An Improved Lower Bound for the Elementary Theories of Trees

Sergei Vorobyov

Max-Planck-Institut für Informatik
Im Stadtwald, D-66123, Saarbrücken, Germany
(e-mail: sv@mpi-sb.mpg.de)

Abstract. The first-order theories of finite and rational, constructor and feature trees possess complete axiomatizations and are decidable by quantifier elimination [15, 13, 14, 5, 10, 3, 20, 4, 2]. By using the uniform inseparability lower bounds techniques due to Compton and Henson [6], based on representing large binary relations by means of short formulas manipulating with high trees, we prove that all the above theories, as well as all their subtheories, are **non-elementary** in the sense of Kalmar, i.e., cannot be decided within time bounded by a k-story exponential function[1] $\exp_k(n)$ for any fixed k. Moreover, for some constant $d > 0$ these decision problems require nondeterministic time exceeding $\exp_\infty(\lfloor dn \rfloor)$ infinitely often.

1 Introduction

Trees are fundamental in Computer Science. Different tree structures are used as underlying domains in automated theorem proving, term rewriting, functional and logic programming, constraint solving, symbolic computation, knowledge representation, data type specification, database theory, compiler construction.

In this paper we address the problem of inherent computational complexity of different first-order theories of trees. Probably the simplest and most widely used theory of trees in the class we consider is the *elementary theory of finite trees TA*, also known as the *theory of term algebras* or the *theory of locally free algebras*, or *Clark's equational theory*. It axiomatizes the usual Herbrand universum, specifying essentially that two terms are equal iff they coincide syntactically. Given a finite or infinite algebraic signature Σ, consider a theory TA axiomatized in the first-order predicate calculus with equality by the axioms:

$(Ax1)$	$\forall \overline{x}, \overline{y}\ (f(\overline{x}) \neq g(\overline{y}))$,	$f \not\equiv g \in \Sigma$.
$(Ax2)$	$\forall \overline{x}, \overline{y}\ (f(\overline{x}) = f(\overline{y}) \Rightarrow \overline{x} = \overline{y})$,	$f \in \Sigma$.
$(Ax3)$	$\forall x, \overline{y}\ (x \neq t(x, \overline{y}))$,	$t(x, \overline{y})$ contains x, distinct from x.

If Σ is finite, one may also add the so-called *Domain Closure Axiom*:

[1] Define $\exp_0(n) = n$ and $\exp_{k+1}(n) = 2^{\exp_k(n)}$;
$\exp_\infty(0) = 0$ and $\exp_\infty(n+1) = 2^{\exp_\infty(n)}$.

$$(DCA) \qquad \forall x \bigvee_{f \in \Sigma} \left(\exists \overline{y}\, x = f(\overline{y}) \right)$$

(i.e., every element of the domain is the value of some function or constant).

For the case of finite signatures, Malcev in 1961–62 [15] proved the decidability of $(Ax1)$–$(Ax3)$ and $(Ax1)$–$(Ax3)$, (DCA) by quantifier elimination. Other proofs are given in [13, 14, 5, 10], also by quantifier elimination. In [24] $(Ax1)$–$(Ax3)$, (DCA) is proved decidable by A. Robinson's model completeness test.

Rational Trees. The *theory of rational trees* is obtained from TA by replacing the acyclicity scheme $(Ax3)$ with the following scheme $(Ax4)$ corresponding to the acceptance of infinite trees [14]:

$$(Ax4) \qquad \forall \overline{x} \exists ! \, \overline{y} \left(\bigwedge_{i=1}^{n} y_i = t_i(\overline{x}, \overline{y}) \right)$$

for every system of equations $\wedge_{i=1}^{n} \; y_i = t_i(\overline{x}, \overline{y})$, where $\overline{x}$, $\overline{y}$ are disjoint lists of variables, and the last conjunction contains no circular subsets $y_{i_1} = y_{i_2} \wedge \ldots \wedge y_{i_{j-1}} = y_{i_j} \wedge y_{i_j} = y_{i_1}$. Maher [14] showed that for infinite signatures $(Ax1)$, $(Ax2)$, $(Ax4)$, and for finite signatures $(Ax1)$, $(Ax2)$, $(Ax4)$, (DCA) constitute complete axiomatizations for rational trees, and proved decidability by quantifier elimination. The last result is also proved for finite signatures by quantifier elimination in [5].

Feature Trees introduced and investigated in [1, 19, 3, 20, 11, 2] allow for non-fixed and variable arities of tree nodes and constitute a convenient formalism for logic and constraint programming, knowledge representation. Intuitively, saying $f(y) \wedge y\{f_1, \ldots, f_n\} \wedge f_1(y, x_1) \wedge \ldots \wedge f_n(y, x_n)$ in the language of feature trees means that y *labeled* f has exactly the *features (or selectors)* $f_1, \ldots, f_n$, and accessing the components of y via f_i's one gets elements x_i.

Formally, fix two finite or countably infinite recursive sets Lab of *labels* and Fea of *features*. A *path* is a word in Fea^*, ε is the empty path. For two paths p and q, pq is a concatenation of paths p, q. A *feature tree* t is a partial function from paths to labels Lab, $t : Fea^* \to Lab$ with a nonempty *prefix-closed domain*, i.e., $pq \in dom(t) \Rightarrow p \in dom(t)$ [1, 20, 4].

