The decision problem for branching time logic

Yuri Gurevich and Saharon Shelah

The Journal of Symbolic Logic / Volume 50 / Issue 03 / September 1985, pp 668 - 681
DOI: 10.2307/2274321, Published online: 12 March 2014

THE JOURNAL OF SYMBOLIC LOGIC
Volume 50, Number 3, Sept. 1985

THE DECISION PROBLEM FOR BRANCHING TIME LOGIC[1,2]

YURI GUREVICH[3] AND SAHARON SHELAH

Abstract. The theory of trees with additional unary predicates and quantification over nodes and branches embraces a rich branching time logic. This theory was reduced in the companion paper to the first-order theory of binary, bounded, well-founded trees with additional unary predicates. Here we prove the decidability of the latter theory.

Contents.

Introduction. The main result of this paper plus the companion paper [GS] is the decidability of the theory TREE of trees with additional unary predicates and quantification over nodes and branches. Here a *tree* is an arbitrary partial ordered set such that for every element x the subset $\{y: y \leq x\}$ is totally ordered. Elements of a tree are called *nodes*. Maximal totally ordered subsets of a tree are called *branches*.

Formally speaking, the language of TREE is the monadic second-order language (with equality) of trees with additional unary predicates. It has individual variables $v_0, v_1, \ldots$ and unary predicate variables $V_0, V_1, \ldots$. Its nonlogical symbols are the binary predicate symbol $<$ and unary predicate constants $P_0, P_1, \ldots$. Formulas $v_i = v_j$, $v_i < v_j$, $V_i(v_j)$, $P_i(v_j)$, $V_i = V_j$ are the atomic formulas of the language. The rest of the formulas in the language are built from the atomic formulas by means of the usual Boolean connectives and the usual quantifiers $\forall, \exists$ for individual as well as unary predicate variables.

Let ϕ be a sentence (a formula without free individual or unary predicate variables) in the language of TREE, and let $P_{i_1}, \ldots, P_{i_k}$ be the unary predicate

Received April 9, 1984; revised May 28, 1984.

[1] The results were obtained in principle during the Jerusalem Logic Year 1980–81 when both authors were fellows in the Institute for Advanced Studies of Hebrew University.

[2] Supported in part by the US-Israel Binational Science Foundation.

[3] Supported in part by NSF grant MCS83-01022

0022-4812/85/5003-0010/$02.40

constants of ϕ. The sentence ϕ is a *theorem* of TREE if it holds in every tree T with unary predicates $P_{i_1}, \ldots, P_{i_k}$ when the unary predicate variables of ϕ range over all branches of T. As usual, the decision problem for TREE is the problem of finding an algorithm which, given a sentence ϕ in the language of TREE, decides whether ϕ is a theorem of TREE or not.

The decision problem for a rich branching time logic reduces to the decision problem for TREE. As a matter of fact TREE is used to define the semantics of branching time logic. In this connection John Burgess posed to us the decision problem for TREE. In the companion paper [GS] we illustrate the relation between TREE and branching time logic, and reduce the decision problem for TREE to the decision problem for the first-order theory of binary, bounded, well-founded trees with additional unary predicates.

Let us recall the necessary definitions from [GS]. We imagine our trees growing upward. If $x < y$ are nodes of a tree and there is no node z with $x < z < y$ then x is the *father* of y and y is a *son* of x. A tree is called *binary* if no node has more than two sons. A branch of a tree is *bounded* if it has a maximal node (a leaf). A tree is *bounded* if all its branches are bounded.

Our definition of well-founded trees is a little more restrictive than the usual one. We say that a tree is *well-founded* if

(i) for every node x the subset $\{y: y \leq x\}$ is well ordered, and

(ii) every pair of nodes has a greatest lower bound.

Note that every well-founded tree has a least node (the root).

REMARK. The additional unary predicates of a tree were called colors in [GS]. In this paper the term "color" is reserved for a different purpose. In other respects the terminology here is compatible with that of [GS].

The main result of this paper is the decidability of the first-order theory of binary, bounded, well-founded trees with additional unary predicates. This paper can be read independently of [GS]. In the rest of the paper all trees are binary and well-founded if the contrary is not stated explicitly.

Even though we deal in this paper with (formally) first-order theories, our methods are typical for dealing with monadic second-order theories. See the survey [Gu] in this connection. This paper is distinguished by an interaction of the automata method and the composition method. Neither of the two methods is used too deeply here, but their interaction is somewhat nontrivial.

In §3 we introduce tree automata. In §4 we reduce the decision problem for the first-order theory of bounded trees with additional unary predicates to the emptiness problem for tree automata. The reduction proceeds essentially by induction on the given first-order formula. (Actually, we deal with so-called *n-theories* rather than formulas.) Since we do not quantify unary predicate symbols (another way to put it is that they are quantified universally but not existentially), we are forced to move negations in all the way to atomic formulas and to deal directly with the case of the universal quantifier. In §3 we prove a composition theorem for trees which allows us to treat the case of the universal quantifier. In §5 we prove the decidability of the emptiness problem for tree automata.

Monadic second-order theories form a better frame for composition theorems [Sh or Gu] than first-order theories. The reason is that subsets are easier to split

than elements. For example, let C be the concatenation $A + B$ of linear orderings A and B. A subset X of C naturally splits into $A \cap X$ and $B \cap X$, whereas an element of C does not split that easily. In order to deal with this difficulty we introduce partial elements in §1.

Finally, let us note that the theory of arbitrary (not necessarily binary or well-founded) trees with additional binary relations of type node-branch and quantification over nodes and branches is undecidable. This theory provides semantics for branching time logic which allows atomic sentences whose truth-value may depend on the future (e.g. "There will be a naval battle tomorrow"). We sketch briefly the undecidability proof.

The idea is to interpret a finitely axiomatizable undecidable arithmetic on the nodes of a distinguished branch B. Every triple $R = (R_1, R_2, R_3)$ of node-branch relations gives a ternary relation R^1 on B as follows:

$(x_1, x_2, x_3) \in R^1$ if and only if there is a branch Y with $(x_i, Y) \in R_i$ for $i = 1, 2, 3$.

It remains to check that on a certain tree (on the tree of lexicographically ordered binary words) we can define addition and multiplication on B this way.

Acknowledgements. We thank John Burgess for posing to us the question of whether TREE is decidable. We thank Andreas Blass for useful comments.

