Projective Quantifier Elimination for Equational Constraint Solving *

Javier Álvez, Montserrat Hermo, and Paqui Lucio

The University of the Basque Country, 20080-San Sebastián, Spain
javier.alvez,montserrat.hermo,paqui.lucio@ehu.es

Abstract. We deal with *(general) equational constraints*, that is, first-order formulas, with equality as its only predicate symbol, over a (finite or infinite) language $\mathcal{L}$ of function symbols. As semantic domain, we consider the algebra of finite terms over $\mathcal{L}$. Solving one of such constraints means to find all the solutions for (i.e. assignments to) its free variables. We present the *Projective Quantifier Elimination (PQE)* technique for solving equational constraints (in particular, deciding their satisfiability) that performs an algebraic-style transformation of expressions instead of handling first-order formulas. PQE is formulated on the basis of three algebraic operations on expressions: complement, intersection and projection. We aim to contribute not only a new style of quantifier elimination for constraint solving, but also a more efficient method for equational constraint solving. PQE avoids unnecessary applications of the costly *Explosion Rule (ER)* that are performed by traditional solving methods.

1 Introduction

We deal with *(general) equational constraints*, that is, first-order formulas, with equality as its only predicate symbol, over a (finite or infinite) language $\mathcal{L}$ of function symbols. As semantic domain, we consider the algebra of finite terms over $\mathcal{L}$. Solving one of such constraints means to find all the solutions for (i.e. assignments to) its free variables. *Equational constraint solving methods* are rewriting processes that transform the input constraint into an equivalent Boolean combination (usually, disjunction) of constraints, in the so-called *solved form*. The solved form represents the solutions of the constraints, since every solved form is satisfiable whenever it is syntactically different from the logical constant for falsehood. The most well-known algorithms for equational solving in this setting (see [5, 8, 6, 10, 13, 14]) and later extensions to richer theories (see [17]) are based on some Quantifier Elimination (QE) technique. A terminating non-deterministic method based on a set of rules is presented in [8, 6, 7], and different deterministic methods are introduced in [5, 10, 13, 14]. All of them (implicitly or explicitly) assume some specific notion of solved form, which allows disequations and (often, but not always) a limited form of universal quantification (see [6] for a good

* This work has been partially supported by the Spanish Project TIN2007-66523 and the Basque Projects S-PE12UN050 and GIU12/26.

comparison of solved forms). In these methods, the elimination of those universal quantifications that are not allowed by the notion of solved form requires the application of the so-called *Explosion Rule (ER)* (following [8]) or the *Weak Domain Closure Axiom* (following [13]). Roughly speaking, an application of the ER, to eliminate a universal variable x (and, hence, the quantifier on x), amounts to conjoining a constraint representing the disjunction $\bigvee_{f\in\mathcal{L}} \exists \overline{v}\ (x \approx f(\overline{v}))$ where $\overline{v}$ are fresh variables and $\mathcal{L}$ is the set of function symbols of a finite language. In most of the solving methods —for efficiency and/or for termination— ER is applied whenever no other rule or transformation step can be applied.

In this paper, we present a new style of QE, called *Projective Quantifier Elimination (PQE)*, to solve equational constraints over the algebra of finite terms of a (finite or infinite) language of function symbols. The PQE technique improves previous results [1–3] by two of the authors. On one hand, it performs an algebraic-style transformation of expressions instead of handling first-order formulas. The expressions we use to represent the matrix (i.e. the formula inside the prefix of quantifiers) are heavily inspired by the notion of *implicit generalization* that was introduced by Lassez and Marriot [12] to study complement problems in the area of machine learning. Since then, different applications of this notion have appeared in many other areas, such as logic and functional programming, model-building, etc. Their computational properties have been extensively studied by R. Pichler (see e.g. [16, 15]). The main power of implicit generalizations is their high expressivity to compactly represent infinite sets of ground terms. In [4], the authors introduce use two notions closely related to implicit generalization, although they left open the problem of deciding their satisfiability. On the other hand, the PQE technique is formulated on the basis of three algebraic operations on such expressions: complement, intersection and projection. The compact representation of the matrix, along with the operation of projection, are the crucial keys to avoid the matrix blow-up produced by ER.

Outline of the paper. In Section 2 we provide basic notation and background to make the paper self-contained. In Section 3, we define the syntax and semantics of the ct-expressions that are the basis of the PQE technique. In Section 4 we explain how to transform a quantifier-free equational constraint into a ct-expression. In Section 5 we define the notion of weak normal form (wnf) for ct-expressions and prove the equivalence results that are expected for a normal form notion. In Section 7 we introduce the crucial operation of projection that allows the elimination of quantifiers through a controlled use of the ER –in the case of finite language– and without any explosion for infinite languages. For that purpose, the notion of normal form (for finite languages) is previously introduced in Section 6. The PQE technique is introduced in Section 8 where its completeness is proved. We provide some concluding remarks in Section 9

2 Preliminaries

Equational constraints are built on a language $\mathcal{L}$ of function symbols, an infinite set of variables $\mathcal{X}$, and the only predicate $\approx$ of equality. A *term* is either

a variable from $\mathcal{X}$ or a function symbol f of arity n from $\mathcal{L}$ applied to a n-tuple of terms, called *subterms*. $\mathcal{T}_{\mathcal{L}}(\mathcal{X})$ denotes the set of all possible terms. If S is any collection of terms, then $\mathtt{Var}(S)$ denotes the set of all variables occurring in S. A term without variables is said to be ground and the set of all ground terms over a language $\mathcal{L}$ is denoted by $\mathcal{T}_{\mathcal{L}}$. Given two terms t_1 and t_2, $t_1 \approx t_2$ is called an *equation* and $t_1 \not\approx t_2$ is an abbreviation for $\neg(t_1 \approx t_2)$ and it is called a *disequation*. A tuple of n terms $(t_1, \ldots, t_n)$ (in particular, variables $(x_1, \ldots, x_n)$) is abbreviated by $\overline{t}$ (resp. $\overline{x}$). We use juxtaposition to denote tuple concatenation. That is, given an n-tuple of terms $\overline{t} = (t_1, \ldots, t_n)$ and an m-tuple of terms $\overline{s} = (s_1, \ldots, s_m)$, the expression $\overline{t}\overline{s}$ denotes the $n + m$-tuple $(t_1, \ldots, t_n, s_1, \ldots, s_m)$. In abuse of notation, we treat tuples as sets whenever order is not relevant. An *equational constraint* is an arbitrary first-order formulas, with equality as its only predicate symbol, over a (finite or infinite) language $\mathcal{L}$ of function symbols, including constant formulas: *true* and *false*. In other words, equational constraints are constructed using constant formulas and equations as atoms, connectives ($\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$) and quantifiers ($\exists$, $\forall$) as usual in first-order syntax. We denote by $\mathcal{C}_{\mathcal{L}}(\mathcal{X})$) the set of all equality constraints over $\mathcal{L}$ and $\mathcal{X}$. $\mathtt{Free}(\varphi)$ is the set of all free variables in a given $\varphi \in \mathcal{C}_{\mathcal{L}}(\mathcal{X})$. Let $\overline{v} = \mathtt{Free}(\varphi)$, $\varphi^{\exists}$ and $\varphi^{\forall}$ are abbreviations for $\exists\overline{v}\ (\varphi)$ and $\forall\overline{v}\ (\varphi)$, respectively. In the sequel, for brevity, we say *constraint* instead of equational constraint.

A *substitution* $\sigma : \mathcal{X} \rightarrow \mathcal{T}_{\mathcal{L}}(\mathcal{X})$ is a mapping from a n-tuple of variables $\overline{x} \subset \mathcal{X}$ called its *domain* (denoted $\mathtt{dom}(\sigma)$) into an n-tuple in $(\mathcal{T}_{\mathcal{L}}(\mathcal{X}))^n$ called its *range* (denoted $\mathtt{range}(\sigma)$). When convenient, we see the mapping σ as a set of associations of terms to variables, namely $\sigma = \{x_1 \leftarrow t_1,\ \ldots,\ x_n \leftarrow t_n\}$, which is usually abbreviated by $\overline{x} \leftarrow \overline{t}$.[1] We denote by $\mathtt{rgVar}(\sigma)$ the set $\mathtt{Var}(\mathtt{range}(\sigma))$. The *empty substitution* (that is, both its domain and range are empty) is denoted by ε. The application of a substitution σ to a term t, denoted by $t\sigma$, is called an *instance* of t. We denote by $\mathtt{ground}_{\mathcal{L}}(t)$ the set of all ground instances of t over the language $\mathcal{L}$. A substitution is called a *renaming* if it is a bijection from $\mathcal{X}$ onto $\mathcal{X}$. The *restriction of a substitution σ to the variables $\overline{x}$*, denoted by $\sigma_{\restriction\overline{x}}$, is defined as $\{\ x \leftarrow t \mid (x \leftarrow t) \in \sigma \text{ and } x \in \overline{x}\ \}$. A substitution σ is said to be an *assignment* (or *ground substitution*) if it is a mapping from $\mathcal{X}$ into $\mathcal{T}_{\mathcal{L}}$. A substitution σ is said to be *linear* if no variable occurs more than once in the tuple of terms $\mathtt{range}(\sigma)$. When a substitution is not linear, we call it non-linear. The *most general unifier* of a set of terms $\{t_1,\ \ldots,\ t_n\}$ (which is denoted by $\mathtt{mgu}(t_1,\ \ldots,\ t_n)$) is an idempotent substitution σ such that $t_i\sigma = t_j\sigma$ for every $1 \leq i,\ j \leq n$ and, for any other substitution θ with the same property, $\theta = \sigma\alpha$ holds for some substitution α. If $\mathtt{mgu}(t_1,\ \ldots,\ t_n)$ does (not) exist, then the terms $t_1,\ \ldots,\ t_n$ are said to be *(non-)unifiable*. The *most general common instance* of a unifiable set of terms $\{t_1,\ \ldots,\ t_n\}$, denoted by $\mathtt{mgi}(t_1,\ \ldots,\ t_n)$, is any $t_i\sigma$ for $1 \leq i \leq n$ where $\sigma = \mathtt{mgu}(t_1,\ \ldots,\ t_n)$.

