Decidability of Term Algebras Extending Partial Algebras

Bakhadyr Khoussainov and Sasha Rubin

Department of Computer Science, University of Auckland, New Zealand

Abstract. Let $\mathcal{A}$ be a partial algebra on a finite signature. We say that $\mathcal{A}$ has decidable query evaluation problem if there exists an algorithm that given a first order formula $\phi(\bar{x})$ and a tuple $\bar{a}$ from the domain of $\mathcal{A}$ decides whether or not $\phi(\bar{a})$ holds in $\mathcal{A}$. Denote by $E(\mathcal{A})$ the total algebra freely generated by $\mathcal{A}$. We prove that if $\mathcal{A}$ has a decidable query evaluation problem then so does $E(\mathcal{A})$. In particular, the first order theory of $E(\mathcal{A})$ is decidable. In addition, if $\mathcal{A}$ has elimination of quantifiers then so does $E(\mathcal{A})$ extended by finitely many definable selector functions and tester predicates. Our proof is a refinement of the quantifier elimination procedure for free term algebras. As an application we show that any finitely presented term algebra has a decidable query evaluation problem. This extends the known result that the word problem for finitely presented term algebras is decidable.

1 Introduction

The (free) algebra of terms plays an important role in many areas of computer science and algebra. It is the unique universal object that can be mapped homomorphically onto any given algebra (over a fixed signature). This provides a bijection between congruences of the term algebra and the class of all algebras (over that signature). Since finite trees can be represented as terms, the algebra of terms appears in computer science. For instance, in automata theory regular languages of trees can be identified with congruences of finite index of the algebra of terms. In logic programming, terms are used as basic objects in unification algorithms. In modern object oriented programming many data structures are stored and manipulated as terms. Other applications are in constraint databases, pattern matching, type theory and the theory of algebraic specification.

In logic and computability, the term algebra attracts much attention due to the fact that its first order theory is decidable. This was first proved by Mal′cev in [15]. His proof uses the method of elimination of quantifiers. This result has been reproved and extended by others in different settings. Rybina and Voronkov in [17] applied the method of elimination of quantifiers to show that the term algebra with queues has a decidable first order theory. Korovin and Voronkov in [8] prove that the existential fragment of the term algebra with the Knuth-Bendix ordering is decidable. Manna, Zhang and Sipma in [16] prove that the term algebra with the length function for terms has a decidable theory using the elimination of quantifiers. They also prove that the theory of the term algebra

© Springer-Verlag Berlin Heidelberg 2005

that involves k-alternations of quantifiers, regardless of the total number of the quantifiers, is at most k-fold exponential. Vorobyov in [18] and Compton and Henson in [4] prove that that the decision problem for the first order theory of the term algebra has a non-elementary lower bound.

In this paper, we extend the decidability result for term algebras to a much more general setting. A *partial algebra* (or simply an *algebra*) $\mathcal{A}$ is a structure whose basic operations are partial functions on A. In case all the functions are total on A, we may stress this and call $\mathcal{A}$ a *total algebra*. Partial algebras naturally occur when one restricts the basic operations of a total algebra $\mathcal{A}$ to some $B \subset A$ that is not closed under the basic operations. In this case the value of a term in $\mathcal{B}$ may be undefined, in which case $\mathcal{B}$ is a partial algebra and not a total algebra.

The algebra freely generated by $\mathcal{A}$, called the *free total extension* of $\mathcal{A}$ and written as $E(\mathcal{A})$, is the total algebra generated by $\mathcal{A}$ with a sort of universal mapping property: every homomorphism from $\mathcal{A}$ into any total algebra $\mathcal{B}$ can be extended to a homomorphism from $E(\mathcal{A})$ into $\mathcal{B}$. We remark that $E(\mathcal{A})$ is a natural object in universal algebra (see [9, Section 28]) as well as in computer science. For instance in the theory of algebraic specification partial algebras are a natural way of treating errors such as division by zero (see [1]); here $E(\mathcal{A})$ are used as models of specifications. The algebra $E(\mathcal{A})$ is also used in providing non-standard models of Clarke's Axioms (see [14]). Also [11] uses $E(\mathcal{A})$ to extend the Myhill-Nerode theorem with 'finitely generated congruence' instead of 'congruence of finite index'.

We say that an algebra $\mathcal{A}$ has a *decidable query evaluation problem* if there exists an algorithm that given a first order formula $\phi(\bar{x})$ and a tuple $\bar{a}$ from the domain of $\mathcal{A}$ decides whether or not $\phi(\bar{a})$ holds in $\mathcal{A}$. In particular if $\mathcal{A}$ has a decidable query evaluation problem then its first order theory is decidable. Our main result states:

If $\mathcal{A}$ has a decidable query evaluation problem then so does $E(\mathcal{A})$.

In particular, the first order theory of $E(\mathcal{A})$ is decidable. In addition, if $\mathcal{A}$ has elimination of quantifiers then so does $E(\mathcal{A})$ extended by finitely many definable selector functions and tester predicates. These results are proved by rewriting a formula Φ of $E(\mathcal{A})$ into a formula Φ' that can be evaluated in A so that $E(\mathcal{A}) \models \Phi$ if and only if $\mathcal{A} \models \Phi'$. The technique is a refinement of the quantifier elimination procedure in [16].

A *finitely presented term algebra* is the quotient of a free term algebra by finitely many ground term equations. The word problem for finitely presented term algebras is decidable [10]. As a corollary to our main result we have, we feel, a clean proof that the first order theory of a finitely presented algebra is decidable [3].

We place our result, that decidability is preserved by the free total extension of $\mathcal{A}$, in the realm of other constructions that preserve decidability. Direct product, disjoint union and under suitable conditions the ω-product preserve decidability of the query evaluation problem. Another example is the construction of the tree-like unfolding from [19] that preserves the monadic second order theory. Finally we mention a construction from [12] that resembles the one in this paper. There a purely relational structure $\mathcal{A}$ is lifted by first extending the signature by adding new function symbols from a functional signature Σ. One then considers the Σ-term algebra generated by constants from A. Finally, the relations of the algebra $\mathcal{A}$ are lifted in a natural way to the domain of the terms. The resulting structure is called the Σ-term power of $\mathcal{A}$. It is proved that if $\mathcal{A}$ has decidable first order theory then so does the Σ-term power of $\mathcal{A}$.