§1. Partial elements and finite fragments of theories. In order to prove composition theorems for fragments of first-order theories (see §§2 and 3) it is convenient to deal with partial elements. In this section we introduce partial elements and adjust the well-known notion of n-theory to the case of partial elements.

DEFINITION. A *partial element* of a structure S is an element of S or the word NIL. We consider only those structures which do not contain the word NIL as an element. To stress that an element x of S differs from NIL we will sometimes say that x is an *actual element* of S.

Recall that the vocabulary (signature, similarity type) of a structure is the set of nonlogical constants of the structure. (The equality sign is a logical constant. It does not belong to any vocabulary.) In this paper vocabularies are finite and consist of predicate symbols and distinguished partial elements. It will be convenient for us to view the distinguished partial elements as a list rather than a set. This way we need not care about the names for distinguished partial elements.

Introducing partial elements allows a liberal definition of *substructures*. Let S be a structure whose partial distinguished elements are $x_1, \ldots, x_l$. With each nonempty subset A of S we associate a substructure of S with universe A. The basic relations of the substructure are of course the basic relations of S restricted to A. The partial distinguished elements of the substructure are $y_1, \ldots, y_l$, where y_i equals x_i if x_i belongs to A and y_i equals NIL otherwise.

NOTATION. If S is a structure of a certain vocabulary σ and $\bar{x} = (x_1, \ldots, x_l)$ is a tuple of l partial elements of S then $(S, \bar{x})$ is a structure whose vocabulary is an extension of σ by l distinguished partial elements.

DEFINITION. Let S be a structure of some vocabulary σ which consists entirely of predicate symbols. Let $\bar{x}$ be a tuple $(x_1, \ldots, x_l)$ of partial elements of S. The 0-theory $\mathrm{Th}^0(S, \bar{x})$ of the structure $(S, \bar{x})$ comprises:

(i) the atomic formulas $P(v_{i_1},\dots,v_{i_k})$ where $P \in \sigma$, $1 \le i_1,\dots,i_k \le l$, every x_{i_j} is an actual element of S and $P(x_{i_1},\dots,x_{i_k})$ holds in S,

(ii) the atomic formulas $v_i = v_j$, where x_i and x_j are the same actual element of S, and

(iii) the atomic formulas $v_i = \mathrm{NIL}$, where x_i is NIL.

DEFINITION. The $(n+1)$-theory $\mathrm{Th}^{n+1}(S)$ of a structure S is the set $\{\mathrm{Th}^n(S,x)\colon x \in S\}$.

Note that x ranges over the actual elements of S in this definition.

DEFINITION. Two structures A and B of the same vocabulary are *n-equivalent* if $\mathrm{Th}^n(A) = \mathrm{Th}^n(B)$.

Ultimately we are interested in the first-order theory of structures without distinguished partial elements.

LEMMA 1. *There is an algorithm with the following property. Let σ be a vocabulary comprising only predicate symbols. Let ϕ be a first-order σ-formula of quantifier depth $\le n$ with k free individual variables. Let S be a σ-structure and $\bar{x}$ be an l-tuple $(x_1,\dots,x_l)$ of distinguished partial elements of S such that $x_{i_1},\dots,x_{i_k}$ are actual elements. Given* $\mathrm{Th}^n(S,\bar{x})$ *and* $(i_1,\dots,i_k)$, *the algorithm computes whether* $\phi(x_{i_1},\dots,x_{i_k})$ *holds in S.*

PROOF. The desired algorithm is defined by an easy recursion in n. #

DEFINITION. Let σ be a vocabulary. The $(0,l)$-*box* for σ is the set of 0-theories $\mathrm{Th}^0(S,\bar{x})$, where S is a σ-structure and $\bar{x}$ is an l-tuple of partial elements of S. The $(n+1,l)$-*box* for σ is the power set of the $(n,l+1)$-box for σ.

LEMMA 2. *There is an algorithm which, given σ, n and l, computes the (n,l)-box for σ. There is an algorithm which, given any member t of the (n,l)-box for σ with $n > 0$ or $l > 0$, computes σ, n and l.*

The proof is easy. #

LEMMA 3. *There is an algorithm with the following property. Let S be a structure, $x_0 = \mathrm{NIL}$, $(x_1,\dots,x_k)$ a tuple of partial elements of S, and f a function from some $\{1,\dots,l\}$ to $\{0,1,\dots,k\}$. Given* $\mathrm{Th}^n(S,x_1,\dots,x_k)$, *the algorithm computes* $\mathrm{Th}^n(S,x_{f1},\dots,x_{fl})$.

PROOF. An easy recursion in n. #

LEMMA 4. *There is an algorithm which, given any* $\mathrm{Th}^{n+1}(S)$, *computes* $\mathrm{Th}^n(S)$. *Hence there is an algorithm which, given any* $\mathrm{Th}^n(S)$, *computes* $\mathrm{Th}^0(S)$.

PROOF. Pick any $\mathrm{Th}^n(S,x)$ in $\mathrm{Th}^{n+1}(S)$ and use the algorithm of Lemma 3 to get rid of x. #

LEMMA 5. *There are two algorithms with the following properties respectively. Let σ^+ be the extension of a vocabulary σ by an additional predicate symbol P, S^+ be a σ^+-structure, and S the σ-reduct of S^+. Given* $\mathrm{Th}^n(S^+)$, *the first algorithm computes* $\mathrm{Th}^n(S)$. *Suppose that P is expressible by a quantifier-free σ-formula ϕ in S^+. Then, given* $\mathrm{Th}^n(S)$ *and ϕ, the second algorithm computes* $\mathrm{Th}^n(S^+)$.

PROOF. The desired algorithms are defined by an easy recursion in n. #

REMARK 1. The algorithms of Lemmas 1 and 3–5 are defined for arbitrary members of the appropriate boxes, not only for genuine n-theories. In particular, the algorithm of Lemma 4, given any member of the $(n+1,l)$-box for some σ, computes a member of the (n,l)-box for σ.

§2. Well-ordered chains with additional unary predicates. This section is auxiliary. We analyze the first-order theory of well-orderings with additional unary predicates and essentially reprove the classical result [DMT] about the decidability of that theory. Our technique is not a novelty either. The decidability of richer theories was proved by similar techniques; see the survey [Gu] in this connection. However, the specific result that we need (Theorem 1 below) is not in the literature, and we prove it here.

Let σ be a vocabulary comprising the binary predicate symbol $<$ and finitely many unary predicate symbols. In this section a *chain* is a σ-structure which is a (total) well-ordering with respect to $<$.