Constraints from $\mathcal{C}_{\mathcal{L}}(\mathcal{X})$ are interpreted in the algebra of all ground terms over a language $\mathcal{L}$, also called the Herbrand structure $\mathcal{H}_{\mathcal{L}}$ over $\mathcal{L}$, whose universe is $\mathcal{T}_{\mathcal{L}}$. We assume that $\mathcal{L}$ contains at least two function symbols one of them

[1] where $\overline{x} = (x_1,\ \ldots,\ x_n)$ and $\overline{t} = (t_1,\ \ldots,\ t_n)$.

with arity 0, because otherwise the Herbrand universe is empty or finite. The structure $\mathcal{H}_\mathcal{L}$ is the standard model of the *free equality theory* $\mathtt{FET}_\mathcal{L}$ (of the language $\mathcal{L}$). Since $\mathtt{FET}_\mathcal{L}$ is an elementary theory[2], every $\varphi \in \mathcal{C}_\mathcal{L}(\mathcal{X})$ is a logical consequence of $\mathtt{FET}_\mathcal{L}$ if and only if $\mathcal{H}_\mathcal{L}$ is a model of φ. The theory $\mathtt{FET}_\mathcal{L}$ (which is also called the theory of term algebra and Clark's equational theory) was originally introduced by Malcev in [14] and was shown to be decidable in [13] (see also [8]). It is also a well-known result that the decision problem of whether a given formula is satisfiable in $\mathcal{H}_\mathcal{L}$ is worst-case non-elementary (see [9, 18]). The inherent complexity of the satisfaction problem of equational constraints for finite signatures is studied in [16]. Given a constraint $\varphi \in \mathcal{C}_\mathcal{L}(\mathcal{X})$, we would like not only to decide if φ is satisfiable (in $\mathcal{H}_\mathcal{L}$), but also to solve φ, that is, to find its solutions. An assignment σ is said to be an *$\mathcal{L}$-solution* of φ if $\mathtt{dom}(\sigma) \supseteq \mathtt{Free}(\varphi)$, $\mathtt{range}(\sigma) \subseteq \mathcal{T}_\mathcal{L}(\mathcal{X})$ and $\mathcal{H}_\mathcal{L} \models \varphi\sigma$. When $\mathcal{L}$ is not relevant, we call them solutions. Consequently, φ is said to be *satisfiable* if it has at least one solution, otherwise φ is said to be *unsatisfiable*. Note that φ is satisfiable if and only if $\mathcal{H}_\mathcal{L} \models \varphi^\exists$. Two constraints φ_1 and φ_2 are said to be *$\mathcal{L}$-equivalent* if $\mathcal{H}_\mathcal{L} \models (\varphi_1 \leftrightarrow \varphi_2)^\forall$, which means that for every assignment σ, $\mathcal{H}_\mathcal{L} \models \varphi_1\sigma$ if and only if $\mathcal{H}_\mathcal{L} \models \varphi_2\sigma$. In other words, both constraints have the same solutions.

3 Expressions on Constrained Terms

In this section, we present the notion of *ct-expression (or constrained terms expression)* to represent sets of tuples of ground terms in a very compact way that is especially suitable for our PQE technique. This notation is strongly inspired by the notion of *implicit generalization* introduced in [12].

Definition 1. *A ct-expression (ct abbreviates constrained term) of arity k is any syntactic object that can be recursively defined as follows*

$$E \ ::= \ \bot \mid \top \mid \bar{t} \setminus\setminus \Theta \mid \ \sim E_1 \mid E_1 \sqcap E_2 \mid E_1 \sqcup E_2$$

where $\bar{t} \in \mathcal{T}_\mathcal{L}^k$ is a k-tuple of terms, Θ is a set of idempotent substitutions such that $\mathtt{dom}(\theta) \subseteq \mathtt{Var}(\bar{t})$ for each $\theta \in \Theta$, and E_1, E_2 are ct-expressions of arity k.[3] *We denote by $\mathcal{E}_\mathcal{L}^k(\mathcal{X})$ the set of all ct-expressions of arity k over language $\mathcal{L}$ and variables $\mathcal{X}$.*

The ct-expressions $\bot$ (empty), $\top$ (total), and $\bar{t} \setminus\setminus \Theta$ are *atomic*.

Example 1. An example of atomic ct-expression over $\mathcal{L} = \{a_{/0}, f_{/1}, g_{/1}\}$ is

$$(z_1, f(z_2), g(z_3), z_4) \setminus\setminus [\ (z_1, z_2, z_3) \leftarrow (a, g(w_1), f(w_2)), (z_1, z_4) \leftarrow (f(w_3), g(a))\ ].$$

When Θ is empty in $\bar{t} \setminus\setminus \Theta$ we simply write $\bar{t}$. Indeed, a tuple of terms is the ct-expression that we associate to an equation in Definition 4 whenever its two terms are unifiable. Negated equations (or disequations) give rise to the general form $\bar{t} \setminus\setminus \Theta$.

Next, we provide a semantic function $[\![\_]\!]_\mathcal{L}$ (relative to a language $\mathcal{L}$) that associates a subset of $\mathcal{T}_\mathcal{L}^k$ to each ct-expression in $\mathcal{E}_\mathcal{L}^k(\mathcal{X})$.

[2] All its models satisfy the same first-order sentences

[3] We consider that $\bot$ and $\top$ have any arity $k \geq 0$.

Definition 2. *The semantic function* $[\![\_]\!]_{\mathcal{L}} :: \mathcal{E}^k_{\mathcal{L}}(\mathcal{X}) \to 2^{(\mathcal{T}_{\mathcal{L}})^k}$ *is inductively defined by*[4]

$$[\![E]\!]_{\mathcal{L}} = \begin{cases} \emptyset & \text{if } E = \bot \\ \mathcal{T}^k_{\mathcal{L}} & \text{if } E = \top \\ \mathtt{ground}_{\mathcal{L}}(\overline{t}) \setminus \bigcup_{\theta \in \Theta} \mathtt{ground}_{\mathcal{L}}(\overline{t}\theta) & \text{if } E = \overline{t} \setminus\!\setminus \Theta \\ \mathcal{T}^k_{\mathcal{L}} \setminus [\![E']\!]_{\mathcal{L}} & \text{if } E = \sim E' \\ [\![E_1]\!]_{\mathcal{L}} \cap [\![E_2]\!]_{\mathcal{L}} & \text{if } E = E_1 \sqcap E_2 \\ [\![E_1]\!]_{\mathcal{L}} \cup [\![E_2]\!]_{\mathcal{L}} & \text{if } E = E_1 \sqcup E_2. \end{cases}$$

Note that $[\![\overline{t}]\!]_{\mathcal{L}} = \mathtt{ground}_{\mathcal{L}}(\overline{t})$ for any $\overline{t} \in \mathcal{T}^k_{\mathcal{L}}$. In particular, $[\![\overline{x}]\!]_{\mathcal{L}} = \mathcal{T}^k_{\mathcal{L}}$ for any k-tuple of distinct variables $\overline{x} \subset \mathcal{X}$.

Definition 3. *Two ct-expressions* $E_1, E_2 \in \mathcal{E}^k_{\mathcal{L}}(\mathcal{X})$ *are (semantically)* $\mathcal{L}$*-equivalent iff* $[\![E_1]\!]_{\mathcal{L}} = [\![E_2]\!]_{\mathcal{L}}$*. It is denoted by* $E_1 \equiv_{\mathcal{L}} E_2$*.*

4 From Constraints to Expressions

The PQE technique transforms any constraint into a ct-expression that represents its solutions. It is well known that any constraint $\varphi \in \mathcal{C}_{\mathcal{L}}(\mathcal{X})$ (as a first order formula) can be transformed into an equivalent prenex form $\mathtt{Prenex}(\varphi)$ where connectives for implication and double-implication are eliminated from the matrix. Hence, $\mathtt{Prenex}(\varphi)$ has the form $Q_1\overline{y}^1 Q_2 \overline{y}^2 \dots Q_n \overline{y}^n(\alpha)$ where α is a quantifier-free formula whose symbols are in $\mathcal{L} \cup \{\approx, \neg, \wedge, \vee\}$, each $Q_i \in \{\forall, \exists\}$, $Q_i \neq Q_{i+1}$ for all $1 \leq i < n$, and $\overline{y}^1\overline{y}^2 \dots \overline{y}^n \subseteq \mathtt{Free}(\alpha)$. Note that the solutions of φ, and also of $\mathtt{Prenex}(\varphi)$, are assignments whose domain is $\mathtt{Free}(\varphi)$. The PQE technique, to eliminate the prefix of quantifiers in $\mathtt{Prenex}(\varphi)$, firstly transforms the matrix α into a ct-expression that represents the solutions of α. Constants $\top$ and $\bot$ are used to represent the empty and total set of solutions, respectively. In particular, when $\mathtt{Free}(\varphi)$ is empty, the PQE technique is applied to decide the satisfiability of φ obtaining either $\top$ or $\bot$. First, we represent equations.