Here is a brief outline of this paper. The next section gives basic definitions, examples, and facts about structures with decidable query evaluation problem. Section 3 presents a formal definition of the free total extensions for partial algebras and some of the properties of these extensions. Section 4 is devoted to proving the main result of this paper. The final section applies the main result to show that each finitely presented term algebra has decidable query evaluation problem.

2 Structures with Decidable Query Evaluation Problem

A *structure* consists of a domain A of elements, and basic operations $f^A, g^A \ldots$ on A and relations $P^A, Q^A, \ldots$ on A. The *signature* of $\mathcal{A}$ is $(f, g, \ldots, P, Q, \ldots)$. We view constants as operations of arity 0. In general, the operations f^A may be partial functions.

Convention: For the purpose of this paper, the domain A is a decidable set from a decidable domain such as the strings over a finite alphabet, or ground terms over a finite ranked alphabet, or natural numbers. In particular all the structures we consider are countable.

Definition 1. The query evaluation problem *for the structure $\mathcal{A}$ is the set $QEP(\mathcal{A})$ of all pairs $(\phi(\bar{x}), \bar{a})$ such that $\mathcal{A} \models \phi(\bar{a})$ where $\phi(\overline{x})$ is a first order formula of $\mathcal{A}$ and $\bar{a}$ is a tuple of elements from A. If there is an algorithm deciding $QEP(\mathcal{A})$ then we say that $\mathcal{A}$* **has decidable query evaluation problem**.

We will abbreviate the phrase 'query evaluation problem' as QEP.

In other words, $\mathcal{A}$ has decidable QEP means that its elementary diagram with constants naming every element in A is decidable. We remark that this definition can be extended to other logics besides first order. Here are several examples of structures with decidable QEP.

Example 1. Every finite structure has decidable QEP.

Recall that a theory T is a set of sentences closed under deduction. A theory T admits effective quantifier elimination if there is an effective procedure that transforms a formula into an equivalent (in T) quantifier free formula. Say that a structure $\mathcal{A}$ admits effective quantifier elimination if its first order theory does. Specific examples include algebraically closed fields, vector spaces over finite fields, term algebras extended with selector functions, etc (see [5] for examples).

Example 2. If $\mathcal{A}$ admits effective elimination of quantifiers and the domain and basic operations of $\mathcal{A}$ are decidable, then $\mathcal{A}$ has decidable QEP.

Automatic structures are relational structures whose predicates are recognised by synchronous finite automata. For precise definitions see [7].

Example 3. Automatic structures have decidable QEP.

For issues on complexity of query evaluation problems for automatic structures see [2], [13], and for examples of automatic structures see [6], [7]. For these structures definable relations are, in fact, recognised by finite automata.

Example 4. Every decidable consistent theory T has a model with decidable QEP.

Indeed, the classical Henkin construction can be made effective; and the constructed model has decidable QEP.

Example 5. If $\mathcal{A}$ has decidable QEP then every structure that is first order definable in $\mathcal{A}$ also has decidable QEP.

Example 6. Let $\mathcal{A}$ and $\mathcal{B}$ be structures of a signature Σ with decidable QEP. Then the following structures have decidable QEP: the product $\mathcal{A}\times\mathcal{B}$ (Feferman, Vaught, 59); the disjoint union $\mathcal{A}\oplus\mathcal{B}$, where the domain of $\mathcal{A}\oplus\mathcal{B}$ is the union of A and B and for each $P\in\Sigma$ the predicate $P^{\mathcal{A}\oplus\mathcal{B}}$ is the union $P^{\mathcal{A}}\cup P^{\mathcal{B}}$ and there is a unary predicate for A.

3 Free Total Extensions of Partial Algebras

Partial Algebras: A *partial algebra*, or simply an *algebra*, is a structure $\mathcal{A} = (A, f^A, g^A, \ldots, h^A)$ consisting of a domain of elements A and finitely many partial functions on A. Constants are viewed as functions of arity 0. Incase all the functions are total we call the structure a *total algebra.* The signature of $\mathcal{A}$ is $(f, g, \ldots, h)$ and is called a *functional signature* since it does not contains symbols for relations. Note that a term, such as $f(g(a, b))$, may not have a value in A, in which case we say that *(the value of) the term is undefined in* $\mathcal{A}$. All structures implicitly have the symbol $=$ for equality. In a partial algebra $\mathcal{A}$ two terms s and t are defined to be equal, written $s = t$, if they are both defined in $\mathcal{A}$ and have the same value in $\mathcal{A}$ (so called existential equality). Tuples of elements a_i of A and tuples of variables x_i are denoted by $\overline{a}$ and $\overline{x}$ respectively.

Term Algebras: Let Σ be a functional signature and C a non-empty domain. Then the set of *ground terms* over C, written $GT_\Sigma(C)$ or simply $GT(C)$, is defined inductively as follows:

- Every element of C is a ground term.
- If f is an n-ary symbol from Σ, and $\overline{t}$ is an n-tuple of ground terms, then $f(\overline{t})$ is a ground term.

The *free term algebra generated by* C is $(GT(C), (f)_{f \in \Sigma})$ where the value of f on $\overline{t}$ is defined as the ground term $f(\overline{t})$. It is a total algebra.

The *depth* of ground terms is defined as follows: terms in C have depth 0; if $f(t_1, \cdots, t_k)$ is not in C, then its depth is 1 more than the maximum of the depths of the t_i.

Free Total Extensions: Let $\mathcal{A}$ be a partial algebra on signature Σ. We define what it means to extend $\mathcal{A}$ to a total algebra $E(\mathcal{A})$ in the free-est possible way. First we give an explicit construction and then the usual one in terms of homomorphisms.

For any ground term $t \in GT(A)$ define its *canonical form* with respect to the structure $\mathcal{A}$, written $t^{(c)}_{\mathcal{A}}$ or simply $t^{(c)}$, by induction as follows.

- If $t = a$ and $a \in A$ then define $t^{(c)}$ as a.
- Assume $t = f(t_1, \ldots, t_n)$, and $t_1^{(c)}, t_2^{(c)}, \ldots, t_n^{(c)}$ have been defined. If each $t_i^{(c)}$ is in A and the value $f(t_1^{(c)}, t_2^{(c)}, \ldots t_n^{(c)})$ is defined in $\mathcal{A}$ and equals $b \in A$ then define $t^{(c)}$ as b. Otherwise $t^{(c)}$ is defined as $f(t_1^{(c)}, t_2^{(c)}, \ldots t_n^{(c)})$.

