varphi(\Gamma_3) \leq \max_\varphi(\Gamma_2) + \mathcal{O}(\max_{C_\varphi}(\Gamma_2)^2)$;
   (c) $\max_{C_\varphi}(\Gamma_3) \leq \max_{C_\varphi}(\Gamma_2)$;
   (d) $\max_{k,j}(\Gamma_3) \leq 2\max_{k,j}(\Gamma_2) + \text{lcm}_j(\Gamma_2)$;
   (e) $\text{lcm}_j(\Gamma_3) \leq \text{lcm}_j(\Gamma_2)$.

Proof. Consider a database $\mathcal{A}$, a formula $\varphi(\boldsymbol{x})$ from Γ_2 and a tuple $\boldsymbol{a}$ such that $\mathcal{A} \models \varphi(\boldsymbol{a})$. A substitution of elements $\boldsymbol{a}$ for variables $\boldsymbol{x}$ may map several variables from $\boldsymbol{x}$ to a single element a in $\boldsymbol{a}$. During Step 4 we replace each constraint (φ, k, j) in Γ_2 with separate constraints for all possible variants of C_φ which can be obtained by identification of some variables in Vars(φ). We also disallow any further identification of variables in the resulting constraints Γ_3. In other words: If $\mathcal{A} \models \varphi(\boldsymbol{a})$ then there is a corresponding homomorphism from $\mathcal{C}_\varphi$ to $\mathcal{A}$. The goal of this step is to obtain a new set Γ_3 which can replace Γ_2, such that all such homomorphisms have to be injective.

Consider a constraint (φ, k, j) in Γ_2. Let $\pi = \{S_1, \dots, S_m\}$ be a partition of Vars(φ). We define a substitution $\theta_\pi : \text{Vars}(\varphi) \to \{y_1, \dots, y_m\}$ as follows. For each $x \in S_i$ we define $\theta_\pi(x)$ to be y_i. Then we define ψ_{θ_π} to be $\varphi_{\theta_\pi} \wedge \text{NotConstants}(\text{Vars}(\varphi_{\theta_\pi})) \wedge \text{INEQ}(\text{Vars}(\varphi_{\theta_\pi}))$, where INEQ(Vars$(\varphi_{\theta_\pi})$) is the conjunction of inequalities of the form $x \neq y$, for each pair of distinct variables $x, y \in \text{Vars}(\varphi_{\theta_\pi})$. We introduce the inequalities to ensure that all variables which are not identified during this step have to be substituted with distinct elements of a database.

For each $t \in \Gamma_2$, where $t = (\varphi, k, j)$, we define Γ_t to be the set of constraints of the form $(\psi_{\theta_\pi}, k_{\theta_\pi}, j)$ for each partition π of Vars(φ). Similarly as in the previous steps we define the numbers k_{θ_π} as

$$\max\{x \in \mathbb{N} \mid x \leq k + j \ \wedge \ x \leq \#(\psi_{\theta_\pi}, \mathcal{A}) \ \wedge \ x \equiv \#(\psi_{\theta_\pi}, \mathcal{A}) \pmod{j}\}.$$

Finally, we define $\mathit{\Gamma}_3$ as the reduced set equivalent to $\bigcup_{t\in\mathit{\Gamma}_2}\mathit{\Gamma}_t$.

The proof of claims 1 and 2 is very similar to the proofs of the respective claims of lemmas 16 and 26. In particular, we use the facts that for each superstructure $\mathcal{B}$ of $\mathcal{A}_{\mathrm{CONST}}$ we have $\sum_\pi \#(\psi_{\theta_\pi},\mathcal{B}) = \#(\varphi,\mathcal{B})$ and $\sum_\pi k_{\theta_\pi} = k+\alpha*j$. Now, we prove claim 3.

(a) Notice that the number of partitions of a set of variables of a formula in $\mathit{\Gamma}_2$ is at most $2^{(s^2)}$, where s is the number of variables in the formula. Clearly, $s \le \max_{C_\varphi}(\mathit{\Gamma}_2)$. Therefore, in the worst case, for each t in $\mathit{\Gamma}_2$ we output $\mathit{\Gamma}_t$ of a size bounded by $2^{(\max_{C_\varphi}(\mathit{\Gamma}_2)^2)}$. Then the claim follows from Lemma 14.
(b) The number of inequalities in the INEQ subformula of a formula φ is bounded quadratically in the number of variables in φ, which is at most $\max_{C_\varphi}(\mathit{\Gamma}_2)$.
(c) Clearly, for every φ and every partition π the size of $\mathcal{C}_{\varphi[\theta_\pi]}$ is at most the size of $\mathcal{C}_\varphi$.
(d) This follows from the definition of $\mathit{\Gamma}_3$ and Lemma 14.
(e) This follows from the definition of $\mathit{\Gamma}_3$ and Lemma 14.

□

Here, we illustrate the effect of Step 4 on the formulas $\varphi_1 = R(x_1,x_2)\wedge R(x_1,x_3)$ and $\varphi_2 = R(x_1,x_2)$ from Example 30.

Example 33. We obtain the following formulas:

Partition π	Formula $\varphi_i[\theta_\pi]$
$\{x_1\},\{x_2\},\{x_3\}$	$\varphi_1^0 = R(y_1,y_2)\wedge R(y_1,y_3)\wedge y_1\neq y_2\wedge y_1\neq y_3\wedge y_2\neq y_3$
$\{x_1,x_2\},\{x_3\}$	$\varphi_1^1 = R(y_1,y_1)\wedge R(y_1,y_2)\wedge y_1\neq y_2$
$\{x_1,x_3\},\{x_2\}$	$\varphi_1^2 = R(y_1,y_2)\wedge R(y_1,y_1)\wedge y_1\neq y_2$
$\{x_1\},\{x_2,x_3\}$	$\varphi_1^3 = \varphi_2^0 = R(y_1,y_2)\wedge y_1\neq y_2$
$\{x_1,x_2,x_3\}$	$\varphi_1^4 = \varphi_2^1 = R(y_1,y_1)$

Consider the constraints $t_1 = (\varphi_1,3,7)$ and $t_2 = (\varphi_2,2,2)$. Then we have $\mathit{\Gamma}_{t_1} = \{t'_i \mid i = 0,1,\ldots,4\}$, where[7] $t'_0 = (\varphi_1^0,0,7)$, $t'_1 = (\varphi_1^1,1,7)$, $t'_2 = (\varphi_1^2,2,7)$, $t'_3 = (\varphi_1^3,3,7)$, $t'_4 = (\varphi_1^4,4,7)$. Also $\mathit{\Gamma}_{t_2} = \{t''_0,t''_1\}$, where $t''_0 = (\varphi_2^0,0,2)$ and $t''_1 = (\varphi_2^1,2,2)$.

Let $\mathit{\Gamma}'$ be the reduced set of constraints which is equivalent to $\mathit{\Gamma}_{t_1}\cup\mathit{\Gamma}_{t_2}$. Notice that φ_1^1 and φ_1^2 are equivalent and hence there is only one constraint, say t_{12}, corresponding to them in $\mathit{\Gamma}'$. Similarly, the formulas φ_1^3 and φ_2^0 and also φ_1^4 and φ_2^1 are equivalent and thus they are replaced with constraints t_{30} and t_{41} in the reduced set $\mathit{\Gamma}'$. Thus, there are the following constraints in $\mathit{\Gamma}'$: t'_0 (with the formula φ_1^0), t_{12} (with φ_1^1), t_{30} (with the formula φ_1^3) and t_{41} (with the formula φ_1^4). Notice that the ordering of $\mathit{\Gamma}'$: $t'_0 < t_{12} < t_{30} < t_{41}$ is correct w.r.t. $\mathcal{A}_{\mathrm{CONST}}$. In particular, there is no tuple (e_1,e_2,e_3) such that $\mathcal{C}_{\varphi_1^3} \models \varphi_1^0(e_1,e_2,e_3)$. This is because the INEQ subformula of φ_1^0 mentions $y_2\neq y_3$ (compare this with Example 30, where we considered the formulas φ_1^0 and φ_1^3 but without the INEQ subformulas).

[7] Note that the numbers k_θ are chosen for the example only since they depend on $\mathcal{A}$ also.

Finally, directly from lemmas 9, 16, 26 and 32 we have the following corollary that relates the set of constraints $\mathit{\Gamma}_3$ to the original input instance $\mathit{\Gamma}$.

