INFORMATION AND COMPUTATION 113, 278–299 (1994)

Deciding Emptiness for Stack Automata on Infinite Trees

DAVID HAREL AND DANNY RAZ

Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot 76100, Israel

We show that the emptiness problem for Büchi stack automata on infinite trees is decidable in elementary time. We first establish the decidability of the emptiness problem for *pushdown* automata on infinite trees. This is done using a pumping-like argument applied to computation trees. We then show how to reduce the emptiness problem for stack automata to the emptiness problem for pushdown automata. Elsewhere, we have used the result to establish the decidability of several versions of nonregular dynamic logic.

1. INTRODUCTION

Automata on infinite trees have attracted much interest ever since the early work of Rabin [R1, R2]. Various acceptance criteria have been proposed, analogous to those for automata on infinite words (ω-automata), and the relationships between them have been investigated. Also, certain problems on such automata, such as emptiness, have been shown to be decidable [R1, R2, HoR].

A particularly interesting direction has been the use of tree automata in proving decidability of various logics. Following Rabin's work on weak S2S [R2], many researchers have used automata-theoretic techniques in obtaining decidability results for logics of programs. A key ingredient in such results is the search for an appropriate class of tree-automata on infinite trees with a decidable emptiness problem. In particular, Vardi and Wolper [VW] have been able to obtain several improved decision procedures for temporal and dynamic logics by reducing the satisfiability problem at hand to the emptiness problem for certain classes of tree-automata, and then finding (efficient) decision procedures for these.

Recently, automata on infinite trees have been generalized to *pushdown* automata on infinite trees, and various acceptance criteria have been investigated [S]. However, decision problems for such extensions do not seem to have been addressed. For example, it was not known whether the emptiness problem for such automata is decidable. In this paper we answer

0890-5401/94 $6.00

a stronger version of this question, by providing an elementary-time decision procedure for the emptiness problem for *stack* automata on infinite trees. Recall that a stack automaton can travel up and down the stack and inspect the symbols appearing therein, but it can make changes only at the top. A good informal description of stack automata can be found in [K] and more updated results and references can be found in [WW]. A stack automaton on infinite trees is a generalization of a pushdown automaton on infinite trees, in the same way as a stack automaton on words is a generalization of a pushdown automaton on words, i.e., in the ability of the head to travel in the stack. We limit ourselves to the Büchi acceptance criterion, namely that a computation is accepting if on every path in it there are infinitely many accepting states.

In the main part of the paper we show that the emptiness problem for Büchi pushdown automata on infinite trees is decidable (Section 3). The proof is elementary in the sense that it does not use any powerful results such as Rabin's Theorem [R1], and is established using a pumping-like argument applied in a novel way to computation trees. We then show how to reduce the emptiness problem for stack automata on infinite trees to the one for pushdown automata on infinite trees (Section 4).

Besides the relevance of this result to automata theory, it turns out that it too has applications in logics of programs, namely to the decidability of variants of nonregular propositional dynamic logic (PDL); see [HPS, H]. In fact, the present result was obtained during our work on such decision problems. Specifically, in [HR] we use it to prove that (i) PDL with the addition of any context-free language accepted by a special kind of input-oblivious pda, termed "simple-minded" therein, is decidable, and (ii) PDL with the addition of any language (even non-context-free) accepted by a deterministic "unique-prefix" stack automaton is decidable.

2. Definitions

Let $[k] = \{1, 2, ..., k\}$. A *k-ary tree* over a set S is a labeling of the set $[k]^*$ by members of S. The empty word λ denotes the root of the tree.

A *stack k-ary ω-tree automaton* (or an STA for short) is a structure

$$M = \langle Q, \Sigma, \Gamma, q_0, z_0, \delta, F \rangle,$$

where Q is a (finite) set of states, Σ is the (finite) input alphabet, Γ is the (finite) stack alphabet, $q_0 \in Q$ is the initial state, $z_0 \in \Gamma$ is the initial stack symbol, and $F \subseteq Q$ is the set of designated accepting states.

The transition function δ is defined as

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\Lambda\}) \to 2^{(Q \times B)^k} \cup 2^{Q \times B},$$

where B is a finite subset of $\{pop, md, mu, sp\} \cup \{push(w) \mid w \in \Gamma^+\}$. Here, *md*, *mu*, and *sp* stand for "move down," "move up," and "stay put," respectively. The transition function reflects the fact that M works on trees with outdegree k that are labeled by Σ. The number of rules in δ is denoted by $|\delta|$. We use Λ as the empty symbol of the stack, describing the stack positions beyond the top of the stack. This is useful when one wants to determine if the head is at the top of the stack.

The *size* of an automaton is measured by the size of its transition function. More formally, if $M = \langle Q, \Sigma, \Gamma, q_0, z_0, \delta, F \rangle$ is a stack k-ary ω-tree automaton, and $l = \max(|w|)$, where the maximum is taken over all words $w \in \Gamma^+$ appearing in the set B in the definition of δ, then the size of M is $k \cdot |\delta| \cdot l$.