For a feature tree and a path $p \in dom(t)$, a *subtree* t/p of t *addressed* by p is the feature tree defined by $t/p \equiv_{df} \{(q, A) \mid (pq, A) \in t\}$ (here we identify a function with its graph). Define $ar(t)$, the *root arity* of a feature tree t, as $\{f \in Fea \mid f \in dom(t)\}$.

A feature tree t is *finite* iff $dom(t)$ is finite. A feature tree t is *rational* iff: 1) t has only finitely many different subtrees, and 2) t is finitely branching: for every $p \in dom(t)$, the root arity of t/p, $ar(t/p)$, is finite.

Let $\mathcal{F}$, $\mathcal{R}$, and $\mathcal{I}$ denote the sets of all finite, rational, and arbitrary feature trees over Lab and Fea; call them the *standard models*. The first-order theories of feature trees in the relational signature $\Sigma(Lab, Fea)$ contain: 1) a unary predicate symbol A for every $A \in Lab$; 2) a binary predicate symbol f for every $f \in Fea$; 3) a unary predicate symbol written $\{f_1, \ldots, f_n\}$ for every finite $\{f_1, \ldots, f_n\} \subseteq Fea$ (traditionally, for $F \subseteq Fea$, one writes xF instead of $F(x)$).

These symbols are interpreted in $\mathfrak{M} \in \{\mathcal{R}, \mathcal{F}, \mathcal{I}\}$ as follows:
– $\mathfrak{M} \models A(t)$ iff $t(\varepsilon) = A$;
– $\mathfrak{M} \models f(t,t')$ iff $t/f = t'$;
– $\mathfrak{M} \models tF$ iff $F = ar(t)$.

Backofen and Smolka [3] axiomatize the theory of rational feature trees without the arity predicates $y\{f_1, \ldots, f_n\}$ and prove its completeness and decidability by quantifier elimination. Smolka and Treinen [20] axiomatize the theory of rational feature trees with arity predicates, and Backofen and Treinen [4] prove its completeness (hence decidability) by using the classical Ehrenfeucht-Fraïssé games, demonstrating that every two models of the theory are elementarily equivalent. Backofen [2] also proves the completeness of the axiomatization of the theory of rational feature trees with arity predicates by quantifier elimination.

Contribution of the Paper. None of the papers cited above suggest any lower or upper complexity bounds for the theories of trees and the proposed quantifier elimination procedures. In [12] Kunen conjectured and gave a sketch of the proof that the decision problem for *TA* (for infinite signatures) is *PSPACE*-complete.

In this paper by applying Compton-Henson's uniform lower bound techniques [6] based on encoding of large binary relations we prove that all theories of finite and rational trees we cited above, as well as *all their subtheories*, are *non-elementary*[2]. We prove the following

Main Theorem. *Let T be either: 1) the theory of finite or rational trees over a signature with at least one non-unary function symbol, or 2) the theory of finite or rational feature trees over the signature with at least one label and two features. Let $T' \subseteq T$ be an arbitrary subtheory of T in the same signature, $Val(T')$ and $Sat(T')$ be the sets of sentences true in* all *and in* some *models of T'. Then:*

1. *The decision problems for $Val(T')$ and $Sat(T')$ are both* ***non-elementary****.*
2. *For some $d > 0$, both $Val(T')$ and $Sat(T')$ do not belong to the complexity class $NTIME(\exp_\infty(dn))$. In other words, both* ***require nondeterministic non-elementary time*** $\exp_\infty(dn)$ ***infinitely often****.* □

The first and the best known proof of a non-elementary lower bound was given by A. Meyer [16] for the weak monadic second-order logic of one successor function (Büchi arithmetic).

Outline of the Paper. After giving a brief account in the next section of Compton-Henson's uniform lower bound method, we proceed to informal explanations followed by the proof of the Main Theorem. The whole idea becomes clear in Sections 3 and 4, where we represent binary relations by trees and explain how to address concisely very deep occurrences in trees.

[2] A problem A is called *elementary* if and only if $A \in NTIME(\exp_m(n))$ for some *fixed* $m \in \omega$, and *non-elementary* otherwise; $NTIME(f(n))$ is the class of problems recognizable by non-deterministic Turing machines within time $f(n)$.

2 Compton-Henson's Lower Bounds Techniques

Compton and Henson in [6] developed a powerful and easy-to-use methods for establishing lower bounds for the decision complexity of logical theories. Their technique is based on the ideas of Trakhtenbrot and Vaught dating back to 50–60, used for proving the *recursive inseparability* of the sets of valid and finitely refutable sets of formulas of the first-order predicate calculus [7, 21, 17]. Compton and Henson refined the idea of recursive inseparability by replacing it with "*inseparable by $NTIME(t(n))$-recognizable sets*".

The lower bounds method of Compton and Henson relies on the ability to express by short formulas of a theory all models, up to a certain size, of a unique binary relation. The larger is the size, the higher is the respective lower bound. Technically, the method of Compton and Henson has a great advantage over the pioneer methods of Meyer, Stockmeyer, Fisher, Rabin [16, 23, 9, 8], since it allows one to avoid tedious encodings of Turing machines. To estimate the size of a binary relation representable by short formulas of a theory is much simpler than to prove that all Turing machine computations up to a certain length are encodable in a theory.