Corollary 34. *Let $\mathit{\Gamma} = \mathit{\Gamma}_{\mathrm{CONST}} \cup \mathit{\Gamma}_{\mathrm{PROP}}$ be a set of constraints and let a database $\mathcal{A}$ satisfy $\mathit{\Gamma}$. There exists a reduced set of extended constraints $\mathit{\Gamma}_3$ in the normal form w.r.t. $\mathcal{A}_{\mathrm{CONST}}$ such that*

1. *$\mathcal{A} \models \mathit{\Gamma}_3$;*
2. *for every $\mathcal{A}'$ which is a superstructure of $\mathcal{A}_{\mathrm{CONST}}$ such that $\mathcal{A}' \models \mathit{\Gamma}_{\mathrm{CONST}}$ and $\mathcal{A}' \models \mathit{\Gamma}_3$ we also have $\mathcal{A}' \models \mathit{\Gamma}$;*
3. *The size of $\mathit{\Gamma}_3$ is at most exponential in the size of $\mathit{\Gamma}$. In particular:*
   (a) The number of constraints in $\mathit{\Gamma}_3$ is at most exponential in the size of $\mathit{\Gamma}$;
   (b) $\max_{\varphi}(\mathit{\Gamma}_3)$ is at most exponential in the size of $\mathit{\Gamma}$;
   (c) $\max_{C_\varphi}(\mathit{\Gamma}_3) \leq \max_{C_\varphi}(\mathit{\Gamma})$;
   (d) $\max_{k,j}(\mathit{\Gamma}_3) \leq 8\max_{k,j}(\mathit{\Gamma}) + 7\mathrm{lcm}_j(\mathit{\Gamma})$;
   (e) $\mathrm{lcm}_j(\mathit{\Gamma}_3) \leq \mathrm{lcm}_j(\mathit{\Gamma})$.

6 Satisfying $\mathit{\Gamma}_{\mathrm{CONST}} \cup \mathit{\Gamma}'_{\mathrm{PROP}}$

From Corollary 34 we have that for a set $\mathit{\Gamma} = \mathit{\Gamma}_{\mathrm{CONST}} \cup \mathit{\Gamma}_{\mathrm{PROP}}$ of modulo constraints and for a database $\mathcal{A}$ satisfying $\mathit{\Gamma}$ it is possible to construct an exponentially bounded substructure $\mathcal{A}_{\mathrm{CONST}}$ of $\mathcal{A}$ such that $\mathcal{A}_{\mathrm{CONST}} \models \mathit{\Gamma}_{\mathrm{CONST}}$ and a set $\mathit{\Gamma}'_{\mathrm{PROP}} = \mathit{\Gamma}_3$ of extended constraints in the normal form w.r.t. $\mathcal{A}_{\mathrm{CONST}}$ such that

1. $\mathcal{A} \models \mathit{\Gamma}_3$;
2. for every superstructure $\mathcal{B}$ of $\mathcal{A}_{\mathrm{CONST}}$, if $\mathcal{B}$ satisfies $\mathit{\Gamma}_{\mathrm{CONST}} \cup \mathit{\Gamma}_3$ then $\mathcal{B}$ also satisfies $\mathit{\Gamma}$.

The normal form of $\mathit{\Gamma}_3$ guarantees the following:

– each formula contains the NotConstants subformula, so that the variables cannot be substituted with elements from $\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$;
– each formula consists of a single connected subformula and does not contain ground atoms and hence, each formula is controllable;
– each formula contains the INEQ subformula, so that distinct variables within the same formula cannot be substituted with the same element of a database.

In order to prove Theorem 7 it is enough to prove that from $\mathit{\Gamma}_3, \mathcal{A}_{\mathrm{CONST}}$ and $\mathcal{A}$ we can construct a database $\mathcal{B}$ of size exponential in $\mathit{\Gamma}$ that satisfies $\mathit{\Gamma}_3 \cup \mathit{\Gamma}_{\mathrm{CONST}}$.

First, we define an order on $\mathit{\Gamma}_3$ that is correct w.r.t. $\mathcal{A}_{\mathrm{CONST}}$..

Definition 35. *We define a partial order $\leq_{\mathrm{part}}$ on the constraints from $\mathit{\Gamma}_3$ as follows: $(\varphi_1, k_1, j_1) \leq_{\mathrm{part}} (\varphi_2, k_2, j_2)$ if there exists a tuple $\boldsymbol{a}$ of elements of $\mathcal{C}_{\varphi_1}$ such that $\mathcal{C}_{\varphi_1} \models \varphi_2(\boldsymbol{a})$.*

We define $\leq$ to be some total order on the constraints from $\mathit{\Gamma}_3$ consistent with $\leq_{\mathrm{part}}$ and, then, we define $<$ to be the strict order implied by $\leq$.

Lemma 36. *Let $\mathit{\Gamma}_3$ be a set of constraints in the normal form w.r.t. $\mathcal{A}_{\mathrm{CONST}}$. Then the order $<$ on $\mathit{\Gamma}_3$ is well-defined and correct w.r.t. $\mathcal{A}_{\mathrm{CONST}}$.*

Proof. First, we prove that $\leq_{\mathrm{part}}$ is well-defined. The only thing which may be not completely obvious is whether $\leq_{\mathrm{part}}$ is antisymmetric.

Suppose that $(\varphi_1, k_1, j_1) \neq (\varphi_2, k_2, j_2)$, $(\varphi_1, k_1, j_1) \leq_{\mathrm{part}} (\varphi_2, k_2, j_2)$ and $(\varphi_2, k_2, j_2) \leq_{\mathrm{part}} (\varphi_1, k_1, j_1)$. Thus, there are homomorphisms $h : \mathcal{C}_{\varphi_2} \to \mathcal{C}_{\varphi_1}$ and $h' : \mathcal{C}_{\varphi_1} \to \mathcal{C}_{\varphi_2}$. Notice that since $\mathit{\Gamma}_3$ is in the normal form w.r.t. $\mathcal{A}_{\mathrm{CONST}}$, φ_1 and φ_2 contain the NotConstants and INEQ subformulas. Therefore h and h' are injective. Moreover, since $\mathcal{C}_{\varphi_1}$ and $\mathcal{C}_{\varphi_2}$ are finite, h and h' are bijections (as functions).

Now we would like to show that h is an isomorphism. It remains to show that h^{-1}, the inverse function of h, preserves structure. Suppose that, for a tuple $\boldsymbol{a}$ of elements of $\mathcal{C}_{\varphi_1}$ and a relation R we have $\mathcal{C}_{\varphi_1} \models R(\boldsymbol{a})$ and $\mathcal{C}_{\varphi_2} \not\models R(h^{-1}(\boldsymbol{a}))$. Thus h maps the tuple $(h^{-1}(\boldsymbol{a}))$ not in R to the tuple $\boldsymbol{a}$ in R. Further, since h is injective, the preimage under h of each tuple of elements of $\mathrm{dom}(\mathcal{C}_{\varphi_1})$ consists of a single tuple. Moreover, h maps tuples in R to tuples in R. Hence, the number of tuples in R in $\mathcal{C}_{\varphi_1}$ is strictly greater than the number of tuples in R in $\mathcal{C}_{\varphi_2}$. However, a similar argument involving h' shows that the number of tuples in R in $\mathcal{C}_{\varphi_1}$ is not greater than the number of tuples in R in $\mathcal{C}_{\varphi_2}$, what gives the desired contradiction. Thus, $\mathcal{C}_{\varphi_2}$ and $\mathcal{C}_{\varphi_1}$ are isomorphic. Moreover, since φ_1 and φ_2 contain the NotConstants and INEQ subformulas we have that φ_1 and φ_2 are equivalent. However, this contradicts the claim that $\mathit{\Gamma}_3$ is reduced. Therefore, we conclude that $\leq_{\mathrm{part}}$ is antisymmetric.

