INFORMATION AND COMPUTATION 83, 41–64 (1989)

Automata on Infinite Objects and Their Applications to Logic and Programming

M. NIVAT AND A. SAOUDI

Université Paris VII, Laboratoire d'informatique Théorique et Programmation, 2, place de Jussieu, 75221 Paris, France

We introduce various types of ω-automata, top-down automata and bottom-up automata on infinite trees. We study the power of deterministic and nondeterministic tree automata and prove that deterministic and non-deterministic bottom-up tree automata accept the same infinite tree sets. We establish a relationship between tree automata, Logic programs, recursive program schemes, and the monadic second-order theory of the tree. We prove that the equivalence of two rational logic programs is decidable. © 1989 Academic Press, Inc.

CONTENTS

INTRODUCTION

The theory of automata is well known to be strongly connected to both logic and computer science. For example, D. Muller (1963) uses automata to characterize a class of circuits and then obtains an analysis method of these circuits. In another example, J. R. Büchi (1960) uses automata to characterize a class of second-order logic formulas and then proves the decidability of the satisfibility problem for these formulas.

In computer science, automata techniques are used for applications to compiling, modeling concurrent systems, and communicating systems. This theory has been extended to infinite trees by many authors (see References). The idea of extending both Büchi's and Muller's automata to infinite trees is due to M. O. Rabin (1969, 1970). Rabin characterizes the monadic second-order theory of the tree (i.e., SkS) in terms of automata and then proves the decidability of SkS. The proof of the decidability of SkS has two

0890-5401/89 $3.00

difficult parts, the closure by complementation and the decidability of the emptiness problem (i.e., the problem whether a given automaton accepts some tree). Rabin's proof is not simple, for this reason L. Harrington and Y. Gurevich (1982) use a special kind of game and reduce the complementation problem to a determinancy result. This gives a simple proof. Likewise, D. Muller and P. Schupp (1984) introduce alternating infinite tree automata and give an alternative simple proof. In another related work, A. Emerson and P. Sistla (1984) prove that for each branching time logic formula, one can construct an automaton on infinite trees that accepts exactly the trees satisfying the formula.

The well-known theory of program schemes [1], describes the program as a syntactic object (i.e., program schema) with a semantic object (i.e., interpretation) such that the interpretation defines the meaning of the program and the scheme, its structure. From this theory, the meaning of a program is defined as a set of trees.

The motivation of our work is to establish the connection between the monadic second-order theory of the tree, logic programs, program schemes, and tree automata. The second idea of this development is to extend the well-known conditions of acceptance on infinite sequences to both top-down and bottom-up tree automata on infinite trees and then study the difference between deterministic and nondeterministic tree automata.

For this, we introduce tree automata with ω-language, called control language, which contain all infinite paths of a so-called accepted run (i.e., computation). We extend the well-known condition on ω-automata to automata on infinite trees and we prove the equivalence between recognizability by top-down tree automata, recognizability by bottom-up tree automata, and recognizability by deterministic bottom-up tree automata. On the other hand, top-down tree automata are more powerful than deterministic top-down tree automata.

We introduce ωi-regular tree grammars and then we characterize SkS in terms of tree grammars. We define a subclass of L_2-formulas (i.e., E_n-formulas), a class of program schemas (i.e., rational programs), and a class of logic programs (i.e., rational logic programs) and then prove the equivalence between computability by rational programs, computatiliby by rational logic programs, computability by a kind of tree machines, and definability by E_n-formulas.

1. Basic Definition

Before defining the classes of infinite tree automata, we will give some basic definitions.

Infinite Words and ω-Languages

An infinite word over Σ is a mapping from $[\omega]$ to Σ. We denote the set of infinite word over Σ by Σ^ω. An ω-language is a set of infinite words. Let L be a language over Σ, we define:

$L^* = \bigcup_{0 \leqslant i < \omega} L^i$

$L^\omega = \{u \in \sum^\omega : u = u_1 u_2 \cdots \text{and } u_i \in L\}$.

$K \cdot L^\omega = \{u : u = u_1 u_2,\ u_1 \in K,\ u_2 \in L^\omega, \text{ and } u \text{ is an infinite word}\}$.

The limit of L is the set of infinite words u such that the set of initial segments of u meets the set L infinitely many times.

An ω-language is called rational iff there exist two sequences $(A_i)_{0 \leqslant i \leqslant n}$ and $(B_i)_{0 \leqslant i \leqslant n}$ of rational (i.e., regular) languages such that $L = \bigcup_{i=1}^n A_i \cdot B_i^\omega$. This defines what Cohen and Gold (1977) call the ω-Kleene closure of regular languages. Let u be an infinite word, we denote by $\mathrm{Inf}(u)$ the set of symbols that occur infinitely often in u.

Trees over a Ranked Alphabet

A ranked alphabet is a structure $\langle \Sigma, r \rangle$, where r is a mapping from Σ to a finite subset of positive integers. If Σ is a ranked alphabet, we denote by Σ_n the set of symbols from Σ having n as their rank. The set of trees over the ranked alphabet Σ is defined as follows:

(i) If $a \in \Sigma_0$ then a is a tree, and

(ii) If $t_1, \ldots, t_n$ are trees and $f \in \Sigma_n$ then $f(t_1, \ldots, t_n)$ is a tree.

Finite and Infinite k-ary Trees

Let Σ be a finite alphabet, the set of k-ary finite trees over Σ, denoted by T_Σ is defined inductively as follows:

(i) If $a \in \Sigma$ then $a \in T_\Sigma$.

(ii) If $t_1, t_2, \ldots, t_k \in T_\Sigma$ and $a \in \Sigma$ then $a(t_1, \ldots, t_k) \in T_\Sigma$.

Let t be a k-ary finite tree, called tree for short, then the domain of t, denoted by $\mathrm{dom}(t)$, is defined inductively as follows:

(i) If $t \in \Sigma$ then $\mathrm{dom}(t) = \{\lambda\}$, where λ denotes the empty word.

(ii) If $t = a(t_1, \ldots, t_k)$ then $\mathrm{dom}(t) = \{\lambda\} \cup (\bigcup_{i=1}^k i \cdot \mathrm{dom}(t_i))$.

Let t be a finite tree over Σ, then the frontier of t, denoted by $\mathrm{Fr}(t)$, is the set $\{u : u \in \mathrm{dom}(t) \text{ and } uj \text{ is not in } \mathrm{dom}(t)\}$.

An infinite tree (i.e., ω-tree) over Σ is a mapping from $D = \{1, \ldots, k\}^*$ to Σ. We denote the set of infinite trees over Σ by T_Σ^ω.

Let $x, u \in D$, then x/u is equal to v, if $x = uv$, and undefined otherwise.

Let t be an infinite tree; then we define an infinite branch of t starting at the root as an infinite word $(t(u_i))_{i\geq 0}$, where:

(i) $u_0 = \lambda$, and

(ii) For each i, there exists a $j_i \in \{1, ..., k\}$ such that $u_i = u_{i-1} j_i$. Let t_i be a finite tree with F_i as a frontier $(i = 1, 2)$. We define the relation between two frontiers as: $F_1 < F_2$ if for each node $y \in F_2$ there exists a node $x \in F_1$ such that $x < y$.