The *algebra of canonical terms with respect to* $\mathcal{A}$ is the total algebra over signature Σ, whose domain is the set of canonical terms $t^{(c)}$ for $t \in GT(A)$, and for which the value of f on $\overline{t}$, with t_i canonical, is simply $f(\overline{t})^{(c)}$.

As we will see in a moment, this algebra is isomorphic to the free total extension of $\mathcal{A}$. First we need some definitions (see [9][section 13]).

Let $\mathcal{C}$ and $\mathcal{B}$ be partial algebras over the same signature. A *homomorphism from* $\mathcal{C}$ *into* $\mathcal{B}$ is a total mapping $h : \mathcal{C} \to \mathcal{B}$ so that whenever $\bar{a} \in C$, $f \in \Sigma$, and $f^{\mathcal{C}}(\bar{a})$ is defined then $f^{\mathcal{B}}(h(\bar{a}))$ is defined and is equal to $hf^{\mathcal{C}}(\bar{a})$. For $B \subset C$, say that $\mathcal{C}$ *extends* $\mathcal{B}$ if for every $\overline{b}$, $f^{\mathcal{B}}(\overline{b})$ is defined and equal to $b \in B$ if and only if $f^{\mathcal{C}}(\overline{b})$ is defined and equal to $b \in B$. Note that this allows the possibility that $f^{\mathcal{C}}(\overline{b})$ is defined (and not in B) while $f^{\mathcal{B}}(\overline{b})$ is undefined. Finally recall that a total algebra $\mathcal{C}$ is *generated* by a set $X \subset C$ if $\mathcal{C}$ is the smallest (under $\subset$) total algebra containing every total subalgebra $\mathcal{B}$ with $X \subset B$. In this case every element of $\mathcal{C}$ is equal to $t(\overline{b})$ for some term t and some tuple of elements $\overline{b}$ from X.

Define a *free total extension of* $\mathcal{A}$ (compare [11]), written $E(\mathcal{A})$, as satisfying the following properties:

1. $E(\mathcal{A})$ is a total algebra extending $\mathcal{A}$.
2. $E(\mathcal{A})$ is generated by the elements of A.
3. Every homomorphism h from $\mathcal{A}$ into a total algebra $\mathcal{B}$ can be extended to a homomorphism of $E(\mathcal{A})$ into $\mathcal{B}$.

Note that $E(\mathcal{A})$ is unique. Here are some examples.

Example 7. Let $\mathcal{C}$ be the partial algebra $(C, (f)_{f \in \Sigma})$ where the f's are undefined everywhere. Then $E(\mathcal{C})$ is the free term algebra generated by C.

Example 8. Let $\mathcal{A}$ be a finite partial algebra. Then $E(\mathcal{A})$ is isomorphic to the quotient of $GT(A)$ by finitely many ground term equations [11].

Proposition 1. *The algebra of canonical terms with respect to $\mathcal{A}$ is isomorphic to $E(\mathcal{A})$.*

Convention. In what follows we will work on this algebra of canonical terms directly, instead of using the abstract definition of $E(\mathcal{A})$. Moreover, in order to distinguish the original domain A of $E(\mathcal{A})$ we introduce a unary predicate A to the language.

4 Main Theorem

The main result is the following theorem.

Theorem 1. *If a partial algebra $\mathcal{A}$ has decidable QEP then so does its free total extension $E(\mathcal{A})$.*

The proof is a mixture of the quantifier elimination procedure for free term algebras and the decision procedure of $\mathcal{A}$. The basic idea is to inductively remove existential quantifiers over variables specified to be outside of A.

We extend the signature $\Sigma \cup \{A\}$ of $E(\mathcal{A})$ to include new operations and relations. To avoid confusion, the operations of Σ are called *constructors*. The new operations consist of:

– unary *selector functions* f_i for every constructor f and $i \leq arity(f)$, and
– unary *tester predicates* IS_f for every constructor f.

From now on we work in this extended signature unless specified otherwise. Sequences of selector functions will be denoted by $L, M, \ldots$ and $L_i, M_i, \ldots$ for $i \in \mathbb{N}$.

Semantics: These new operations have the following semantics. Let t be a canonical term. If $t \notin A$ then there is a unique constructor f and canonical terms $\overline{s}$ so that $t = f(\overline{s})$. In this case, define $f_i(t)$ as s_i (for $i \leq arity(f)$) and define $\mathrm{IS}_f(t)$ as $\top$. Also define $g_i(t) = t$ and $\mathrm{IS}_g(t)$ as $\perp$ for every constructor $g \neq f$ (for $i \leq arity(g)$). On the other hand, if $t \in A$, then define $g_i(t)$ as t and $\mathrm{IS}_g(t)$ as $\perp$ for every constructor g (for $i \leq arity(g)$. Finally $A(t)$ holds if and only if $t \in A$. Note the selector functions f_i and the tester predicates IS_f are definable in the language of $E(\mathcal{A})$.

Terms: A term t over the extended signature of $E(\mathcal{A})$ with free variables amongst $\overline{x}$ will be written $t(\overline{x})$. The expression

$$t[x_1/s_1(\overline{v}), \cdots, x_k/s_k(\overline{v})]$$

denotes the term t where each of the mentioned x_i from $\overline{x}$ has been replaced with the corresponding term $s_i(\overline{v})$. For instance, if $t = f(h_2(x_1), x_2)$, then $t[x_1/g(v_1)]$ is the term $f(h_2(g(v_1)), x_2)$.

Literals: To be sure, every literal in the language of $E(\mathcal{A})$ is either an equation of terms $t = s$, a disequation of terms $t \neq s$, or a tester predicate applied to a term $\mathrm{IS}_f(t)$.

We list some basic properties of $E(\mathcal{A})$ that will be used implicitly in showing the correctness of the the algorithm provided in the next section. Here and unless specified otherwise, equivalence is in $E(\mathcal{A})$.