Now, we show for $\varphi_1, \varphi_2 \in \mathit{\Gamma}_3$ that $\mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi_1} \models \varphi_2(\boldsymbol{e})$ implies $\mathcal{C}_{\varphi_1} \models \varphi_2(\boldsymbol{e})$. Suppose that there is some tuple $\boldsymbol{e}$ such that $\mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi_1} \models \varphi_2(\boldsymbol{e})$ Thus, there is a homomorphism $h : \mathcal{C}_{\varphi_2} \to \mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi_1}$ that respects the NotConstants and INEQ clauses of φ_2. Hence each tuple of each relation R in $\mathcal{C}_{\varphi_2}$ is mapped by h either to a tuple in R in $\mathcal{A}_{\mathrm{CONST}}$ or to a tuple in R in $\mathcal{C}_{\varphi_1}$. However, the former cannot happen. This is since no element corresponding to a variable of φ_2 can be mapped to $\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$ (because φ_2 contains the NotConstants subformula) and since there is no tuple with constants only in R in $\mathcal{C}_{\varphi_2}$ ($\mathit{\Gamma}_3$ is in the 2nd normal form and hence φ_2 does not contain ground atoms). Thus, in fact, h is a homomorphism from $\mathcal{C}_{\varphi_2}$ to $\mathcal{C}_{\varphi_1}$.

Finally, we prove that any order $<$ that is consistent with $\leq_{\mathrm{part}}$ is also correct w.r.t. $\mathcal{A}_{\mathrm{CONST}}$. Suppose $<$ is not correct w.r.t. $\mathcal{A}_{\mathrm{CONST}}$. Then there are two constraints (φ_1, k_1, j_1) and (φ_2, k_2, j_2) in $\mathit{\Gamma}_3$ such that (1) $(\varphi_1, k_1, j_1) \leq_{\mathrm{part}} (\varphi_2, k_2, j_2)$ and (2) $\mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi_2} \models \varphi_1(\boldsymbol{e})$. Hence, since (2) $\mathcal{C}_{\varphi_2} \models \varphi_1(\boldsymbol{e})$ and thus $(\varphi_2, k_2, j_2) \leq_{\mathrm{part}} (\varphi_1, k_1, j_1)$, a contradiction with (1). □

We also need the following lemma.

Lemma 37. *Let (φ, k, j) be a constraint in the normal form w.r.t. $\mathcal{A}_{\mathrm{CONST}}$ such that $\mathcal{C}_\varphi$ has exactly n automorphisms.* [8] *Then $\#(\varphi, \mathcal{C}_\varphi) = n$. Moreover, for every database $\mathcal{B} \supseteq \mathcal{A}_{\mathrm{CONST}}$ we have $\#(\varphi, \mathcal{B}) = \alpha * n$, for some $\alpha \in \mathbb{N}$.*

[8] Recall that an automorphism of a structure $\mathcal{C}$ is an isomorphism from $\mathcal{C}$ to itself.

Proof. Clearly, if $\mathcal{C}_\varphi \models \varphi(\boldsymbol{a})$ then for each automorphism σ of $\mathcal{C}_\varphi$ we have $\mathcal{C}_\varphi \models \varphi(\sigma(\boldsymbol{a}))$. Hence, $\#(\varphi, \mathcal{C}_\varphi) \geq n$. Moreover, we also have that $\#(\varphi, \mathcal{C}_\varphi) \leq n$. This is since φ contains the INEQ and NotConstants subformulas and thus each mapping θ that substitutes variables in φ to the elements of $\mathcal{C}_\varphi$ and respects the INEQ and NotConstants subformulas must map distinct variables in φ to distinct elements that cannot be constants. Therefore, since the elements of $\mathcal{C}_\varphi$ comprise the variables and constants of φ itself, θ is an isomorphism from $\mathcal{C}_\varphi$ to $\mathcal{C}_\varphi$, and hence an automorphism.

Consider now the set Φ of tuples satisfying φ in some database $\mathcal{B} \supseteq \mathcal{A}_{\text{CONST}}$. For some $\boldsymbol{e_1}, \boldsymbol{e_2}$ in Φ, we say that $\boldsymbol{e_1} \equiv \boldsymbol{e_2}$ if $\sigma(\boldsymbol{e_1}) = \boldsymbol{e_2}$ for some automorphism σ of $\mathcal{C}_\varphi$. Clearly, $\equiv$ is an equivalence relation of Φ. As φ contains the INEQ and NotConstants subformulas, $\equiv$ partitions Φ into equivalence classes such that each class has exactly n tuples. □

Notice however, that the above lemma would not be true if conjunctive queries with projections were allowed. We illustrate this in the following example.

Example 38. Consider the following query: $\varphi(x) = \exists y_1 \exists y_2 R(x, y_1) \wedge R(x, y_2)$. Clearly, C_φ has 2 automorphisms[9] (i.e. σ_1 that is the identity on $\{x, y_1, y_2\}$ and σ_2 that swaps y_1 and y_2), but there is only one tuple (i.e. x) such that $C_\varphi \models \varphi(x)$.

Now, we are ready to present a single step of the construction of the small database $\mathcal{B}$ from $\mathcal{A}_{\text{CONST}} \subseteq \mathcal{A}$ and Γ_3.

Lemma 39. *Let (φ, k, j) be a constraint in Γ_3 and let $\mathcal{B}$ be a database such that $\mathcal{A}_{\text{CONST}} \subseteq \mathcal{B}$ and $\mathcal{B} \models \{t \in \Gamma_3 \mid t < (\varphi, k, j)\}$. There exists a database $\mathcal{B}' \supseteq \mathcal{B}$ such that $\mathcal{B}' \models \{t \in \Gamma_3 \mid t \leq (\varphi, k, j)\}$. The size of the database $\mathcal{B}'$ is at most* $\text{size}(\mathcal{B}) + \text{size}(\mathcal{C}_\varphi) * \delta$, *where $\delta \leq k + j$.*

Example 40. Consider constraints $t_1 = (\varphi_1, 2, 2)$ and $t_2 = (\varphi_2, 1, 6)$ with $\varphi_1 = R(v, z_1) \wedge R(v, z_2) \wedge R'(a, b, z_1) \wedge \text{NotConstants}(v, z_1, z_2) \wedge \text{INEQ}(v, z_1, z_2)$ and $\varphi_2 = R(x, y) \wedge \text{NotConstants}(x, y) \wedge \text{INEQ}(x, y)$. Clearly: $t_1 < t_2$. There are two tuples $\boldsymbol{x}$ such that $\mathcal{C}_{\varphi_1} \models \varphi_2(\boldsymbol{x})$, namely (v, z_1) and (v, z_2). Let $\mathcal{B}$ be a database presented schematically at Figure 3 such that $\mathcal{B} \models \{t_1\}$. We construct the database $\mathcal{B}' \models \{t_1, t_2\}$. In order to satisfy t_2 three copies of $\mathcal{C}_{\varphi_2}$ should be added.

Proof (of Lemma 39). Let n be the number of automorphisms of $\mathcal{C}_\varphi$. According to Lemma 37, the number $m = \#(\varphi, \mathcal{B})$ is a multiple of n.

Let m' be the smallest number such that m' is a multiple of n (including 0) and $m' + m = k + \alpha * j$, for some $\alpha \in \mathbb{N}$. Notice that the number m' exists since Γ_3 is satisfiable. Moreover, its value is bounded by $k + j * n$. Indeed, let m'' be the smallest multiple of n which is at least $k - m$. Therefore $m'' \leq k + n$. Then consider the following set M defined as $\{m'' + \beta * n \mid \beta = 0, 1, \ldots, j - 1\}$. Notice

[9] The canonical structure $\mathcal{C}_\varphi$ for a query φ of the form $\exists \boldsymbol{x} \psi(\boldsymbol{x}, \boldsymbol{y})$, where ψ is a projection-free CQ is defined as $\mathcal{C}_\psi$.

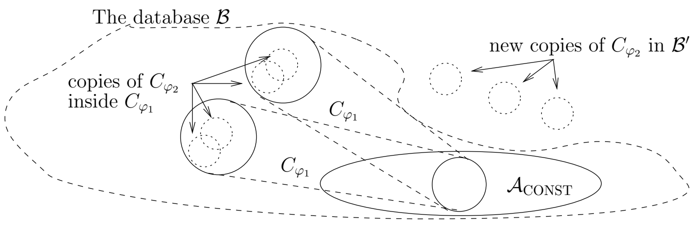

The database $\mathcal{B}$
new copies of C_{φ_2} in $\mathcal{B}'$
copies of C_{φ_2} inside C_{φ_1}
C_{φ_1}
C_{φ_1}
$\mathcal{A}_{\text{CONST}}$

Fig. 3. The database $\mathcal{B}$ satisfying the constraint t_1 from Example 40.

that, for every multiple N of n that is greater than $k + j * n$, there is a number in M that is congruent modulo j to N. Thus, m' is one of the numbers in M.