The set of *stack configurations* is $S = z_0\Gamma^* \uparrow \Gamma^*$, and the *initial stack configuration* s_0 is $z_0\uparrow$. For a stack configuration $s = z_0\gamma_1 \uparrow \gamma_2$, the *depth* of s is $d(s) = |\gamma_1| + |\gamma_2|$ and the *head position* is $h(s) = |\gamma_1|$. A *node configuration* N is a pair $(q, s) \in Q \times S$, and the *initial node configuration* n_0 is (q_0, s_0). The depth of a node configuration is the depth of its stack.

Let $\mathscr{H} : S \to \Gamma \cup \{\Lambda\}$ be given by $\mathscr{H}(z_0\gamma_1 z \uparrow \gamma_2) = z$, where $z \in \Gamma \cup \{\Lambda\}$ and $\mathscr{H}(z_0 \uparrow \gamma) = z_0$. This describes the letter read by the stack head.

In order to define the effect of δ on stack configurations, we define the partial function $\mathscr{B} : (S \times B) \to S$. This function gives the new content of the stack and is defined as follows:

- $\mathscr{B}(z_0\gamma z\uparrow, pop) = z_0\gamma\uparrow$.
- $\mathscr{B}(z_0\gamma\uparrow, push(w)) = z_0\gamma w\uparrow$.
- $\mathscr{B}(z_0\gamma_1 z \uparrow \gamma_2, md) = z_0\gamma_1 \uparrow z\gamma_2$.
- $\mathscr{B}(z_0\gamma_1 \uparrow z\gamma_2, mu) = z_0\gamma_1 z \uparrow \gamma_2$.
- $\mathscr{B}(z_0\gamma\uparrow, mu) = z_0\gamma\Lambda\uparrow$.
- $\mathscr{B}(z_0\gamma\Lambda\uparrow, md) = z_0\gamma\uparrow$.
- $\mathscr{B}(s, sp) = s$.

An *ε-path* from (q_1, s_1) to (q_l, s_l) is a sequence $((q_1, s_1), ..., (q_l, s_l))$, such that for all $1 < i \leqslant l$, we have $(q_i, b_i) \in \delta(q_{i-1}, \varepsilon, \mathscr{H}(s_{i-1}))$ and $\mathscr{B}(s_{i-1}, b_i) = s_i$. Let N_ε be the set of ε-paths. An ε-path *signals* F if there is $1 \leqslant j \leqslant l$ such that $q_j \in F$ and $d(s_j) = h(s_j)$.

A *computation* of M on the infinite tree t over Σ is an extension of t, defined as $C : [k]^* \to (\Sigma \times N_\varepsilon)$, such that, in addition to a symbol from Σ, each node is labeled by an ε-path. The tree C must satisfy the following condition:

- If N is some ε-path to (q, s), then for all u in $[k]^*$ such that $C(u) = (a, N)$, there exists $((q_1, b_1), ..., (q_k, b_k)) \in \delta(q, a, \mathscr{H}(s))$, such that for all $1 \leqslant j \leqslant k$, $C(uj) = (a_j, N_j)$, and N_j is some ε-path from $(q_j, \mathscr{B}(s, b_j))$.

A computation C is said to be *Büchi accepting*, or just *accepting* for short, if $C(\lambda) = (a, N)$, where $a \in \Sigma$ and N is an ε-path from (q_0, s_0), and every path in C contains infinitely many ε-paths that signal F. A tree t is *accepted* by M if there exists an accepting computation of M on t.

The *emptiness problem* is, given an automaton M, to determine if M accepts some tree.

An automaton $M = \langle Q, \Sigma, \Gamma, q_0, z_0, \delta, F \rangle$ is termed *simple* if it has no ε-*moves*, $|\Sigma| = 1$, and for all *push*(w) in its instructions, $|w| = 1$.

Let $M = \langle Q, \Sigma, \Gamma, q_0, z_0, \delta, F \rangle$ be an STA such that for all *push*(w) in its instructions, $|w| = 1$. (By adding new states and ε-moves, every STA can be transformed into an STA that accepts the same trees and follows this condition.) We now define a simple automaton corresponding to M. Let $M' = \langle Q', \Sigma', \Gamma, q_0, z_0, \delta', F \rangle$, be defined as follows:

- $Q' = Q \cup \{q_\varepsilon \mid q \in Q\}$.
- $\Sigma' = \{a\}$.
- for every $((p_1, b_1) \cdots (p_k, b_k)) \in \delta(q, \sigma, \gamma))$, where $\sigma \neq \varepsilon$, we have $((p_1, b_1) \cdots (p_k, b_k)) \in \delta'(q, a, \gamma)$ and $((p_1, b_1) \cdots (p_k, b_k)) \in \delta'(q_\varepsilon, a, \gamma)$.
- for every $(p, b) \in \delta(q, \varepsilon, \gamma)$ we have $\underbrace{((p_\varepsilon, b) \cdots (p_\varepsilon, b))}_{k} \in \delta'(q, a, \gamma)$ and $\underbrace{((p_\varepsilon, b) \cdots (p_\varepsilon, b))}_{k} \in \delta'(q_\varepsilon, a, \gamma)$.