Definition 4. *Let* $\alpha \in \mathcal{C}_{\mathcal{L}}(\mathcal{X})$ *be quantifier-free and* $\overline{v} = \mathtt{Free}(\alpha)$*. We associate each equation* $t_1 \approx t_2$ *in* α *to the ct-expression, denoted by* $E^{\overline{v}}_{t_1 \approx t_2}$*, defined as follows*

- $E^{\overline{v}}_{t_1 \approx t_2} = \bot$ *if* t_1 *and* t_2 *are non-unifiable*
- $E^{\overline{v}}_{t_1 \approx t_2} = \overline{v}\sigma\rho$ *where* $\sigma = \mathtt{mgu}(t_1, t_2)$ *and* ρ *is the renaming* $(\overline{v} \cap \mathtt{rgVar}(\sigma)) \leftarrow \overline{w}$ *for some fresh tuple of distinct variables* $\overline{w}$*.*

Example 2. Let us suppose that $Free(\alpha) = (x_1, x_2, x_3, x_4)$ and that α contains the equation $f(x_1, a) \approx f(g(x_2), x_4)$. Then, the most general unifier is $\sigma = \{x_1 \leftarrow g(x_2), x_4 \leftarrow a\}$ and we choose a fresh w_1 to make $\rho = \{x_2 \leftarrow w_1\}$. Therefore, the atomic ct-expression that we associate to the above equation (in α) is $(x_1, x_2, x_3, x_4)\sigma\rho$, which is the tuple of terms $(g(w_1), w_1, x_3, a)$.

[4] where $\setminus, \cup, \cap$ respectively stand for set-theoretic operations of difference, union and intersection.

The following remark states the relation between equations and tuples of terms (exceptionally $\bot$) established in Definition 2.

Remark 1. From Definition 4 it is obvious that the solutions of an equation $t_1 \approx t_2$ are exactly those assignments θ such that $\mathtt{dom}(\theta) = \overline{v}$ and $\overline{v}\theta \in \mathtt{ground}_{\mathcal{L}}(\overline{v}\sigma\rho)$. When $t_1 \approx t_2$ has no solutions, the associated ct-expression is $\bot$ (which represents the empty set of ground terms).

An equation is represented by a tuple of terms, which is just the particular case of $\bar{t} \between \Theta$ (in the above Definition 1) when Θ is the empty set. Our goal is to combine all the equations in a quantifier-free constraint α, according to the first-order connectives, to obtain a ct-expression whose semantics is the set of all solutions of α. Negation (in particular, disequations) gives rise to ct-expressions $\bar{t} \between \Theta$ with non-empty Θ.

Definition 5. *Let $\alpha \in \mathcal{C}_{\mathcal{L}}(\mathcal{X})$ be quantifier-free and let $\overline{v} = \mathtt{Free}(\alpha)$ be a k-tuple for some $k \geq 0$. We denote by $E^{\overline{v}}_{\alpha}$ the ct-expression of arity k that is recursively constructed (from α) as follows: $E^{\overline{v}}_{true} = \top$, $E^{\overline{v}}_{false} = \bot$, $E^{\overline{v}}_{t_1 \approx t_2}$ is the ct-expression of arity k associated in Definition 4 to the equation $t_1 \approx t_2$, $E^{\overline{v}}_{\neg\beta} = \sim E^{\overline{v}}_{\beta}$, $E^{\overline{v}}_{\alpha_1 \wedge \alpha_2} = E^{\overline{v}}_{\alpha_1} \sqcap E^{\overline{v}}_{\alpha_2}$, and $E^{\overline{v}}_{\alpha_1 \vee \alpha_2} = E^{\overline{v}}_{\alpha_1} \sqcup E^{\overline{v}}_{\alpha_2}$.*

Proposition 1. *Let $\alpha \in \mathcal{C}_{\mathcal{L}}(\mathcal{X})$ be quantifier-free and let $\overline{v} = \mathtt{Free}(\alpha)$ be a k-tuple for some $k \geq 0$. For each assignment θ: θ is an $\mathcal{L}$-solution of α if and only if $\overline{v}\theta \in [\![E^{\overline{v}}_{\alpha}]\!]_{\mathcal{L}}$.*

Proof. Structural induction on α. The base step is given by Remark 1. □

Example 3. Let us consider the following constraint α

$$f(x_1, a) \approx f(g(x_2), x_4) \wedge (\ g(x_4) \not\approx g(a) \vee f(g(x_4), g(x_2)) \not\approx f(g(a), x_1)\ )$$

In Example 2 we associate to the equation $f(x_1, a) \approx f(g(x_2), x_4)$ the 4-tuple $(g(w_1), w_1, x_3, a)$. According to Definition 5, the ct-expression $E^{x_1,x_2,x_3,x_4}_{\alpha}$ is

$$(g(w_1), w_1, x_3, a) \sqcap (\ \sim(x_1, x_2, x_3, a) \sqcup \sim(g(w_2), w_2, x_3, a)\ ).$$

We omit sub/super-indices in ct-expressions whenever they are not relevant. We are already able to represent the matrix of any constraint in prenex form. In Section 8, we deal with the quantification of ct-expressions.

5 Weak Normal Form

In this section we define the notion of weak normal form and prove some equivalence results that are useful along the rest of the paper. Let us first introduce the required syntactical conditions.

Definition 6. *We say that a substitution θ is minimal*[5] *if and only if $w \in \mathtt{rgVar}(\theta \setminus \{x \leftarrow w\})$ for all $(x \leftarrow w) \in \theta$. We call θ' the minimal version of θ if θ' is minimal and $\theta = \theta' \cup \rho$ for some renaming ρ.*

[5] in the sense that θ minimally renames variables

Example 4. $\theta_1 = (v_1, v_2) \leftarrow (w_1, f(w_1))$ is minimal, whereas $\theta_2 = (v_1, v_2) \leftarrow (w_1, a)$ is not minimal. The minimal version of θ_2 is $v_2 \leftarrow a$.

Obviously, the minimal version of a minimal substitution is itself. A minimal substitution θ only replaces the variables that are essential to reflect coincidences of (sub-)terms in $\texttt{range}(\theta)$.

Proposition 2. *The ct-expressions $\overline{t} \setminus\!\setminus \Theta$ and $\overline{t} \setminus\!\setminus \Theta'$, such that Θ' consists of the minimal versions of all substitutions in Θ, are $\mathcal{L}$-equivalent for any $\mathcal{L}$.*

Proof. $\texttt{ground}_{\mathcal{L}}(\overline{t}\theta) = \texttt{ground}_{\mathcal{L}}(\overline{t}\theta')$ if θ' is the minimal version of θ. □

Proposition 3. *For any $\mathcal{L}$ and any atomic ct-expression $\overline{t} \setminus\!\setminus \Theta$, if $\varepsilon \in \Theta$ then $\overline{t} \setminus\!\setminus \Theta \equiv_{\mathcal{L}} \bot$.*

Proof. It is a direct consequence of $\texttt{ground}_{\mathcal{L}}(\overline{t}) = \texttt{ground}_{\mathcal{L}}(\overline{t}\varepsilon)$. □

Definition 7. *A ct-expression $E^{\overline{x}} \in \mathcal{E}^k_{\mathcal{L}}(\mathcal{X})$ is in weak normal form (shortly, wnf) if and only if $E^{\overline{x}}$ is either a constant ($\bot$ or $\top$) or a ct-expression of the form*

$$E^{\overline{x}} \;=\; (\overline{t}_1 \setminus\!\setminus \Theta_1)^{\overline{x}} \sqcup \ldots \sqcup (\overline{t}_n \setminus\!\setminus \Theta_n)^{\overline{x}}$$

where $n \geq 1$ and $\theta_i \neq \varepsilon$ and θ_i is minimal for every $(\overline{t}_i \setminus\!\setminus \Theta_i)^{\overline{x}}$ (which has arity k) and every $\theta_i \in \Theta_i$.

In abuse of notation, we sometimes refer to a ct-expression E in wnf as the set of all the atoms $\overline{t}_i \setminus\!\setminus \Theta_i$ that compounds E.

Next, in order to show that any ct-expression can be transformed into weak normal form, we introduce some auxiliary results. First, we show that any atomic ct-expression can be transformed into an equivalent atomic ct-expression in wnf.

Definition 8. *Let $E \in \mathcal{E}^k_{\mathcal{L}}(\mathcal{X})$ be any atomic ct-expression, we define*

$$\texttt{wnf}(E) = \begin{cases} \bot & \text{if either } E = \bot \text{ or } E = \overline{t} \setminus\!\setminus \Theta \text{ and } \varepsilon \in \Theta' \\ \top & \text{if } E = \top \\ \overline{t} \setminus\!\setminus \Theta' & \text{if } E = \overline{t} \setminus\!\setminus \Theta \text{ and } \varepsilon \notin \Theta' \end{cases}$$

where $\Theta' = [\theta' \mid \theta \in \Theta$ and θ' is the minimal version of $\theta]$.