We define the database $\mathcal{B}'$ as the union of the database $\mathcal{B}$ and $\delta = \frac{m'}{n}$ copies of the canonical structure $\mathcal{C}_\varphi$, with constants from each copy of $\mathcal{C}_\varphi$ identified with respective elements of $\text{dom}(\mathcal{A}_{\text{CONST}})$. Formally, let $\mathcal{C}^1_\varphi, \ldots, \mathcal{C}^\delta_\varphi$ be the isomorphic copies of $\mathcal{C}_\varphi$. We define $\mathcal{B}'$ to be $\mathcal{B} \uplus \mathcal{C}^1_\varphi \uplus \ldots \uplus \mathcal{C}^\delta_\varphi$. Hence, the size of $\mathcal{B}'$ is at most $\text{size}(\mathcal{B}) + \text{size}(\mathcal{C}_\varphi) * \delta$.

Now, we show that $\mathcal{B}' \models \{t \in \Gamma_3 \mid t \leq (\varphi, k, j)\}$. First, we show that $\mathcal{B}' \models (\varphi, k, j)$. Let us count the number of tuples $\boldsymbol{b}$ such that $\mathcal{B}' \models \varphi(\boldsymbol{b})$. It follows from Lemma 29 that each such a tuple $\boldsymbol{b}$ is contained either in $\text{dom}(\mathcal{B})$ or in the domain of some $\mathcal{C}^i_\varphi$. Clearly, there are exactly m tuples consisting of elements of $\text{dom}(\mathcal{B})$ only, and there are exactly m' tuples (by Lemma 37), such that the elements of each of them are all contained in some $\mathcal{C}^i_\varphi$. Hence finally, there are exactly $m + m'$ tuples satisfying φ and thus $\mathcal{B}' \models (\varphi, k, j)$.

Now, we need to prove that by extending the structure we did not spoil one of the old constraints. We claim that for each constraint $(\varphi', k', j') \in \Gamma_3$, such that $(\varphi', k', j') < (\varphi, k, j)$, the number of tuples satisfying (φ', k', j') in $\mathcal{B}'$ is exactly the same as in $\mathcal{B}$. From Lemma 29 it follows that each new tuple $\boldsymbol{e}$, such that $\mathcal{B}' \models \varphi'(\boldsymbol{e})$, must be contained in some new copy of $\mathcal{C}_\varphi$. But this would mean that $\mathcal{A}_{\text{CONST}} \uplus \mathcal{C}_\varphi \models \varphi'(\boldsymbol{b})$. However, since $(\varphi', k', j') < (\varphi, k, j)$, this would mean that $<$ is not correct w.r.t. $\mathcal{A}_{\text{CONST}}$, a contradiction. $\square$

The above proof would no longer be correct if we allowed projections in the queries. This is since Lemma 37 was used there, and, as we have seen, this lemma does not generalize to the scenario with projections. But actually, the very idea of Lemma 39 – that having a satisfiable set of constraints in the normal form we can extend any database satisfying some set of *small* (in the order $\leq$) constraints, to a new database satisfying the *small* constraints and a new greater one – fails if projections are allowed. Consider the following example.

Example 41. Let $\varphi_1(x) = \exists y R(x, y)$ and $\varphi_2(x, y) = R(x, y)$. Clearly, $\mathcal{C}_{\varphi_1}$ and $\mathcal{C}_{\varphi_2}$ are isomorphic. Now, consider the constraints $t_1 = (\varphi_1(x), 1, 2)$ and $t_2 = (\varphi_2(x, y), 0, 2)$. Our algorithm builds the satisfying database using disjoint copies of canonical structures for queries. In this case, however, such database does not exist – there is no database with an even number of copies of $\mathcal{C}_{\varphi_2}$ and an odd number of copies of $\mathcal{C}_{\varphi_1}$, such that all these copies are disjoint. But there is a database satisfying both constraints t_1 and t_2: $R(a,a) \wedge R(a,b)$.

Proof (of Proposition 8). Suppose that a database $\mathcal{A}$ satisfies the set $\Gamma = \Gamma_{\mathrm{CONST}} \cup \Gamma_{\mathrm{PROP}}$. By application of Lemma 9 there exists a substructure $\mathcal{A}_{\mathrm{CONST}}$ of $\mathcal{A}$ of size at most exponential in Γ that satisfies Γ_{CONST}. By application of Corollary 34 there exists a reduced set of constraints Γ_3 in the normal form w.r.t. $\mathcal{A}_{\mathrm{CONST}}$ such that

1. $\mathcal{A} \models \Gamma_3$;
2. for every $\mathcal{A}'$ which is a superstructure of $\mathcal{A}_{\mathrm{CONST}}$ such that $\mathcal{A}' \models \Gamma_{\mathrm{CONST}}$ and $\mathcal{A}' \models \Gamma_3$ we also have $\mathcal{A}' \models \Gamma$;
3. The size of Γ_3 is at most exponential in the size of Γ. In particular:
   (a) The number of constraints in Γ_3 is at most exponential in the size of Γ;
   (b) $\max_{\varphi}(\Gamma_3)$ is at most exponential in the size of Γ;
   (c) $\max_{C_\varphi}(\Gamma_3) \leq \max_{C_\varphi}(\Gamma)$;
   (d) $\max_{k,j}(\Gamma_3) \leq 8\max_{k,j}(\Gamma) + 7\mathrm{lcm}_j(\Gamma)$;
   (e) $\mathrm{lcm}_j(\Gamma_3) \leq \mathrm{lcm}_j(\Gamma)$.

Let $\Gamma_3 = \{t_1, \ldots, t_n\}$ such that for $i, j \in \{1, \ldots, n\}$, if $i < i'$ then $t_i < t_{i'}$. We define a sequence of databases $\mathcal{A}_i$, for $i = 0, \ldots, n$, such that the database $\mathcal{A}_i$ satisfies $\{t'_i \in \Gamma_3 \mid t'_i \leq t_i\}$. We start from the database $\mathcal{A}_0 = \mathcal{A}_{\mathrm{CONST}}$. Then, for all $i = 1, \ldots, n$, we process the constraint t_i and construct a new database $\mathcal{A}_i$ from the database $\mathcal{A}_{i-1}$ using Lemma 39. From this lemma we have that finally $\mathcal{B} = \mathcal{A}_n$ satisfies Γ_3.

We prove now that $\mathcal{B}$ is exponential in the size of Γ. Let s be the size of Γ. From Lemma 9 the size of $\mathcal{A}_{\mathrm{CONST}}$ is at most exponential in s. While processing t_i in Γ_3 we add at most $\max_{k,j}(\Gamma_3) + \mathrm{lcm}_j(\Gamma_3)$ copies of the canonical structure for the formula in t_i (Lemma 39). From Corollary 34 it follows that the size of each such canonical structure is polynomial in s. Moreover, we claim that the value $\max_{k,j}(\Gamma_3) + \mathrm{lcm}_j(\Gamma_3)$ is at most exponential in s. This is since the value of $\max_{k,j}(\Gamma)$ is at most 2^s (since the numbers are in binary). Further, $\mathrm{lcm}_j(\Gamma) \leq \max_{k,j}(\Gamma)^{|\Gamma|}$ (since this is a product of at most $|\Gamma|$ numbers $\leq \max_{k,j}(\Gamma)$). Hence, $\mathrm{lcm}_j(\Gamma) \leq (2^s)^s \leq 2^{(s^2)}$. From Corollary 34 we have $\max_{k,j}(\Gamma_3) \leq 8\max_{k,j}(\Gamma) + 7\mathrm{lcm}_j(\Gamma)$. Hence, $\max_{k,j}(\Gamma_3) = \mathcal{O}(2^{(s^2)})$. Moreover, $\mathrm{lcm}_j(\Gamma_3) \leq \mathrm{lcm}_j(\Gamma) = \mathcal{O}(2^{(s^2)})$ Hence, for each $t \in \Gamma_3$, we add at most exponentially many tuples to relations in $\mathcal{B}$. Finally, since the number of constraints in Γ_3 is at most exponential as well, the size of $\mathcal{B}$ is exponential.

The only thing which would still be in doubt is if in the process of constructing $\mathcal{B}$ that satisfies Γ_3 we did not spoil anything concerning the constant constraints Γ_{CONST} which were satisfied in $\mathcal{A}_{\mathrm{CONST}}$.