A Limit of Finite Tree Sets

We say that t_1 is an initial tree of t_2 iff (i) $\mathrm{dom}(t_1) \subseteq \mathrm{dom}(t_2)$, and (ii) for each $u \in \mathrm{dom}(t_1 : t_1(u) = t_2(u)$. t_1 is called a proper initial tree of t_2 (i.e., $t_1 < t_2$) if t_1 is an initial tree of t_2 and $F_1 < F_2$. Let L be a set of finite trees; we define a limit of L in the sense of Rabin as:

$$\mathrm{Rlim}(L) = \{t \in T_\Sigma^\omega / \text{there is } \{(t_i)\}_{0\leq i\leq \omega} \subset L : t_i < t_{i+1} \text{ and } t_i < t\}.$$

Sets of Infinite Trees as Expressions of Sets of Finite Trees

Let $X = \{x_1, ..., x_n\}$ be a set of variables such that $X \cap \Sigma = \varnothing$ and let $(L_i)_{0\leq i\leq n}$ be a sequence of sets of finite trees on $\Sigma \cup X$ such that $L_i \cap X = \varnothing$ and L_i contains a finite tree such that all variables occurring as values of the frontier node are from X. We define $L_0 \cdot \langle L_1, ..., L_n \rangle$ as the set of trees that are obtained by taking t in L_0 and substituting elements of L_i for all occurrences of x_i on t. $L_0\langle L_1, ..., L_n\rangle$ is formally defined as follows:

(i) If $t = x_i$ then $t \cdot \langle L_1, ..., L_n \rangle = L_i$.

(ii) If $t \in \Sigma$ then $t(L_1, ..., L_k) = \{t(t_1, ..., t_k) : t_i \in L_i\}$.

(iii) If $t = a(t_1, ..., t_k)$ then $t\langle L, ..., L_n\rangle = a(t_1\langle L_1, ..., L_n\rangle, ..., t_k\langle L_1, ..., L_n\rangle)$.

(iv) $L \cdot \langle L_1, ..., L_n \rangle = \bigcup_{t\in L} t \cdot \langle L_1, ..., L_n \rangle$.

Let $L_0 \cdot \langle L_1, ..., L_n\rangle^p = (L_0 \cdot \langle L_1, ..., L_n\rangle^{p-1}) \cdot \langle L_1, ..., L_n\rangle$, where $L_0 \cdot \langle L_1, ..., L_n\rangle^0 = L_0$, $L_0 \cdot \langle L_1, ..., L_n\rangle^\omega = \{t \in T_\Sigma^\omega : \exists (t_n)_{0\leq n\leq \omega} t = \mathrm{Rlim}(\{t_n\})$, $t_0 \in L_0$, and $t_p \in t_{p-1} \cdot \langle L_1, ..., L_n\rangle\}$.

A projection is a mapping from a set Σ to a set Δ. A projection determines a mapping from the set of trees on Σ to the set of trees on Δ.

Let L be a set of infinite trees, then the adherence of L is the set of all infinite trees having their initial subtrees in the set of initial subtrees of L. The center of L is the set of initial subtrees of the adherence of L. Let x^+ be the set of finite k-ary trees over $\{x\}$ and x^ω be the infinite k-ary tree over $\{x\}$, then the adherence of $a(b^+, ..., b^+)$ is $\{a(b^\omega, ..., b^\omega)\}$ and the center of $a(b^+, ..., b^+)$ is equal to $a(b^+, ..., b^+) + a$. Let $x = (x_1, ..., x_n)$, then $p_i(x) = x_i$.

2. Automata on Infnite Sequences

Before defining top-down and bottom-up automata on infinite trees, we will define ω-automata (i.e., finite automata on infinite sequences) to clarify the passage from words to trees. The idea of using automata for recognizing infinite sequences is due to the late Büchi (1960). Büchi used ω-automata to prove the decidability of the monadic second-order theory of natural numbers with the successor relation, which is called S1S.

Definition 2.1. A *Büchi automaton* is a structure $M = \langle Q, \Sigma, q_0, \delta, F\rangle$, where:

(i) Q is a finite set of states,

(ii) Σ is the input alphabet,

(iii) q_0 is the initial state,

(iv) $\delta: Q \times \Sigma \to 2^Q$ is the transition function, and

(v) $F \subseteq Q$ is the set of designated states.

A computation of M over an infinite word u is a mapping $C: [\omega] \to Q$ satisfying the conditions:

(i) $C(0) = q_0$, and

(ii) for each $i \in [\omega]$, $C(i+1) \in \delta(C(i), u(i))$.

M accepts the infinite word u if and only if there exists a designated state which occurs infinitly often in this computation.

In 1963, D. Muller (1963) introduced a new condition for accepting infinite sequences and used it, with finite state automata, to analyze asynchronous circuits.

Definition 2.2. A *Muller automaton* is a structure $M = \langle Q, \Sigma, q_0, \delta, F\rangle$, where:

(i) Q, Σ, q_0, δ are defined as before, and

(ii) F is the family of designated sets of states.

A computation C is accepted in the sense of Muller if and only if the set of states ocurring infinitely often in this computation belongs to F.

Now we shall unify Büchi's automata and Muller's automata into ω-automata.

Definition 2.3. An *ω-automaton* is a structure $M = \langle Q, \Sigma, q_0, \delta, F\rangle$, where Q is the set of states, Σ is the input alphabet, q_0 is the initial state, $\delta: Q \times \Sigma \to 2^Q$ is the transition function, and F is the set of designated sets of states.

Let C be a computation; then we define $\mathrm{Inf}(C)$, $\mathrm{Fin}(C)$, and $\mathrm{Occ}(C)$ as follows:

(1) $\mathrm{Inf}(C) = \{q: \mathrm{Card}(C^{-1}(q)) = \omega\}$

(2) $\mathrm{Fin}(C) = \{q: 0 \leqslant \mathrm{Card}(C^{-1}(q)) < \omega\}$

(3) $\mathrm{Occ}(C) = \{q: \mathrm{Card}(C^{-1}(q)) \neq 0\}$.

A computation C of an ω-automaton is called i-accepted if there is a designated set of states H, such that the condition C_i is satisfied, where C_i is defined for $i = 1, ..., 9$ as follows:

C_1: $\mathrm{Inf}(C) \cap H \neq \varnothing$

C_2: $\mathrm{Inf}(C) \subseteq H$

C_3: $\mathrm{Inf}(C) = H$

C_4: $\mathrm{Occ}(C) \cap H \neq \varnothing$

C_5: $\mathrm{Occ}(C) \subseteq H$

C_6: $\mathrm{Occ}(C) = H$

C_7: $\mathrm{Fin}(C) \cap H = \varnothing$

C_8: $\mathrm{Fin}(C) \subseteq H$

C_9: $\mathrm{Fin}(C) = H$.

We use the term ωi-automaton to denote an ω-automaton with acceptance condition defined according to condition i. An ωi-automaton is said to be complete if and only if for each state q and each input sumbol a, M can read a from the state q. M is called deterministic if and only if for each state q and each input symbol a, we have $\mathrm{Card}(\delta(q, a)) \leqslant 1$.

Remarks. (1) A Büchi automaton can be viewed as an $\omega 1$-automaton with one designated set of states.

(2) An $\omega 3$-automaton is a Muller's automaton.

(3) For each computation C, we have $\mathrm{Fin}(C) = Q - \mathrm{Inf}(C)$.

It follows from the last remark, that conditions C_7, C_8, and C_9 are respectively equivalent to the conditions:

(i) $H \subseteq \mathrm{Inf}(C)$.

(ii) $Q - H \subseteq \mathrm{Inf}(C)$.

(iii) $Q - H = \mathrm{Inf}(C)$.