Proposition 4. *For all $\mathcal{L}$ and all atomic $E \in \mathcal{E}^k_{\mathcal{L}}(\mathcal{X})$, $\texttt{wnf}(E) \equiv_{\mathcal{L}} E$.*

Proof. It is a consequence of Definition 8 and Propositions 2 and 3. □

Next, we prove that the complement of an atomic ct-expression in wnf can be transformed into an equivalent ct-expressions in wnf.

Definition 9. *Let $E \in \mathcal{E}^k_{\mathcal{L}}(\mathcal{X})$ be an atomic ct-expressions in wnf, we define*

$$\texttt{wnf}(\sim E) = \begin{cases} \top & \text{if } E = \bot \\ \bot & \text{if } E = \top \\ \texttt{wnf}(\overline{x} \setminus\!\setminus [\overline{x} \leftarrow \overline{t}]) \sqcup \bigsqcup\limits_{\theta \in \Theta} \overline{t}\theta & \text{if } E = \overline{t} \setminus\!\setminus \Theta \end{cases}$$

where $\overline{x}$ is a k-tuple of fresh variables

It is worth to note that whenever $\overline{t}$ is a tuple of distinct variables the above $\mathtt{wnf}(\overline{x} \setminus\!\setminus [\overline{x} \leftarrow \overline{t}])$ is $\overline{x} \setminus\!\setminus [\varepsilon] = \bot$.

Example 5. Let $\mathcal{L} = \{a_{/0}, f_{/1}, g_{/1}\}$ and the ct-expression

$$E = (z_1, f(z_2), g(z_3), z_4) \setminus\!\setminus [\ (z_1, z_2) \leftarrow (a, g(w)), z_4 \leftarrow g(a)\ ].$$

Then, $\mathtt{wnf}(\sim E) = (x_1, x_2, x_3, x_4) \setminus\!\setminus [(x_1, x_2, x_3, x_4) \leftarrow (z_1, f(z_2), g(z_3), z_4)] \sqcup (a, f(g(w)), g(z_3), z_4) \sqcup (z_1, f(z_2), g(z_3), a)$.

Proposition 5. *For all $\mathcal{L}$ and all atomic $E \in \mathcal{E}^k_{\mathcal{L}}(\mathcal{X})$ in wnf, $\mathtt{wnf}(\sim E) \equiv_{\mathcal{L}} \sim E$.*

Proof. It is a direct consequence of Definitions 2 and 9. □

Next, we prove that the intersection of atomic ct-expressions in wnf can be transformed into an equivalent ct-expression in wnf.

Definition 10. *Let E_1, E_2 be two atomic ct-expressions in wnf, we define*

$$\mathtt{wnf}(E_1 \sqcap E_2) = \begin{cases} \bot & \text{if either } E_1 = E_2 = \bot \text{ or } E_1 = \overline{t}_1 \setminus\!\setminus \Theta_1, E_2 = \overline{t}_2 \setminus\!\setminus \Theta_2 \\ & \quad \text{and } \mathtt{mgu}(\overline{t}_1, \overline{t}_2) \text{ does not exist} \\ E_2 & \text{if } E_1 = \top \\ E_1 & \text{if } E_2 = \top \\ \overline{s} \setminus\!\setminus \Theta & \text{if } E_1 = \overline{t}_1 \setminus\!\setminus \Theta_1, E_2 = \overline{t}_2 \setminus\!\setminus \Theta_2, \overline{s} = \mathtt{mgi}(\overline{t}_1, \overline{t}_2) \text{ and} \\ & \quad \Theta = [\theta_{\restriction \mathtt{Var}(\overline{s})} \mid \theta \text{ is the minimal version of } \mathtt{mgu}(\overline{s}, \overline{t}_1\theta_1) \\ & \qquad \text{for } \theta_1 \in \Theta_1 \text{ or } \mathtt{mgu}(\overline{s}, \overline{t}_2\theta_2) \text{ for } \theta_2 \in \Theta_2] \end{cases}$$

Example 6. Let $\mathcal{L} = \{a_{/0}, f_{/1}, g_{/1}\}$ and the two ct-expressions

$$E_1 = (\underbrace{f(v_1), v_1, v_2, v_2}_{\overline{t}_1}) \setminus\!\setminus [\ \underbrace{(v_1, v_2) \leftarrow (w_1, w_1)}_{\theta_1}, \underbrace{v_2 \leftarrow f(w_2)}_{\theta_2}\ ]$$

$$E_2 = (\underbrace{z_1, z_2, g(z_3), z_4}_{\overline{t}_2}) \setminus\!\setminus [\ \underbrace{(z_1, z_2) \leftarrow (a, g(w))}_{\theta'_1}, \underbrace{z_4 \leftarrow g(a)}_{\theta'_2}\ ]$$

Then, $\mathtt{wnf}(E_1 \sqcap E_2) = (f(v_1), v_1, g(z_3), g(z_3)) \setminus\!\setminus [\ z_3 \leftarrow a\ ]$ because

$$\mathtt{mgu}((f(v_1), v_1, g(z_3), g(z_3)), \overline{t}_2\theta'_2)_{\restriction v_1, z_3, v_3} = z_3 \leftarrow a$$

is the only unifier –of the four possible candidates– that exists.

Proposition 6. *For all $\mathcal{L}$ and all atomic $E_1, E_2 \in \mathcal{E}^k_{\mathcal{L}}(\mathcal{X})$ in wnf, $\mathtt{wnf}(E_1 \sqcap E_2) \equiv_{\mathcal{L}} E_1 \sqcap E_2$.*

Proof. By Definitions 2 and 10. □

On the basis of all the above results, we can now prove that any ct-expression can be transformed into an equivalent one in wnf.

Theorem 1. *Let $\mathcal{L}$ be a language. Any ct-expression $E \in \mathcal{E}^k_{\mathcal{L}}(\mathcal{X})$ can be transformed into an $\mathcal{L}$-equivalent ct-expression $\mathtt{wnf}(E) \in \mathcal{E}^k_{\mathcal{L}}(\mathcal{X})$ that is in wnf.*

Proof. By structural induction on E. The base cases are given by Propositions 4, 5 and 6. The inductive step relies on the following two definitions:

$$\mathtt{wnf}(E \sqcap E') = \bigsqcup_{\substack{F \in \mathtt{wnf}(E) \\ F' \in \mathtt{wnf}(E')}} \mathtt{wnf}(F \sqcap F')$$

$\mathtt{wnf}(\sim E) = \mathtt{wnf}(\mathtt{wnf}(\sim E_1) \sqcap \ldots \sqcap \mathtt{wnf}(\sim E_n))$ where $\mathtt{wnf}(E) = E_1 \sqcup \ldots \sqcup E_n$.□

The notion of wnf works different for infinite and finite languages. Whereas for infinite languages checking the emptiness of $\mathtt{wnf}(E)$ suffices to ensure the emptiness of E, that is not enough in the case of finite languages.

Theorem 2. *Let $\mathcal{L}$ be an infinite language. For any ct-expression $E \in \mathcal{E}^k_{\mathcal{L}}(\mathcal{X})$, $[\![E]\!]_{\mathcal{L}} = \emptyset$ if and only if $\mathtt{wnf}(E) = \bot$.*

Proof. Since $\mathcal{L}$ is an infinite language, there are infinitely-many function symbols in $\mathcal{L}$, whereas only a finite subset of them occurs in $\mathtt{wnf}(E)$. Let us consider any $\overline{t} \setminus\!\setminus \Theta \in \mathtt{wnf}(E)$ and any ground instance $\overline{t}\sigma$ of $\overline{t}$ such that $\mathtt{range}(\sigma)$ use symbols from $\mathcal{L}$ that do neither occur in $\overline{t}$ nor in Θ. Obviously $\overline{t}\sigma \in [\![\overline{t} \setminus\!\setminus \Theta]\!]_{\mathcal{L}}$, and, hence, $\overline{t}\sigma \in [\![\mathtt{wnf}(E)]\!]_{\mathcal{L}}$. □

However, when $\mathcal{L}$ is a finite language (that is, $\mathcal{L}$ has finitely-many function symbols), the syntactic conditions required by the wnf notion do not allow to decide whether the set of ground terms represented by a ct-expression is empty. For example, if $\mathcal{L} = \{a_{/0}, g_{/2}, f_{/1}\}$, then the ct-expression

$$E = g(v) \setminus\!\setminus [v \leftarrow a, v \leftarrow f(w), v \leftarrow g(w_1, w_2)]$$

represents the empty set of ground terms although E is in wnf. It is easy to see that none of the substitutions $v \leftarrow a$, $v \leftarrow f(w)$, $v \leftarrow g(w_1, w_2)$ is ε and all of them are minimal. Hence, it is necessary to provide a different condition to check the emptiness of the represented set of ground terms. In Section 6, we introduce a restriction of the notion of weak normal form that overcomes that problem.

6 Normal Form

In this section, we introduce the notion of normal form and prove the equivalence result that, in particular, provides a syntactic check of the emptiness of the set of terms represented by a ct-expression. First,we first extend the notion of term instance to *atomic ct-expressions instance*.