Clearly M' is simple.

CLAIM 1. *M accepts some tree t iff M' has an accepting computation on its unique input tree.*

Proof. The proof is straightforward, based upon the fact that M' "guesses" simultaneously both the input tree and the computation of M. It is left to the reader. ∎

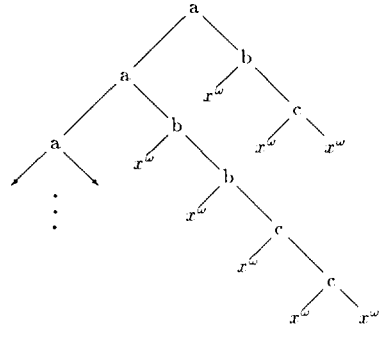

FIGURE 1

Since M' has no ε-*moves*, all the nodes in its computations are of the form $(a, (q, s))$, where (q, s) is a node configuration. The projection of this computation on the node configurations yields an infinite tree over $Q \times S$, which we call an *S-computation* of M.

We say that two node configurations n and m are *connected* in M if there is an S-computation C of M such that $n, m \in C$ and there is a path from n to m in C. If n and m are connected, and there is a node (q, s) on the path from n to m in C such that $q \in F$, we say that the pair n, m *signals* F.

An STA that uses only the symbol *sp* from B is simply a Büchi k-ary ω-tree automaton, as defined in [VW]. An STA that uses only the symbols *sp*, *push*(w), and *pop* from B is a *pushdown* k-ary ω-tree automaton (PTA for short). This definition is similar to that appearing in [S]. Clearly, if $k = 1$, the infinite trees become infinite sequences.

EXAMPLE 1. Let $k = 2$ and $M = \langle Q, \Sigma, \Gamma, q_A, z_0, \delta, F \rangle$, where $Q = \{q_A, q_B, q_C, q_X, q_\varepsilon\}$, $\Sigma = \{a, b, c, x\}$, $\Gamma = \{z_0, z\}$, $F = \{q_A, q_X\}$, and

$$\delta(q_A, a, \gamma) = \{(q_A, push(z)), (q_B, push(z))\}$$
$$\delta(q_B, b, z) = \{(q_X, sp), (q_B, md)\}$$
$$\delta(q_B, \varepsilon, z_0) = \{(q_\varepsilon, mu)\}$$

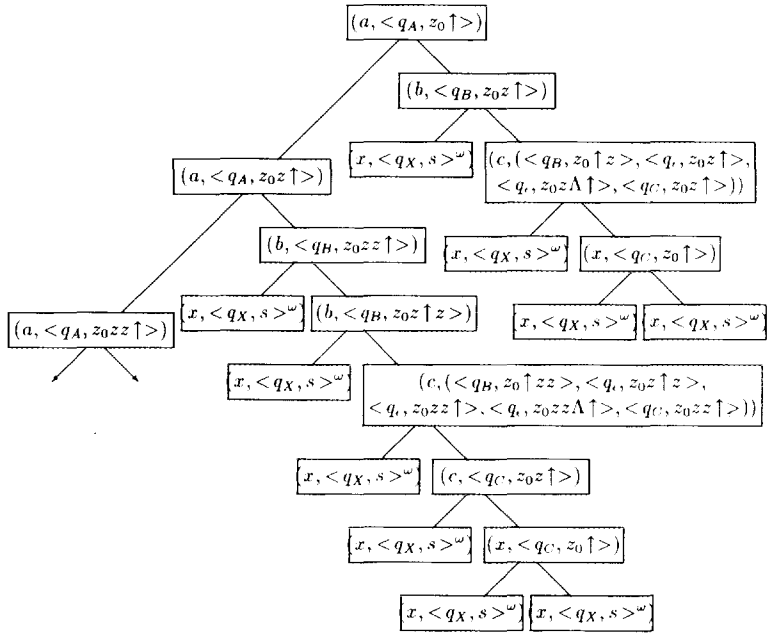

FIGURE 2

$$\delta(q_\varepsilon, \varepsilon, z) = \{(q_\varepsilon, mu)\}$$
$$\delta(q_\varepsilon, \varepsilon, \Lambda) = \{(q_C, md)\}$$
$$\delta(q_C, c, z) = \{(q_X, sp), (q_C, pop)\}$$
$$\delta(q_C, x, z_0) = \{(q_X, sp), (q_X, sp)\}$$
$$\delta(q_X, x, \gamma) = \{(q_X, sp), (q_X, sp)\}.$$

Denote by x^ω the unique tree over $\{x\}$, and let t be the tree of Fig. 1. Since M is deterministic, it has only one computation on t; this computation is described in Fig. 2. It is not difficult to see that this computation is an accepting one.