One of the (numerous) result in [6] useful to settle the nondeterministic time lower bounds is the following Theorem 5. To formulate it we need to define the interpretations of classes of binary relations in a theory.

Definition1 (Binary Relations). A *binary relation* is a model for the first-order language with equality and the unique binary predicate symbol $P(x, y)$. The *size* of a binary relation is the cardinality of its carrier. □

The interpretations used in [6] are based on a kind of reducibility slightly different from the standard log-lin reducibility (cf., [23, 22]):

Definition2 (Reset Log-Lin Reducibility, [6], pp. 10–11). A function is *reset log-lin computable* iff it is computable by a deterministic Turing machine within logarithmic space and linear time, the machine is allowed to reset its input tape k times to the beginning of the tape (in one step), where k is fixed for all inputs.

Let $\mathcal{L}_1 \subseteq A_1^*$ and $\mathcal{L}_2 \subseteq A_2^*$ be two languages over finite alphabets. We write $\mathcal{L}_1 \leq_{r\text{-}log\text{-}lin} \mathcal{L}_2$ and say that $\mathcal{L}_1$ is *reset log-lin reducible* to $\mathcal{L}_2$ if and only if there exists a reset log-lin computable function $f : A_1^* \to A_2^*$ such that

$$x \in \mathcal{L}_1 \;\Leftrightarrow\; f(x) \in \mathcal{L}_2 \text{ for every } x \in A_1^*. \qquad \square$$

Obviously, if $\mathcal{L}_1 \leq_{r\text{-}log\text{-}lin} \mathcal{L}_2$, then any corresponding reducing function f is *linearly bounded*, i.e., for some constant $c \in \omega$, $|f(x)| \leq c|x|$ for all $x \in A_1^*$. The $\leq_{r\text{-}log\text{-}lin}$ relation is transitive [6]. The following is straightforward:

Proposition3. *Suppose, $\mathcal{L}_1 \leq_{r\text{-}log\text{-}lin} \mathcal{L}_2$ and $t : \omega \to \omega$ is monotone increasing. Then $\mathcal{L}_2 \in NTIME(t(n)) \;\Rightarrow\; \mathcal{L}_1 \in NTIME(t(dn)+dn)$ for some $d > 0$.* □

Definition 4 (Interpretations of Binary Relations). Let $\mathcal{C}_1, \mathcal{C}_2, \ldots$ be classes of binary relations and T be a theory in a language L. Suppose that for each $n \in \omega \backslash \{0\}$ there are formulas $\delta_n(x,u)$, $\pi_n(x,y,u)$ of the language L, **reset log-lin computable from n expressed in unary notation**, such that for every binary relation $\mathfrak{A} \in \mathcal{C}_n$ there is a model $\mathfrak{B}$ of T and an element m of $\mathfrak{B}$ such that the model

$$\langle \delta_n^{\mathfrak{B}}(x,m);\ \pi_n^{\mathfrak{B}}(x,y,m) \rangle^3$$

is isomorphic to $\mathfrak{A}$. Then the sequence $\{I_n \mid n \in \omega \backslash \{0\}\}$ of couples of formulas $I_n = \langle \delta_n, \pi_n \rangle$ is called an *interpretation* of the classes $\mathcal{C}_n$ in the theory T. □

Theorem 5 (Compton-Henson [6], Theorem 6.2, p. 38). *Let $t(n)$ be a time resource bound such that for some d between 0 and 1 one has $t(dn) = o(t(n))$. Let $\mathcal{C}_n$ be the class of binary relations on sets of size at most $t(n)$ and T be a theory. If there is an interpretation of the classes $\mathcal{C}_n$ in T, then T is hard via reset log-lin reductions for the complexity class*

$$\mathbb{C} \ = \ \bigcup_{c>0} NTIME(t(cn)).$$

*This lower bound is **hereditary** in the sense that for each subtheory $T' \subseteq Val(T)$, both $Sat(T')$ and $Val(T')$ are hard for the above complexity class $\mathbb{C}$ via reset log-lin reductions.* □

Note that the iterated exponential functions $\exp_k(n)$, $\exp_\infty(n)$ all satisfy the condition of the theorem. We use Theorem 5 with the resource bound $\exp_\infty(n)$, which yields the intractability of a theory T as follows:

Proposition 6. *Suppose, Theorem 5 applies to a theory T and the time resource bound $t(n) = \exp_\infty(n)$. Then:*

1. *T is non-elementary.*
2. *$T \notin NTIME(\exp_\infty(dn))$ for some $d > 0$, i.e., T requires nondeterministic non-elementary time infinitely often.*
3. *1 and 2 above are true for $Val(T')$ and $Sat(T')$ for every $T' \subseteq T$.* □

Proof. 1) Suppose that T is elementary, i.e., $T \in NTIME(exp_m(n))$ for some *fixed* $m \in \omega$. For every $A \in \mathbb{C}$, by $\mathbb{C}$-hardness of T, we have $A \leq_{r\text{-}log\text{-}lin} T$. Hence, by Proposition 3, $A \in NTIME(\exp_m(dn))$ for some $d > 0$. Therefore, $\mathbb{C} \subseteq \bigcup_{d>0} NTIME(\exp_m(dn)) \subseteq NTIME(\exp_{m+1}(n))$. A contradiction, since $\mathbb{C}$ properly contains $NTIME(\exp_{m+1}(n))$ [4].