Definition 11. *An atomic ct-expression $E \in \mathcal{E}^k_{\mathcal{L}}(\mathcal{X})$ is in normal form (shortly, nf) if and only if E is in wnf and, if $E = \overline{t} \setminus\!\setminus \Theta$, then every $\theta \in \Theta$ is non-linear. A compound ct-expression $E \in \mathcal{E}^k_{\mathcal{L}}(\mathcal{X})$ is in nf if and only if E is of the form $E_1 \sqcup \ldots \sqcup E_n$ where, for each $1 \leq i \leq n$, E_i is a non-constant atomic ct-expression.*

We are going to explain how to transform any ct-expression that is already in wnf into nf. This transformation is an a adaptation to our framework and our purposes of the algorithm *uncover* presented in [12]. This algorithm was designed to decide whether an implicit generalization (similar to a ct-expression) can be transformed (or not) into an equivalent *explicit generalization* (in our terminology, a union of terms which may, in general, be non-ground).

Definition 12. *The instance of* $\overline{t} \setminus\!\setminus \Theta$ *with a substitution* β *such that* $\mathtt{dom}(\beta) \subseteq \mathtt{Var}(\overline{t})$*, denoted by* $(\overline{t} \setminus\!\setminus \Theta)\beta$*, is the ct-expression* $\mathtt{wnf}((\overline{t} \setminus\!\setminus \Theta) \sqcap (\overline{t}\beta))$.

Since $\mathtt{wnf}(E)$ is either one of constant ct-expressions ($\bot$,$\top$) or the union of (at least one) atomic ct-expressions of the form $\overline{t} \setminus\!\setminus \Theta$ in wnf, it suffices to define the normal form for a $\overline{t} \setminus\!\setminus \Theta$ in wnf.

Definition 13. *Let* $\mathcal{L}$ *be finite. Let* $E = \overline{t} \setminus\!\setminus \Theta \in \mathcal{E}^k_{\mathcal{L}}(\mathcal{X})$ *in wnf. If* E *is in nf (according to Definition 11) then* $\mathtt{nf}_{\mathcal{L}}(E) = E$*. Otherwise, let* $\theta \in \Theta$ *be linear and let* $v \leftarrow f(\overline{s}) \in \theta$*, both arbitrarily chosen.*[6] *We recursively define* $\mathtt{nf}_{\mathcal{L}}(\overline{t} \setminus\!\setminus \Theta)$ *as follows*

$$\bigsqcup \{\mathtt{nf}_{\mathcal{L}}((\overline{t} \setminus\!\setminus \Theta)\sigma_g) \mid g \in \mathcal{L}, g \neq f\} \;\sqcup\; \mathtt{nf}_{\mathcal{L}}((\overline{t} \setminus\!\setminus \Theta')\{v \leftarrow f(\overline{u})\})$$

where

- $\sigma_g = \{v \leftarrow g(\overline{z})\}$ *for some tuple* $\overline{z}$ *of fresh variables, and*
- $\Theta' = (\Theta \setminus \{\theta\}) \cup \{(\theta \setminus \{v \leftarrow f(\overline{s})\}) \cup (\overline{u} \leftarrow \overline{s})\}$ *for some fresh tuple* $\overline{u}$.

It is worthy to note that nf depends on the (finite) language whereas wnf does not depends on the language. The union of ct-expressions (in Definition 13) ranging on the symbols of the language corresponds to the use of the Explosion Rule in traditional quantifier elimination methods for solving equational constraints.

Example 7. Let $\mathcal{L} = \{a_{/0}, f_{/1}\}$ and

$$E = (x_1, x_2) \setminus\!\setminus [\underbrace{(x_1, x_2) \leftarrow (w_1, w_1)}_{\theta_1}, \underbrace{(x_1, x_2) \leftarrow (f(f(w_2)), a)}_{\theta_2}]$$

Suppose that θ_2 and $x_1 \leftarrow f(f(w_2))$ are chosen. Then, $\sigma_a = \{x_1 \leftarrow a\}$ and the former component of $\mathtt{nf}_{\mathcal{L}}(E)$ is $\mathtt{nf}_{\mathcal{L}}(((x_1, x_2) \setminus\!\setminus \{\theta_1, \theta_2\})\{x_1 \leftarrow a\}) = \mathtt{nf}_{\mathcal{L}}(((x_1, x_2) \setminus\!\setminus \{\theta_1, \theta_2\}) \sqcap (a, x_2)) = \mathtt{nf}_{\mathcal{L}}((a, x_2) \setminus\!\setminus x_2 \leftarrow a)$. Further, let $\overline{u} = x_3$ (x_3 is fresh), we have that $(x_1, x_2)\{x_1 \leftarrow f(x_3)\} = (f(x_3), x_2)$ and $\Theta' = \{\theta_1, (x_3, x_2) \leftarrow (f(w_2), a)\}$. Since the instance $((x_1, x_2) \setminus\!\setminus \Theta')\{x_1 \leftarrow f(x_3)\} = (f(x_3), x_2) \setminus\!\setminus [(x_3, x_2) \leftarrow (w_3, f(w_3)), (x_3, x_2) \leftarrow (f(w_2), a)]$, then $\mathtt{nf}_{\mathcal{L}}(E)$ is the union of both $\mathtt{nf}_{\mathcal{L}}((a, x_2) \setminus\!\setminus x_2 \leftarrow a)$ and $\mathtt{nf}_{\mathcal{L}}((f(x_3), x_2) \setminus\!\setminus [(x_3, x_2) \leftarrow (w_3, f(w_3)), (x_3, x_2) \leftarrow (f(w_2), a)])$.

Theorem 3. *Let* $\mathcal{L}$ *be a finite language. For any ct-expression* E, $\mathtt{nf}_{\mathcal{L}}(E) \equiv_{\mathcal{L}} E$*. In particular,* $[\![E]\!]_{\mathcal{L}} = \emptyset$ *if and only if* $\mathtt{nf}_{\mathcal{L}}(E) = \bot$.

Proof. This result follows from Theorem 1 and Definition 13, by induction. □

This provides an alternative proof of correctness for the algorithm *uncover* [12].

[6] Note that, since $\overline{t} \setminus\!\setminus \Theta$ is in wnf but not in nf, it should be at least one $\theta \in \Theta$ that is linear and minimal. This two properties ensure the existence of $v \leftarrow f(\overline{s})$ in θ.

7 Projection

In this section, we provide the projection operation that enables the elimination of quantifiers. Projection depends on whether the language is finite or infinite. Indeed, the case of an infinite language is much simpler.

Definition 14. *Let $\mathcal{L}$ be an infinite language. Let $\bar{t}\ \bar{t}' \setminus\!\setminus \Theta$ be an atomic ct-expression in wnf, $\overline{v} = \mathtt{Var}(\bar{t})$ and $\overline{w} = \mathtt{Var}(\bar{t}')$. We say that Δ is the $\mathcal{L}$-projection of Θ on $\bar{t}$ if and only if Δ exactly contains all the $\theta_{\restriction \overline{v}}$ such that $\theta \in \Theta$, $\mathtt{rgVar}(\theta_{\restriction \overline{v}}) \cap \mathtt{rgVar}(\theta_{\restriction \overline{w}}) = \emptyset$ and either $\theta_{\restriction \overline{w}} = \varepsilon$ or $\theta_{\restriction \overline{w}}$ is a renaming.*

For a finite language, normal form (hence, ER) is used in a controlled way.

Definition 15. *Let $\mathcal{L}$ be a finite language. Let $\bar{t}\ \bar{t}' \setminus\!\setminus \Theta$ be a ct-expression in wnf, $\overline{v} = \mathtt{Var}(\bar{t})$ and $\overline{w} = \mathtt{Var}(\bar{t}')$. Let $\Gamma = \{\theta \mid \theta \in \Theta$ and $\mathtt{rgVar}(\theta_{\restriction \overline{v}}) \cap \mathtt{rgVar}(\theta_{\restriction \overline{w}}) = \emptyset\}$. We say that Δ is the $\mathcal{L}$-projection of Θ on $\bar{t}$ if and only if Δ is the maximal set of substitutions such that:*

(a) *$\theta_{\restriction \overline{v}} \in \Delta$ if $\theta \in \Gamma$ and $\mathtt{nf}_{\mathcal{L}}((\bar{t}\ \bar{t}' \setminus\!\setminus \Theta)\theta_{\restriction \overline{v}}) = \bot$.*

(b) *$\theta_{\restriction \overline{v}}\sigma \in \Delta$ if $\theta \in \Theta \setminus \Gamma$, $\overline{z} = \mathtt{rgVar}(\theta_{\restriction \overline{v}}) \cap \mathtt{rgVar}(\theta_{\restriction \overline{w}})$, $\mathtt{nf}_{\mathcal{L}}((\bar{t}\ \bar{t}' \setminus\!\setminus \Theta)\theta_{\restriction \overline{v}}\sigma) = \bot$, and $\sigma \in \{\mathtt{mgu}(\bar{t}\ \bar{t}'\theta, \overline{r})_{\restriction \overline{z}} \mid (\overline{r} \setminus\!\setminus \Omega) \in \mathtt{nf}_{\mathcal{L}}(\bar{t}\ \bar{t}' \setminus\!\setminus \Gamma)$ for some $\Omega\}$.*

Example 8. Let $\mathcal{L}_1 = \{a_{/0}, f_{/1}\}$ and let E be the ct-expression

$$(\underbrace{f(v_1)}_{\bar{t}}, \underbrace{v_1, v_2, v_2}_{\bar{t}'}) \setminus\!\setminus [\ \underbrace{(v_1, v_2) \leftarrow (w_1, w_1)}_{\theta_1}, \underbrace{v_2 \leftarrow f(w_2)}_{\theta_2}\ ].$$