DEFINITION. A chain with l distinguished partial elements will be called an *l-chain.*

DEFINITION. Let C be a chain and $\bar{x}$ be an l-tuple $(x_1, \ldots, x_l)$ of partial elements of C. The *color* of an element y of the l-chain $(C, \bar{x})$ is the 0-theory of the singleton substructure $\{y\}$ of C.

LEMMA 1. *There is a uniform in σ algorithm with the following property. Let C be a singleton l-chain. If at least one of the l distinguished partial elements is the actual element of C then, given* $\mathrm{Th}^0(C)$ *and n, the algorithm computes* $\mathrm{Th}^n(C)$. *If $l = 0$ or $l > 0$ but all l distinguished partial elements are equal to* NIL *then, given n and the color of the unique element of C, the algorithm computes* $\mathrm{Th}^n(C)$.

The proof is clear. #

DEFINITION. Let A be a subchain of a chain C. For every partial element x of C we define the *projection* $x \mid A$ of x into A. If $x \in A$ then $x \mid A = x$; otherwise $x \mid A = \mathrm{NIL}$. If $\bar{x}$ is a tuple $(x_1, \ldots, x_l)$ of partial elements of C then $\bar{x} \mid A$ is the tuple $(x_1 \mid A, \ldots, x_l \mid A)$. With respect to §1 the l-chain $(A, \bar{x} \mid A)$ is a substructure of the l-chain $(C, \bar{x})$. If A is a segment (initial segment, final segment) of C then $(A, \bar{x} \mid A)$ is a *segment* (*initial segment*, *final segment*) of $(C, \bar{x})$.

DEFINITION. An l-chain C is a *sum* of l-chains A and B if C can be split (partitioned) into an initial segment isomorphic to A and the corresponding final segment isomorphic to B. If C is a sum of A, B we write $A + B = C$.

LEMMA 2. *There is an algorithm* PLUS *with the following property. If A, B, C are l-chains and $A + B = C$ then*

$$\mathrm{Th}^n(A)\ \mathrm{PLUS}\ \mathrm{Th}^n(B) = \mathrm{Th}^n(C).$$

We will use an alias $+$ for PLUS.

PROOF. The algorithm PLUS is defined by recursion in n. The case $n = 0$ is obvious.

$\mathrm{Th}^{n+1}(C)$ is the union of the sets $\{\mathrm{Th}^n(C, x): x \in A\}$ and $\{\mathrm{Th}^n(C, x): x \in B\}$. The first set is

$$\{\mathrm{Th}^n(A, x) + \mathrm{Th}^n(B, \mathrm{NIL}): x \in A\} = \{t + \mathrm{Th}^n(B, \mathrm{NIL}): t \in \mathrm{Th}^{n+1}(A)\}.$$

(Use the algorithm of Lemma 3 in §1 to compute $\mathrm{Th}^n(B, \mathrm{NIL})$ from $\mathrm{Th}^n(B)$.) The second set is computed similarly. #

DEFINITION. Let α be a nonzero ordinal, and let A be an l-chain. An l-chain C is an *α-multiple* of A if C can be partitioned into α successive segments isomorphic to A. It is easy to see that if $\alpha > 1$, then A has α-multiples iff all l distinguished partial

elements of A are equal to NIL. All α-multiples of A are isomorphic. This justifies the notation $\alpha \cdot A$ for α-multiples of A.

LEMMA 3. *Let A be an l-chain with all l distinguished partial elements equal to* NIL. *For every integer $k \geq 2^n$ the l-chains $k \cdot A$ and $2^n \cdot A$ are n-equivalent.*

PROOF. Use Ehrenfeucht games [Eh]. The lemma obviously reduces to the case of singleton A, which was treated e.g. in the lemma of §1 of [Gu2]. #

DEFINITION. Suppose that A is an l-chain such that $2 \cdot A$ (exists and) is n-equivalent to A. An unbounded l-chain C is a *pseudomultiple* of A with respect to n if it has a cofinal set D such that D contains the first element of C and for every pair $x < y$ of elements of D, the segment $[x, y)$ of C is n-equivalent to A.

LEMMA 4. *There is a uniform in σ algorithm with the following property. Let A be an l-chain with $A + A$ n-equivalent to A. Given* $\mathrm{Th}^n(A)$, *the algorithm computes the n-theory of any pseudomultiple of A with respect to n. Hence all pseudomultiples of a given l-chain with respect to n are n-equivalent.*

PROOF. The desired algorithm is recursive in n. The case $n = 0$ is trivial. In order to deal with the case of $n + 1$ let C be a pseudomultiple of A with respect to n and let $D \subseteq C$ witness that C is a pseudomultiple of A with respect to n. For every $x \in C$ select an initial segment $A_x = [c, d)$ of C, where $c = \min(C)$ and $x < d \in D$. The corresponding final segment $C_x = [d, \infty)$ is a pseudomultiple of A. Then

$$\mathrm{Th}^{n+1}(C) = \{\mathrm{Th}^n(C, x): x \in C\} = \{\mathrm{Th}^n(A_x, x) + \mathrm{Th}^n(C_x, \mathrm{NIL}): x \in C\}$$
$$= \{t + \mathrm{Th}^n(C, \mathrm{NIL}): t \in \mathrm{Th}^{n+1}(A)\}. \quad \#$$

LEMMA 5. *There is a uniform in σ algorithm* MULT *with the following property. Let A be an l-chain with all l distinguished partial elements equal to* NIL. *Then*

$$\mathrm{MULT}(\mathrm{Th}^n(A)) = \mathrm{Th}^n(\omega \cdot A).$$

PROOF. Use the algorithm PLUS of Lemma 2 to compute the n-theory of $B = 2^n \cdot A$. By Lemma 3, $B + B$ is n-equivalent to B. Use the algorithm of Lemma 4 to compute the n-theory of $\omega \cdot B$, i.e. of $\omega \cdot A$. #

We write $\omega \cdot \mathrm{Th}^n(A)$ for $\mathrm{MULT}(\mathrm{Th}^n(A))$.

LEMMA 6. *For every n, every l-chain C is n-equivalent to an l-chain of ordinal type $< \omega^\omega$.*

PROOF (by induction on the ordinal type α of C). The only interesting case is when α is limit. By Theorem 1.1 in [Sh], there is a cofinal $D \subseteq C$ such that for all $x < y$ in D the segment $[x, y)$ of C is n-equivalent to a fixed l-chain A. Thus C has a final segment which is a pseudomultiple of A with respect to n. By Lemma 4 this final segment can be replaced by $\omega \cdot A$. Thereafter both the corresponding initial segment and each copy of A can be replaced by l-chains of ordinal type $< \omega^\omega$. #

DEFINITION. Let C be an l-chain. An element x of C is an *n-leader* in C if there is no $y < x$ such that the final segments $[x, \infty)$ and $[y, \infty)$ of C are n-equivalent.