[3] As usual, this is the model with the carrier consisting of all elements x of $\mathfrak{B}$ satisfying $\mathfrak{B} \models \delta_n(x,m)$ and the binary predicate $P(x,y)$ defined on the elements x, y of the carrier satisfying $\mathfrak{B} \models \pi_n^{\mathfrak{B}}(x,y,m)$.

[4] By the following well-known result from [18]: Let $t_1(n)$ and $t_2(n)$ be functions such that $\lim_{n\to\infty} \frac{t_1(n+1)}{t_2(n)} = 0$. Then there is a problem in $NTIME(t_2(n)) \backslash NTIME(t_1(n))$.

2) Let $A \in \mathbb{C}$ belong to $NTIME(\exp_\infty(c_2 n)) \setminus NTIME(\exp_\infty(c_1 n))$ for some $c_2 > c_1 > 0$ [5]. By $\mathbb{C}$-hardness of T, $A \leq_{r\text{-}log\text{-}lin} T$ via a function f such that $|f(x)| \leq d|x|$ for some $d > 0$. We claim that $T \notin NTIME(\exp_\infty(c'n))$ for every $c' < c_1/d$. If not, then by Proposition 3, $A \in NTIME(\exp_\infty(c'dn) + dn) \subseteq NTIME(\exp_\infty(c_1 n))$, which contradicts to the choice of A. □

3 Trees Representing Binary Relations

We apply Theorem 5 to the theories of trees with the time resource bound $t(n) = \exp_\infty(n)$ thus obtaining the hereditarily non-elementary lower bounds.

The model $\mathfrak{B}$ in which we will interpret classes of binary relations according to Definition 4 will always be the same and equal to the standard model of the corresponding theory. For the theory of term algebras TA this is the usual Herbrand universum over a functional signature.

We can represent an *arbitrary* binary relation of size up to $t(n)$ by a tree of height $2 \cdot t(n)$ as follows. Suppose, we have means allowing us:

- to say succinctly[6] that a tree p represents a pointer to an occurrence at depth $t(n)$ (in some other tree); let the formula $Pnt_n(p)$ express this fact;
- to specify concisely that a tree p is a pointer as above addressing an occurrence of a tree s in a tree u; let this be expressed by the formula $St_n(u, p, s)$.

This is enough for dealing with binary relations of size up to $t(n)$. In fact, every binary relation of size up to $t(n)$ can be represented by a tree u of height at most $2 \cdot t(n)$ and bounded branching (two is enough) as follows. We represent elements of a relation as pointers at depth $t(n)$, and for every two elements i, j represented by pointers p, q respectively we say that i and j are related if and only if the subtree of u addressed first by p and then by q (which is a subtree of t at depth $2 \cdot t(n)$) equals 1 for some distinguished tree denoted 1. So, the formulas $\delta_n(x, u)$ and $\pi_n(x, y, u)$ from Definition 4 will be simply $Pnt_n(x) \wedge \exists s St_n(u, x, s)$ and $\exists s \Big[St_n(u, x, s) \wedge St_n(s, y, 1) \Big]$ respectively[7].

In the next sections we show how to write reset log-lin computable short formulas $Pnt_n(p)$ and $St_n(u, p, s)$ for addressing in subtrees down to depth $t(n) = \exp_\infty(n)$.

4 How to Address Deep Occurrences

Informally, we exploit the easy fact that a full binary tree contains $\exp_\infty(n+1) = 2^{\exp_\infty(n)}$ subtrees at depth $\exp_\infty(n)$.

[5] See the previous footnote.

[6] By using short formulas, cf., Definition 4.

[7] Of course, not every such tree represents a binary relation, but every binary relation of size at most $t(n)$ is represented this way. By Definition 4, this is enough. A tree representing a binary relation has to have the branching degree at least two, and two suffices. This corresponds to the assumption that the language contains at least one k-ary function symbol with $k \geq 2$, or two features.

Suppose, we already know how to address succinctly in any tree down to depth $t(n)$. We show that this is enough to succinctly address down to depth $2^{t(n)}$. Let a tree u be given and we would like to address its subtrees at depth $2^{t(n)}$. Since we are able to address down to depth $t(n)$ we can describe a tree p with the following properties (we later call it a pointer):

1) down to depth $t(n)$ the tree p represents a full binary tree;

2) the subtrees of p at depth $t(n)$ form (in some order, no matter which) a sequence of pairs of trees $\langle s_i, v_i \rangle$ for $i = 0, \ldots, 2^{t(n)}$;

3) for every $i = 1, \ldots, 2^{t(n)}$ the tree s_i specifies a selector (i.e., an argument position) in v_{i-1} such that v_i is an immediate subtree of v_{i-1} at that position.

4) the selector s_0 specifies a position of an immediate subtree v_0 in u.

Then the tree $v_{2^{t(n)}}$ is a subtree of u at depth $2^{t(n)}$, as needed. Notice that to specify a subtree at depth $2^{t(n)}$ we used only the ability to look at most at depth $t(n)$.

5 Proof Plan, Iterative Definitions

Using the ideas sketched above we proceed to the definition of the formulas $\delta_n(x, u)$ and $\pi_n(x, y, u)$ from Definition 4, reset log-lin computable from n, needed to interpret binary relations, as requested by Theorem 5.