We are going to check that the $\mathcal{L}_1$-projection of $\{\theta_1, \theta_2\}$ on the 1-tuple $f(v_1)$ consists of a unique substitution $v_1 \leftarrow a$, whereas the $\mathcal{L}_2$-projection of $\{\theta_1, \theta_2\}$ on $f(v_1)$ is empty for any language $\mathcal{L}_2$ that extends $\mathcal{L}_1$ with at least one symbol. For infinite $\mathcal{L}_2$ it is very easy to check that Δ —as defined in Definition 14— is empty. For finite languages, let us first consider $\theta_1 \in \Theta \setminus \Gamma$. We look for a candidate σ such that $\theta_{1\restriction v_1}\sigma$ satisfies conditions of Definition 15(b). Then, $\mathtt{rgVar}(\theta_{1\restriction v_1}) \cap \mathtt{rgVar}(\theta_{1\restriction v_2}) = \{w_1\} \neq \emptyset$, $\mathtt{nf}_{\mathcal{L}_1}((f(v_1), v_1, v_2, v_2) \setminus\!\setminus [\theta_2]) = (f(v_1), v_1, a, a)$, and

$$\sigma = \mathtt{mgu}((f(v_1), v_1, v_2, v_2)\{(v_1, v_2) \leftarrow (w_1, w_1)\}, (f(v_1), v_1, a, a)) = w_1 \leftarrow a.$$

So that, $\mathtt{nf}_{\mathcal{L}_1}(E\theta_{1\restriction v_1}\{w_1 \leftarrow a\}) = \mathtt{nf}_{\mathcal{L}_1}((f(a), a, v_2, v_2) \setminus\!\setminus [v_2 \leftarrow a, v_2 \leftarrow f(w_2)])$, which obviously yields $\bot$. However, $\mathtt{nf}_{\mathcal{L}_2}(E\theta_{1\restriction v_1}\{w_1 \leftarrow a\}) \neq \bot$ for any finite language $\mathcal{L}_2$ that extends $\mathcal{L}_1$. To sum up $\theta_{1\restriction v_1}\sigma = v_1 \leftarrow a$ is in the $\mathcal{L}_1$-projection, but not in the $\mathcal{L}_2$-projection. Let us now consider $\theta_2 \in \Gamma$, since $\theta_{2\restriction v_1} = \varepsilon$, then $\mathtt{nf}_{\mathcal{L}_i}(\bar{t}\ \bar{t}' \setminus\!\setminus [\theta_1\theta_2]\theta_{2\restriction v_1}) = \mathtt{nf}_{\mathcal{L}_i}(\bar{t}\ \bar{t}' \setminus\!\setminus [\theta_1\theta_2]) \neq \bot$ for any $i \in \{1, 2\}$. By Definition 15(a), $\theta_{2\restriction v_1}$ is not in the $\mathcal{L}_i$-projection of $\{\theta_1, \theta_2\}$ on $f(v_1)$, for any $i \in \{1, 2\}$.

Theorem 4. *Let $\mathcal{L}$ be any language. Let $\bar{t}\ \bar{t}' \setminus\!\setminus \Theta$ be an atomic ct-expression in wnf, and let k and k' be the respective arities of $\bar{t}$ and $\bar{t}'$. If Δ is the $\mathcal{L}$-projection of Θ on $\bar{t}$, then for all $\overline{s} \in \mathcal{T}^k_{\mathcal{L}}$:*

$\overline{s} \in [\![\bar{t} \setminus\!\setminus \Delta]\!]_{\mathcal{L}}$ if and only if there exists $\overline{s}' \in \mathcal{T}^{k'}_{\mathcal{L}}$ such that $\overline{s}\ \overline{s}' \in [\![\bar{t}\ \bar{t}' \setminus\!\setminus \Theta)]\!]_{\mathcal{L}}$.

Proof. For infinite $\mathcal{L}$ the result easily follows from the fact that, according to Definition 14, for all $\theta \in \Theta$: $\theta_{\restriction \overline{v}} \in \Delta$ if and only if $[\![\overline{t}\ \overline{t}' \setminus\!\setminus \theta_{\restriction \overline{w}}]\!]_{\mathcal{L}} = \emptyset$.
For finite $\mathcal{L}$, the right-to-left implication is proved by contradiction. That is, we assume that there exists $\overline{s} \in \mathcal{T}_{\mathcal{L}}^{k}$ and $\overline{s}' \in \mathcal{T}_{\mathcal{L}}^{k'}$ such that $\overline{s}\ \overline{s}' \in [\![\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)]\!]_{\mathcal{L}}$ but $\overline{s} \notin [\![\overline{t} \setminus\!\setminus \Delta]\!]_{\mathcal{L}}$. Then, $\overline{s}\ \overline{s}' \in [\![\overline{t}\ \overline{t}']\!]_{\mathcal{L}}$ and $\overline{s}\ \overline{s}' \notin [\![\overline{t}\ \overline{t}'\theta]\!]_{\mathcal{L}}$ for every $\theta \in \Theta$. Further, since $\overline{s} \in [\![\overline{t}]\!]_{\mathcal{L}}$, then $\overline{s} \in [\![\overline{t}\delta]\!]_{\mathcal{L}}$ for some $\delta \in \Delta$. However, since Δ is the $\mathcal{L}$-projection of Θ on $\overline{t}$, every δ in Δ satisfies that $\mathtt{nf}_{\mathcal{L}}((\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)\delta) = \emptyset$. By Theorem 3 and Definition 12, that means $[\![(\overline{t}\ \overline{t}' \setminus\!\setminus \Theta) \sqcap (\overline{t}\ \overline{t}'\delta)]\!]_{\mathcal{L}} = \emptyset$. This is a contradiction because $\overline{s}\ \overline{s}'$ is in such intersection. Note that $\mathtt{dom}(\delta) \subseteq \mathtt{Var}(\overline{t})$ and $\overline{s}' \in [\![\overline{t}']\!]_{\mathcal{L}}$.
For the reverse implication, we consider some fixed $\overline{s} \in \mathcal{T}_{\mathcal{L}}^{k}$ such that $\overline{s} \in [\![\overline{t} \setminus\!\setminus \Delta]\!]_{\mathcal{L}}$. Let $\sigma_0 = \mathtt{mgu}(\overline{s}, \overline{t})$. It is obvious that $[\![(\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)\sigma_0]\!]_{\mathcal{L}} \subseteq [\![(\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)]\!]_{\mathcal{L}}$, hence to ensure the existence of at least one $\overline{s}' \in \mathcal{T}_{\mathcal{L}}^{k'}$ such that $\overline{s}\ \overline{s}' \in [\![\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)]\!]_{\mathcal{L}}$ it suffices to check that $[\![(\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)\sigma_0]\!]_{\mathcal{L}} \neq \emptyset$. Let $\overline{v} = \mathtt{Var}(\overline{t})$, $\overline{w} = \mathtt{Var}(\overline{t}')$, and $\Gamma = \{\theta \mid \theta \in \Theta$ and $\mathtt{rgVar}(\theta_{\restriction \overline{v}}) \cap \mathtt{rgVar}(\theta_{\restriction \overline{w}}) = \emptyset\}$. By Definition 12, the instance $(\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)\sigma_0 = \mathtt{wnf}((\overline{t}\ \overline{t}' \setminus\!\setminus \Theta) \sqcap (\overline{t}\ \overline{t}')\sigma_0)$ which, by Definition 10, is the ct-expression $(\overline{t}\ \overline{t}')\sigma_0 \setminus\!\setminus \Lambda$ where

$$\Lambda = [\text{ minimal version of } \mathtt{mgu}(\overline{s}\ (\overline{t}'\sigma_0), (\overline{t}\ \overline{t}')\theta)_{\restriction \overline{w}} \mid \theta \in \Theta].$$