PROPOSITION 2.4. *For each ω-language L and for $i \in \{1, ..., 9\}$, the following conditions are equivalent*:

(i) *L is accepted by an ωi-automaton*;

(ii) *L is a projection of an ω-language accepted by a deterministic ωi-automaton.*

Proof. Let L be an ω-language accepted by the ωi-automaton $M = \langle Q, \Sigma, q_0, \delta, F \rangle$, we shall exhibit a deterministic ωi-automaton $\bar{M} = \langle \bar{Q}, \bar{\Sigma}, \bar{q}_0, \bar{\delta}, \bar{F} \rangle$ accepting K, and a projection Π such that $L = \Pi(K)$.

Construction. (1) $\bar{Q} = Q$

(2) $\bar{q}_0 = q_0$

(3) $\bar{\Sigma} = \Sigma \times Q$

(4) For each $\langle a, q \rangle \in \Sigma \times Q$ and $q_1 \in Q$, if $q_2 \in \delta(q_1, a)$ then $q_2 = \bar{\delta}(q_1, \langle a, q_2 \rangle)$

(5) $\bar{F} = F$.

Now let Π be the projection such that, for each $\langle x, q \rangle$ $\Pi(\langle x, q \rangle) = x$. It is clear that $L = \Pi(K)$. To finish the proof one can use the fact that the family of ω-languages accepted by nondeterministic ωi-automata is closed under projection. ∎

Now we shall study the power of ωi-automata ($i = 1, 9$), by comparing their power to both Büchi automata and Muller automata. For this we firstly recall some results.

THEOREM 2.5 (Mc Naughton, 1966). *For each ω-language L, the following conditions are equivalent:*

(i) *L is accepted a determinstic $\omega 3$-automaton,*

(ii) *L is a finite boolean combination of ω-languages accepted by deterministic Büchi automata, and*

(iii) *$L = \bigcup_{i=1}^{n} L_i \cdot K_i^{\omega}$, where L_i and K_i are regular languages*

The equivalence between (i) and (iii) is well known as a conjecture of Muller.

THEOREM 2.6 (J. R. Büchi, 1960). *The family of ω-languages accepted by nondeterministic $\omega 1$-automata is a boolean algebra.*

THEOREM 2.7 (D. Muller, 1963). *The family of ω-languages accepted by a deterministic $\omega 3$-automata is a boolean algebra.*

We shall prove the equivalence between Büchi's condition and the condition (i.e., condition C_{10}):

$$H \subseteq \mathrm{Inf}(C).$$

THEOREM 2.8. *For each ω-language L, the following conditions are equivalent*:

(i) *L is accepted by a nondeterministic (resp. deterministic) $\omega 1$-automaton, and*

(ii) *L is accepted by a nondeterministic (resp. deterministic) $\omega 10$-automaton.*

Proof. Let L be an ω-language accepted by the $\omega 1$-automaton $M = \langle Q, \Sigma, q_0, \delta, F\rangle$. Now we shall exhibit an $\omega 10$-automaton $M_1 = \langle Q_1, \Sigma, q_1, \delta_1, F_1\rangle$ accepting L.

Construction. (1) $Q_1 = Q$

(2) $q_1 = q_0$

(3) $\delta_1 = \delta$

(4) $F_1 = \{\{q\}: q \in H, \text{ for some } H \in F\}$.

It is clear that M and M_1 accept the same ω-language. Let M be an $\omega 10$-automaton accepting L; now we shall exhibit an $\omega 1$-automaton accepting L. Let $F = \{F_1, ..., F_n\}$. Assume that $\varnothing$ is not a member of F.

Construction. (1) $Q_1 = P(F_1) \times \cdots \times P(F_n) \times Q$

(2) $q_1 = (\varnothing, ..., \varnothing, q_0)$

(3) For each $q \in Q_1$ and $a \in \Sigma$ do

$\delta_1(q, a) = \varnothing$
$q' = p_{n+1}(q)$
For each $q'' \in \delta(q', a)$ do
For each $i \in [n]$ do
If $p_i(q) = F_i$ then $Q_i = \varnothing$ else $Q_i = F_i \cap p_i(q) \cup \{q'\}$
$\bar{q} = (Q_1, ..., Q_n, q'')$
od;
$\delta_1(q, a) = \delta_1(q, a) \cup \{\bar{q}\}$
od;
od;

Note that (3) is a procedure defining the transition function δ_1.

(4) $F_1 = \{H_1: \text{for each } q \in H_1 \text{ and for some } i \in [n],\ p_i(q) = F_i\}$.

M visits all states of F_i infinitely many times if and only if M_1 visits H_1 infinitely many times.

To see that M and M_1 accept the same set of infinite trees, we note the following: If C is a computation of M then $p_{n+1}(C)$ is a computation of M, and every computation C of M is obtained as $p_{n+1}(C')$, for some computa-

tion C' of M_1. Intuitively, M_1 keeps track of its compuation by recording the visited state in the track associated with F_i, when the visited state belongs to F_i, and setting the track associated with F_i to empty when all states of F_i are visited. This implies that M_1 and M accept the same ω-language.

Assume that $\varnothing \in F$; then it suffices to take M_1 the automaton obtained by substituting F for 2^Q in M. To finish the proof, one can remark that if M is deterministic then M_1 is also deterministic. ▮

PROPOSITION 2.9. *For each ω-language L, the following conditions are equivalent*:

(i) *L is accepted by a deterministic Büchi automaton,*

(ii) *L is accepted by a deterministic ω10-automaton,*

(iii) *L is accepted by a deterministic ω8-automaton,*

(iv) *L is accepted by a deterministic ω7-automaton, and*

(v) *L is a limit of a rational (i.e., regular) languages.*

Proof. The equivalence between (i) and (v) is due to S. Eilenberg (1974). Since the family of ω-languages accepted by determinstic Büchi automata is closed under union, the equivalence between (i) and (ii) holds from Theorem 2.8. Obviously, (iii) is equvalent to (iv), and (ii) is equivalent to (iii). ▮

PROPOSITION 2.10. *For each ω-language L, the following conditions are equivalent*:

(i) *L is accepted by a deterministic Muller automaton,*

(ii) *L is accepted by a deterministic ω9-automaton,*

(iii) *L is accepted by a nondeterministic ω9-automaton, and*

(iv) *L is accepted by nondeterminstic Büchi automaton.*

Proof. Mc Naughton (1966) proves that for each deterministic Muller's automaton, an equivalent Büchi automaton can be constructed. This implies that (i) is equivalent to (iv). To finish the proof, one can use the fact that each ω-language accepted by a nondeterministic ωi-automaton is a projection of an ω-language accepted by a deterministic ωi-automaton and the fact that regular ω-languages are closed under projection. ▮

Let DRi be the family of sets accepted by complete deterministic ωi-automata. Our first aim is to generalize these conditions to tree automata and compare them with both Büchi's automata and Muller automata on inifinite trees. For this we shall recall some results obtained

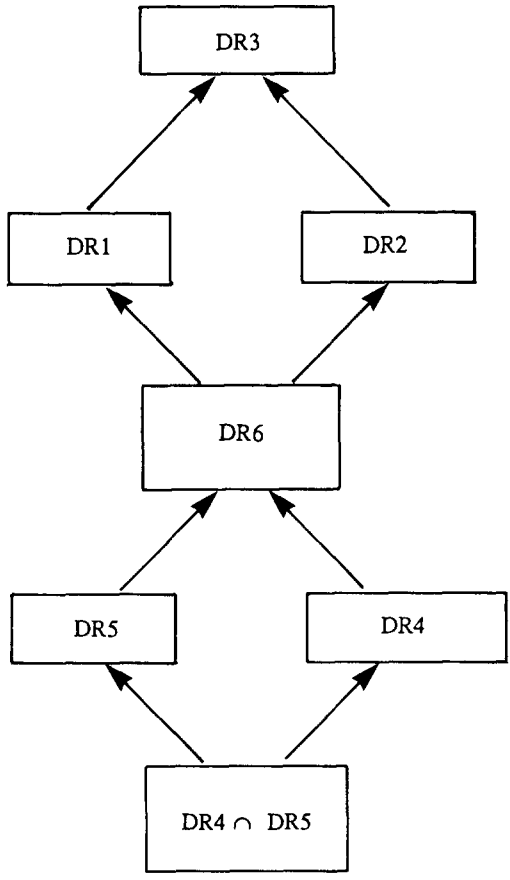

FIGURE 1

by M. Takahashi and H. Yamasaki (1983). They obtained a hierarchy of deterministic ω-languages, which can be represented by Fig. 1.