DEFINITION. Let C be an l-chain. If x is the maximal element of C then x is an *n-preleader*. A nonmaximal element x of C is an n-preleader of C if x is an n-leader in every proper initial segment of C which contains x.

LEMMA 7. *If $x_1 < \cdots < x_k$ are nonmaximal n-preleaders of an l-chain C then k is bounded by the cardinality of the (n, l)-box for σ.*

PROOF. The segments $[x_1, x_k], \ldots, [x_k, x_k]$ have different n-theories. #

DEFINITION. Let $x_1 < \cdots < x_k$ be the n-preleaders of an l-chain C. The *extended n-theory* $\mathrm{ETh}^n(C)$ of C is the sequence (of length k) of the n-theories of the segments $[x_1, x_2), [x_2, x_3), \ldots, [x_{k-1}, x_k), [x_k, \infty)$ of C.

LEMMA 8. *There is a uniform in σ algorithm with the following property. Given the extended n-theory of an l-chain A and the color of the unique element of a singleton l-chain B, the algorithm computes* $\mathrm{ETh}^n(A + B)$.

The proof is clear. #

LEMMA 9. *There is a uniform in σ algorithm with the following property. Let C be an unbounded l-chain and I be the set of extended n-theories t such that for every $x \in C$ there is an initial segment A of C with $x \in A$ and* $\mathrm{ETh}^n(A) = t$. *Given I, the algorithm computes* $\mathrm{ETh}^n(C)$.

PROOF. Pick $u = (t_1, \ldots, t_k)$ in I in such a way that

(i) there are $p < q \leq k$ such that $\sum\{t_i : p \leq i \leq k\} = \sum\{t_j : q \leq j \leq k\}$, and

(ii) the index q is minimal.

(By Lemma 7 the number of n-preleaders is bounded. Hence cofinally often we have n-preleaders that are not n-leaders. Hence there exists a member of I which satisfies (i). Hence there exists a member of I which satisfies (i) and (ii).)

Let $a = \min(C)$, let D be a cofinal subset of C such that $\mathrm{ETh}^n[a, x) = u$ for all $x \in D$, and put $d = \min(D)$. Without loss of generality I contains every $\mathrm{ETh}^n[a, x)$ with $x \geq d$. By the minimality of q the first $q - 1$ n-preleaders of $[a, d)$ are exactly the n-preleaders of C. For every $x \in D$ let fx be the first n-preleader of $[a, x)$ which fails to be an n-leader in $[a, x)$. It remains only to compute $\mathrm{Th}^n[fd, \infty)$.

Clearly, $fx < fy$ if $x < y$ are elements of D. The set $\{fx : x \in D\}$ is cofinal in C. For, if b is an upper bound for this set then $\{fx : b < x \in D\}$ is an infinite set of n-leaders in $[a, b]$, which contradicts Lemma 7. Without loss of generality, $x < fy$ if $x < y$ are elements of D. Let c be the pth n-preleader of $[a, d)$, and let $s = \mathrm{Th}^n[c, fd) = \sum\{t_i : p \leq i < q\}$.

For all $x \in D$, $\mathrm{Th}^n[c, fx) = \mathrm{Th}^n[c, fd) = s$ and $\mathrm{Th}^n[fx, x) = \mathrm{Th}^n[c, x)$. Hence for all $x < y$ in D,

$$\begin{aligned}\mathrm{Th}^n[fx, fy) &= \mathrm{Th}^n[fx, x) + \mathrm{Th}^n[x, fy)\\ &= \mathrm{Th}^n[c, x) + \mathrm{Th}^n[x, fy) = \mathrm{Th}^n[c, fy) = s.\end{aligned}$$

By Lemma 4, $\mathrm{Th}^n[fd, \infty) = \omega \cdot s$. #

DEFINITION. An *l-chain automaton* is a quadruple $M = (Q, \delta, q_{\mathrm{in}}, F)$, where Q is a finite set (of *states*), q_{in} is an element of Q (the *initial state*), F is a subset of Q (the set of *final states*), and δ (the *transition function*) is the union of two functions:

$$\delta\text{-NEXT}: Q \times l\text{-COLOR} \to Q \quad \text{and} \quad \delta\text{-LIM}: \mathrm{PowerSet}(Q) \to Q.$$

Here l-COLOR is the set of possible colors of elements of l-chains.

DEFINITION. The *run* of an l-chain automaton $M = (Q, \delta, q_{\mathrm{in}}, F)$ on an l-chain C is the function $r: C \to Q$ such that

(i) if x is the first element of C then $r(x) = q_{\mathrm{in}}$,

(ii) if x is the successor of an element y then $r(x) = \delta(r(y), \mathrm{color}(y))$, and

(iii) if x is a limit element and $I = \{q$: for every $y < x$ there is z with $y < z < x$ and $r(z) = q\}$, then $r(x) = \delta(I)$.

The function $r^1(x) = \delta(r(x), \text{color}(x))$ is the *associate run* of M on C.

THEOREM 1. *For all l and n there is an l-chain automaton M such that if r^1 is the associate run of M on an l-chain C, a is the first element of C and x is an element of C, then $r^1(x)$ is the extended n-theory of the segment $[a, x]$ of C. Moreover, there is a uniform in σ algorithm which, given l and n, constructs an appropriate automaton M.*

PROOF. The algorithms of Lemmas 8 and 9 provide the transition function of the desired automaton #

The decidability of the first-order theory of well-orderings with additional unary predicates easily follows from Theorem 1, but we will not work out the details. Theorem 1 is the only result of this section which is used later.

§3. A composition theorem for trees. This is again an auxiliary section.

Let τ be a vocabulary comprising the binary predicate symbol $<$, the unary predicate symbols Root, Left, Right, and a finite number of additional unary predicate symbols.

In this section a *tree* is a τ-structure T such that

(i) T is a binary well-founded tree with respect to $<$,

(ii) $T \models \text{Root}(x)$ iff x is the root of T, and

(iii) the predicates Left and Right partition all sons (i.e. all nodes which have fathers) in T into *left sons* (satisfying Left) and *right sons* (satisfying Right) in such a way that for every pair of brothers, one brother is a left son and the other is a right son.