Note that $\overline{t}\ \overline{t}'\sigma_0 = \overline{s}\ (\overline{t}'\sigma_0)$. We are going to prove, by contradiction, that the two equivalent expressions $(\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)\sigma_0$ and $\overline{s}\ \overline{t}'\sigma_0 \setminus\!\setminus \Lambda$ represent a non-empty set of ground tuples. For that suppose that $[\![(\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)\sigma_0]\!]_{\mathcal{L}} = [\![\overline{s}\ \overline{t}'\sigma_0 \setminus\!\setminus \Lambda]\!]_{\mathcal{L}} = \emptyset$. Then, for all $\overline{s}' \in [\![\overline{t}'\sigma_0]\!]_{\mathcal{L}}$ there exists some $\theta \in \Theta$ such that $\overline{s}\ \overline{s}' \in [\![\overline{t}'\lambda]\!]_{\mathcal{L}}$ where $\lambda = \mathtt{mgu}(\overline{s}\ \overline{t}'\sigma_0, \overline{t}\ \overline{t}'\theta)_{\restriction \overline{w}}$. Now, let us consider some arbitrary fixed $\overline{s}' \in [\![\overline{t}'\sigma_0]\!]_{\mathcal{L}}$, we are going to see that the existence of such $\theta \in \Theta$ leaves to a contradiction. We proceed by cases on θ.
For the first case, suppose that $\theta \in \Gamma$. Since $\overline{s} \in [\![\overline{t} \setminus\!\setminus \Delta]\!]_{\mathcal{L}}$, $\theta_{\restriction \overline{v}}$ can not be in Δ, hence by Definition 15(a), $[\![(\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)\theta_{\restriction \overline{v}}]\!]_{\mathcal{L}} \neq \emptyset$. This contradicts our hypothesis $[\![(\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)\sigma_0]\!]_{\mathcal{L}} = [\![\overline{s}\ \overline{t}'\sigma_0 \setminus\!\setminus \Lambda]\!]_{\mathcal{L}} = \emptyset$. To realize such contradiction, note that any element in $[\![(\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)\theta_{\restriction \overline{v}}]\!]_{\mathcal{L}}$ should belong to $[\![\overline{t}\ \overline{t}'\theta_{\restriction \overline{v}}]\!]_{\mathcal{L}}$ and, the existence of the associated $\lambda = \mathtt{mgu}(\overline{s}\ \overline{t}'\sigma_0, \overline{t}\ \overline{t}'\theta)_{\restriction \overline{w}}$ enables that such element is also in $[\![\overline{s}\ \overline{t}'\sigma_0]\!]_{\mathcal{L}}$, hence it also belongs to $(\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)\sigma_0$ or equivalently to $[\![\overline{s}\ \overline{t}'\sigma_0 \setminus\!\setminus \Lambda]\!]_{\mathcal{L}}$.
For the second case, suppose $\theta \in \Theta \setminus \Gamma$ and name by $\overline{z}$ the non-empty tuple of all common variables in the ranges of $\theta_{\restriction \overline{v}}$ and $\theta_{\restriction \overline{w}}$. If $\mathtt{nf}_{\mathcal{L}}(\overline{t}\ \overline{t}' \setminus\!\setminus \Gamma)$ does not contain some disjunct of the form $\overline{r} \setminus\!\setminus \Omega$ such that $\overline{s}\ \overline{s}' \in [\![\overline{r}]\!]_{\mathcal{L}}$, then there exist some $\theta' \in \Gamma$ such that $\overline{s}\ \overline{s}' \in [\![\overline{t}\overline{t}'\theta']\!]_{\mathcal{L}}$. Then, one gets a contradiction by applying the above first case to $\theta' \in \Gamma$ (as the above $\theta \in \Gamma$). Hence, to complete the proof we can assume the existence of some $\overline{r}$ (which is not necessarily ground) such that $\overline{s}\ \overline{s}' \in [\![\overline{r}]\!]_{\mathcal{L}}$ and, for some Ω, there is a ct-expression $\overline{r} \setminus\!\setminus \Omega$ in the union of atomic ct-expressions given by $\mathtt{nf}_{\mathcal{L}}(\overline{t}\ \overline{t}' \setminus\!\setminus \Gamma)$. Let $\sigma = \mathtt{mgu}(\overline{t}\ \overline{t}'\theta, \overline{r})$. Since $\overline{s} \in [\![\overline{t} \setminus\!\setminus \Delta]\!]_{\mathcal{L}}$, the substitution $\theta_{\restriction \overline{v}}\sigma \notin \Delta$. Therefore, by Definition 15(b), $[\![(\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)\theta_{\restriction \overline{v}}\sigma]\!]_{\mathcal{L}} \neq \emptyset$. This contradict our hypothesis $[\![(\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)\sigma_0]\!]_{\mathcal{L}} = [\![\overline{s}\ \overline{t}'\sigma_0 \setminus\!\setminus \Lambda]\!]_{\mathcal{L}} = \emptyset$, because the existence of $\overline{r}$ and σ ensures that $[\![(\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)\theta_{\restriction \overline{v}}\sigma]\!]_{\mathcal{L}} \subseteq [\![(\overline{t}\ \overline{t}' \setminus\!\setminus \Theta)\sigma_0]\!]_{\mathcal{L}}$. □

8 Projective Quantifier Elimination

In this section, we provide the PQE technique and prove that it is complete.

Definition 16. *Let $\varphi = Q_1\overline{y}^1 Q_2\overline{y}^2 \dots Q_n\overline{y}^n(\alpha) \in \mathcal{C}_{\mathcal{L}}(\mathcal{X})$ (for some $n \geq 0$) in prenex form and let $\overline{x} = \texttt{Free}(\varphi)$ be a k-tuple. We associate to φ the prenex ct-expression of arity k: $P^{\overline{x}}_{\varphi} = Q_1\overline{y}^1 Q_2\overline{y}^2 \dots Q_n\overline{y}^n(E^{\overline{x}\,\overline{y}^1\cdots\overline{y}^n}_{\alpha})$ where $E^{\overline{x}\,\overline{y}^1\cdots\overline{y}^n}_{\alpha}$ is the ct-expression associated to α by Definition 5.*

Note that $P^{\overline{x}}_{\varphi} = E^{\overline{x}}_{\alpha}$ when $n = 0$. We omit the sub/super-index in (prenex) ct-expressions whenever they are not relevant. Let us extend the semantic function to prenex ct-expressions.

Definition 17. *Let $\overline{v}$ be a k-tuple, $\overline{y}$ an m-tuple, and $P^{\overline{v}\,\overline{y}}$ any prenex ct-expression of arity $k+m$. We define*

$$[\![\exists\overline{y}(P^{\overline{v}\,\overline{y}})]\!]_{\mathcal{L}} = \{\overline{s} \mid \overline{s} \in \mathcal{T}^k_{\mathcal{L}} \text{ and } \overline{s}\,\overline{r} \in [\![P^{\overline{v}\,\overline{y}}]\!]_{\mathcal{L}} \text{ for some } \overline{r} \in \mathcal{T}^m_{\mathcal{L}}\}$$
$$[\![\forall\overline{y}(P^{\overline{v}\,\overline{y}})]\!]_{\mathcal{L}} = \{\overline{s} \mid \overline{s} \in \mathcal{T}^k_{\mathcal{L}} \text{ and } \overline{s}\,\overline{r} \in [\![P^{\overline{v}\,\overline{y}}]\!]_{\mathcal{L}} \text{ for all } \overline{r} \in \mathcal{T}^m_{\mathcal{L}}\}$$

Proposition 7. *Let $P^{\overline{x}}_{\varphi}$ be associated to φ as in Definition 16. For all assignment θ: θ is an $\mathcal{L}$-solution of φ if and only if $\overline{x}\theta \in [\![P^{\overline{x}}_{\varphi}]\!]_{\mathcal{L}}$.*

Proof. An easy induction on the length n of the prefix of quantifiers. The case $n = 0$ trivially follows from Proposition 1. The induction hypothesis along with Definition 17 suffice to complete the proof. □

In QE techniques, it is usual to reduce the elimination of the universal quantifier to the existential one, using the classical equivalence $\forall\overline{x}(\varphi) \equiv \neg\exists\overline{x}(\neg\varphi)$.

Proposition 8. *For every atomic ct-expression $E^{\overline{v}\,\overline{y}}$, there exists $n \geq 1$ atomic ct-expressions $E^{\overline{v}}_1, \dots, E^{\overline{v}}_n$ such that $\forall\overline{y}(E^{\overline{v}\,\overline{y}}) \equiv_{\mathcal{L}} E^{\overline{v}}_1 \sqcup \cdots \sqcup E^{\overline{v}}_n$.*

Proof. First, by Definition 17 and Theorem 1, $[\![\forall\overline{y}(E^{\overline{v}\,\overline{y}})]\!]_{\mathcal{L}} = [\![\sim(\exists\overline{y}(\sim E^{\overline{v}\,\overline{y}}))]\!]_{\mathcal{L}} = [\![\sim(\exists\overline{y}(\texttt{wnf}(\sim E^{\overline{v}\,\overline{y}})))]\!]_{\mathcal{L}}$. Now, let $F^{\overline{v}\,\overline{y}}_1, \dots, F^{\overline{v}\,\overline{y}}_m$ the atomic ct-expression such that $[\![\sim(\exists\overline{y}(\texttt{wnf}(\sim E^{\overline{v}\,\overline{y}})))]\!]_{\mathcal{L}} = [\![\sim(\exists\overline{y}(F^{\overline{v}\,\overline{y}}_1 \sqcup \cdots \sqcup F^{\overline{v}\,\overline{y}}_m))]\!]_{\mathcal{L}}$. Then, $\forall\overline{y}(E^{\overline{v}\,\overline{y}}) \equiv_{\mathcal{L}} \sim(\exists\overline{y}(F^{\overline{v}\,\overline{y}}_1) \sqcup \cdots \sqcup \exists\overline{y}(F^{\overline{v}\,\overline{y}}_m))$. By Theorem 4, there exists m atomic ct-expression $G^{\overline{v}}_i$ ($i \in \{1, \dots, m\}$) such that $\exists\overline{y}(F^{\overline{v}\,\overline{y}}_i) \equiv_{\mathcal{L}} G^{\overline{v}}_i$. Therefore, by Theorem 1, there exists $n \geq 1$ atomic ct-expressions $E^{\overline{v}\,\overline{y}}_1, \dots, E^{\overline{v}\,\overline{y}}_n$ such that $\forall\overline{y}(E^{\overline{v}\,\overline{y}}) \equiv_{\mathcal{L}} \texttt{wnf}(\sim(G^{\overline{v}}_1 \sqcup \cdots \sqcup G^{\overline{v}}_m)) \equiv_{\mathcal{L}} E^{\overline{v}}_1 \sqcup \cdots \sqcup E^{\overline{v}}_n$. □