3. TOP-DOWN AND BOTTOM-UP AUTOMATA ON INFINITE TREES

It is well known that for finite trees, the regular tree grammars, the top-down tree automata, and the deterministic bottom-up tree automata define the same sets of finite trees. We shall prove that this is also the case for infinite trees.

DEFINITION 3.1. A *top-down k-ary ω-tree automaton* is a structure $M = \langle Q, \Sigma, q_0, \delta, F \rangle$, where Q, Σ, q_0, F are defined as before, and $\delta: Q \times \Sigma \to 2^{Q^k}$ is the transition function.

A *computation* of a top-down k-ary ω-tree automaton M on the infinite tree t is the mapping from $[k]^*$ to Q, where $C(\lambda) = q_0$ and for each node u, $((C(u1), ..., C(uk)) \in \delta(C(u), t(u))$. A computation C is called i-accepted if for each branch of this computation, there is a designated set of states S such that this branch satisfies the condition C_i. We define a *Di-automaton* to be a top-down k-ary ω-tree automaton with i-acceptance. M is called deterministic if for each state q and each input symbol a, there is at most one transition in $\delta(q, a)$.

PROPOSITION 3.2. *For each* $i \in \{1, ..., 9\}$, *the family of sets accepted by deterministic Di-automata is properly included in the family of those accepted by Di-automata.*

Proof. Let $K = a(b^\omega, b^\omega, ..., b^\omega, c^\omega) + a(c^\omega, b^\omega, ..., b^\omega)$, where x^ω is the unique infinite tree on $\{x\}$. One can easily construct a *Di*-automaton accepting K. Now, assume that K is accepted by a deterministic *Di*-automaton M. Let $C1$ (resp $C2$) be the unique computation of M on the tree $a(b^\omega, ..., c^\omega)$ (resp. $a(c^\omega, ..., b^\omega)$). $C1$ and $C2$ are i-accepted. Let $C1 = q_0(A1, ..., A2)$ and $C2 = q_0(A3, ..., A4)$. Since M is deterministic, the tree $q_0(A1, ..., A4)$ is a computation of M on the tree $a(b^\omega, ..., b^\omega)$, which is i-accepted; this contradicts the fact that $a(b^\omega, ..., b^\omega)$ is not a member of K. ∎

PROPOSITION 3.3. *For each* $i \in \{1, ..., 9\}$, *the family of sets accepted by Di-automata is closed under union.*

This can be easily proved using the classical construction. It depends on the fact that for each pair (M_1, M_2) of *Di*-automata, one can easily construct a *Di*-automaton which can simulate M_1 and M_2.

DEFINITION 3.4a. A *bottom-up k-ary ω-tree automaton* is a structure $M = \langle Q, \Sigma, \delta, T, F \rangle$, where Q, Σ, F are defined as before, $\delta: Q^k \times \Sigma \to 2^Q$ is the transition function, and T is the set of terminal states.

A *computation* of M on the tree t is a mapping from D to Q, such that $C(\lambda) \in T$, and for each node u, $C(u) \in \delta((C(u1), ..., C(uk)), t(u))$. An *Ai*-automaton is a bottom-up automaton which accepts by terminal state at the root and by i-acceptance on each infinite branch. An *Ai*-automaton is called deterministic if and only if for each tuple q, and each input symbol a, there is at most one transition from (q, a).

We will now define some classes of regular tree grammars, which generate the same classes as those accepted by *Di*-automata.

DEFINITION 3.4b. An *ω-regular tree grammar is a structure* $G = \langle V, \Sigma, v_0, R, V_{\text{inf}} \rangle$, where V is the set of nonterminal symbols, Σ is the

alphabet, v_0 is the start symbol, R is the set of rewriting rules of the form $v \to f(v_1, ..., v_k)$, and where $f \in \Sigma$ and v, $v_i \in V$, and V_{inf} is the set of designated sets of nonterminals.

An ω-regular tree grammar is said to be deterministic if and only if when $v \to a(v_1, ..., v_k)$ and $v \to a(\bar{v}_1, ..., \bar{v}_k)$ are rules then $v_i = \bar{v}_i$.

Let us define the notion of a derivation tree associated with an infinite tree for a given ω-regular tree grammar. An infinite derivation of G is an infinite sequence $(t_i)_{i \geqslant 0}$ satisfying the conditions:

(i) $t_0 = v_0$, and

(ii) for each $i \geqslant 0$, $t_i \Rightarrow t_{i+1}$ if and only if there exists a node $u \in \text{dom}(t_i)$ and a rule $v \to t$ satisfying the conditions:

(1) $t_i(u) = v$;

(2) $\text{dom}(t_{i+1}) = \text{dom}(t_i) \cup u \cdot \text{dom}(t)$;

(3) for each $x \in \text{dom}(t_{i+1})$, we have the conditions:

- $t_{i+1}(x) = t_i(x)$, if $x \in \text{dom}(t_i) - \{u\}$
- $t_{i+1}(x) = t(x/u)$ otherwise.

We suppose that $\bigcup_{i \geqslant 0} \text{dom}(t_i)$ is the domain of an infinite tree and let T be the limit of the sequence $(\bar{t}_i)_{i \geqslant 0}$, where $\bar{t}$ is the restriction of t to $\text{dom}(t) - \text{Fr}(t)$. One can associate the sequence $(d_i)_{i \geqslant 0}$ with the sequence $(t_i)_{i \geqslant 0}$, which is defined as

(1) $d_0 = \langle T(\lambda), v_0 \rangle$,

(2) For each $i \geqslant 0$ we have the conditions:

- $\text{dom}(d_i) = \text{dom}(t_i)$
- $d_{i+1}(x) = d_i(x)$, if $u \in \text{dom}(d_i)$
- $d_{i+1}(x) = \langle T(x), t_{i+1}(x) \rangle$, otherwise.
- Range $(d_i) = \Sigma \times V$.

The derivation tree, associated to the infinite derivation (t_i), is the infinite tree $p_2(\sigma)$, where σ is the limit of the sequence (d_i).

EXAMPLE 1. Let $k = 2$ and $G = \langle V, \Sigma, v_0, R, V_{\text{inf}} \rangle$ be the regular tree grammar where $V = \{v_0, v_1, v_2, v_3, v_4\}$, $\Sigma = \{a, b\}$, $V_{\text{inf}} = \{\{v_1\}, \{v_2\}\}$, and $R = \{v_0 \to a(v_1, v_2) + a(v_3, v_4),\ v_1 \to b(v_1, v_1),\ v_2 \to c(v_2, v_2),\ v_3 \to c(v_3, v_3),\ v_4 \to c(v_3, v_4)\}$.