To define the necessary constructing blocks for $\delta_n(x, u)$ and $\pi_n(x, y, u)$ like $Pnt_n(p)$ and $St_n(t, p, s)$ above, we will proceed by giving simultaneous iterative definitions of the form

$$\begin{cases} \mathbf{P}_0(\mathbf{x}) & = \boldsymbol{\Psi}(\mathbf{x}) \\ \mathbf{P}_{n+1}(\mathbf{x}) & = \boldsymbol{\Phi}(\mathbf{x}, \mathbf{P}_n), \end{cases} \tag{1}$$

defining (vectors of) new auxiliary predicates $\mathbf{P}$ for every $n \in \omega$.

The definition (1) starts from the basic case, where $\mathbf{P}_0(\mathbf{x})$ are defined for all $\mathbf{x}$ by $\boldsymbol{\Psi}(\mathbf{x})$, containing no $\mathbf{P}$. Whenever $\mathbf{P}_n(\mathbf{x})$ are defined, the next iteration $\mathbf{P}_{n+1}(\mathbf{x})$ is defined by $\Phi(\mathbf{x}, \mathbf{P}_n)$ by using $\mathbf{P}_n(\mathbf{x})$ from the previous iteration.

The predicates $\mathbf{P}_n$ do not form the part of the language, and for every fixed $n \in \omega$ they can be eliminated by unfolding the iterative definition down to the basic case. Unfortunately, the straightforward unfolding yields formulas growing *exponentially* as n grows[8].

However, under very general assumptions on the form of $\boldsymbol{\Psi}$, $\boldsymbol{\Phi}$, this iterative unfolding can be done cleverly, so as $\mathbf{P}_n(\mathbf{x})$ grow only linearly with n. This is known as the *standard abbreviation trick* due to Meyer, Stockmeyer, Fisher, Rabin, which allows for rewriting a formula with multiple occurrences of the same subformula (but with different parameters) as an equivalent formula containing *just one* occurrence of the subformula, and *only fixed* number of variables. This standard abbreviation trick is described in full detail by Stockmeyer [22], pp. 189–190, also by Ferrante and Rackoff [8], Chapter 7, pp. 153–161, and further

[8] Consider $P_{n+1}(x, y) = \exists z(P_n(x, z) \wedge P_n(z, y))$ growing as 2^n.

by Compton and Henson in Section 3 of [6]. This trick allows one for the iterative definitions as above, to keep the size of $\mathbf{P}_n(\mathbf{x})$ proportional to n.

In Section 8 we proceed semi-formally by just writing simultaneous iterative definitions of the form (1) above. We justify that such definitions indeed yield reset log-lin computable formulas in Section 9, where we briefly describe Compton-Henson's abbreviation techniques, following Section 3 of [6].

6 Snapshots and Pointers

Here we formalize the ideas presented in Section 4. In this and the following two sections we speak about terms. It should be clear that the same can be conducted for feature trees, as we discuss in the end of Section 8.

Definition 7 (Snapshots). Fix two arbitrary different terms L and R and call them the *left* and the *right selectors* respectively.

A *snapshot* is a term of the form $f(s,t)$ where s is a selector L or R, and f is a binary function symbol[9]. Define the following relations on selectors:

$$Succ_L(x,y) \equiv_{df} \exists uvw\Big[x = f(u,f(v,w)) \wedge y = f(L,v)\Big] \tag{2}$$

$$Succ_R(x,y) \equiv_{df} \exists uvw\Big[x = f(u,f(v,w)) \wedge y = f(R,w)\Big] \tag{3}$$

$$Succ(x,y) \equiv_{df} Succ_L(x,y) \vee Succ_R(x,y) \tag{4}$$

So, in $Succ(x,y)$ the second argument of y is determined by its first argument and the second argument of x. Notice that if $f(s_0,t_0) = p_0, p_1, \ldots, p_m = f(s_m,t_m)$ is a sequence of snapshots such that $Succ(p_{i-1},p_i)$ for $1 \leq i \leq m$, then t_m is a subterm of t_0 at depth m, and the word $s_1 \ldots s_m \in \{L,R\}^*$ is the corresponding path leading from t_0 to t_m.

Definition 8 ($\exp_\infty(n)$-pointer). Let $n \in \omega$. An $\exp_\infty(n)$-pointer (cf., Section 4) is a term p with the following properties:

1. down to depth $\exp_\infty(n)$ the term p represents a full binary tree constructed of the binary function symbol f;
2. the subterms of p at depth $\exp_\infty(n)$, $\{p_1, \ldots, p_{\exp_\infty(n+1)}\}$ (there are $\exp_\infty(n+1)$ such subterms) are snapshots linearly ordered by the reflexive-transitive closure of the $Succ$ relation.

Call the first and the last elements in the order mentioned in 2 the *initial* and the *final snapshots* of the pointer p. □

[9] We could have used any $k > 2$-ary symbol by filling out its $k-2$ extra argument positions with some term, say L.