Lemma 1. *The algebra $E(\mathcal{A})$ satisfies the following properties:*

1. $t_1 = t_2$ *implies that* $t_1 \in A$ *if and only if* $t_2 \in A$.
2. $f(\overline{t}) \notin A \wedge f(\overline{s}) \notin A$ *implies that* $f(\overline{t}) = f(\overline{s})$ *is equivalent to* $\bigwedge t_i = s_i$.
3. $f(\overline{t}) \notin A \wedge f(\overline{s}) \notin A$ *implies that* $f(\overline{t}) \neq f(\overline{s})$ *is equivalent to* $\bigvee t_i \neq s_i$.
4. *for* $f \neq g$, $[f(\overline{t}) \notin A \wedge g(\overline{s}) \notin A]$ *implies that* $f(\overline{t}) = g(\overline{s})$ *is equivalent to* $\perp$.
5. *for* $f \neq g$, $[f(\overline{t}) \notin A \wedge g(\overline{s}) \notin A]$ *implies that* $f(\overline{t}) \neq g(\overline{s})$ *is equivalent to* $\top$.
6. $s \notin A \wedge f(\overline{t}) \notin A$ *implies that the equation* $s = f(\overline{t})$ *is equivalent to* $\bigwedge f_i s = t_i \wedge IS_f(s)$. *Similarly it implies that the disequation* $s \neq f(\overline{t})$ *is equivalent to* $\bigvee f_i s \neq t_i \vee \neg IS_f(s)$.

4.1 Quantifier Elimination

(Un)limited Quantification: An *unlimited quantification* is one of the form $Qz \notin A$ and a *limited quantification* is one of the form $Qz \in A$, where $Q \in \{\exists, \forall\}$. As a shorthand we write $Q^u z$ for unlimited quantification and $Q^l z$ for limited quantification. Also we write $\exists^u \overline{x}$ for $\exists^u x_1 \cdots \exists^u x_m$ where $\overline{x} = (x_1, \cdots, x_m)$. Note that a quantification may be neither limited nor unlimited.

From now on, let $\overline{x}$ denote existentially quantified unlimited variables and let $\overline{y}$ denote unlimited parameters. Similarly let $\overline{x}'$ denote quantified limited variables and let $\overline{y}'$ denote limited parameters.

The following technical lemma shows how to remove unlimited quantification. Its proof will be the focus of this subsection.

Lemma 2. *Every formula of the form* $\exists^u \overline{x}\, \chi(\overline{x}, \overline{y}, \overline{y}')$ *where all quantifications in* χ *are limited, is equivalent in* $E(\mathcal{A})$ *to a formula of the form* $\chi'(\overline{y}, \overline{y}')$, *where all quantifications in* χ' *are also limited.*

Proof. We may assume $\mathcal{A}$ is not a total algebra for otherwise $E(\mathcal{A})$ is isomorphic to $\mathcal{A}$ in which case the lemma is trivial.

Consider a formula of the form

$$\exists^u \overline{x}\, \chi(\overline{x}, \overline{y}, \overline{y}'),$$

where all the quantifications in χ are limited. We will describe a procedure that transforms this formula into an equivalent formula where all the quantifications are limited.

The procedure will use the following techniques implicitly. We can always assume that all quantified variables are distinct by renaming if neccessary.

disjunctive splitting: The process of replacing a formula of the form $\exists x (B \vee C)$ by its logical equivalent $(\exists x\, B) \vee (\exists x\, C)$.

disjunctive normal form: Every quantifier free formula can be written as $\bigvee (\bigwedge B_{i,j})$ where the $B_{i,j}$ are literals.

formula normal form: Every formula can be expressed as

$$Q_k v_k \cdots Q_1 v_1 \psi,$$

where the Q_i are blocks of quantifiers of the same type (namely $\exists$ or $\forall$) and ψ is quantifier free in disjunctive normal form.

We now explain the concepts of type and type completion that will be used throughout the algorithm.

Types and Type Completions: A *type* of a term s is one of the following formulae:

- $s \in A$, or
- $s \notin A \wedge \mathrm{IS}_g(s) \wedge \bigwedge_{f \neq g} \neg \mathrm{IS}_f(s)$, where g is some constructor.

Note that a term has finitely many types. This definition is used in the following important concept [16].

Say that a conjunction of literals B is *type-completed* if for every subterm s in B, exactly one type of s is expressed in B. Note that a conjunction of literals can be extended to finitely many non-equivalent in $E(\mathcal{A})$ type-completed formulae. For example, a type completion of the formula $f(x) = y$ is the conjunction of $f(x) = y$ and

$$[x \in A] \wedge [f(x) \notin A \wedge \mathrm{IS}_f(f(x)) \wedge \bigwedge_{g \neq f} \neg \mathrm{IS}_g(f(x))] \wedge [y \notin A \wedge \mathrm{IS}_h(y) \wedge \bigwedge_{g \neq h} \neg \mathrm{IS}_g(y)].$$

We remark that a type completion may not be satisfiable (for instance if $f \neq h$ in the example above) Indeed in the algorithm such type completions are identified and replaced with $\perp$.

The *type completion of a quantifier free formula* $\phi = \bigvee \psi_i$, where each ψ_i is a conjunction of literals, is the equivalent quantifier free formula

$$\bigvee_{i,k} \psi'_{i,k},$$

where $\{\psi'_{i,k} \mid k\}$ consist of all the non-equivalent type completions of ψ_i. Here $\bigvee_{i,k} \psi'_{i,k}$ is *the type completion of* ϕ and is called *type-completed.* Note that each $\psi'_{i,k}$ is a conjunction of literals.

The *type completion of a formula* Φ is defined by the following procedure. First put Φ into formula-normal-form. So Φ is of the form

$$Q_p v_p \cdots Q_1 v_1 \, \psi,$$

where each Q_i is either $\forall$ or $\exists$. Now ensure that every quantifier is either limited or unlimited as follows. We proceed by induction on p. Suppose by induction that we have transformed the formula Φ into an equivalent formula Ψ with the property that Ψ is in formula-normal-form and all its quantifiers are either limited or unlimited. Now $\exists v_{p+1} \Psi$ is replaced by

$$\left[\exists^l v_{p+1} \Psi \vee \exists^u v_{p+1} \Psi\right].$$

Similarly, $\forall v_{p+1} \Psi$ is replaced by

$$\left[\forall^l v_{p+1}\Psi \wedge \forall^u v_{p+1}\Psi\right].$$

Now put the result into formula-normal-form. This completes the inductive step. Finally type-complete the quantifier free part. So the formula is now of the form

$$Q_k^{\alpha_k} v_k \cdots Q_1^{\alpha_1} v_1 \psi,$$

where ψ is $\bigvee_i \psi_i$, and each ψ_i is a type-completed conjunction of literals, and $\alpha_i \in \{l, u\}$. Note that it is equivalent to the original formula Φ.