Let $\delta = v_0 \Rightarrow a(v_1, v_2) \Rightarrow a(b(v_1, v_1), v_2) \Rightarrow a(b(v_1, v_1), c(v_2, v_2)) \Rightarrow \cdots$. The infinite derivation of $a(b^\omega, c^\omega)$ is $v_0(v_1^\omega, v_2^\omega)$.

A tree t is i-generated by G if there is an infinite derivation in G, having as a derivation tree (i.e., a trace of this derivation), a tree satisfying condi-

tion (i) according to V_{inf} instead of F. An *ωi-regular tree grammar* is an ω-regular tree grammar, using the ith condition to generate trees.

PROPOSITION 3.5. *For each* $i \in \{1, ..., 9\}$ *and for each set* K *of infinite trees, the following conditions are equivalent*:

(1) K *is generated by an ωi-regular tree grammar, and*

(2) K *is accepted by a Di-automaton.*

The equivalence between (1) and (2) is due to the natural correspondence between regular tree grammars and tree automata.

PROPOSITION 3.6. *For each* $i \in \{1, ..., 9\}$ *and for each set* K *of infinite trees, the following conditions are equivalent*:

(i) K *is accepted by a deterministic Di-automaton, and*

(ii) K *is generated by a deterministic ωi-regular tree grammar.*

PROPOSITION 3.7. *For each* $i \in \{1, ..., 9\}$ *and for each set* L *of infinite trees, the following conditions are equivalent*:

(i) L *is accepted by a Di-automaton, and*

(ii) L *is a projection of a set accepted by a deterministic Di-automaton.*

Proof. Since the family of sets accepted by *Di*-automata is closed under projection, it follows that condition (ii) implies condition (i). Let $M = \langle Q, \Sigma, q_0, \delta, F \text{ or } K \rangle$ be a *Di*-automaton accepting L. Now we shall exhibit a deterministic *Di*-automaton $M_1 = \langle Q_1, \Sigma_1, \delta_1, F_1 \text{ or } K_1 \rangle$ and a projection p, such that the set accepted by M is a projection by p of the set accepted by M_1.

Construction. (i) $Q_1 = Q$

(ii) $\Sigma_1 = \Sigma \times Q \times [m]$, where $m = \text{Max}(\text{Card}(\delta(q, a)))$

(iii) $q_1 = q_0$

(iv) For each $q \in Q$ and $a \in \Sigma$ do

If $\delta(q, a) = \{(q_1^1, ..., q_1^k), ..., (q_n^1, ..., q_n^k)\}$ then $\delta_1(q, \langle a, q, i \rangle) = (q_i^1, ..., q_i^k)$

(v) $F_1 = F$ or $K_1 = K$.

It is clear that $L^\omega(M_1) = p_1(L^\omega(M))$. ∎

THEOREM 3.8. *For each* $i \in \{1, ..., 9\}$ *and for each set* K *of infinite trees, the following conditions are equivalent*:

(1) *K is accepted by a Di-automaton,*

(2) *K is accepted by an Ai-automaton, and*

(3) *K is accepted by a deterministic Ai-automaton.*

Proof. The equivalence between (1) and (2) is a consequence of the facts:

(i) The family of sets accepted by *Di*-automata is closed by union (i.e., Proposition 3.3).

(ii) Each *Di*-automaton can be simulated by an *Ai*-automaton with one terminal state and each *Ai*-automaton with one terminal state can be simulated by a *Di*-automaton.

Let $M = \langle Q, \Sigma, \delta, T, F \rangle$ be an *Ai*-automaton accepting *K*. Now we shall exhibit a deterministic *Ai*-automaton $M' = \langle Q', \Sigma, \delta', T', F' \rangle$ equivalent to *M*.

Construction. (1) $Q' = Q \times Q \times \cdots \times Q$

(2) for each $\{q_1, ..., q_k, q, q'\} \subseteq Q$ land $a \in \Sigma$ do

If $q \in \delta(\langle q_1, ..., q_k \rangle, a)$ then $\langle q, q' \rangle = \delta'((\langle q_1, q \rangle, \langle q_2, q \rangle, ..., \langle q_k, q' \rangle), a)$

(3) $T' = T \times Q$

(4) $F' = \{A \subseteq Q' : p_1(A) \in F\}$.

If C' is an *i*-accepted computation of M', then $p_1(C')$ is a computation of *M* which is also *i*-accepted. For each *i*-accepted computation *C* of *M* on *t*, one can easily exhibit an *i*-accepted computation of M' on *t*. ∎

A *Rabin automaton* is a structure $M = \langle Q, \Sigma, q_0, \delta, \Omega \rangle$, where Q, Σ, q_0, δ are defined as before and $\Omega = \{(U_i, L_i) : 1 \leqslant i \leqslant n, L_i, U_i \subseteq Q\}$. A Rabin's automaton *M* accepts an infinite tree iff it has a computation such that for each infinite branch Π of this computation, there is a pair (U_i, L_i) such that $\mathrm{Inf}(\Pi) \cap U_i \neq \varnothing$ and $\mathrm{Inf}(\Pi) \cap L_i = \varnothing$.

It is well known that the emptiness problem for Rabin automata is non-trival. M. O. Rabin improved his original proof of the decidability of this emptiness problem, but even the second proof does not yield a simple effective criterion for deciding emptiness. R. Hosseley (1972) provided such criterion by showing that Rabin's automaton accepts an infinite tree if and only if there is a computation of the automaton containing a certain kind of finite subtree. In R. Hosseley and C. Rackoff (1972) the emptiness problem of Rabin's automata is reduced to the emptiness problem of finite tree automata.

Let us mention that A. Mostowski (1982) proved that nondeterministic top-down Rabin's automata and deterministic bottom-up Rabin automata accept the same sets. In A. Mostowski (1984) defines a standard form of

Rabin automaton is defined and it is proved that each Rabin automaton is equivalent to a Rabin automaton in standard form.

PROPOSITION 3.9. *For each* $i \in \{1, 2, ..., 9\}$, *the family of sets accepted by deterministic Di-automata is not closed under union.*

Proof. Let $L_1 = a(b^\omega, ..., b^\omega, c^\omega)$ and $L_2 = a(c^\omega, ..., c^\omega, b^\omega)$; then one can easily construct a deterministic *Di*-automaton M_i $(i = 1, 2)$ which recognizes L_i. We have seen that $L_1 \cup L_2$ cannot be accepted by any deterministic *Di*-automaton. ∎

4. CONTROLLED TREE AUTOMATA ON INFINITE TREES

In this section, we will introduce some kind of tree automata, called controlled tree automata, and then we compare them with the above tree automata.

DEFINITION 4.1. A *controlled k-ary ω-tree automaton*, called *C*-automaton for short, is a structure $M = \langle Q, \Sigma, q_0, \delta, K \rangle$, where Q, Σ, q_0, δ are defined as for top-down tree automata and K is an ω-language on Q, called a control language.

A *Ci*-automaton is a *C*-automaton, where the control ω-language is accepted by a deterministic ωi-automaton. A computation C of the *Ci*-automaton M, is called accepted if the control language of M contains all infinites sequences lying on it's branches.

THEOREM 4.2. *For each* $i \in \{1, ..., 9\}$ *and for each set K of infinite trees, the following conditions are equivalent*:

(1) *K is accepted by a nondeterministic* (*rep. deterministic*) *Di-automaton,*

(2) *K is accepted by a nondeterministic* (*resp. deterministic*) *Ci-automaton.*

Proof. That part (1) implies (2) is obvious. Let $M_c = \langle Q, \Sigma, q_0, \delta, L \rangle$ be a *Ci*-automaton accepting K and let $A = \langle S, Q, s_0, \delta_c, F \rangle$ be the deterministic ωi-automaton accepting L.