7 The Predicates Pnt_n, Min_n, Max_n, $Match_n$, St_n

We are going to define *inductively* the following predicates (cf., Definition 8):

- $Pnt_n(p)$ — p is an $\exp_\infty(n)$-pointer;
- $Min_n(t,s)$ — s is a minimal wrt $Succ$ subterm of t among all subterms of t at depth $\exp_\infty(n)$;
- $Max_n(t,s)$ — s is a maximal wrt $Succ$ subterm of t among all subterms of t at depth $\exp_\infty(n)$.

Suppose, p is an $\exp_\infty(n)$-pointer. Then $Min_n(p,s)$ and $Max(p,u)$ mean that s and u are the initial and the final snapshots of p.

With the predicates Pnt_n, Min_n, Max_n defined for every $n \in \omega$ we can explicitly define the following useful predicate:

$$Match_n(t,p) \equiv_{df} Pnt_n(p) \wedge \exists sux\Big\{Min_n(p,s) \wedge \\ \wedge\Big[s = f(L,u) \wedge t = f(u,x) \vee s = f(R,u) \wedge t = f(x,u)\Big]\Big\}. \quad (5)$$

Intuitively, $Match_n(t,p)$ means that the initial snapshot s of the $\exp_\infty(n)$-pointer p matches t in the sense that $Succ(f(x,t),s)$.

With the predicates Pnt_n, Min_n, Max_n, and $Match_n$ at hand we can also explicitly define for all $n \in \omega$ the most important for our purposes predicate

$$St_n(t,p,s) \equiv_{df} Pnt_n(p) \wedge Match_n(t,p) \wedge \exists ux\Big[Max_n(p,u) \wedge u = f(x,s)\Big], \quad (6)$$

which expresses the fact that a $\exp_\infty(n)$-pointer p addresses the subterm s at depth $\exp_\infty(n+1)$ in a term t.

8 Iterative Definitions of Min_n, Max_n, Pnt_n

Here are the simultaneous iterative definitions of Min_n, Max_n, Pnt_n. In the right-hand sides of these definitions all the occurrences of St_n and $Match_n$ should be understood as abbreviations for the right-hand sides of (5) and (6).

$$\begin{aligned}
Min_0(p,s) &= \exists xy\Big(p = s = f(x,y) \wedge [x = L \vee x = R]\Big), && (7)\\
Min_{n+1}(p,s) &= \exists a\Big[Pnt_n(a) \wedge St_n(p,a,s)\Big] \wedge \\
&\quad \forall bt\Big[Pnt_n(b) \wedge St_n(p,b,t) \Rightarrow \neg Succ(t,s)\Big], && (8)\\
Max_0(p,s) &= \exists xy\Big(p = s = f(x,y) \wedge [x = L \vee x = R]\Big), && (9)\\
Max_{n+1}(p,s) &= \exists a\Big[Pnt_n(a) \wedge St_n(p,a,s)\Big] \wedge
\end{aligned}$$

$$\forall bt\Big[Pnt_n(b) \wedge St_n(p,b,t) \Rightarrow \neg Succ(s,t)\Big], \tag{10}$$

$$Pnt_0(t) \;=\; \exists x\Big[t = f(L,x) \vee t = f(R,x)\Big], \tag{11}$$

$$\begin{aligned} Pnt_{n+1}(t) \;=\; & \forall p\Big\{Match_n(t,p) \Rightarrow \exists u St_n(t,p,u) \wedge \\ & \Big[\neg Max_n(t,p) \Rightarrow \\ & \exists qv\Big(Match_n(t,q) \wedge St_n(t,q,v) \wedge Succ(u,v)\Big)\Big]\Big\}. \end{aligned} \tag{12}$$

Finally, we make reference to the Meyer-Stockmeyer-Fisher-Rabin-Compton-Henson abbreviation trick described in Section 9 to get the final step in our proof:

Lemma 9. *For every $n \in \omega\backslash\{0\}$ the formulas*

$$\begin{aligned} \delta_n(x,u) \;&\equiv_{df}\; Pnt_{n-1}(x) \wedge \exists s St_{n-1}(u,x,s), \\ \pi_n(x,y,u) \;&\equiv_{df}\; \exists s\Big[St_{n-1}(u,x,s) \wedge St_{n-1}n(s,y,1)\Big] \end{aligned}$$

interpreting binary relations up to size $\exp_\infty(n)$ (cf., Definition 4) are reset log-lin computable from n expressed in unary notation. □

Our Main Theorem is now an immediate corollary to Theorem 5, Proposition 6, and Lemma 9. To extend the above proof to the case of feature trees it suffices to replace terms and term equalities by feature trees and feature formulas in the definitions of $Succ_L$ in (2), of $Succ_R$ in (3), of $Match_n$ in (5), of St_n in (6), of Min_0 in (7), of Max_0 in (9), and of Pnt_0 in (11). This is relatively straightforward. For example, for L we select the unique feature tree satisfying $f(x) \wedge x\{f_1\} \wedge f_1(x,x)$. Instead of term equalities with the binary function symbol $x = f(y,z)$ we use the feature tree notation $f(x) \wedge x\{f_1,f_2\} \wedge f_1(x,y) \wedge f_2(x,z)$.

9 Iterative Definitions Yielding Short Formulas

In this section we briefly and semi-formally present results from Section 3 of [6] necessary for the justification of the claim in Lemma 9. The interested reader should consult Chapter 7 of [8] for the explanation of the abbreviation trick.

We are going to enrich the usual first-order language with the possibilities to write *explicit* and *(simultaneous) iterative definitions*.