Lemma 42. *Let $\mathcal{A}$ be a database such that $\mathcal{A} \models \Gamma_3 \cup \Gamma_{\mathrm{CONST}}$, let $\mathcal{A}_{\mathrm{CONST}}$ be a substructure of $\mathcal{A}$ such that $\mathcal{A}_{\mathrm{CONST}} \models \Gamma_{\mathrm{CONST}}$ and such that Γ_3 is in the normal form w.r.t. $\mathcal{A}_{\mathrm{CONST}}$. Let $\mathcal{B}$ be a database as constructed in the previous paragraphs. Then $\mathcal{B} \models \Gamma_{\mathrm{CONST}}$.*

We prove Lemma 42 below, but first, we would like to present some intuitions. Clearly, we have to show that there is no new tuple $\boldsymbol{a}$ of elements of $\mathrm{dom}(\mathcal{B})$ containing an element not in $\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$, such that $\boldsymbol{a}$ satisfies a formula in a constraint from Γ_{CONST}. The main intuitive reason for it is that the structure $\mathcal{B}$ has been built from copies of some substructures of $\mathcal{A}$. These copies are disjoint in $\mathcal{B}$ (in the sense that all elements which do not correspond to constants are distinct), although they were not necessarily disjoint in $\mathcal{A}$. However, since neither inequalities nor negation are allowed, if there is such a new tuple $\boldsymbol{a}$ in $\mathcal{B}$, then, saying informally, there has already been some version of $\boldsymbol{a}$ in $\mathcal{A}$ (possibly with some elements identified). We formalize this intuition in the rest of this section.

Definition 43. *We say that* $\mathcal{C}$ *is* a positive substructure *of* $\mathcal{D}$ *if* $\mathrm{dom}(\mathcal{C}) \subseteq \mathrm{dom}(\mathcal{D})$*, for each constant symbol* c *of* $\mathcal{C}$ *we have* $c^{\mathcal{C}} = c^{\mathcal{D}}$ *and for each relation* R *we have* $R^{\mathcal{C}} \subseteq R^{\mathcal{D}}$*.*

We make use of the following observation:

Observation 44. Let $(\varphi, k, j) \in \Gamma_3$. If $\mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi}$ is a substructure of $\mathcal{B}$ then $\mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi}$ is (isomorphic to) a positive substructure of $\mathcal{A}$.

Proof (of the observation). Suppose $\mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi}$ is a substructure of $\mathcal{B}$. Then there exists a minimal number i such that $\mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi}$ is isomorphic to a positive substructure of $\mathcal{A}_i$ (in particular $\mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi}$ is not isomorphic to a positive substructure of $\mathcal{A}_{i-1}$). Consider the constraint $(\varphi', k', j') \in \Gamma_3$ that we processed at the i-th step of the construction of $\mathcal{B}$. Since i is minimal, we extended the database $\mathcal{A}_{i-1}$ with at least one isomorphic copy of the canonical structure $\mathcal{C}_{\varphi}$. Hence, $\mathcal{C}_{\varphi}$ must be isomorphic to a positive substructure of $\mathcal{C}_{\varphi'}$. Now there are 2 cases:

Case 1: $k' > 0$. Since $\mathcal{A} \models \{(\varphi', k', j')\}$ we know that $\#(\varphi', \mathcal{A}) = k' + \alpha * j'$. Hence, there is at least one tuple satisfying φ' in $\mathcal{A}$. Therefore, since φ' contains the INEQ and NotConstants subformulas, $\mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi'}$ is isomorphic to a positive substructure of $\mathcal{A}$, and hence $\mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi}$ is isomorphic to a positive substructure of $\mathcal{A}$.

Case 2: $k' = 0$. Recall that we extended $\mathcal{A}_{i-1}$ with some number of copies of $\mathcal{C}_{\varphi'}$ because $\#(\varphi', \mathcal{A}_{i-1}) \neq \alpha * j'$ for any $\alpha \in \mathbb{N}$, including $\alpha = 0$. Thus, there is at least one tuple satisfying φ' in $\mathcal{A}_{i-1}$. Hence, as φ' contains the INEQ and NotConstants subformulas, $\mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi'}$ is isomorphic to a positive substructure of A_{i-1}. Therefore, since $\mathcal{C}_{\varphi}$ is isomorphic to a positive substructure of $\mathcal{C}_{\varphi'}$, $\mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi}$ is isomorphic to a positive substructure of A_{i-1}. However, this contradicts the minimality of i and thus the case $k' = 0$ is impossible. $\square$

Proof (of Lemma 42). For each φ, for each substructure $\mathcal{B}'$ of $\mathcal{B}$ of the form $\mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi}$ such that $\mathrm{GRAPH}(\mathcal{B}')$ is a connected component of $\mathrm{GRAPH}(\mathcal{B})$, fix a positive substructure $\mathcal{A}'$ of $\mathcal{A}$ of the form $\mathcal{A}_{\mathrm{CONST}} \uplus \mathcal{C}_{\varphi}$. The existence of $\mathcal{A}'$ is guaranteed by Observation 44. Let $h_{\mathcal{B}'} : \mathcal{B}' \to \mathcal{A}'$ be the obvious isomorphism.

We define a mapping $h\colon \mathcal{B} \to \mathcal{A}$. For $c \in \mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$ put $h(c) = c$. For $a \notin \mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$ put $h(a) = h_{\mathcal{B}'}(a)$, where $\mathcal{B}'$ is such that $a \in \mathrm{dom}(h_{\mathcal{B}'})$. Notice that for each $a \in \mathrm{dom}(\mathcal{B})$ the value $h(a)$ is well-defined. This is clear for $a \in \mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$. For $a \notin \mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$ notice that there is exactly one $\mathcal{B}'$ such that $a \in \mathrm{dom}(h_{\mathcal{B}'})$. Clearly, there is at least one since all connected components of GRAPH($\mathcal{B}$) are of the form GRAPH($\mathcal{C}_\varphi$) for some $(\varphi, k, j) \in \Gamma_3$. Moreover, there is at most one since GRAPH($\mathcal{B}'$) is a connected component of GRAPH($\mathcal{B}$) and a cannot be in two distinct connected components of GRAPH($\mathcal{B}$).

Now, suppose that $\mathcal{B} \not\models \Gamma_{\mathrm{CONST}}$. So, there exists a tuple $\boldsymbol{a}$ of elements of $\mathcal{B}$, containing some element e not in $\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$ such that for a constraint $(\psi, k, j) \in \Gamma_{\mathrm{CONST}}$ it holds $\mathcal{B} \models \psi(\boldsymbol{a})$. However, since $h(e)$ is defined as $h_{\mathcal{B}'}(e)$, where $h_{\mathcal{B}'}$ is an isomorphism between $\mathcal{B}'$ and $\mathcal{A}'$, the tuple $h(\boldsymbol{a})$ contains at least one element (i.e. $h(e)$) not in $\mathrm{dom}(\mathcal{A}_{\mathrm{CONST}})$. Moreover, $\mathcal{A} \models \psi(h(\boldsymbol{a}))$. This is since:

- h preserves structure – notice that there are no tuples in relations of $\mathcal{B}$ with elements from distinct connected components of GRAPH($\mathcal{B}$), and for each $\mathcal{B}'$, $h_{\mathcal{B}'}$ is an isomorphism.
- Negation and inequality are not allowed in constraints from Γ_{CONST}.

Therefore, we conclude that $\mathcal{A} \not\models \Gamma_{\mathrm{CONST}}$, a contradiction. This completes the proofs of Lemma 42 and Proposition 8. □

7 From the Intermediate Result to the Main Result

In this section we use some of the ideas from [2] to show how Theorem 7 implies Theorem 5. We start with the following definition and lemma from [2].