Theorem 5. *Let $\mathcal{L}$ be any language. For all $n \geq 0$ and all prenex ct-expression $P^{\overline{x}} = Q_1\overline{y}^1 Q_2\overline{y}^2 \dots Q_n\overline{y}^n(E^{\overline{x}\,\overline{y}^1\cdots\overline{y}^n})$, there exists $m \geq 1$ atomic ct-expressions $E^{\overline{x}}_i$ in wnf such that $P^{\overline{x}}_{\varphi}$ and $E^{\overline{x}}_1 \sqcup \cdots \sqcup E^{\overline{x}}_m$ are $\mathcal{L}$-equivalent.*

Proof. By induction on the length n of the prefix of quantifiers in P_{φ}. The case $n = 0$ follows from Theorem 1. For the inductive step, we can assume that $E^{\overline{x}\,\overline{y}^1\cdots\overline{y}^{n-1}\overline{y}^n}$ is an atomic ct-expression in wnf. Such assumption relies on Theorem 1 along with the reduction of $\forall$ to $\exists$ by two complements, and

the distributive property of $\exists$ over $\sqcup$. By Theorem 4 or Proposition 8, respectively depending on whether Q_n is $\exists$ or $\forall$, we have $m \geq 1$ ($m = 1$ for $Q_n = \exists$) atomic ct-expressions $F_i^{\overline{x}\,\overline{y}^1 \ldots \overline{y}^{n-1}}$ such that $Q_n \overline{y}^n (E^{\overline{x}\,\overline{y}^1 \ldots \overline{y}^{n-1}\overline{y}^n})$ is $\mathcal{L}$-equivalent to $F_1^{\overline{x}\,\overline{y}^1 \ldots \overline{y}^{n-1}} \sqcup \cdots \sqcup F_m^{\overline{x}\,\overline{y}^1 \ldots \overline{y}^{n-1}}$. Therefore $P_\varphi^{\overline{x}}$ is $\mathcal{L}$-equivalent to $Q_1 \overline{y}^1 Q_2 \overline{y}^2 \ldots Q_{n-1} \overline{y}^{n-1} (F_1^{\overline{x}\,\overline{y}^1 \ldots \overline{y}^{n-1}} \sqcup \cdots \sqcup F_m^{\overline{x}\,\overline{y}^1 \ldots \overline{y}^{n-1}})$. The induction hypothesis applied to the latter prenex ct-expression gives the desired ct-expressions $E_i^{\overline{x}}$ such that $E_1^{\overline{x}} \sqcup \cdots \sqcup E_m^{\overline{x}}$ is $\mathcal{L}$-equivalent to the latter and hence also to $P_\varphi^{\overline{x}}$. $\square$

Corollary 1. *Let $\mathcal{L}$ be any language. For all $\varphi \in \mathcal{C}_\mathcal{L}(\mathcal{X})$ such that $\mathtt{Free}(\varphi) = \overline{x}$, there exists $m \geq 1$ atomic ct-expressions $E_i^{\overline{x}}$ in wnf such that: $\mathcal{H}_\mathcal{L} \models \varphi\theta$ if and only if $\overline{x}\theta \in [\![E_1^{\overline{x}} \sqcup \cdots \sqcup E_m^{\overline{x}}]\!]_\mathcal{L}$ for every assignment θ such that $\mathtt{dom}(\theta) = \overline{x}$.*

Example 9. Continuing Example 8, the following prefix ct-expression

$$\exists y_1 y_2 y_3 (((f(v_1), v_1, v_2, v_2)) \between [(v_1, v_2) \leftarrow (w_1, w_1), v_2 \leftarrow f(w_2)])^{x\ y_1 y_2 y_3})$$

is $\mathcal{L}_1$-equivalent to $(f(v_1) \between [v_1 \leftarrow a])^x$ and $\mathcal{L}_2$-equivalent to $f(v_1)^x$.

9 Conclusion

We have presented a complete QE technique for equational constraint solving that, in particular, provides a decision procedure for the free equality theory of any (finite or infinite) language. Complete QE techniques for solving quantified first-order formulas (in decidable theories) is one of the main current challenges in automated reasoning. A remarkable application of them is the improvement of the quantifier reasoning of SMT solvers, especially because its valuable application for solving quantified verification conditions (see e.g. [11]). As a complete decision procedure, PQE provides a good method for proving quantified verification conditions on algebraic datatypes whose model are the (free) term algebra. Moreover, as a solutions generator, PQE can be applied in the improvement of programming languages capabilities (such as e.g. constructive negation). For that purposes efficiency is essential. Using our terminology, traditional methods (see [5, 8, 6, 10, 13, 14]) keep the matrix in nf, whereas in the PQE technique the matrix is kept in wnf. This difference is crucial because the transformation into nf involves ER applications, whereas wnf does not. Unlike traditional methods, PQE uses nf not to explode the matrix but only to calculate the projection. Moreover, such calculation sometimes requires the computation of the nf of some subexpression (of the matrix), though most often only need to decide whether a nf is equivalent to $\bot$. For that, one does not need to calculate the whole expression in nf. Actually, as soon as a solution appears one can stop reporting that the nf is not $\bot$. Hence, minimal explosion is performed. Experimental results using a prototype and formal complexity analysis are near future work. We also plan to study how to obtain similar algebraic-style QE procedures for other decidable (equality) theories, e.g. the theory of the algebra of (infinite) rational terms.

References

1. J. Álvez and P. Lucio. Equational constraint solving using quasi-solved forms. In M. Kohlhase, editor, *8th International workshop on Unification, UNIF'04, at IJCAR 2004. Cork, Ireland. July 2004*, pages 61–79, 2004.
2. J. Álvez and P. Lucio. Equational constraint solving via a restricted form of universal quantification. In J. Dix and S. J. Hegner, editors, *Foundations of Information and Knowledge Systems, Proceedings of the 4th International Symposium, FoIKS 2006, Budapest, Hungary, February 14-17, 2006*, volume 3861 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 2006.
3. J. Álvez and P. Lucio. A new proposal of quasi-solved form for equality constraint solving. *Electr. Notes Theor. Comput. Sci.*, 206:23–40, 2008.
4. W. L. Buntine and H.-J. Bürckert. On solving equations and disequations. *J. ACM*, 41(4):591–629, 1994.
5. A. Colmerauer and T.-B.-H. Dao. Expressiveness of full first order constraints in the algebra of finite or infinite trees. In R. Dechter, editor, *CP*, volume 1894 of *Lecture Notes in Computer Science*, pages 172–186. Springer, 2000.
6. H. Comon. Disunification: A survey. In J. Lassez and G. Plotkin, editors, *Computational Logic - Essays in Honor of Alan Robinson*, pages 322–359, 1991.
7. H. Comon and C. Kirchner. Constraint solving on terms. In H. Comon, C. Marché, and R. Treinen, editors, *CCL*, volume 2002 of *Lecture Notes in Computer Science*, pages 47–103. Springer, 1999.
8. H. Comon and P. Lescanne. Equational problems and disunification. *J. Symb. Comput.*, 7(3/4):371–425, 1989.
9. K. J. Compton and C. W. Henson. A uniform method for proving lower bounds on the computational complexity of logical theories. *Ann. Pure Appl. Logic*, 48(1):1–79, 1990.
10. K. Djelloul. A full first-order constraint solver for decomposable theories. In S. Autexier, J. A. Campbell, J. Rubio, V. Sorge, M. Suzuki, and F. Wiedijk, editors, *AISC/MKM/Calculemus*, volume 5144 of *Lecture Notes in Computer Science*, pages 93–108. Springer, 2008.
11. Y. Ge, C. Barrett, and C. Tinelli. Solving quantified verification conditions using satisfiability modulo theories. In F. Pfenning, editor, *CADE*, volume 4603 of *Lecture Notes in Computer Science*, pages 167–182. Springer, 2007.
12. J.-L. Lassez and K. Marriott. Explicit representation of terms defined by counter examples. *J. Autom. Reasoning*, 3(3):301–317, 1987.
13. M. J. Maher. Complete axiomatizations of the algebras of finite, rational and infinite trees. In *LICS '1988: Proceedings of the Third Annual Symp. on Logic in Computer Science, 5-8 July 1988, Edinburgh, Scotland, UK*, pages 348–357. IEEE Computer Society, 1988.
14. A. I. Malcev. Axiomatizable classes of locally free algebras. In B. F. Wells, editor, *The Metamathematics of Algebraic Systems (Collected Papers: 1936-1967)*, volume 66 of *Studies in Logic and the Foundations of Mathematics*, chapter 23, pages 262–281. North-Holland, 1971.
15. R. Pichler. Explicit versus implicit representations of subsets of the herbrand universe. *Theor. Comput. Sci.*, 290(1):1021–1056, 2003.
16. R. Pichler. On the complexity of equational problems in cnf. *J. Symb. Comput.*, 36(1-2):235–269, 2003.
17. T. Rybina and A. Voronkov. A decision procedure for term algebras with queues. *ACM Trans. Comput. Log.*, 2(2):155–181, 2001.

18. S. G. Vorobyov. An improved lower bound for the elementary theories of trees. In M. A. McRobbie and J. K. Slaney, editors, *CADE-13: Proceedings of the 13th International Conference on Automated Deduction*, volume 1104 of *Lecture Notes in Computer Science*, pages 275–287, London, UK, 1996. Springer-Verlag.