Limited and unlimited literals: Suppose that ψ_i is a (not necessarily type-completed) conjunction of literals with the property that there is a *unique* (up to equivalence in $E(A)$) type-completed conjunction of literals ψ_i' equivalent in $E(\mathcal{A})$ to ψ_i. For example if ψ_i is $x \in A \wedge f(x) \notin A$ then ψ_i' is $x \in A \wedge f(x) \notin A \wedge IS_f(f(x)) \wedge_{g\neq f} \neg IS_g(f(x))$. In most cases, $\psi_i = \psi_i'$ will be type-completed itself.

Call a term t *limited (with respect to ψ_i)* if ψ_i' contains the literal $t \in A$, and *unlimited (with respect to ψ_i)* if ψ_i' contains the literal $t \notin A$. An equation or disequation is (un)limited in ψ_i if both sides of it are (un)limited in ψ_i. A tester predicate $\mathrm{IS}_g(t)$ is (un)limited (with respect to ψ_i) if t is (un)limited (with respect to ψ_i). For the rest of the proof, when a term or (dis)equation is called limited, implicitly it is with respect to the type-completed ψ_i' in which it occurs. Recall a quantification Qv is called limited if it is of the form $(Qv \in A)$, also written $Q^l v$. If it is of the form $Qv \notin A$, also written $Q^u v$, it is unlimited.

The Algorithm: We are now ready to describe the algorithm. It takes as input a formula Φ of $E(\mathcal{A})$ of the form $\exists^u \overline{x}\, \chi(\overline{x}, \overline{y}, \overline{y}')$, where every quantification in χ is limited. So we may write Φ as $\exists^u \overline{x}\, Q_k^l x_k' \cdots Q_1^l x_1'\ \psi(\overline{x}, \overline{x}', \overline{y}, \overline{y}')$. Also, recall our notation for variables: $\overline{x}$ denotes existentially quantified unlimited variables and $\overline{y}$ unlimited parameters; similarly, $\overline{x}'$ denote quantified limited variables and let $\overline{y}'$ denote limited parameters.

The algorithm proceeds in steps that transform the input formula to an equivalent output formula with additional syntactic properties. After describing each step we prove, unless obvious, termination and correctness of that step.

Step 1. Type Completion.

Input: An $E(\mathcal{A})$-formula.

Output: An equivalent type completed formula.

So the formula is now of the form

$$\exists^u \overline{x}\, Q_k^l x_k' \cdots Q_1^l x_1' \bigvee_i \psi_i(\overline{x}, \overline{x}', \overline{y}, \overline{y}'), \qquad (\star)$$

where each ψ_i is a type-completed conjunction of literals.

Step 2. Put Every Term into Term-Normal-Form.

Input: A type-completed formula in form $(\star)$.

Output: An equivalent type-completed formula in form $(\star)$ for which every term is in term-normal-form.

A term is in *term-normal-form* if it is of the form

$$t(v_1, \cdots, v_{j+k})[v_1/L_1 w_1, \cdots, v_j/L_j w_j]$$

where $t(\overline{v})$ is a term built from constructors only. Here L_i is a sequence of selectors applied to the variable w_i.

This step proceeds by pushing selectors past constructors as follows.

For each ψ_i do the following until no more apply. Pick some t occurring as a subterm in ψ_i. There are two cases.

Case 1: t Is Limited. Then for every constructor f, replace $f_j(t)$ in ψ_i by t, and

Case 2: t Is Unlimited. Let g be the unique constructor for which $\mathrm{IS}_g(t)$ is a conjunct of ψ_i.

- For every $f \neq g$, replace $f_j(t)$ in ψ_i by t, and
- Say t is of the form $g(\overline{s})$ for some $\overline{s}$. Then replace $g_j(t)$ in ψ_i by s_j.

Termination: After applying each case the number of selectors in the formula decreases. Hence this step can be iterated only a finite number of times.

Correctness: By Lemma 1, the transformation preserves the equivalence of the formulas. Since only terms have changed, the resulting formula is still in formula-normal-form. Also, since every term in the resulting formula is already a term in the original formula, the result is also type-completed.

Step 3. Remove Selectors from All Unlimited Quantified Variables.

Input: A type-completed formula Φ in form $(\star)$ for which every term is in term-normal-form.

Output: An equivalent type-completed formula for which every term is in term-normal-form and there are no selectors infront of unlimited quantified variables. That is one of the form

$$\exists^u \overline{x}\, Q_k^l x_m' \cdots Q_1^l x_1' \bigvee_i \psi_i(\overline{x}, \overline{x}', \overline{y}, \overline{y}'), \qquad (\dagger)$$

where each ψ_i is type-completed and no unlimited quantified variable x in ψ_i has a selector in front of it.

Recall $\overline{x}$ is the collection of unlimited quantified variables. For each $x \in \overline{x}$ do the following until no more apply:

- Pick an $x \in \overline{x}$ for which Lx occurs in Φ where L is some non-empty block of selectors.
- Replace Φ by

$$\exists \overline{v} \bigvee_f \left[f(\overline{v}) \notin A \wedge (\exists^u \overline{x}\, Q_k^l x_k' \cdots Q_1^l x_1' \bigvee_i \psi_i)[x/f(\overline{v})] \right],$$

  where f varies over all constructors. Now remove the existential quantifier $\exists^u x$.
- Apply step 1 and then step 2.

Termination: The result of substituting $f(\overline{v})$ for x and then putting the terms in normal form results in every selector of the form Lx being transformed into one of the form $L'v_i$ where the length of L' is smaller than the length of L. And since steps 1 and 2 do not introduce selectors, the size of the largest block of selectors in front of unlimited quantified variables strictly decreases with each iteration.

Correctness: Once the procedure terminates no unlimited quantified variable x has a selector infront of it. That the output formula is equivalent follows from the fact that the following are equivalent in $E(\mathcal{A})$ for every formula Θ:

- $\exists^u x\, \Theta$.
- $\exists^u x \bigvee_f (\mathrm{IS}_f(x) \wedge x \notin A \wedge \Theta)$.
- $\exists \overline{v}\, \exists^u x \bigvee_f (x = f(\overline{v}) \wedge \mathrm{IS}_f(x) \wedge x \notin A \wedge \Theta)$.

Step 4a. Put Every (Dis)equation into Literal-Normal-Form.
Input: A formula of the form (†).
Output: An equivalent formula of the form (†) in which every equation and disequation is in literal-normal-form.

An *unlimited (dis)equation is in literal-normal-form* if it is of the form

$$L_1 v_1 \Delta L_2 v_2$$

for some (possibly empty) L_i, and $\Delta \in \{=, \neq\}$. Here the v_i and $L_i v_i$ are unlimited.