DEFINITION. A substructure S of a tree T is a *stem* for T if

(1) $x < y \in S$ implies $x \in S$, and

(2) every totally ordered subset of S is bounded in S if it bounded in T.

Note that if C is a totally ordered subset of a stem S for a tree T and $b = \sup(C)$ in T then $b \in S$ and $b = \sup(C)$ in S.

DEFINITION. Let l be a natural number. An *l-tree* is a tree with l distinguished partial elements.

DEFINITION. Let T be a tree, S a stem for T and x a partial element of T. We define the *projection* $x \mid S$ of x into S. If $x = \text{NIL}$ then $x \mid S = \text{NIL}$. Otherwise $x \mid S$ is the supremum of $\{a \in S: a \leq x\}$. If $\bar{x}$ is a tuple $(x_1, \ldots, x_l)$ of partial elements of T then $\bar{x} \mid S$ is the tuple $(x_1 \mid S, \ldots, x_l \mid S)$ of partial elements of S, and $(S, \bar{x} \mid S)$ is a *stem* for the l-tree $(T, \bar{x})$.

DEFINITION. Let T be a tree, S a stem for T and a an element of S. The *graft* of T at a with respect to S is the tree G_a with the following properties. The universe of G_a is $\{x \in T: x \mid S = a\}$. G_a is the substructure of T with respect to the vocabulary $\tau - \{\text{Root}\}$, and $G_a \models \text{Root}(x)$ iff $x = a$.

DEFINITION. Let T be a tree, S a stem for T, a an element of S, and G_a the graft of T at a with respect to S. For every partial element x of T we define the projection $x \mid G_a$ of x into G_a. If x is an element of G_a then $x \mid G_a$ is x; otherwise $x \mid G_a$ is NIL. If $\bar{x}$ is a tuple $(x_1, \ldots, x_l)$ of partial elements of T then $\bar{x} \mid G_a$ is the tuple $(x_1 \mid G_a, \ldots, x_l \mid G_a)$ of partial elements of G_a, and $(G_a, \bar{x} \mid G_a)$ is the *graft* of the l-tree $(T, \bar{x})$ at the node a with respect to S.

DEFINITION. Let T be an l-tree and S a stem for T. Then S^n is the expansion of S by the unary predicates

$$P_t = \{a \in S: \mathrm{Th}^n(G_a) = t\},$$

where t ranges over the (n, l)-box for τ and G_a is the graft of T at a with respect to S.

THEOREM 1. *There is a uniform in τ algorithm with the following property. Let T be an l-tree and S a stem for T. Given* $\mathrm{Th}^n(S^n)$, *the algorithm computes* $\mathrm{Th}^n(T)$.

PROOF. The case $n = 0$ is straightforward. Let x, y be distinguished partial elements of T, let $a = x \mid S$, $b = y \mid S$, let $G = G_a$, and let Q be a unary predicate symbol in τ which differs from Root, Left and Right. Use the following obvious facts:

$T \models x = \mathrm{NIL}$ iff $S \models a = \mathrm{NIL}$,

$T \models x = y$ iff $S \models a = b$ and $G \models x = y$,

$T \models \mathrm{Root}(x)$ iff $S \models \mathrm{Root}(a)$ and $G \models \mathrm{Root}(x)$,

$T \models x < y$ iff either $S \models a = b$ and $G \models x < y$ or $S \models a < b$ and $G \models \mathrm{Root}(x)$,

$T \models \mathrm{Left}(x)$ iff either $G \models \mathrm{Left}(x)$ or $S \models \mathrm{Left}(a)$ and $G \models \mathrm{Root}(x)$,

$T \models \mathrm{Right}(x)$ iff either $G \models \mathrm{Right}(x)$ or $S \models \mathrm{Right}(a)$ and $G \models \mathrm{Root}(x)$,

$T \models Q(x)$ iff $G \models Q(x)$.

The case of $n + 1$. Given $\mathrm{Th}^{n+1}(S^{n+1})$, we would like to compute $\mathrm{Th}^{n+1}(T)$. It suffices to construct an auxiliary algorithm which, given an element $\mathrm{Th}^n(S^{n+1}, b)$ of $\mathrm{Th}^{n+1}(S^{n+1})$, computes all n-theories $\mathrm{Th}^n((S, y \mid S)^n)$ with $y \mid S = b$, where $(S, y \mid S)$ is the corresponding stem for the $(l + 1)$-tree (T, y). For, the auxiliary algorithm allows us to compute $\{\mathrm{Th}^n((S, y \mid S)^n): y \in T\}$. Then, calling recursively the main algorithm we can compute $\{\mathrm{Th}^n(T, y): y \in T\} = \mathrm{Th}^{n+1}(T)$.

Let $u = \mathrm{Th}^n(S^{n+1}, b)$ be an arbitrary element of $\mathrm{Th}^{n+1}(S^{n+1})$. Using the algorithm of Lemma 4 in §1 we can compute $\mathrm{Th}^0(S^{n+1}, b)$, which provides $\mathrm{Th}^{n+1}(G_b)$. Let $s = \mathrm{Th}^n(G_b, y)$ be an arbitrary element of $\mathrm{Th}^{n+1}(G_b)$. It turns out that u and s uniquely define $v = \mathrm{Th}^n((S, y \mid S)^n)$. Moreover, v is computable from u and s.

To compute v from u and s we use the algorithms of Lemma 5 in §1. Let t range over the $(n, l + 1)$-box for τ, and let r range over the $(n + 1, l)$-box for τ. Then u describes (S, b) with predicates P_r, and v describes (S, b) with predicates P_t. It suffices to express every $P_t(a)$ as a quantifier-free statement about (S, b) with predicates P_r. The algorithms of Lemmas 3 and 4 in §1 allow us, given r, to compute $t = f(r)$ such that if G is an l-tree and $r = \mathrm{Th}^{n+1}(G)$ then $\mathrm{Th}^n(G, \mathrm{NIL}) = f(r)$. Obviously, every P_r is included into $P_{f(r)}$.

If $t = s$, then

$$P_t(a) \leftrightarrow t = \mathrm{Th}^n(G_a, y \mid G_a) \leftrightarrow a = b \quad \text{because } y \mid G_a \neq \mathrm{NIL} \text{ iff } a = b.$$

If $t \neq s$, then

$$P_t(a) \leftrightarrow a \neq b \quad \& \quad \bigvee \{P_r(a): f(r) = t\}. \quad \#$$

§4. Tree automata. In this section we introduce tree automata and reduce the decision problem for the first-order theory of binary, bounded, well-founded trees with additional unary predicates to the emptiness problem for tree automata. The notation and terminology of §3 are used.