Definition 45. *Let R be a TreeQL-program and let d be a DTD such that R does not typecheck with respect to d. Then:*

- *there is a path $\boldsymbol{v} = v_1, \ldots, v_k$ in the program R where*
  1. *v_1 is a child of the root; and*
  2. *v_{i+1} is a child of v_i;*
  3. *$lab(v_i) = (\sigma_i, \varphi_i(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i))$, for $i \in \{1, \ldots, k\}$;*
  4. *v_k has precisely n children with labels $(\delta_1,\ \psi_1(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k,\ \boldsymbol{y}_1)), \ldots, (\delta_n,\ \psi_n(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k,\ \boldsymbol{y}_n))$ in that order;*

  such that
- *there is a database $\mathcal{A}$ with elements $\boldsymbol{a} = \boldsymbol{a}_1, \ldots, \boldsymbol{a}_k$ for which the following holds*
  1. *$\mathcal{A}$ satisfies $\varphi_i(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_i)$, for each $i = 1, \ldots, k$;*
  2. *$\delta_1^{m_1} \ldots \delta_n^{m_n} \notin d(\sigma_k)$ where $m_i = |\{\boldsymbol{b} \mid \mathcal{A} \models \psi_i(\boldsymbol{a}, \boldsymbol{b})\}|$, for all $i = 1, \ldots, n$.*

We say that $(\boldsymbol{v}, \mathcal{A}, \boldsymbol{a})$ is a breakpoint *for R and d.*

Lemma 46 ([2]). *Let $\delta_1, \ldots, \delta_n$ be symbols and let $V = (k_1, j_1), \ldots, (k_n, j_n)$ be a vector of n pairs of natural numbers. We denote by L_V the language of all words of the form: $\delta_1^{k_1+\alpha_1 * j_1} \ldots \delta_n^{k_n+\alpha_n * j_n}$ where each $\alpha_i \in \mathbb{N}$, $1 \leq i \leq n$. For each regular language L over alphabet $\{\delta_1, \ldots, \delta_n\}$, there exists a finite set Vec of vectors of pairs of natural numbers as above such that $\delta_1^* \ldots \delta_n^* - L = \bigcup_{V \in Vec} L_V$.*

Moreover, the values of the numbers in Vec are bounded by the number of states of the deterministic automaton recognizing the complement of L.

If R is a TreeQL program, v is a node of R, and d is a DTD then by $\overline{L_d(v)}$ we denote the complement of the language specified by d for the Σ-label of v.

Lemma 47. *Let R be a TreeQL program and let d be a DTD.*

1. *If R does not typecheck w.r.t. d then there is a node v_k of R, there is a vector V of n pairs of natural numbers, where n is the number of children of v_k, and there is a database $\mathcal{A}$ such that $L_V \subseteq \overline{L_d(v_k)}$, and such that the children of a node of $R(\mathcal{A})$ of the form (v_k, θ) form a word in L_V.*
   Moreover, the size of V is polynomial in the size of R and d.
2. *Let v_k be a node of R and let V be a vector of n pairs of natural numbers, where n is the number of children of v_k. If $L_V \subseteq \overline{L_d(v_k)}$ and there is a database $\mathcal{A}$ such that the children of a node of $R(\mathcal{A})$ of the form (v_k, θ) form a word in L_V, then R does not typecheck w.r.t. d.*
3. *Let v_k be a node of R and let V be a vector of n pairs of natural numbers, where n is the number of children of v_k. The containment test $L_V \subseteq \overline{L_d(v_k)}$ can be done in PSPACE.*

Proof. 1. The following proof comes from [2]. We use the notation introduced in Definition 45. Assume that the program R does not typecheck w.r.t. DTD d, then there exists a breakpoint $(\boldsymbol{v}, \mathcal{A}, \boldsymbol{a})$ for R and d. Let L be the language defined by the regular expression $d(\sigma_k)$ and let N be the node of $R(\mathcal{A})$ of the form (v_k, θ) where the error occurs. So, the word $\delta_1^{m_1} \ldots \delta_n^{m_n}$, formed by the children of N, is not in L. Thus, it is in the regular language $\delta_1^* \ldots \delta_n^* - L$. From Lemma 46 it follows that there exists a set of vectors Vec such that $\delta_1^* \ldots \delta_n^* - L = \bigcup_{V \in \text{Vec}} L_V$. Since $(\boldsymbol{v}, \mathcal{A}, \boldsymbol{a})$ is a breakpoint, there is a vector V in Vec such that the children of N form a word w in L_V. Note also that, if a vector V is in Vec, then L_V is contained in $\overline{L_d(v_k)}$. To prove the claim 1 it remains to show the bound on the size of V. From Lemma 46 the numbers in V are bounded by the number S of states of the DFA for $\overline{L_d(v_k)}$. Clearly, S is at most exponential in the size of DTD d. This is since the DFA for $\overline{L_d(v_k)}$ can be computed from $d(\sigma_k)$ with at most exponential cost. Hence, the sizes of the binary representations of the numbers in V are polynomial in the size of d.

2. Consider a node N of the form (v_k, θ) in the output tree $R(\mathcal{A})$. Notice that if L_V is contained in $\overline{L_d(v_k)}$ and if the children of N form a word w in L_V then $w \in \overline{L_d(v_k)}$. Hence, $w \notin L_{d(\sigma_k)}$ and thus R does not typecheck w.r.t. d.
3. We present a *brute-force* algorithm to do the containment test in polynomial space. Let $L = \overline{L_d(v_k)}$ and let S be the number of states of the DFA for

L. First, notice that if some word w of the form $\delta_1^{k_1+\alpha_1*j_1}\dots\delta_n^{k_n+\alpha_n*j_n}$ is not in L, where some $\alpha_i > S$, then there is a number $\beta_i \leq S$ such that $\delta_1^{k_1+\alpha_1*j_1}\dots\delta_i^{k_i+\beta_i*j_i}\dots\delta_n^{k_n+\alpha_n*j_n}$ is not in L either. Consider the state s of the DFA for L that is reached after reading the prefix $\delta_1^{k_1+\alpha_1*j_1}\dots\delta_i^{k_i+\alpha_i*j_i}$. Clearly, since $\alpha_i > S$, the same state s is reached after reading a word $\delta_1^{k_1+\alpha_1*j_1}\dots\delta_i^{k_i+\beta_i*j_i}$, for some $\beta_i \leq S$. Hence, it is enough to check all words in L_V with values of $\alpha_1,\dots,\alpha_n$ bounded by S. Although such words may be exponentially long, we represent them with all numbers written in binary. Each of these words can be verified to be in L in PSPACE. Simply, while reading a word w, we remember the set of reachable states of the NFA $\mathbb{A}$ for $d(\sigma_k)$. At the end, we check whether a final state of $\mathbb{A}$ can be reached. If yes then $w \in L$. Otherwise, $w \notin L$. Notice that the size of $\mathbb{A}$ is polynomial in the size of $d(\sigma_k)$ and thus we can remember each subset of the states of $\mathbb{A}$ in polynomial space. □

In the proof of Theorem 5 we need also the following lemma.

Lemma 48. *Let R be a TreeQL program, let v_k be a node of R, let V be a vector of n pairs of natural numbers, where n is the number of children of v_k, and let $\mathcal{A}$ be a database. There there is a set of modulo constraints $\Gamma_{R,v_k,V}$ such that $\mathcal{A} \models \Gamma_{R,v_k,V}$ if and only if there is a node N in $R(\mathcal{A})$ which is of the form (v_k,θ) and the children of N form a word in L_V. Moreover, the size of $\Gamma_{R,v_k,V}$ is polynomial in the size of R and V.*

Proof. We define $\Gamma_{R,v_k,V}$ to be a set of modulo constraints over the vocabulary given by the program R with the new constant symbols $\boldsymbol{a} = \boldsymbol{a_1},\dots,\boldsymbol{a_k}$.

For formulas $\varphi_1,\dots,\varphi_k$ in the nodes $v_1,\dots,v_k$ that form a path from the root of R to v_k, we define the following constraint t_0: $(\bigwedge_{i=1,\dots,k}\varphi_i(\boldsymbol{a_1},\dots,\boldsymbol{a_i}),1,0)$. Notice that $\mathcal{A} \models t_0$ if and only if $\mathcal{A}$ satisfies $\varphi_i(\boldsymbol{a_1},\dots,\boldsymbol{a_i})$, for each $i = 1,\dots,k$.

For the l-th child of v_k, with a formula ψ_l, we define the following constraint $t_l = (\psi_l(\boldsymbol{a},\boldsymbol{y_l}),k_l,j_l)$, where (k_l,j_l) is the l-th pair of numbers in V.

Notice that $\mathcal{A} \models \{t_1,\dots,t_n\}$ if and only if $\delta_1^{\alpha_1}\dots\delta_n^{\alpha_n} \in L_V$, where $\alpha_l = |\{\boldsymbol{b} \mid \mathcal{A} \models \psi_l(\boldsymbol{a},\boldsymbol{b})\}|$, for all $l = 1,\dots,n$.