Let L^* be a language L extended with new predicate symbols, P be a predicate symbol from the extension. An *explicit definition* is a figure of the form $[P(\mathbf{x}) \equiv \theta]$, where P is a *defined predicate*, θ is a *defining* prenex formula of L^* containing no occurrences of P. For a formula ψ of L^* the figure $[P(\mathbf{x}) \equiv \theta]\,\psi$ is a formula of L^* that means that $P(\mathbf{x})$ is defined in ψ as θ. Equivalently, in the second-order logic this can be interpreted as $\forall P\Big(\forall x(P(x) \Leftrightarrow \theta) \Rightarrow \psi\Big)$.

An *iterative definition* $[P(\mathbf{x}) \equiv \theta]_n$, where θ is called the *operator formula*, n written in *unary notation*, and P possibly occurring in θ, is inductively defined as follows:

1) $[P(\mathbf{x}) \equiv \theta]_0$ is equivalent to the explicit definition $[P(\mathbf{x}) \equiv \exists x(x \neq x)]$,

2) $[P(\mathbf{x}) \equiv \theta]_{n+1}$ is equivalent to $\Big[P(x) \equiv [P(x) \equiv \theta]_n \theta\Big]$.

Since the index n is written in unary, the length of the iterative definition is *of the same order* as the nested explicit definitions it replaces.

Similarly, one can define *simultaneous iterative definitions* of the form, where θ_i's can contain any of P_j's:

$$\begin{bmatrix} P_1(\mathbf{x}_1) & \equiv & \theta_1 \\ & \cdots & \\ P_k(\mathbf{x}_k) & \equiv & \theta_k \end{bmatrix}_n$$

Iterative and explicit definitions are made *first-class constructs* of the language L^*. Thus, if ϕ is a formula of L^* and $\mathcal{D}$ is an explicit or iterative definition then $\mathcal{D}\,\phi$ is a formula of L^*. But it is required that the iteration number subscripts are written in *unary*. The explicit and iterative definitions do not really extend the expressive power of the language, since they can always be eliminated. Theorem 10 below assures that this elimination can be done by *giving at most a linear growth of the length of formulas*. This is exactly what is needed in defining the interpretations of classes of binary relations in a theory.

Let L^* be a language L extended with new predicate symbols $P_1, \ldots, P_k$, and l be a fixed natural number. A *prescribed set of formulas* over L is a set of figures of the form

$$\mathcal{D}_1 \cdots \mathcal{D}_i \cdots \mathcal{D}_m \; \psi,$$

where ψ is a prenex formula of L^* without any explicit or iterative definitions, every $P_1, \ldots, P_k$ defined in some $\mathcal{D}_j$, and for each i:

– either $\mathcal{D}_i$ is an explicit definition with all predicates in its defining formula appearing as defined predicates in the previous definitions $\mathcal{D}_j$ $(j < i)$,

– or $\mathcal{D}_i$ is a (simultaneous) iterative definition with operator formulas of length $\leq l$, possibly containing the defined predicates in the previous definitions $\mathcal{D}_j$ $(j < i)$ and in $\mathcal{D}_i$ itself.

Additionally, every variable in ψ and in defining formulas is quantified just once.

Theorem 10 (Compton-Henson, Theorem 3.4, p. 19). *Let L be a first-order language. For each prescribed set of formulas over L there is a reset log-lin reduction taking each formula in the set to an equivalent formula of L.* □

This theorem considerably simplifies proofs, since its provides quite liberal form of writing explicit and simultaneous iterative definitions.

In practice, it is also very convenient (as we did) to split an iterative definition into two parts, the (non-false) basic case and the inductive step by writing iterative definitions $\mathcal{D}$ of the form:

$$\mathcal{D}: \quad P_0(\mathbf{x}) \equiv \Psi(\mathbf{x}) \text{ and } P_{n+1}(\mathbf{x}) \equiv \Phi[\mathbf{x}, P_n],$$

and define $\mathcal{D}_0$ to be $[P \equiv \Psi]$ and $\mathcal{D}_{n+1}$ to be $[P \equiv \mathcal{D}_n\, \Phi]$.

Such iterative definitions with the non-false basic case can be easily transformed (by using boolean connectives) into the canonical form of iterative definition above. In fact, define θ to be

$$[\forall \mathbf{u} \neg P(\mathbf{u}) \Rightarrow \Psi(\mathbf{x})] \wedge [\exists \mathbf{v} P(\mathbf{v}) \Rightarrow \Phi(\mathbf{x}, P)]$$

We claim that every iterative definition $\mathcal{D}_n$ is equivalent to $[P \equiv \theta]_{n+1}$. In fact, for $n = 0$ we have $[P \equiv \theta]_1 = [P \equiv [P \equiv \exists x(x \neq x)]\theta] = [P \equiv \Psi] = \mathcal{D}_0$, as needed. Let the claim be true for $n = n_0$. We prove it for $n + 1$:

$$\mathcal{D}_{n+1} = [P \equiv \mathcal{D}_n \Phi] = [P \equiv [P \equiv \theta]_{n+1} \Phi] \stackrel{(*)}{=} [P \equiv [P \equiv \theta]_{n+1} \theta] = [P \equiv \theta]_{n+2},$$

as needed. The equality (*) is also validated by induction by using the fact that $\Psi(x)$ is not tautologically false.