In §2 we have defined the color of a point of an l-chain. In a similar way we define the *color* of a node x of an l-tree T: it is the 0-theory of the singleton substructure $\{x\}$ of T. The set of all possible colors of nodes of l-trees will be denoted COLORS(l).

DEFINITION. An *l-tree automaton* is a quadruple (Q, Δ, q_{in}, F), where Q is a finite set (of *states*), q_{in} belongs to Q (the *initial state*), F is a subset of Q (the set of *final states*), and Δ (the *transition function*) is the union of two functions

$$\Delta\text{-NEXT:} \quad Q \times \text{COLORS}(l) \to \text{PowerSet}(Q \times Q),$$

and

$$\Delta\text{-LIM:} \quad \text{PowerSet}(Q) \to \text{PowerSet}(Q).$$

DEFINITION. A *run* of an l-tree automaton (Q, Δ, q_{in}, F) on an l-tree T is a pair of functions $r_1: T \to Q$ and $r_2: T \to Q \times Q$ such that

(1) $r_1(x) = q_{in}$ if x is the root of T,

(2) $r_2(x) \in \Delta(r_1(x), \text{color}(x))$,

(3) if y is the left (right) son of x then $r_1(y)$ is the left (right) component of $r_2(x)$, and

(4) if x is a limit node and I is the set of $q \in Q$ such that for every $y < x$ there is z with $y < z < x$ and $r_1(z) = q$, then $r_1(x) \in \Delta(I)$.

DEFINITION. A run (r_1, r_2) of an l-tree automaton on an l-tree is *accepting* if

(1) for every node x without a left (right) son the left (right) component of $r_2(x)$ is a final state, and

(2) for every unbounded branch B, if I is the set of states which appear cofinally often in the sequence $\langle r_1(x): x \in B \rangle$, then $\Delta(I)$ meets F.

DEFINITION. An l-tree automaton *accepts* an l-tree if it has an accepting run on the l-tree.

It is easy to see that an l-tree automaton accepts a given l-tree if and only if it has a winning strategy against an adversary (called Pathfinder) in the following acceptance game. In a play of the game Pathfinder builds an initial segment of a branch, and the automaton builds what can be called a run on that segment of a branch.

Step 0. Set x_0 equal to the root and q_0 equal to q_{in}. The automaton chooses (L_0, R_0) in $\Delta(q_0, \text{color}(x_0))$.

Step $\alpha + 1$. Pathfinder chooses Left or Right. Suppose without loss of generality that Left is chosen. If x_α does not have a left son then the game is over. If $L_\alpha \in F$ then the automaton won; else Pathfinder won. If x_α has a left son set $x_{\alpha+1}$ equal to the left son of x_α. Then the automaton chooses $(L_{\alpha+1}, R_{\alpha+1})$ in $\Delta(q_\alpha, \text{color}(x_{\alpha+1}))$.

Step α *for a limit* α. Let $B = \{x_\beta: \beta < \alpha\}$ and $I = \{q:$ for every $\beta < \alpha$ there is γ with $\alpha < \gamma < \beta$ and $q_\gamma = q\}$. If B is a branch then the game is over; the automaton won if $\Delta(I)$ meets F; else Pathfinder won. Otherwise set $x_\alpha = \sup(B)$. The automaton chooses $q_\alpha \in \Delta(I)$ and (L_α, R_α) in $\Delta(q_\alpha, \text{color}(x_\alpha))$.

THEOREM 1. *For all n, l and every member of the* (n, l)*-box for* τ *there is an l-tree automaton* A_t *which accepts an arbitrary l-tree T if and only if* $\text{Th}^n(T) = t$. *Moreover, there is a uniform in* τ *algorithm which, given t, constructs an appropriate* A_t.

PROOF. We prove only the first statement of Theorem 1, but the proof will provide the desired algorithm.

The case $n = 0$ is easy. The desired automaton A_t "knows" which of the l partial elements are equal to NIL and which are actual elements with respect to t. Each state of A_t codes (consistent with t) information about these actual elements: which of them have been seen, which have been left aside, and which will be seen. If (r_1, r_2) is

an accepting run of A_t on an l-tree T and $x \in T$, then the information coded in $r_1(x)$ faithfully reflects the situation in the acceptance game when the automaton reaches x.

We suppose that the first statement of Theorem 1 is proved for n, and we prove it for $n+1$.

LEMMA 1. *For every t in the $(n, l+1)$-box for τ there is an l-tree automaton E_t which accepts an arbitrary l-tree T if and only if $t \in \mathrm{Th}^{n+1}(T)$.*

PROOF. The desired automaton E_t runs a copy of A_t. Since the colors of a given l-tree do not contain the needed quantifier-free information about the $(l+1)$st distinguished element, E_t guesses the needed information in a consistent fashion. #

DEFINITION. Let x be a node of an l-tree T. T_x, T_x^{L} and T_x^{R} are l-trees with a common root x. The three l-trees are substructures of T with respect to $\tau - \{\mathrm{Root}\}$. T_x comprises the nodes $y \geq x$. T_x^{L} (resp. T_x^{R}) comprises the nodes y such that either $y = x$ or y is a descendent of the left (right) son of x.

LEMMA 2. *For every l there are l-tree automata C, L, R such that*

(1) *each of the three automata accepts all l-trees, and*

(2) *if (r_1, r_2) is an accepting run of C (resp. L, R) on an arbitrary l-tree T and $x \in T$, then $r_2(x)$ codes the n-theory of (T_x, x) (resp. $(T_x^{\mathrm{L}}, \mathrm{NIL})$, $(T_x^{\mathrm{R}}, \mathrm{NIL})$) with respect to an a priori fixed coding.*

PROOF. After reading the color of a node x the automaton C guesses $t = \mathrm{Th}^n(T_x, x)$ and starts a copy of A_t to verify that $\mathrm{Th}^n(T_x, x)$ is indeed t. If C happens to have more than one copy of some A_t in the same state, it merges them into one copy.