3. Emptiness for Pushdown Automata

In this section we prove that the emptiness problem for pushdown ω-tree automata (PTAs) is decidable. The proof uses pumping-like arguments applied to computation trees, in which the depth of the stack is a crucial ingredient.

3.1. *Definitions*

Let M be a PTA, and let n be a node in an S-computation C of M. Denote by N_F^n the set of nodes $n' = (q', s')$ such that:

1. n' is a descendant of n in C;
2. for each descendant $m = (p, t)$ of n in C, which is also an ancestor of n', we have $p \notin F$.

Denote by Π_n^m the set of nodes n' such that:

1. n' is on the path from n to m in C;
2. for all $n'' \neq n'$ on the path from n' to m, we have $d(n'') > d(n')$.

Intuitively, N_F^n contains all the nodes that would have been nodes also if M were an PTA acting on *finite* trees. Π_n^m contains all the nodes on the path from n to m that have a shorter stack than all their successors along that path. If $d(m) > d(n)$, the set Π_n^m is nonempty and $|\Pi_n^m| \geq d(m) - d(n)$.

Claim 2. *For any node n in an accepting S-computation C of some PTA M, the set N_F^n is finite.*

Proof. Let n be a node in an accepting S-computation C of M and assume by way of contradiction that N_F^n is infinite. We have that for each

$k \geqslant 1$ there exists a path $n = n_0, n_1, ..., n_k$ of nodes in C such that for all $1 \leqslant j \leqslant k$ the node n_j is a succesor of n_{j-1} and $n_j \notin F$. Hence, by König's Lemma, there exists an infinite path in C that has only finitely many appearances of nodes that signal F. This is a contradiction to our assumption that C is an accepting S-computation of M. ∎

Denote by S'^{+}_n the set of nodes n' such that:

1. n' is a descendant of n in C;
2. $d(n') = d(n)$;
3. for each node n'' on the path from n to n' in C, we have $d(n'') > d(n)$;
4. the pair (n, n') signals F.

Denote by S'^{-}_n the set of nodes n' such that:

1. n' is a descendant of n in C;
2. $d(n') = d(n)$;
3. for each node n'' on the path from n to n' in C, we have $d(n'') > d(n)$;
4. the pair (n, n') does *not* signal F.

Let $S^+_n = \{q \mid \exists s.(q, s) \in S'^{+}_n\}$, let $S^-_n = \{q \mid \exists s.(q, s) \in S'^{-}_n\}$, and let $S_n = (S^+_n, S^-_n)$. Clearly, $|S^+_n| \leqslant |Q|$, and $|S^-_n| \leqslant |Q|$, so $|S_n| \leqslant |Q|^2$ and the number of different such sets is at most $2^{|Q|^2}$.

Figure 3 shows part of an S-computation, viewed in terms of stack depth. The definitions of Π^m_n, S^+_n, S^-_n, and S_n are best understood with the aid of such diagrams. In this figure, Π^m_λ has been emphasized, $S^+_n = \{q, p, ...\}$, $S^-_n = \{r, ...\}$ and $S_m = (\{y, x, ...\}, \{w, ...\})$.

An *l-depth cut S-computation*, or DC_l for short, is a mapping $C : [k]^* \to N \cup \{\perp\}$ such that:

- $C(u) = \perp$ iff for all $1 \leqslant j \leqslant k$ we have $C(uj) = \perp$;
- For all u in $[k]^*$ such that $C(u) = (q, s)$, there exists $((q_1, b_1), ..., (q_k, b_k)) \in \delta(q, a, \mathscr{H}(s))$, such that for all $1 \leqslant j \leqslant k$, if $C(uj) \neq \perp$ then $C(uj) = (q_i, \mathscr{B}(s, b_j))$;
- $C(\lambda) \neq \perp$, and for each u such that $C(u) \neq \perp$ and $d(C(u)) < l$, it is the case that for each $1 \leqslant j \leqslant k$, $C(uj) \neq \perp$.

Intuitively, an l-depth cut S-computation is simply an infinite subtree of an S-computation, with the same root as that of the S-computation. However, since we deal with infinite trees we mark all the nodes not in that subtree with the special symbol $\perp$. Note that by the third clause in its definition, all the "leaves" of an l-cut are guaranteed to be of depth at least l. A DC_l C