Finally, $\Gamma_{R,v_k,V}$ is defined as $\{t_0,\dots,t_n\}$. Notice that since the formulas in the constraints in $\Gamma_{R,v_k,V}$ are from R and the numbers in the constraints in $\Gamma_{R,v_k,V}$ are from V the size of $\Gamma_{R,v_k,V}$ is polynomial in the size of R and V. □

Proof (of Theorem 5). The following NEXPTIME algorithm decides the complement of the problem of typechecking. Let R be a TreeQL program and let d be a DTD.

1. Guess a node v_k in the program R. Let S be the number of states of the DFA for $\overline{L_d(v_k)}$ and let n be the number of children of v_k. Then guess a vector V of n pairs of natural numbers, with each number less or equal to S.
2. Verify whether $L_V \subseteq \overline{L_d(v_k)}$ and then whether $\Gamma_{R,v_k,V}$ is satisfiable.

From claim 3 of Lemma 47 and from Theorem 7 it follows that the verification phase is in NEXPTIME. From Lemma 48 and from claim 2 of Lemma 47 it follows that for each v_k and V such that $L_V \subseteq \overline{L_d(v_k)}$ if there is a database $\mathcal{A}$ such that $\mathcal{A} \models \Gamma_{R,v_k,V}$ then R does not typecheck w.r.t. DTD d. Conversely, if R does not typecheck w.r.t. d then from claim 1 of Lemma 47 it is possible to guess v_k and V such that $L_V \subseteq \overline{L_d(v_k)}$ and there is a database $\mathcal{A}$ such that the children of a node of $R(\mathcal{A})$ of the form (v_k, θ) form a word in L_V. Then $\mathcal{A} \models \Gamma_{R,v_k,V}$ from Lemma 48. □

8 The Lower Bound

In this section we show that the problem of satisfiability of a set of modulo constraints is NEXPTIME-hard. Then we use this result to show coNEXPTIME-hardness of the typechecking problem.

Theorem 49. *The problem of satisfiability of a set of modulo constraints is NEXPTIME-hard, even if the numbers in the modulo constraints are represented in unary.*

Proof. We use a reduction from the following tiling problem. *An instance of the tiling problem* I is a finite set of tiles T, a set of quadruples of tiles $A \subseteq T^4$, two tiles $s, e \in T$ and a natural number n in unary. *An exponential square tiling for* I is a mapping $\delta : \{1, \ldots, 2^n\} \times \{1, \ldots, 2^n\} \to T$ satisfying the following constraints:

- the first tile in the first row is s: $\delta(1, 1) = s$;
- the first tile in the last row is e: $\delta(2^n, 1) = e$;
- all quadruples in the tiling are in A: for all $1 \leq i < 2^n, 1 \leq j < 2^n, (\delta(i, j), \delta(i+1, j), \delta(i, j+1), \delta(i+1, j+1)) \in A$

It is NEXPTIME-hard to decide whether, for a given tiling instance I, there exists an exponential square tiling [4].

Let I be an instance of the tiling problem. We will construct an instance Γ_I of the satisfiability problem for modulo constraints such that Γ_I is satisfiable if and only if there exists an exponential square tiling for I.

Let us denote $|T| - 1$ by d. We use the constants $c_0, c_1, \ldots, c_d$, unary relations R_{01}, R_{0d} and a relation R of arity $2n + 1$. The relation R will encode the tiling for I. The role of the constants and auxiliary relations R_{01}, R_{0d} is to characterize the domain of the satisfying database, and thus, to provide a convenient tool to describe the relation R.

We now describe the desired interpretation of the relations R_{01} and R_{0d}. In short, we require that R_{0d} is the set $\{c_0, \ldots, c_d\}$ and R_{01} is the set $\{c_0, c_1\}$. The constant c_i represents the i-th tile in T, for $i \in \{0, 1, \ldots, d\}$. Additionally, c_0 and c_1 are used as binary digits in order to encode numbers of columns and rows in the tiling. We use the following modulo constraints:

1. $(R_{01}(c_0) \wedge R_{01}(c_1), 1, 0)$;
2. $(R_{01}(x), 2, 0)$;
3. $(\bigwedge_{i=0,\ldots,d} R_{0d}(c_i), 1, 0)$;

4. $(R_{0d}(x), d+1, 0)$.

Now, we present the constraints concerning R. From our point of view only some tuples in R are interesting. We say that a tuple $(x_1, \ldots, x_{2n}, z)$ of elements of a database $\mathcal{A}$ is *valid* if for all i in $\{1, \ldots, 2n\}$, x_i is one of the two elements from R_{01} and z is an element from R_{0d}. Each valid tuple in R defines the value of $\delta(i,j)$ (i.e. the tile used in the i-th column and j-th row of the tiling) for some $i, j \in \{1, \ldots, 2^n\}$. The first n arguments of R serve as a binary encoding of i and the next n arguments as a binary encoding of j. The last argument represents the tile $\delta(i,j)$.

5. "there are no two distinct tuples describing the same column and row in the tiling": for all $i, j \in \{0, 1, \ldots, c\}$, such that $i \neq j$:

$$(R(x_1, \ldots, x_n, y_1, \ldots, y_n, c_i) \wedge R(x_1, \ldots, x_n, y_1, \ldots, y_n, c_j), 0, 0)$$

We have to ensure that each possible tile in the tiling is defined — we require the existence of all possible $2^{(2n)}$ valid tuples in R. We cannot express this directly, since the numbers in the constraints should be in unary and the size of the constraints should be polynomial. Instead, we use the Chinese Remainder Theorem. Let $p_1, \ldots, p_k$ be the first k primes, where k is the smallest number such that the product $\Pi_{i=1}^{k} p_i$ is greater than $2^{(2n)}$. Obviously, $k \leq 2n$ (note that for each i we have $p_i \geq 2$).

Let r_i be $2^{(2n)}$ modulo p_i, for $i \in \{1, \ldots, k\}$. The constraints 6 say that the number of valid tuples is congruent with r_i, modulo p_i, for every i. Obviously, if we have 2^{2n} valid tuples, the constraints are satisfied. The Chinese Remainder Theorem now shows that all solutions to these constraints (i.e. the numbers of the tuples satisfying the formulas in these constraints) are congruent with 2^{2n} modulo $\Pi_{i=1}^{k} p_i$, which implies that there must be at least 2^{2n} valid tuples. Since only the two elements from R_{01} (i.e. c_0 and c_1) are allowed as the first $2n$ arguments, and the first $2n$ arguments determine the $(2n+1)$-th argument, $2^{(2n)}$ is also the maximal number of valid tuples in R. Moreover, since $k \leq 2n$, it follows from the prime number theorem that p_k is $\mathcal{O}(n \log n)$. Hence, all the numbers r_i and p_i are $\mathcal{O}(n \log n)$.

6. for $i \in \{1, \ldots, k\}$:

$$(R(x_1, \ldots, x_n, y_1, \ldots y_n, z) \wedge R_{0d}(z) \wedge \bigwedge_{j=1}^{n} (R_{01}(x_j) \wedge R_{01}(y_j)), r_i, p_i)$$

The following constraints require that R describes a correct tiling. For simplicity, we write c_t, where t is a tile in T, to denote the constant representing the tile t.

7. “the first tile in the first row is s”: $(R(c_0, \dots, c_0, c_0, \dots c_0, c_s), 1, 0)$;
8. “the first tile in the last row is e”: $(R(c_0, \dots, c_0, c_1, \dots c_1, c_e), 1, 0)$;
9. “there are no quadruples in the tiling from the complement of A”: for all $r, s \in \{1, \dots, n\}$ and for all quadruples $(t_1, t_2, t_3, t_4) \in T^4 \setminus A$:

$$(R(x_1, x_2, \dots, x_{r-1}, c_0, c_1, \dots, c_1, y_1, y_2, \dots, y_{s-1}, c_0, c_1, \dots, c_1, c_{t_1}) \wedge$$

$$R(x_1, x_2, \dots, x_{r-1}, c_1, c_0, \dots, c_0, y_1, y_2, \dots, y_{s-1}, c_0, c_1, \dots, c_1, c_{t_2}) \wedge$$

$$R(x_1, x_2, \dots, x_{r-1}, c_0, c_1, \dots, c_1, y_1, y_2, \dots, y_{s-1}, c_1, c_0, \dots, c_0, c_{t_3}) \wedge$$

$$R(x_1, x_2, \dots, x_{r-1}, c_1, c_0, \dots, c_0, y_1, y_2, \dots, y_{s-1}, c_1, c_0, \dots, c_0, c_{t_4}),\ 0,\ 0).$$

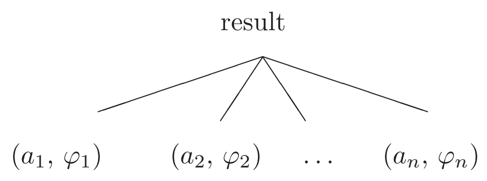
This ends the construction of the set of constraints $\mathit{\Gamma}_I$. Clearly, this reduction can be done in polynomial time and it is easy to prove its correctness. □

Theorem 50. *The problem of typechecking a TreeQL program with projection-free CQs*[10] *w.r.t. a DTD with arbitrary regular expressions is coNEXPTIME-hard.*

Proof. We reduce the satisfiability problem of a set of modulo constraints with all numbers represented in unary to the complement of the typechecking problem.