The automata L and R are defined similarly. #

LEMMA 3. *For every l there is an l-tree automaton U such that*

(1) *U accepts all l-trees, and*

(2) *if (r_1, r_2) is an accepting run of U on an l-tree T and $x \in T$ then $r_2(x)$ codes the n-theory of the $(l+1)$-tree (T, x) with respect to an a priori fixed coding.*

PROOF. Let T be an arbitrary l-tree. For every x let S_x be the stem $\{y: y \leq x\}$ for the $(l+1)$-tree (T, x). The algorithm of Theorem 1 in §3 computes $\mathrm{Th}^n(T, x)$ from $\mathrm{Th}^n(S_x^n)$.

Note that S_x^n is an $(l+1)$-chain for a certain vocabulary σ. By Theorem 1 in §2 there is a deterministic $(l+1)$-chain automaton M such that if r' is the associate run of M on S_x^n then $r'(x)$ is the extended n-theory $\mathrm{ETh}^n(S_x^n)$. The usual n-theory $\mathrm{Th}^n(S_x^n)$ is easily computable from $\mathrm{ETh}^n(S_x^n)$.

It is easy to see that U is able to exploit M provided it is able to compute the n-theories of (T_x, x), $(T_x^{\mathrm{L}}, \mathrm{NIL})$ and $(T_x^{\mathrm{R}}, \mathrm{NIL})$. Now we use Lemma 2. U incorporates the automata C, L, R in such a way that a run of U is accepting only if the corresponding runs of C, L, R are accepting. #

We are ready now to finish the proof of Theorem 1. Let t be a member of the (n, l)-box for τ. The desired automaton A_t incorporates automata E_s (see Lemma 1) for each $s \in t$. A run of A_t is accepting only if the corresponding runs of the automata E_s are accepting. Hence A_t accepts an l-tree T only if $\mathrm{Th}^{n+1}(T)$ includes t.

A_t incorporates also a modified version of the automaton U of Lemma 3. Recall that each state q of U codes a member s of the $(n, l+1)$-box for τ. If $s \notin t$, make q a dead-end state. As a result, if A_t accepts an l-tree T then $\mathrm{Th}^{n+1}(T)$ is included in t. #

COROLLARY 1. *There is a uniform in τ algorithm which, given any member t of any $(n,0)$-box for τ, constructs a tree automaton A'_t such that an arbitrary tree T is accepted by A'_t if and only if T is bounded and* $\mathrm{Th}^n(T) = t$.

PROOF. Given t, use the algorithm of Theorem 1 to construct a tree automaton $A_t = (Q, \Delta, q_{\mathrm{in}}, F)$ which accepts exactly the trees T with $\mathrm{Th}^n(T) = t$. We change A_t into the desired A'_t.

First let us hint the idea. If A_t accepts an unbounded tree, then after reading a certain unbounded branch A_t may put itself into a final state q. We would like to replace q by a nonfinal state q' without altering A_t too much.

Here is a formal construction. Let Q' be the extension of Q by new elements q', where $q \in F$, and let f be a mapping from Q' to Q such that f is the identity on Q and $fq' = q$ for $q \in F$. The desired automaton A'_t equals $(Q', \Delta', q_{\mathrm{in}}, F)$, where Δ' is the extension of Δ with respect to the following two clauses:

(1) If $q \in F$ and s is a color of a node of a tree then $\Delta'(q', s) = \Delta(q, s)$.

(2) If $I \subseteq Q$, $I' \subseteq Q'$, f maps I' onto I, $\Delta(I) = J$ and $\Delta'(I') = J'$ then f maps J' onto J and J' avoids F.

It is easy to see that A'_t accepts only bounded trees and that A'_t accepts every bounded tree accepted by A_t. #

COROLLARY 2. *There is a uniform in τ decision algorithm for the first-order theory of bounded trees with an oracle for the emptiness problem for tree automata.*

PROOF. Given a first-order sentence ϕ in the vocabulary τ, compute the quantifier depth n of ϕ. Use the algorithm of Lemma 1 in §1 to compile a list L of those members t of the $(n,0)$-box for τ which imply ϕ. Use the algorithm of Corollary 1 to construct automata A'_t for t in L. The sentence ϕ is a theorem of the first-order theory of bounded trees iff none of the automata A'_t accepts any tree. #

§5. The emptiness problem. Recall that our trees are special structures of a certain vocabulary τ (see the beginning of §3). In this section we construct a uniform in τ decision algorithm for the emptiness problem for tree automata (defined in §4). This algorithm and the algorithm of Corollary 2 in §4 yield a decision algorithm for the first-order theory of binary, bounded, well-founded trees with additional unary predicates.

DEFINITION. A tree automaton is *liberal* if it accepts at least one tree.

LEMMA 1. *A tree automaton $(Q, \Delta, q_{\mathrm{in}}, F)$ is liberal if there is a color t (an element of* COLORS(0) *in terms of* §4) *such that $\Delta(q_{\mathrm{in}}, t)$ meets $F \times F$.*

PROOF. The automaton accepts a singleton tree. #

LEMMA 2. *Let $A = (Q, \Delta, q_{\mathrm{in}}, F)$ be a tree automaton and $q \in Q - F$. If the tree automata $A_1 = (Q, \Delta, q_{\mathrm{in}}, F \cup \{q\})$ and $A_2 = (Q, \Delta, q, F)$ are liberal, then so is A.*

PROOF. Fix trees T_1, T_2 accepted by A_1, A_2 respectively. Let (r_1, r_2) be an accepting run of A_1 on T_1. It is easy to see that A accepts the tree obtained from T_1 as follows.

If a node x of T_1 does not have a left (right) son and the left (right) component of $r_2(x)$ equals q, then graft a copy of T_2 in such a way that the root of the graft is the left (right) son of x. If B is an unbounded branch of T_1 and $\Delta(B)$ contains q but avoids F, then graft a copy of T_2 in such a way that the root of the graft is the supremum of B. #

LEMMA 3. *Let A be a tree automaton $(Q, \Delta, q_{\mathrm{in}}, F)$ such that $\Delta(Q - F)$ meets F. Let $(q_0, \ldots, q_{m-1})$ be a list of all members of $Q - F$, where $q_0 = q_{\mathrm{in}}$ if q_{in} is nonfinal.*

Suppose that all automata

$$A_k = (Q, \Delta, q_k, F \cup \{q_{k+1}\})$$

are liberal, where $q_m = q_0$. *If* q_{in} *is final, suppose that the automaton*

$$A_{\text{in}} = (Q, \Delta, q_{\text{in}}, F \cup \{q_1\})$$

is liberal. Then A is liberal.