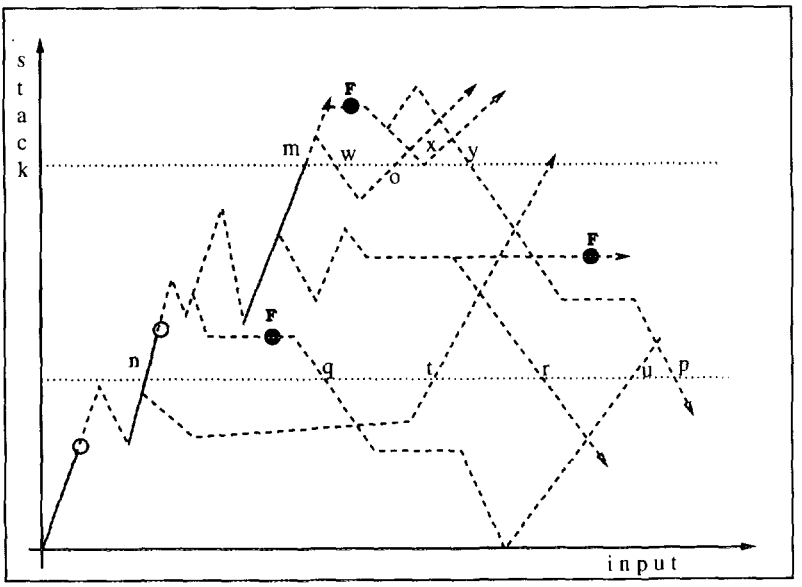

FIGURE 3

is said to be *accepting* if $C(\lambda)$ is an initial configuration, and every infinite path in it contains infinitely many node configurations with states in F, or infinitely many $\perp$s. It is clear that if C is an accepting S-computation of M, then for every l, C is an accepting l-depth cut S-computation of M.

Let C be a DC_l of M for some l, and let $n=(q_n, s_n)$ and $m=(q_m, s_m)$ be nodes in C. We call n and m a *nice pair* if $n \neq \perp$, $m \neq \perp$, $n \in \Pi_\lambda^m$, $d(m)-d(n)=j>0$, $S_n=S_m$, and the same rule of δ applies to both n and m in C. As we shall see, a nice pair can be made the subject of a pumping-like argument, mainly by virtue of the equality of S_n and S_m, which guarantees that the new tree obtained by pumping is indeed an S-computation.

3.2. *The Main Operators*

Let $T(x, t_1, t_2, ..., t_k)$ be the k-ary tree rooted at x, with the k-ary trees $t_1, t_2, ..., t_k$ rooted at its successors. For a node n in a tree t, let $ST(n)$ denote the subtree of t rooted at n, let $S_i(n)$, for $1 \leqslant i \leqslant k$, denote the ith offspring of n in t, and let $ST_i(n)$ denote the subtree rooted at $S_i(n)$.

Define $\mathcal{O}_n^+ : Q \to N$, such that $\mathcal{O}_n^+(q)$ is some node $(q, s) \in C$ with $(q, s) \in S'^+_n$. If no such node exists, $\mathcal{O}_n^+(q)$ is undefined. Note that by the definition of S_n^+, such a node always exists for any $q \in S_n^+$. In the same way, define $\mathcal{O}_n^- : Q \to N$, such that $\mathcal{O}_n^-(q)$ is some node $(q, s) \in C$ with

$(q, s) \in S'^{-}_n$. If no such node exists, $\mathcal{O}^{-}_n(q)$ is undefined. Again, for all $q \in S^{-}_n$, this mapping is defined.

For a stack configuration $s = z_0\gamma_1 \uparrow \gamma_2$, the function $\mathcal{I}_{l,\gamma}(s)$ is the result of inserting the word γ to the right of the symbol that appears at the lth position of s. Similarly, the function $\mathcal{D}_{l,j}(s)$ is the result of deleting from s the letters in positions $l+1, ..., l+j$.

Let n and m be a nice pair in a DC_l. We abbreviate $d(n)$ by d_n, $d(m)$ by d_m, $d(m) - d(n)$ by j, and the word in positions $d_n + 1, ..., d_m$ in m's stack by $\gamma_{n,m}$.

The following are efined for $u \in [k]^*$ by recursion on the structure of the tree $[k]^*$:

- $U_{n,m}(u) = T(\mathcal{D}_{d_n, j}(u), t_1, ..., t_k)$, where for $1 \leqslant i \leqslant k$ we have:

$$
\begin{aligned}
t_i &= U_{n,m}(ST_i(u)) && \text{if } d(S_i(u)) > d_m \\
t_i &= ST(\mathcal{O}^{-}_n(q_i)) && \text{if } d(S_i(u)) \leqslant d_m \text{ and } q_i \in S^{-}_n \\
t_i &= ST(\mathcal{O}^{+}_n(q_i)) && \text{if } d(S_i(u)) \leqslant d_m \text{ and } q_i \notin S^{-}_n \\
t_i &= ST_i(u) && \text{if } S_i(u) = \perp
\end{aligned}
$$

- $P_{n,m}(u) = T(\mathcal{I}_{d_n, \gamma_{n,m}}(u), t_1, ..., t_k)$, where for $1 \leqslant i \leqslant k$ we have:

$$
\begin{aligned}
t_i &= P_{n,m}(ST_i(u)) && \text{if } d(S_i(u)) > d_n \\
t_i &= ST(\mathcal{O}^{-}_m(q_i)) && \text{if } d(S_i(u)) \leqslant d_n \text{ and } q_i \in S^{-}_m \\
t_i &= ST(\mathcal{O}^{+}_m(q_i)) && \text{if } d(S_i(u)) \leqslant d_n \text{ and } q_i \notin S^{-}_m \\
t_i &= ST_i(u) && \text{if } S_i(u) = \perp.
\end{aligned}
$$

Here, $q_i = \text{state}(S_i(u))$.