Returning back to the iterative definitions in Sections 7 and 8, we see that the prescribed definitions for $\delta_n(x, u)$ and $\pi_n(x, y, u)$ given in Lemma 9 have the form (to be completely accurate, we have to transform all formulas to the prenex form and be careful about using different variables in quantifications):

$$\delta_n(x, u)] = [\mathcal{D}] \Big(Pnt(x) \wedge \exists s St(u, x, s) \Big),$$
$$\pi_n(x, y, u)] = [\mathcal{D}] \, \exists s \Big[St(u, x, s) \wedge St(s, y, 1) \Big],$$

where $[\mathcal{D}]$ is $\begin{bmatrix} Min = \ldots \\ Max = \ldots \\ Pnt = \ldots \end{bmatrix}_{n-1} \Big[Match = \ldots \Big] \, \Big[St = \ldots \Big]$.

So, Theorem 10 applies proving Lemma 9. □

Acknowledgments. I am grateful to an anonymous referee of one of my papers, who pointed out the connection with the non-elementary theory of a pairing function (cf., Chapter 8 of [8]). The result we presented here is, however, stronger, since it gives *hereditary* non-elementary lower bounds.

References

1. H. Aït-Kaci, A. Podelski, and G. Smolka. A feature constraint system for logic programming with entailment. *Theor. Comput. Sci.*, 122:263–283, 1994. Preliminary version: 5th Intern. Conf. Fifth Generation Computer Systems, June 1992.
2. R. Backofen. A complete axiomatization of a theory with feature and arity constraints. *J. Logic Programming*, 24:37–71, 1995.
3. R. Backofen and G. Smolka. A complete and recursive feature theory. *Theor. Comput. Sci.*, 146:243–268, 1995. Also: Report DFKI–RR–92–30, 1992.
4. R. Backofen and R. Treinen. How to win a game with features. In *Constraints in Computational Logics'94*, volume 845 of *Lect. Notes Comput. Sci.*, pages 320–335. Springer-Verlag, 1994.

5. H. Comon and P. Lescanne. Equational problems and disunification. *J. Symb. Computation*, 7:371–425, 1989.
6. K. J. Compton and C. W. Henson. A uniform method for proving lower bounds on the computational complexity of logical theories. *Annals Pure Appl. Logic*, 48:1–79, 1990.
7. Yu. L. Ershov, I. A. Lavrov, A. D. Taimanov, and M. A. Taitslin. Elementary theories. *Russian Math. Surveys*, 20:35–105, 1965.
8. J. Ferrante and C. W. Rackoff. *The computational complexity of logical theories*, volume 718 of *Lect. Notes Math.* Springer-Verlag, 1979.
9. M. J. Fisher and M. O. Rabin. Super-exponential complexity of Presburger arithmetic. In *SIAM–AMS Proceedings*, volume 7, pages 27–41, 1974.
10. W. Hodges. *Model Theory*, volume 42 of *Encyclopedia of Mathematics and its Applications*. Cambridge Univ. Press, 1993.
11. J. Jaffar and M. J. Maher. Constraint logic programming: A survey. *J. Logic Programming*, 19 & 20:503–581, 1994.
12. K. Kunen. Answer sets and negation as failure. In J.-L. Lassez, editor, *4th International Conference on Logic Programming*, volume 1, pages 219–228. MIT Press, 1987.
13. K. Kunen. Negation in logic programming. *J. Logic Programming*, 4:289–308, 1987.
14. M. J. Maher. Complete axiomatizations of the algebras of finite, rational, and infinite trees. In *3rd Annual IEEE Symp. on Logic in Computer Science LICS'88)*, pages 348–357, 1988.
15. 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 Pub. Co., 1971.
16. A. R. Meyer. Weak monadic second-order theory of successor is not elementary-recursive. In R. Parikh, editor, *Logic Colloquium: Symposium on Logic Held at Boston, 1972–1973*, volume 453 of *Lect. Notes Math.*, pages 132–154. Springer-Verlag, 1975.
17. P. Odifreddi. *Classical recursion theory*, volume 125 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Pub. Co., 1989. Second Edition, 1992.
18. J. I. Seiferas, M. J. Fisher, and A. R. Meyer. Separating nondeterministic time complexity classes. *J. ACM*, 25(1):146–167, 1978.
19. G. Smolka. Feature constraint logics for unification grammars. *J. Logic Programming*, 12:51–87, 1992.
20. G. Smolka and R. Treinen. Records for logic programming. *J. Logic Programming*, 18:229–258, 1994. Also: Report DFKI-RR–92–23, 1992.
21. R. M. Smullyan. *Theory of Formal Systems*. Princeton University Press, revised edition, 1961.
22. L. J. Stockmeyer. *The complexity of decision problems in automata theory and logic*. PhD thesis, MIT Lab for Computer Science, 1974. (Also /MIT/LCS Tech Rep 133).
23. L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: preliminary report. In *5th Symp. on Theory of Computing*, pages 1–9, 1973.
24. S. Vorobyov. Theory of finite trees revisited: Application of model-theoretic algebra. Technical Report CRIN–94–R–135, Centre de Recherche en Informatique de Nancy, October 1994.

This article was processed using the LaTeX macro package with LLNCS style