PROOF. For each $k < m$ fix a tree S_k accepted by A_k and an accepting run $r^k = (r_1^k, r_2^k)$ of A_k on S_k. If q_{in} is final fix a tree S_{in} accepted by A_{in} and fix an accepting run $r^{\text{in}} = (r_1^{\text{in}}, r_2^{\text{in}})$ of A_{in} on S_{in}. We construct embedded trees $T_0, T_1, \ldots$ and runs $R^0, R^1, \ldots$ of A on $T_0, T_1, \ldots$ respectively.

If q_{in} is final set $T_0 = S_{\text{in}}$ and $R^0 = r^{\text{in}}$; otherwise set $T_0 = S_0$ and $R^0 = r^0$. Suppose that T_n and R^n are constructed. Let k be the residue of $n + 1$ modulo m.

If x is a node of T_n without a left (right) son and the left (right) component of $R_2^n(x)$ equals q_k, then graft a copy of S_k on T_n in such a way that the root of the graft is the left (right) son of x. If B is an unbounded branch of T_n and $\Delta(B)$ contains q_k but avoids F, then graft a copy of S_k on T_n in such a way that the root of the graft is the supremum of B. Compose R^{n+1} from R^n and r^k in the natural way.

It is easy to see that the union of the runs R^n is an accepting run of A on the union of trees T_n. #

LEMMA 4. *If an automaton* $A = (Q, \Delta, q_{\text{in}}, F)$ *is liberal, then any automaton* $A' = (Q', \Delta', q_{\text{in}}, F)$, *where* Q' *extends* Q *and* Δ' *extends* Δ, *is liberal.*

PROOF. Accepting runs of A are accepting runs of A'. #

Let LIB be the smallest set of tree automata which contains every automaton satisfying the condition of Lemma 1 and which is closed under the operations described in Lemmas 2, 3 and 4.

LEMMA 5. *The decision problem of whether a given tree automaton belongs to* LIB *is decidable.*

The proof is easy. #

THEOREM 1. LIB *contains every liberal tree automaton.*

PROOF. Assume the contrary and fix a liberal automaton $A = (Q, \Delta, q_{\text{in}}, F)$ outside LIB with a minimum number m of nonfinal states. Let T be a tree accepted by A and let $r = (r_1, r_2)$ be an accepting run of A on T. Without loss of generality we suppose that $r_1(x) \in Q - F$ for every node x of T which is not the root. (Just throw away all nodes y such that $y \geq x$ for some x with $r_1(x)$ in F.)

Claim 1. *T is not singleton.*

Proof. Use Lemma 1. #

Claim 2. *T does not have leaves.*

Proof. Assume, towards a contradiction, that x is a leaf of T. Let $q = r_1(x)$ and A_1, A_2 be as in Lemma 2. A_1 accepts T; hence A_1 is liberal. By the minimality of A, A_1 belongs to LIB. By Lemma 1, A_2 belongs to LIB. By the definition of LIB, A belongs to LIB, which is impossible. #

Claim 3. *For every p in* $Q - F$ *and every x in T there is* $y > x$ *in T with* $r_1(y) = p$.

Proof. Suppose that p and x witness the contrary. Let $q = r_1(x)$, and let A_1, A_2 be as in Lemma 2. A_1 accepts T; by the minimality of A the automaton A_1 belongs to LIB. A proper reduct of A_2 (obtained by omitting p) accepts the subtree $\{y: x \leq y\}$ of

T. By the minimality of A the reduct belongs to LIB. By the clause related to Lemma 4 in the definition of LIB, A_2 belongs to LIB. By the clause related to Lemma 2 in the definition of LIB, A belongs to LIB. #

Let $q_0, \ldots, q_{m-1}$ be a list of all elements of $Q - F$, where $q_0 = q_{\text{in}}$ if q_{in} is not final.

Claim 4. *$\Delta(Q - F)$ meets F.*

Proof. Choose an arbitrary node x_0 in T. If $r_1(x_\alpha) = q_k$ and l is the residue of $k + 1$ modulo m, choose an arbitrary node $x_{\alpha+1} > x_\alpha$ with $r_1(x_{\alpha+1}) = q_l$. If α is a limit ordinal and the sequence $\langle x_\beta : \beta < \alpha \rangle$ is bounded in T, choose an arbitrary bound x_α for the sequence. Consider the branch B through all nodes x_α. Since A accepts T, $\Delta(Q - F)$ meets F. #

Let $A_0, \ldots, A_{m-1}$ and A_{in} be as in Lemma 3.

Claim 5. *All automata $A_0, \ldots, A_{m-1}$ and A_{in} belong to* LIB.

Proof. First we show that all these automata are liberal. A_{in} accepts T; hence it is liberal. For $k < m$ pick a node x in T with $r_1(x) = q_k$. A_k accepts the subtree $\{y : y \geq x\}$ of T.

By the minimality of A, all the automata in question belong to LIB. #

By the definition of LIB and Claims 4 and 5, A belongs to LIB, which is impossible. Theorem 1 is proved. #

THEOREM 2. *There is a uniform in τ decision algorithm for the emptiness problem for tree automata.*

PROOF. By Theorem 1 every liberal tree automaton belongs to LIB. By Lemmas 1–3, every automaton in LIB is liberal. Now use Lemma 4. #

REFERENCES

[DMT] J. E. DONER, A. MOSTOWSKI and A. TARSKI, *The elementary theory of well-ordering—a metamathematical study*, ***Logic Colloquium '77***, North-Holland, Amsterdam, 1978, pp. 1–54.

[Eh] A. EHRENFEUCHT, *An application of games to the completeness problem for formalized theories*, ***Fundamenta Mathematicae***, vol. 49 (1961), pp. 129–141:

[Gu] Y. GUREVICH, *Monadic second-order theories*, ***Model-theoretical logics*** (J. Barwise and S. Feferman, editors), Springer-Verlag, Berlin (to appear).

[Gu2] ———, *Toward logic tailored for computational complexity*, ***Logic Colloquium '83***, Lecture Notes in Mathematics, Springer-Verlag, Berlin (to appear).

[GS] Y. GUREVICH and S. SHELAH, *To the decision problem for branching time logic*, ***Foundations of logic and linguistics: Problems and their solutions***, Plenum Press (to appear).

[Sh] S. SHELAH, *The monadic theory of order*, ***Annals of Mathematics***, ser. 2, vol. 102 (1975), pp. 379–419.

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
UNIVERSITY OF MICHIGAN
ANN ARBOR, MICHIGAN 48109

INSTITUTE OF MATHEMATICS
THE HEBREW UNIVERSITY
JERUSALEM 91904, ISRAEL