$U_{n,m}$ and $P_{n,m}$ actually represent *pumping* arguments, where P stands for *pump*, and U stands for *unpump*. Let n_1 be the successor of n on the path from n to m in C, and let m_1 be the corresponding successor of m (i.e., if n_1 is the ith successor of n, then m_1 is the ith successor of m).

Denote by $C_{U_{n,m}}$ and $C_{P_{n,m}}$ the following trees over $N \cup \{\perp\}$:

- $C_{U_{n,m}}$ is identical to C except that $ST(n_1)$ is replaced by $U_{n,m}(m_1)$ as the successor of n.

- $C_{P_{n,m}}$ is identical to C except that $ST(m)$ is replaced by $T(C(m), t_1, ..., t_k)$, where for $1 \leqslant i \leqslant k$ we have $t_i = P_{n,m}(ST_i(n))$ if $d(S_i(n)) > d_n$, and $t_i = ST_i(m)$ otherwise.

Note that in $C_{P_{n,m}}$, we replace all offspring of m that have larger stack depth with the appropriate subtrees, while in $C_{U_{n,m}}$ we change only one offspring. This is done mainly for technical reasons, but the heart of the pump

and unpump operators is essentially the same. We now illustrate these notions. In the S-computation shown in Fig. 4, the nodes n and m are a nices pair. Figure 5 demonstrates the action of the unpump operator; in it, the appropriate successors of m and n have been conjoined, and both q and r in S'_m have been replaced by q and r in S'_n. Note that the node u has not been changed because it is not in S'_n. Figure 6 shows the action of the pump operator; in it, the successors of n have been replaced by the appropriate successors of m, and both q and r in its subtree have been replaced by q and r in S'_m.

CLAIM 3. *If C is an accepting S-computation of M, then so is $C_{U_{n,m}}$.*

Proof. In order to show that $C_{U_{n,m}}$ is an S-computation of M it suffices to show that nodes that have been changed, or those whose successors have been changed, still follow the rules of M. Considering n, since in C the same rule was applied to both m and n, this same rule is applicable to n in $C_{U_{n,m}}$. For every node n' that is a descendant of n in C and has $d(n') > d_m + 1$, since the operator $\mathscr{D}_{d_n,j}$ does not change the top of its stack, the same rules as in C will still be applicable in $C_{U_{n,m}}$.

For every node n' that is a descendant of n in C and has $d(n') = d_m + 1$ there are two possibilities. If all its successors have $d \geqslant d_m + 1$, the previous argument is true (i.e., $\mathscr{D}_{d_n,j}$ still does not change the top of the stack, hence the same rules as in C will still be applicable in $C_{U_{n,m}}$.) If, however, n' has