Now we shall exhibit a *Di*-automaton $M_1 = \langle Q_1, \Sigma, q_1, \delta_1, F_1 \rangle$ accepting K.

Construction. (1) $Q_1 = S \times Q$

(2) $q_1 = (s_0, q_0)$

(3) For each $(s, q) \in Q_1$ and $a \in \Sigma$ do

If $(q_1, ..., q_k) \in \delta(q, a)$ then $((\delta_c(s, q), q_1), ..., (\delta_c(s, q), q_k)) \in \delta_1((s, q), a)$.

(4) $F_1 = \{A \subseteq Q_1 : p_1(A) \in F\}$.

One can easily prove that M_c and M_1 are equivalent (i.e., they accept the same set). Note that if M_c is deterministic, M_1 is also deterministic. ∎

Remark. It follows from the last theorem that the diagram defined for deterministic ω-automata in Fig. 1 is valid for both deterministic and nondeterministic *Di*-automata.

DEFINITION 4.3. A *Ci-regular k-ary ω-tree grammar* is a structure $G = \langle V, \Sigma, v_0, R, K \rangle$, where:

(i) V, Σ, v_0, R are defined as before, and

(ii) $K \subseteq V^\omega$ is a set accepted by a deterministic ωi-automaton.

An infinite tree t is generated by a *Ci*-regular tree grammar if and only if t has a derivation tree such that all its infinite branches belong to K. From results obtained before, one can easily prove the result:

PROPOSITION 4.4. *For each infinite tree set L and for each $i \in \{1, ..., 9\}$, the following conditions are equivalent*:

(i) *L is generated by a nondeterministic (resp. deterministic) Ci-regular tree grammar, and*

(ii) *L is accepted by a nondeterministic (resp. deterministic) Di-automaton.*

Now we shall give characterizations of rational infinite tree sets in terms of automata and tree grammars. For this, we first define the class of rational infinite tree sets.

A set L of infinite trees is said to be rational if and only if there is a sequence $(L_i)_{0 \leq i \leq n}$ of regular sets of finite trees such that $L = L_0 \langle L_1, ..., L_n \rangle^\omega$.

THEOREM 4.5. *For each infinite tree set L, the following conditions are equivalent*:

(i) *L is rational,*

(ii) *L is accepted by D1-automaton,*

(iii) *L is accepted by a D10-automaton,*

(iv) *L is accepted by a D7-automaton, and*

(v) *L is accepted by a D8-automaton.*

The equivalence between (i) and (ii) is proved in M. Nivat and A. Saoudi (1985). The quivalence between (ii), (iii), (iv), and (v) is easy to prove using results obtained in the last section.

Remark. The equivalence between (i) and (ii) is a Kleene's theorem generalized to infinite tree sets accepted by a $D1$-automaton.

Now we shall give a Kleene theorem corresponding to the family of infinite tree sets accepted by a $D4$-automaton.

THEOREM 4.6. *For each infinite tree set L, the following conditions are equivalent:*

(i) *L is accepted by a complete nondeterministic (resp. deterministic) $D4$-automaton, and*

(ii) *$L = L_0\langle T_\Sigma^\omega, \ldots, T_\Sigma^\omega\rangle$, for some set of finite trees L_0 accepted by nondeterministic (resp. deterministic) top-down tree automata.*

The proof can be obtained from constructions used in M. Nivat and A. Saoudi (1985) to prove the equivalence between rational sets and those accepted by a nondeterministic $D1$-automaton.

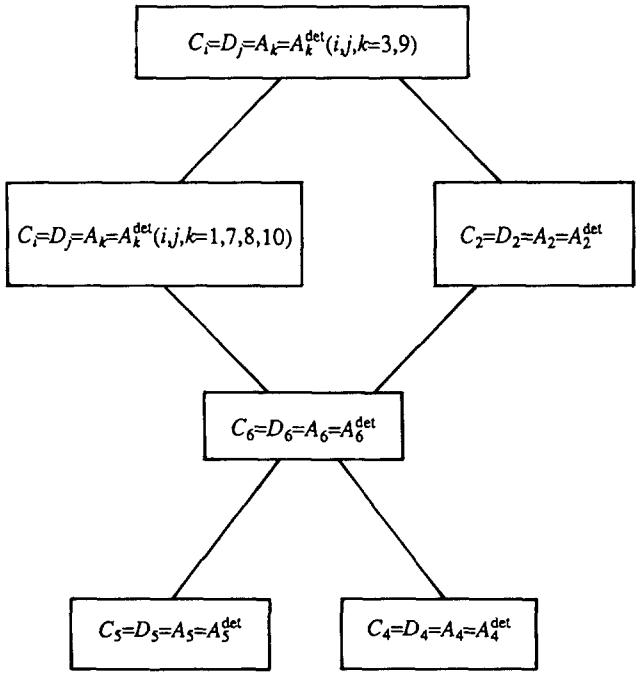

FIGURE 2

FIGURE 3

Let X_i (resp. X_i^{det}) be the family of sets accepted by nondeterministic (resp. deterministic) X_i-automaton over infinite trees, where $i=1, 10$ and $X=D, A, C$. Then the followng charts (i.e., Figs. 2 and 3) summarize the previous results.

5. Rational Program Schemes and Logic Programming

A *rational program schema*, called rational program for short, is a structure $P=\langle \Sigma, V, v_1, E\rangle$, where Σ is the set of functional symbols, V is the set of variables, v_1 is the start symbol, and E is the set of equations of the form: $v_i=t_1+\cdots+t_n$, where t_i is a term on $\Sigma \cup V$. A rational program can be viewed as a recursive program, where each variable identifies a nondeterministic procedure, and v_1 is the principal procedure. We can also view a rational program as regular tree grammar.

EXAMPLE 2. Let B be the program:

$$v_1=f(v_2, v_2)+f(v_3, v_3)$$
$$v_2=f(v_2, v_2)$$
$$v_3=b$$

then the solution of P is $\langle\{f(b,b), f^\omega\}, \{f^\omega\}, \{b\}\rangle$, where $k=2$, $\Sigma_2=\{f\}$, and $\Sigma_0=\{b\}$. The set of computations of P is $f^\omega+f(b,b)$.

A finite (resp. infinite) computation of the program P is a finite (resp. infinite) tree on Σ, obtained as an element of the first component of the solution of E.

A rational program is called *deterministic* if E is of the form: $v_i=t_i$. Note that a deterministic rational program has at most one computation. Two rational program are said to be C-equivalent (resp. D-equivalent) if they compute the same set of finite (resp. infinite) objects (i.e., trees). Two programs are said to be equivalent if the compute the same things. We will give a forma definition of a rational logic program.

DEFINITION 5.1. A *rational logic program* is a structure $S = \langle P, \Sigma, X, F, C, G \rangle$, where:

(1) P is the set of unary predicates,

(2) Σ is the set of functional symbols,

(3) X is the set of individual variables,

(4) F is the set of facts of the form $A(a) \leftarrow$, where $a \in \Sigma$ and $A \in P$,

(5) C is the set of the clauses of form: $A(f(x_1, ..., x_n)) \leftarrow B_1(x_1), ..., B_n(x_n)$, and

(6) G is a goal of the form $\leftarrow B(x)$.

We will give an example of rational logic program which computes the set of trees on $\{f, a, b, g\}$.

EXAMPLE 3.