A *limited (dis)equation is in literal-normal-form* if each term is in term-normal-form

$$t(v_1, \cdots, v_{j+k})[v_1/L_1 y_1, \cdots, v_j/L_j y_j],$$

with the additional property that the $v_{j+1}, \cdots, v_{j+k}, L_1 y_1, \cdots, L_j y_j$ are limited (that is, stated in ψ_i to be in A).

Throughout this step ensure that every literal $t \Delta s$ satisfies $t \in A$ if and only if $s \in A$ by applying the following steps whenever possible.

- An equation between terms t and s with $t \in A$ and $s \notin A$ is replaced with $\perp$.
- A disequation between terms t and s with $t \in A$ and $s \notin A$ is replaced with $\top$.

Hence every (dis)equation is either limited or unlimited.

For the unlimited (dis)equations repeat the following steps in each ψ_i until none can be applied; and then finally put the result into formula-normal-form, and type complete it.

- An unlimited equation of the form $f(\overline{t}) = g(\overline{s})$ is replaced with $\perp$ if $f \neq g$ and with $\bigwedge_i t_i = s_i$ if $f = g$.
- An unlimited disequation of the form $f(\overline{t}) \neq g(\overline{s})$ is replaced with $\top$ if $f \neq g$ and with $\bigvee_i t_i \neq s_i$ if $f = g$.
- An unlimited equation of the form $Ly = f(\overline{t})$, *with y unquantified*, is replaced with $\bigwedge_i f_i L y = t_i$.

– An unlimited disequation of the form $Ly \neq f(\overline{t})$, *with y unquantified*, is replaced with $\bigvee_i f_i Ly \neq t_i$.

Termination: In general each item removes a literal $t\varDelta s$ and replaces it with (a boolean combination of) a set of literals $\{t_i \varDelta' s_i\}$. The relevant property here is that the largest number of constructors appearing in a term from $t\varDelta s$ is strictly greater than the largest number of constructors appearing in a term from any of the $t_i \varDelta' s_i$.

Correctness: The procedure produces a formula that is equivalent to the original one as seen from Lemma 1. The formula is of the form (†) since the only introduced selectors are in front of unquantified unlimited variables from $\overline{y}$. Now if $t\varDelta s$ is a resulting unlimited literal then it does not contain constructors. Also both t and s are in term-normal-form since each operation preserves being in term-normal-form. Hence $t\varDelta s$ is in literal-normal-form.

Now we deal with the easier case of limited literals. Suppose t is limited (with respect to some ψ_i of the input formula). Note that t is already in term-normal-form. Now if some v_j or $L_j y_j$ occurring in t were unlimited (with respect to ψ_i) then t would also be unlimited, and so ψ_i is replaced with $\perp$. Hence every limited (dis)equation in the formula obtained is in literal-normal-form.

Step 4b. Put Tester Predicates into Literal-Normal-Form.

Input: A formula of the form (†) in which every equation and disequation is in literal-normal-form.

Output: An equivalent formula in formula-normal-form, for which every literal is in literal-normal-form, and there are no selectors infront of unlimited quantified variables. Also it has the property that no literal mentions both a quantified unlimited variable from $\overline{x}$ and a limited variable.

A *tester predicate is in literal-normal-form* if it is of the form $\mathrm{IS}_g(Lv)$ for some g, where L is a possibly empty block of selectors, v is a variable and Lv is unlimited (that is, stated in ψ_i not to be in A).

We put every tester predicate into literal normal form by applying the following steps to every term t in ψ_i. Say t is of the form $f(\overline{s})$ for some $\overline{s}$ and f.

– If t is limited then replace $\mathrm{IS}_g(t)$ by $\perp$ for every g.
– If t is unlimited then replace $\mathrm{IS}_g(t)$ by $\top$ if $f = g$ and by $\perp$ otherwise.

Termination is clear in this case.

Correctness: By Lemma 1 the resulting formula is equivalent to the input formula. Every tester predicate is in normal form since each term in the input was in term-normal-form. Since in this step only tester predicates are removed the (dis)equations are still in literal-normal-form, and there are no selectors infront of unlimited quantified variables x from $\overline{x}$. We remark that although $\varPhi$ is no longer necessarily type-completed, each disjunct ψ_i has a unique type-completion ψ'_i up to equivalence in $E(\mathcal{A})$. Recall that we call a term (un)limited in ψ_i if it is (un)limited in ψ'_i. So if x occurs in a term, that term is unlimited. Moreover if x occurs in an equation or disequation, then it is of the form $x\varDelta Ly$, or $x\varDelta x_1$, where x_1 is also from $\overline{x}$ and y is unlimited and unquantified. Similarly if x occurs in a tester predicate, it is of the form $\mathrm{IS}_g(x)$ for some g. Hence no literal mentions both x and a limited variable.

Step 5. Separate the Unlimited Quantifiers from the Limited Quantifiers.

Input: A formula Φ of the form

$$\exists^u \overline{x}\, Q^l_k x'_k \cdots Q^l_1 x'_1\, \psi(x'_k, \cdots, x'_1, \overline{x}, \overline{y}, \overline{y}'),$$

and with the properties resulting from the previous step.

Output: An equivalent formula in disjunctive normal form where each conjunction consists of formulae of the form

$$Q^l_k x'_k \cdots Q^l_1 x'_1\, \mu(x'_k, \cdots, x'_1, \overline{y}, \overline{y}'),\ \ \exists^u \overline{x}\, \delta(\overline{x}, \overline{y}) \ \text{ and } \ \epsilon(\overline{y}, \overline{y}').$$

Here

- μ is a possibly empty quantifier free formula with the property that every literal in μ is limited and mentions some variable from $\bar{x}'$.
- δ is a possibly empty conjunct of literals, and every literal in it is unlimited and mentions some variable from $\overline{x}$.
- ϵ is a possibly empty conjunct of literals.

In other words literals not in the scope of some quantifier are separated out. This can be done since by the previous step no literal mentions both some x and some limited variable x'_j or y'.

Step 6a. Remove Equations from δ That Mention x.

Input: The formula resulting from the previous step.

Output: An equivalent formula of the same form with the additional property that there are no equations in any of the δ.

Repeat the following in every δ until no more apply.

- Replace $x = x$ by $\top$ and $x \neq x$ by $\perp$.
- If an equation $x = Ly$ is a conjunct in δ, then replace δ by $\delta[x/Ly]$.