Let $\mathit{\Gamma} = \{(\varphi_i, k_i, j_i) \,|\, 1 \le i \le n\}$ be a set of modulo constraints, with all numbers represented in unary. We construct a TreeQL program P and a DTD d such that P does not typecheck w.r.t. d if and only if there exists a database $\mathcal{A}$ satisfying $\mathit{\Gamma}$. Then, the claim will follow from Theorem 49.

The TreeQL program is defined as follows:



Now, we define a DTD d with regular expressions over the alphabet $\mathit{\Sigma} = \{a_1, a_2, \dots, a_n\}$. For each $i \in \{1, 2, \dots, n\}$, let r_i be a regular expression describing the language:

$$\{a_i^{m_i} \,|\, m_i \ne k_i + \alpha_i * j_i \text{ for each } \alpha_i \in \mathbb{N}\}.$$

Since k_i and j_i are in unary, it is easy to construct such an expression of a size bounded polynomially w.r.t. $k_i + j_i$ (e.g. for $k_i = 3$ and $j_i = 5$ we have the regular expression $(\epsilon + a_i + a_i a_i + a_i a_i a_i a_i)(a_i a_i a_i a_i a_i)^*$).

Let us denote $\mathit{\Sigma}_p^q = \{a_i \in \mathit{\Sigma} \,|\, p \le i \le q\}$ if $p \le q$, and $\mathit{\Sigma}_p^q = \emptyset$ otherwise. For each $i \in \{1, 2, \dots, n\}$ we define s_i to be the regular expression $(\mathit{\Sigma}_1^{i-1})^* \cdot r_i \cdot$

[10] The definition of projection-free CQs is in Section 2.

$(\Sigma_{i+1}^n)^*$. Intuitively, if for some database $\mathcal{A}$ the children of the root node of $P(\mathcal{A})$ form a word in s_i then $\mathcal{A}$ does not satisfy the modulo constraint (φ_i, k_i, j_i).

Finally, the DTD d is defined as follows: $d(\text{result}) = s_1 + s_2 + \ldots + s_n$ and $d(a_i) = \epsilon$ (for each $i \in \{1, 2, \ldots, n\}$).

Suppose the TreeQL program P does not typecheck w.r.t the DTD d. Hence, there is a database $\mathcal{A}$ such that $P(\mathcal{A})$ does not satisfy d. Consequently, the children of the root of $P(\mathcal{A})$ form a word w not in the language of s_i, for each $i \in \{1, 2, \ldots, n\}$. Hence, the word w is of the form:

$$a_1^{k_1+j_1*\alpha_1} a_2^{k_2+j_2*\alpha_2} \ldots a_n^{k_n+j_n*\alpha_n},$$

for some $\alpha_1, \ldots, \alpha_n \in \mathbb{N}$. Clearly, $\mathcal{A}$ satisfies the set of constraints Γ.

Conversely, if there is a database $\mathcal{A}$ such that $\mathcal{A} \models \Gamma$ then $P(\mathcal{A})$ does not satisfy d and consequently P does not typecheck w.r.t. d. □

9 Conclusions

We have proved that the typechecking problem for TreeQL programs containing projection-free conjunctive queries and DTDs with arbitrary regular expressions is coNEXPTIME-complete. Our main technical contribution consists of proving NEXPTIME-completeness of the problem of satisfiability of a set of modulo constraints with projection-free conjunctive queries. In our opinion the notion of modulo constraints is natural and simple. However, in spite of this, very little is known about the complexity of satisfiability of modulo constraints. In [2] it was shown that if we allow projection-free conjunctive queries with negations and inequalities in the constraints then the problem is decidable, but the algorithm, presented there, has non-elementary complexity. To make things worse in presence of projections we do not even know whether the problem is decidable. And, at the same time, we cannot disprove the following conjecture:

Conjecture. For each satisfiable set of modulo constraints with conjunctive queries (possibly with projections) there is a satisfying database of exponential size.

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[2] Noga Alon, Tova Milo, Frank Neven, Dan Suciu, and Victor Vianu. Typechecking XML views of relational databases. *ACM Trans. Comput. Log.*, 4(3):315–354, 2003.

[3] Noga Alon, Tova Milo, Frank Neven, Dan Suciu, and Victor Vianu. XML with data values: typechecking revisited. *J. Comput. Syst. Sci.*, 66(4):688–727, 2003.

[4] Bogdan S. Chlebus. Domino-Tiling Games. *J. Comput. Syst. Sci.*, 32(3):374–392, 1986.

[5] James Clark and Makoto Murata. Relax NG. http://www.relaxng.org.

[6] Heinz-Dieter Ebbinghaus and Joerg Flum. *Finite model theory.* Springer, 1999.

[7] Joost Engelfriet and Heiko Vogler. Macro tree transducers. *J. Comput. Syst. Sci.*, 31(1):71–146, 1985.

[8] Mary Fernandez, Dan Suciu, and Wang-Chiew Tan. SilkRoute: trading between relations and XML, 2000.

[9] Alain Frisch and Haruo Hosoya. Towards Practical Typechecking for Macro Tree Transducers. In Marcelo Arenas and Michael I. Schwartzbach, editors, *DBPL*, volume 4797 of *Lecture Notes in Computer Science*, pages 246–260. Springer, 2007.

[10] Sebastian Maneth, Alexandru Berlea, Thomas Perst, and Helmut Seidl. Xml type checking with macro tree transducers. In Chen Li, editor, *PODS*, pages 283–294. ACM, 2005.

[11] Sebastian Maneth, Thomas Perst, and Helmut Seidl. Exact XML Type Checking in Polynomial Time. In Schwentick and Suciu [18], pages 254–268.

[12] Wim Martens and Frank Neven. On the complexity of typechecking top-down XML transformations. *Theor. Comput. Sci.*, 336(1):153–180, 2005.

[13] Wim Martens and Frank Neven. Frontiers of tractability for typechecking simple XML transformations. *J. Comput. Syst. Sci.*, 73(3):362–390, 2007.

[14] Wim Martens, Frank Neven, and Marc Gyssens. On Typechecking Top-Down XML Tranformations: Fixed Input or Output Schemas. *Inf. Comput.* to appear.

[15] Wim Martens, Frank Neven, Thomas Schwentick, and Geert Jan Bex. Expressiveness and complexity of XML Schema. *ACM Trans. Database Syst.*, 31(3):770–813, 2006.

[16] Tova Milo, Dan Suciu, and Victor Vianu. Typechecking for XML transformers. *J. Comput. Syst. Sci.*, 66(1):66–97, 2003.

[17] Makoto Murata, Dongwon Lee, Murali Mani, and Kohsuke Kawaguchi. Taxonomy of xml schema languages using formal language theory. *ACM Trans. Internet Techn.*, 5(4):660–704, 2005.

[18] Thomas Schwentick and Dan Suciu, editors. *Database Theory - ICDT 2007, 11th International Conference, Barcelona, Spain, January 10-12, 2007, Proceedings*, volume 4353 of *Lecture Notes in Computer Science*. Springer, 2007.

[19] W3C. Extensible Markup Language (XML) 1.0. http://www.w3.org/TR/xml, 1999.

[20] W3C. XML Schema Part 1: Structures. http://www.w3.org/TR/xmlschema-1, 2004.

[21] Piotr Wieczorek. Complexity of Typechecking XML Views of Relational Databases. In Schwentick and Suciu [18], pages 239–253.