(1) **Facts**:
$Tree(a) \leftarrow$
$Tree(b) \leftarrow$

(2) **Clausal procedures**:
$Tree(f(x_1, x_2)) \leftarrow Tree(x_1), Tree(x_2)$
$Tree(g(x_1, x_2)) \leftarrow Tree(x_1), Tree(x_2)$

(3) **Goal**:
$\leftarrow Tree(x)$

The meaning of a rational logic program P is the set of its computations (i.e., trees) deducible from P and satisfying the predicate X. Intuitively, a rational logic program P can be viewed as a classical program such that the set of its facts defines the inputs, the set of its clausal procedures defines the statements, and the goal defines the output. For more details about the interpretation of Horn clauses see R. Kowalski (1972).

S is called deterministic if when $P(t_1) \leftarrow B_1(x_1), ..., B_n(x_n)$, $P(t_2) \leftarrow A_1(x_1), ..., A_p(x_p) \in C$ then $t_1 = t_2$, $n = p$, $A_i = B_i$. On the other hand, a deterministic logic program has at most one computation.

Now we shall prove that a rational program and rational logic program have the same computing power.

THEOREM 5.2. *For each set K of trees, the following conditions are equivalent*:

(1) *K is computed by a nondeterministic (resp. deterministic) rational program, and*

(2) *K is computed by a nondeterministic (resp. deterministic) rational logic program.*

Proof. Since a rational program can be viewed as a regular tree grammar, one can easily transform it to an equivalent one in normal form. By normal form, we mean that each rule is of the form $A \to f(B_1, ..., B_n)$ or $A \to a$. It suffices to give the correspondence between rational logic programs and rational tree grammars in normal form:

(1) $A(a) \leftarrow$ iff $A \to a$ is a rule

(2) $A(f(x_1, ..., x_n)) \leftarrow B_1(x_1), ..., B_n(x_n)$ is a clause iff $A \to f(B_1, ..., B_n)$ is a rule

(3) $\leftarrow A(x)$ is a goal iff A is an axiom.

To finish the proof, one can use the facts:

(i) $B \Rightarrow^* t$ if and only if $B(t)$, and

(ii) $B \Rightarrow^\omega t$ if and only if $B(t)$. ∎

It is well known that the equivalence problem for two finite tree automata is decidable. This gives us the dedidability of the C-equivalence of two rational logic programs. Let us recall some results due to M. O. Rabin (1969), who characterizes SkS in terms of tree automata and then obtains the decidability of SkS. It follows from Rabin's results that the equivalence of two $D3$-automata (i.e., Rabin's automata) is decidable.

PROPOSITION 5.3. *It is solvable, whether for two rational logic programs* P_1 *and* P_2:

(1) P_1 *is C-equivalent to* P_2,

(2) P_1 *is D-equivalent to* P_2, *or*

(3) P_1 *is equivalent to* P_2.

Proof. The C-equivalence problem of two rational logic programs, can be reduced to the equivalence problem of two finite tree automata on finite trees, which is solvable. The D-equivalence problem of two rational logic programs, can be reduced to the equivalence problem of two Rabin automata, which is solvable. ∎

6. COMPUTABLE INFINITE TREE SETS AND E_n -FORMULAS

Let $T = \langle D, v, s_1, ..., s_k, \leqslant \rangle$, where v is the constant for the empty word, $s_1, ..., s_k$ is the successor function such that $s_i(u) = ui$, and $\leqslant$ is the prefix relation on the tree domain.

Let $\Sigma = \{a_1, ..., a_n\}$ be an alphabet and $P_1, ..., P_n$ be the unary predicates such that $P_i(u) =$ True iff $t(u) = a_i$, where t is an infinite tree on Σ.

An L_2-*formula* is a formula built up constants, successor functions, unary predicates, individual variables ranging over D, set variables ranging over subsets of D, quantifiers, and connectors. Rabin (1969) proves that for each L_2-formula, one can construct a nondeterministic Rabin automaton (i.e., $D3$-automaton), that accepts exactly the trees satisfying this formula.

A tree satisfying a formula ψ is called a model of ψ. We denote by Model(ψ) the set of infinite trees satisfying ψ. Now we will define a subclass of L_2-formulas ψ such that:

(1) Model(ψ) is accepted by some Di-automata,

(2) Model(ψ) is computed by a rational logic program, and

(3) Model(ψ) is computed by a rational program.

An E_n-formula is a formula of the form: $\exists X_1 \cdots \exists X_n$: Part$(X_1, ..., X_n) \wedge$ Root$(X_{i_0}) \wedge$ Trans$(X_1, ..., X_n)$, where:

(i) Part$(X_1, ..., X_n) = \forall x \bigwedge_{i \neq j} X_i x \rightarrow \neg X_j x$

(ii) Root$(X_{i_0}) = X_{i_0} v$

(iii) Trans$(X_1, ..., X_n) = \forall x \bigwedge_{i,j} (X_i x \wedge P_j x) \rightarrow T_{i,j}$, where $T_{i,j} = \bigvee_{l=1}^{n_{ij}} X_{1,l} s_1 x \wedge \cdots \wedge X_{k,l} s_k x$.

Note that x is an individual variable, X_i is a set variable, and Xx means that x belongs to X.

Remark. By interpreting X_i as a state of a nondeterministic top-down tree automaton, an E_n-formula represents the transtitions of this automaton and defines exactly trees having at least one computation (i.e., run). ▮

An E_n-formula ψ is reduced to ϕ if and only if ψ and ϕ have the same models and the set of set variables of ψ contains the set of set variables of ϕ. An E_n-formula ψ is called minimal if it cannot be reduced to a smaller one.

THEOREM 6.1. *For each infinite tree set L, the following conditions are equivalent*:

(1) *L is definable by an E_n-formula,*

(2) *L is accepted by a Di-automaton with $F = 2^Q$,*

(3) *L is generated by an ωi-regular tree grammar with $V_{\inf} = 2^V$,*

(4) *L is computed by a rational program with $\Sigma = \Sigma_k$,*

(5) *L is computed by a rational logic program, with $\Sigma = \Sigma_k$, and*

(6) *L is equal to the adherence of a finite set accepted by a top-down tree automaton.*

Proof. (1) is equivalent to (2) by the natural corresppondence between top-down tree automata and E_n-formula. The equivalence between (2), (3)., and (4), and (5) is a consequence of Theorem 5.2 and Proposition 3.4. The equivalence between (2) and (6) can be proved without difficulty. ▮

It is well known from W. Brainerd (1968) that for each bottom-up finite tree automaton, there is an equivalent minimal automaton. To prove the effectiveness of constructing a minimal E_n-formula which is equivalent to a given E_n-formula, we need some results.

PROPOSITION 6.2. *The set of initial subtrees of a set accepted by a Rabin automaton is accepted by a finite tree automaton.*

One can take a Rabin automaton and reduce it by eliminating states which are not reachable from the initial state and then use the emptiness algorithm of Rackoff and Hosseley (1972) to eliminate states which cannot accept any tree. From the resulting automaton, it suffices to consider any state as a terminal state for accepting the initial subtrees of the initial set.

PROPOSITION 6.3. *The center of a set accepted by a Rabin automaton is accepted by a finite tree automaton on finite trees.*

The proof is left as an exercise.