Some literals may be transformed into literals of the form $L_1 y_1 \Delta L_2 y_2$. So put these new literals into the corresponding ϵ.

Termination: Since each stage removes the variable x from δ, this process eventually stops.

Correctness: After termination it is still the case that every literal in δ is unlimited and mentions some variable from $\overline{x}$. And the corresponding ϵ may have gained more literals. So the output formula has the same form but there are no equations left in δ since every equation mentioned some variable from $\overline{x}$.

Step 6b. Replace Each $\exists^u \overline{x}\, \delta$ with $\top$.

Input: The formula resulting from the previous substep.

Output: An equivalent formula without any unlimited quantification.

From the previous step there are no equations in any of the δ; the disequations in δ are of the form $x \neq Ly$ or $x \neq x_i$ for some other unlimited quantified variable x_i from $\overline{x}$.

Let $\overline{b}$ be an arbitrary instantiation of canonical terms for the parameters $\overline{y}$. We need to show that

$$E(\mathcal{A}) \models \exists^u \overline{x}\, \delta(\overline{x}, \overline{b})$$

First evaluate all selectors Lb by applying the definition of the selectors to the canonical terms from $\overline{b}$. Let $s \in \mathbb{N}$ be the maximum of the depth of the subterms (of the sentence) that do not mention variables from $\overline{x}$. For each f let $n_f \in \mathbb{N}$ be the number of distinct $x \in \overline{x}$ so that $\mathrm{IS}_f(x)$ is a conjunct of δ. Choose $k \in \mathbb{N}$ larger than s and with the property that for every constructor $f \in \Sigma$ there are at least n_f distinct canonical terms of depth k that start with an f. This can be done since $\mathcal{A}$ is not a total algebra.

Now let $\overline{a}$ be distinct canonical terms of depth k that satisfy the type data in δ. Then $E(\mathcal{A}) \models \delta(\overline{a}, \overline{b})$. Indeed an unlimited literal $a_i \neq a_j$ holds in $E(\mathcal{A})$ by assumption that the elements of $\overline{a}$ are distinct. An unlimited literal $a \neq b$ holds in $E(\mathcal{A})$ since the depth k of the term a is at least s which is greater than the depth of the term b. This completes the description and correctness of the algorithm.

Tracing through the proof, we see that we have transformed a formula of $E(\mathcal{A})$,

$$\exists^u \overline{x}\, Q^l_k x'_k \cdots Q^l_1 x'_1 \bigvee \psi_i(\overline{x}, \overline{x}', \overline{y}, \overline{y}'),$$

into one of the form

$$\bigvee_j [\epsilon_j(\overline{y}, \overline{y}') \wedge Q^l_k x'_k \cdots Q^l_1 x'_1\, \mu_j(\overline{x}', \overline{y}', \overline{y})],$$

with only limited quantification, and where every literal in μ_j is limited and mentions some variable of $\overline{x}'$. This completes the proof of Lemma 2.

4.2 Corollaries and the Main Result

We can also give a characterisation of the definable relations of $E(\mathcal{A})$.

Theorem 2. *There is a procedure that given a formula $\Phi(\overline{y}, \overline{y}')$ of $E(\mathcal{A})$ returns an equivalent formula $\Phi'(\overline{y}, \overline{y}')$ with the property that every quantification is limited. In particular Φ' has the form*

$$\bigvee_j [\epsilon_j(\overline{y}, \overline{y}') \wedge Q^l_m x'_m \cdots Q^l_1 x'_1\, \mu_j(\overline{x}', \overline{y}', \overline{y})]$$

where ϵ_j is a type-completed conjunct of literals, μ_j is a type-completed quantifier free formula, and every literal in μ_j is limited and mentions some variable from $\overline{x}' = \cup_i \overline{x}'_i$.

Proof. Given a formula Φ of $E(\mathcal{A})$, first replace the formula with its type-completion. So it is now of the form

$$Q_m v_m \cdots Q_1 v_1 \psi,$$

where every quantifier is either limited or unlimited. Now pick an innermost formula of the form

$$Q^u \overline{x}\, Q^l_k x'_k \cdots Q^l_1 x'_1\, \psi(\overline{x}, \overline{x}', \overline{y}, \overline{y}'),$$

where the quantifiers Q^l_i are limited, ψ is in disjunctive normal form $\bigvee_i \psi_i$, where each ψ_i, a conjunction of literals, is type-completed. Note that k may be 0. Also, we may assume that $Q^u\overline{x}$ is $\exists^u\overline{x}$, for if it were $\forall^u\overline{x}$ then replace it with $\neg\exists^u\overline{x}\neg$, and push the second $\neg$ inward as usual. Applying the lemma results in a formula with no unlimited quantification. Now repeat this process until there are no more unlimited quantifiers in Φ. Finally although the lemma may result in a formula containing a conjunct of literals B that is not type-completed, it is the case that B is equivalent in $E(\mathcal{A})$ to a type-completed conjunction of literals B'. So replace B by B'. This completes the proof.

A formula $\Phi(\overline{y}, \overline{y}')$ of $E(A)$ is called an $\mathcal{A}$-*formula* if it is type-completed, and of the form

$$Q^l_k x'_k \cdots Q^l_1 x'_1\, \mu(\overline{x}', \overline{y}', \overline{y}),$$

where every literal in μ is limited. Recall this means that each term is limited and of the form

$$t(v_1, \cdots, v_{j+k})[v_1/L_1 y_1, \cdots, v_j/L_j y_j],$$

where each of $v_{j+1}, \cdots, v_{j+k}, L_1 y_1, \cdots, L_j y_j$ is limited, and $t(\overline{v})$ consists of constructors only.

Observe that if χ_1 and χ_2 are $\mathcal{A}$-formulae, then they are equivalent in $\mathcal{A}$ if and only if they are equivalent in $E(\mathcal{A})$. Also the subformulae from Theorem 2 of the form

$$Q^l_k x'_k \cdots Q^l_1 x'_1\, \mu(\overline{x}', \overline{y}', \overline{y}),$$

are $\mathcal{A}$-formulae. Hence we have the next corollary.

Corollary 1. *If $\mathcal{A}$ admits elimination of quantifiers, then so does $E(\mathcal{A})$.*

We now restate, and are ready to prove the main theorem.