PROPOSITION 6.4. *For each E_n-formula ψ, a minimal equivalent formula can be effectively computed.*

Proof. Using Theorem 6.1, one can construct a *Di*-automaton M, with $F = 2^Q$ accepting $\text{Model}(\psi) = L$. Since the center of L is accepted by a bottom-up automaton on finite trees, one can minimize the obtained automaton and then obtain an equivalent bottom-up automaton with the same states. Finally, from the last automaton we can construct a minimal formula having L as the set of its models. ▮

THEOREM 6.5. *Whether an L_2-formula is equivalent to an E_n-formula can be solved.*

Proof. Let ψ be an E_n-formula such that $\text{Model}(\psi) = L$, then ψ is equivalent to an E_n-formula if and only if $\text{Adh}(\text{Center}(L))$ is equal to L. This problem can be reduced to the equivalence problem of two *Di*-automata, which is solvable. ▮

On the other hand, the last theorem gives us a procedure for testing if a set of models of an L_2-formula is computable by a rational program.

ACKNOWLEDGMENTS

It is a pleasure to thank the anonymous referee for their helpful comments and suggestions.

RECEIVED October 29, 1987; ACCEPTED November 17, 1988

REFERENCES

ARNOLD A., AND NIVAT, M. (1980a), Formal computations of nondeterministic recursive programm schemes, *Math. Systems Theory* **13**, 219–236.

ARNOLD, A., AND NIVAT, M. (1980b), The metric space of infinite trees: Algebraic and topological properties, *Fund. Inform.* **4**, 445–476.

BEN-ARI, MANNA, Z., AND PNUELLI, A. (1981), The temporal logic of branching time *in* "ACM Symposium on Principles of Programming Languages," pp. 164–176.

BRAINERD, W. (1968), The minimization of tree automata, *Inform. and Control* **13**, 484–491.

BÜCHI, J. R. (1960), On a decision method in restricted second order arithmetic, *in* "Proceedings, Cong. Logic Method and Philos. of Sci., Univ. Press, Stanford, CA.

CHANDRA, A., KOZEN, D., AND STOCKMEYER, L. (1981), Alternation, *J. Assoc. Comput. Mach.* **28**, 114–133.

COHEN, R. S., AND GOLD, A. Y., (1977), Theory of ω-languages I and II: A study of various models of ω-type generation and recognition, *J. Comput. System Sci.* **15**, 185–208.

CHOUEKA, Y. (1974), Theories of ω-automata on ω-tapes: A simplified approach, *J. Comput. System Sci.* **8**, 107–114.

COURCELLE, B. (1985), Fundamental properties of infinite trees, *Theoret. Comput. Sci.* **25**, 95–169.

EILENBERG, S. (1974), "Automata, Languages and Machines," Vol. A, Academic Press, New York/London.

EMERSON, A. AND SISTLA, P. (1984), Deciding branching time logic, *in* "Proceedings, 16th ACM Symp. on Theory of Computing," pp. 14–24.

ENGELFRIET, J. (1975), "Tree Automata and Tree Grammars," Report, Daimi FN-10, University of Aarhus.

GUESSARIAN, I. (1983), On push-down tree automata, *Math. Systems Theory* **16**, 237–263.

GUERIVICH, Y., AND HARRINGTON, L., (1982), Trees, automata, and games *in* "Proceedings, 14th ACM Symp. on Theory of Computing," pp. 237–263.

HAREL, D. (1979), "First-Order Dynamic Logic," Lecture Notes in Computer Science, Vol. 68, Springer-Verlag, Berlin.

HOSSELEY, R., AND RACKOFF, C. (1972), The emptiness problem for automata on infinite trees, *in* "13th IEEE on Switching and Automata Theory, pp. 121–124.

HOSSELEY, R. (1972), Finite tree automata and ω-automata, M.S. thesis, MAC TR 102, MIT.

KFOURY, A. J. (1985), Definability by deterministic and nondeterministic Programs, *Inform. and Control* **65**, Nos. 2/3, 98–121.

KOWALSKI, R. (1972), Algorithm = Logic + Control, *Comm. ACM* **22**, No. 7, 424–436.

MEYER, A. R., AND PARIKH, R. (1981), Definability in dynamic logic, *J. Comput. Sci.* **23**, 279–298.

MOSTOWSKI, A. W. (1982), Determinancy of sinking automata on infinite trees and inequalities between various Rabin's pair indices *Inform. Proc. Lett.* **15**, No. 4, 159–163.

MOTOWSKI, A. W. (1984), Regular Expressions for Infinite Trees and Standard Form of Automata," Lecture Notes in Computer Science, Vol. 208, Springer-Verlag, Heidelberg.

MULLER, D. E. (1968), Infinite sequences and finite machines, *in* "Proceedings, 4th IEEE on Switching Circuit Theory and Logical Design," pp. 3–16.

MULLER, D. E., AND SCHUPP, P. E. (1984), Alternating automata on infinite objects, determinancy and Rabin's theorem, *in* "Actes de l'école de printemps d'Informatique Théorique," Lecture Notes in Computer Science, Vol. 192 (D. Perrin and M. Nivat, Eds.), pp. 100–107, Springer-Verlag, Berlin.

MULLER, D., SAOUDI, A., AND SCHUPP, P. (1986), Alternating automata, the weak monadic theory and its complexity, *in* "13th Inter. Colloquium Automata, Languages and Programming," Lecture Notes in Computer Science, Vol. 226, Springer-Verlag, Berlin.

MC NAUGHTON, (1966), Testing and generating infinite sequences by a finite automaton, *Inform. and Control*, **9**, 521–530.

NIVAT, M. (1979), Infinite words, infinite trees and infinite computations, *in* "Foundations of Computer Science III, part 2," Math. Center Tract 109, Math Centrum, Amsterdam.

NIVAT, M., AND PERRIN, D. (1984), Ensembles reconnaissables de mots bi-infinis, *in* "Proceedings, 14th ACM Symp. on Theory of Computing," pp. 47–59.

NIVAT, M., AND SAOUDI, A. (1985), "Rational, Recognizable and Computable Languages," Report L.I.T.P. 85-75, University of Paris 7.

PARK, D. (1981), Concurrency and automata on infinite sequences, *in* "5th GI Conference," Lecture Notes in Computer Science, Vol. 104, pp. 167–183, Springer-Verlag, Berlin.

RABIN, M. O. (1969), Decidability of second theories and automata on infinite trees, *Trans. Amer. Math. Soc.* **141**, 1–35.

RABIN, M. O. (1970), Weakly definable relations and special automata, *in* "Math. Logic and Foundations of Set Theory" Y. Bar Hillel, Ed.), pp. 1–23, North-Holland, Amsterdam.

SAOUDI, A. (1984), Infinite tree languages recognized by ω-automata, *Inform. Proc. Lett.* **18**, 15–19.

SAOUDI, A. (1986), Varitétés d'automates descendants d'arbres infinis, *Theoret. Comput. Sci.* **43**, 1–21.

SCHUTZENBERGER, M. P. (1973), Sur les relations rationelles fonctionelles, *in* "Col. Automata, Languages, and Programming" (M. Nivat Ed.), pp. 103–114, North-Holland, Amsterdam.

TAKAHASHI, M. (1985), The greatest fixed point and rational ω-tree languages, Report L.I.T.P. 85–69, University of Paris 7.

TAKAHASHI, M., AND YAMASAKI, H. (1983), A note on ω-regular languages, *Theoret. Comput. Sci.* **23**, 217–225.

THATCHER, B. J. W., AND WRIGHT, J. B. (1968), Generalized finite automata theory with applications to a decision problem of second order logic, *Math. Systems Theory* **2**, 57–81.

THOMAS, W. (1982), A hierarchy of sets of infinite trees, *in* "GI Conference," Lecture Notes in Computer Science, Vol. 145, pp. 335–342, Springer-Verlag, Berlin.