Theorem 3. *If a partial algebra $\mathcal{A}$ has decidable QEP then so does its free total extension $E(\mathcal{A})$.*

Proof. Given a formula $\Phi(\overline{v})$ and a tuple of elements $\overline{w}$ from $E(\mathcal{A})$. Apply Theorem 2 and transform Φ into Φ'. Now form the sentence $\Phi'(\overline{w})$. This sentence consists of quantifier free sentences $\epsilon(\overline{a}, \overline{a}')$, and sentences of the form

$$Q^l_k v'_k \cdots Q^l_1 v'_1 \bigvee_i \psi_i(\overline{v}', \overline{a}', \overline{a}),$$

where each ψ_i consists of limited literals and type data. Here $\overline{a}'$ and $\overline{a}$ are amongst $\overline{w}$, and moreover $\overline{a}' \subset A$ and $\overline{a} \cap A = \emptyset$. Also $\overline{v}' = \cup_j v'_j$.

Now evaluate $\epsilon(\overline{a}, \overline{a}')$. This is done by first applying the definition of the selector functions and tester predicate and then applying the fact that $t_1 = t_2$, where the t_i are canonical terms not in A, if and only if t_1 and t_2 are syntactically equal.

This leaves an $\mathcal{A}$-sentence that is evaluated using the algorithm for the theory of $\mathcal{A}$.

5 Application

Recall that $GT_\Sigma(C)$ denotes the (total) algebra of ground terms generated by the non-empty set C of constants from Σ. We start with the following definition.

Definition 2. *Let E be a set of* ground equations*; that is equations of the form $t = s$ where t and s are ground terms. Consider the quotient of GT_Σ by the smallest congruence generated by E. This is a (total) algebra (over Σ) that we will denote by $\mathcal{A}_E$. Call a total algebra* **finitely presented** *if it is of the form $\mathcal{A}_E$ for some* finite *set E of ground equations.*

Easy examples include GT_Σ itself and every finite algebra. Decision problems for finitely presented algebras in a given variety of algebras have received much attention. For instance, the word problem in finitely presented semigroups or groups (in the variety of semigroups or groups) is, in general, undecidable. However in the variety of *all* algebras, the situation is different. For instance Kozen in [10] considers the uniform word problem, the finiteness problem, the subalgebra membership problem and the triviality problem: all are decidable in polynomial time. Comon in [3] proves that the first order theory of any finitely presented term algebra is decidable. The proof uses algebraic techniques combined with quantifier elimination methods. Our main theorem can now be applied to give another and, we think, simpler proof to decide the first order theory for finitely presented term algebras.

The relationship between finitely presented algebras and free total extensions is described in the next theorem (implicit in [11]).

Theorem 4. *A total algebra is a finitely presented term algebra if and only if it is the free total extension of a finite partial algebra.*

So we immediately have the following application of Theorem 1.

Theorem 5. *Let $\mathcal{A}_E$ be a finitely presented algebra. Then it has decidable QEP. In particular its first order theory is decidable.*

Acknowledgement

We thank the referees for their helpful comments used in improving the presentation and proof of the results.

References

1. E. Astesiano, M. Bidoit, H. Kirchner, B. Krieg-Brückner, P.D. Mosses, D. Sannella, A. Tarlecki: CASL: The Common Algebraic Specification Language. *Theoretical Computer Science* 286:2 (2002) 153–196
2. A. Blumensath, E. Gradel: Automatic structures. *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science (LICS) 2000* 51–62
3. H. Comon: Complete axiomatization of some quotient term algebras: *Theoretical Computer Science* 122:1-2 (1993) 165–200
4. K. Compton, C. Henson: A uniform method for proving lower bounds on computational complexity of logical theories. *Annals of Pure and Applied Logic* 48 (1990) 1–79
5. W. Hodges: *Model theory.* Cambridge University press (1993)
6. B. Khoussainov, S. Rubin, F. Stephan: Automatic Linear Orders and Trees. *Transactions on Computational Logic (TOCL) special issue (selected papers from the LICS 2003 conference)*, editor Phokion G. Kolaitis (accepted)
7. B. Khoussainov, A. Nerode: Automatic Presentations of Structures. D. Lievant (editor) *Proceedings of the conference on Logic and Computational Complexity (1994)* vol. 960 of LNCS (1995) 367–393
8. K. Korovin, A. Voronkov: A decision procedure for the existential theory of term algebra with the Knuth-Bendix ordering. In *Proceedings IEEE Conference on Logic in Computer Science* (2000) 291–302
9. G. Grätzer: *Universal Algebra.* D. Van Nostrand Co., Inc., Princeton, N.J.-Toronto, Ont.-London (1968)
10. D. Kozen: Complexity of finitely presented algebras. In *Proceedings of the 9th ACM symposium on theory of computing* (1977) 164–177
11. D. Kozen: Partial automata and finitely generated congruences: an extension of Nerode's theorem. In J. Crossley, J. Remmel, R. Shore, M. Sweedler editors. Logical methods: in honour of Anil Nerode's Siztieth Birthday, Birkhauser (1993) 490–511
12. V. Kuncak, M. C. Rinard: Structural sybtyping of non-recursive types is decidable. In *Proceedings IEEE Conference on Logic in Computer Science* (2003) 96–107
13. M. Lohrey: Automatic structures of bounded degree. *In proceedings of 10th International conference Logic for programming, artificial intelligence and reasoning*, editors M. Vardi and A. Voronkov, vol. 2850 of LNCS (2003) 346–360
14. M. Maher: A CLP view of logic programming. In *Algebraic and Logic Programming* (1992) 364–383
15. A. Mal′cev: Axiomatizable classes of locally free algebras of various types. In *The Metamathimatics of Algebraic Systems. Anatoliĭ Ivanovič Mal′cev. Collected papers: 1936-1967,* B. Wells III, Ed. Vol. 66. North Holland. Chapter 23 (1971) 262–281
16. T. Zhang, H. Sipma, Z. Manna: Term algebras with length function and bounded quantifier alternation. *the* 17^{th} *International Conference on Theorem Proving in Higher Order Logics (TPHOLs'04)* Volume 3223 of LNCS (2004) 321–336
17. T. Rybina, A. Voronkov: *A Decision procedure for term algebras with queues.* ACM Transaction on Computational Logic 2:2 (2001) 155–181
18. S. Vorobyov: An improved lower bound for the elementary theories of trees. In *Proceedings of the 13th Intl. Conference ib automated deduction*, Volume 1104 of LNCS (1996) 275–287
19. I. Walukiewicz: Monadic second order logic on tree-like structures. *Theoretical computer science* 275:1-2 (2002) 311–346