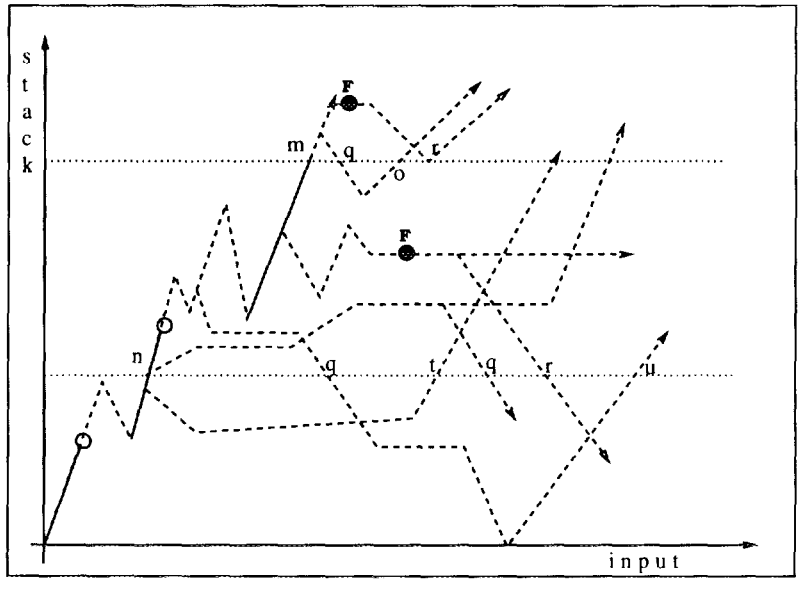

FIGURE 4

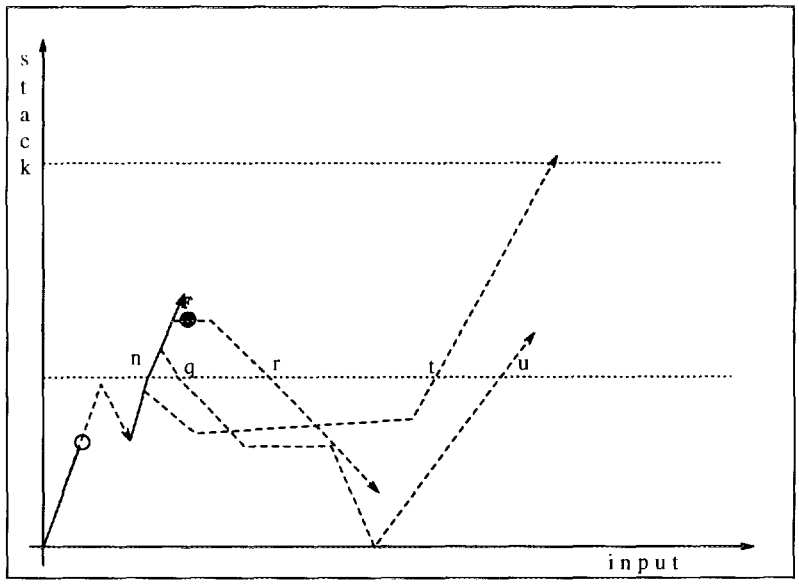

FIGURE 5

successors of $d \leqslant d_m$, then from $S_n = S_m$ and from the behavior of $\mathscr{D}_{d_n,j}$ on stacks, the same rule as in C will still be applicable in $C_{U_{n,m}}$. The suffix of every infinite path in $C_{U_{n,m}}$ is the suffix of an infinite path in C, or is such a suffix when $\mathscr{D}_{d_n,j}$ operates on the stack in its node confifurations. In any case, if in any infinite path in C there are infinitely many apprarances of states in F, the same will hold for $C_{U_{n,m}}$, and the claim follows. ∎

CLAIM 4. *If C is a DC_l of M, for some l, then so is $C_{P_{n,m}}$.*

Proof. Essentially the same as that of Claim 3. ∎

3.3. *Deciding Emptiness*

We now prove that emptiness for pushdown automata on infinite trees is decidable. To that end, we need a definition for the limit of a sequence of trees.

Let $\{t_i\}_{i=1}^{\infty}$ be a sequence of infinite k-ary trees over some set S. If for each $u \in [k]^*$ there exist j_u and a fixed $s_u \in S$, such that for all $i \geqslant j_u$ we have $t_i(u) = s_u$, we take the limit of $\{t_i\}_{i=1}^{\infty}$ to be the tree t defined by $t(u) = s_u$ for each u. We also write $\lim_{i \to \infty}(t_i) = t$.

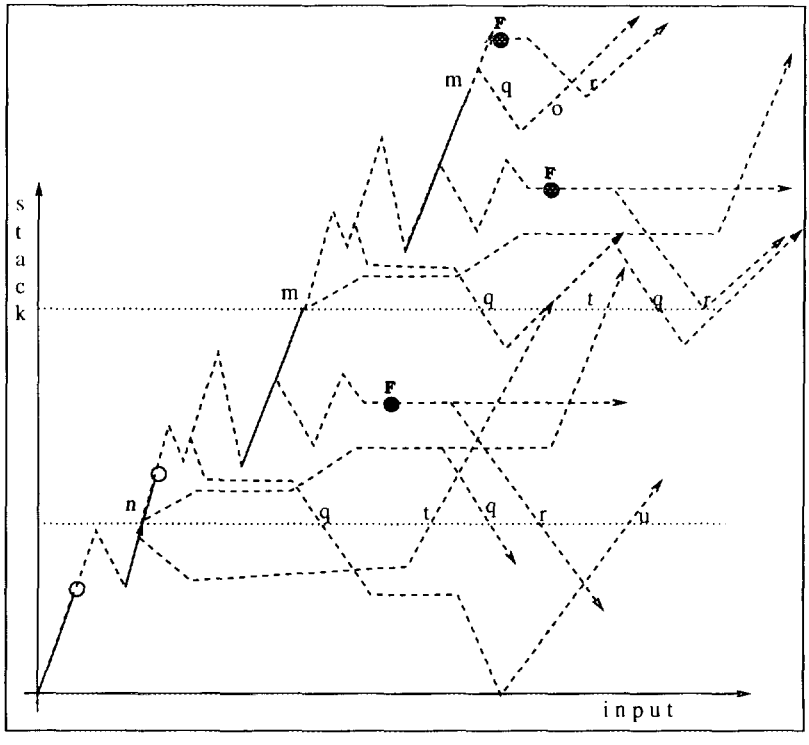

FIGURE 6

Let n be a node in an accepting S-computation C of M. Let a_C^n be the maximal depth of the set N_F^n, defined by:

$$a_C^n = \max_{n' \in N_F^n \text{ of } C \cup \{n\}} (d(n') - d(n)).$$

The set of nodes of maximal depth in N_F^n is called A_C^n, more formally $A_C^n = \{n' \in N_F^n \mid d(n') - d(n) = a_C^n\}$. We also associate a node n' in A_C^n with the path from n to it in the following way: Let $k = (|\Gamma| \cdot |Q|)^{a_C^n}$ and $d_n = d(n)$; define $P_n^{n'} = (q_0, \gamma_0, ..., q_{a_C^n}, \gamma_{a_C^n})$, where γ_i is the letter in position $d_n + i$ in the stack of n', and q_i is the state of the (unique) node in $\Pi_\lambda^{n'}$ of depth $d_n + i$. We associate with n the tuple $x_C^n = (a_C^n, m_1, ..., m_k)$ where m_j is the number of different nodes n'' in A_C^n such that $P_n^{n''}$ is the jth vector in lexicographic order in the set $(Q \times \Gamma)^{a_C^n}$. Since C is an accepting computation, for every $n \in C$ the set N_F^n is finite; hence x_C^n is well defined.

First, we prove two technical claims that deal with the effect of the operator $U_{n,m}$ on the set A_C^n.

CLAIM 5. *Let n be a node in an accepting computation C of M. Assume that for any node n' on the path from n to some node in A^n_C the following hold*: (1) $d(n') > d(n)$; *and* (2) *any node n'' on the path from n to n' has a different stack configuration from the one of n'. Let m be a node in A^n_C such that P^m_n is minimal among the nodes of A^n_C, and let n_0, m_0 be a nice pair on the path from n to m. Then $x^n_{C_{U_{n_0,m_0}}} < x^n_C$.*

Proof. Consider all the nodes m in the subtree rooted by n in $C_{U_{n_0,m_0}}$ such that no node on the path from n to m signals F. We say that such a node m is of type 0 if it is not a descendant of n_1 (here n_1 is the appropriate node from the definition of $C_{U_{n,m}}$). In this case the pair n, m does not signal F in $C_{U_{n_0,m_0}}$ iff n, m does not signal F in C (by the definition of $C_{U_{n_0,m_0}}$). We say that such a node m is of type 1 if it is a descendant of n_1, and each node m' on the path from n_1 to m has $d(m') \geqslant d(n_1)$. In this case the pair n, m does not signal F in $C_{U_{n_0,m_0}}$ iff there exists a node m'' in C, with $d(m'') > d(m)$, such that the pair n, m'' does not signal F in C. This follows from the fact that $d(m_0) > d(n_0)$ and from the definition of $U_{n,m}$. Otherwise, such a node m is said to be of type 2. In this case we claim that if the pair n, m does not signal F in $C_{U_{n_0,m_0}}$ then there exists a node m'' in C, where $P^{m''}_n$ in C is equal to P^m_n in $C_{U_{n_0,m_0}}$. To verify this, consider the first node s on the path from n_1 to m with $d(s) < d(n_1)$. By the definition of $U_{n,m}$, since n_1, s does not signal F, the subtree of $C_{U_{n_0,m_0}}$ rooted at s is exactly the subtree of C rooted at some node s' in S'^-_n; hence the claim follows.

Now, if $A^n_{C_{U_{n_0,m_0}}}$ contains any node of type 1, then $a^n_{C_{U_{n_0,m_0}}} < a^n_C$ and we are done. If $A^n_{C_{U_{n_0,m_0}}}$ does not contain any node of type 2, then either $a^n_{C_{U_{n_0,m_0}}} < a^n_C$, or there are strictly less nodes in $A^n_{C_{U_{n_0,m_0}}}$ with $P^{n'}_n$ than in A^n_C. We are left with the case that $a^n_{C_{U_{n_0,m_0}}} = a^n_C$, and $A^n_{C_{U_{n_0,m_0}}}$ contains some nodes of type 2. Note that by the conditions of the Claim, every node m in A^n_C with $P^m_n = P^{n'}_n$ is not in the subtree rooted by n_1, hence in this case too $x^n_{C_{U_{n_0,m_0}}} < x^n_C$. ∎

CLAIM 6. *If for every node configuration n that appears in some accepting S-computation C' of M, n also appears in some acepting S-computation C of M in which $a^n_C < |\delta| \cdot 2^{|Q|^2}$, and M has an accepting S-computation, then M has an accepting S-computation C_0, such that for every node $m \in C_0$, we have $a^m_{C_0} < |\delta| \cdot 2^{|Q|^2}$.*

Proof. Using induction on the depth of the stack configuration, it is easy to prove that if for every node configuration n that appears in some accepting S-computation C' of M, n also appears in some accepting S-computation C of M in which $a^n_C < |\delta| \cdot 2^{|Q|^2}$, then for every node configuration

m, that appears in some accepting S-computation C' of M, m also appears in some accepting S-computation C_m^+ of M in which for every node m' in N_F^m we have that $a_{C_m^+}^{m'} < |\delta| \cdot 2^{|Q|^2}$.

Now, to construct C_0, we start with some accepting S-computation C of M, and, recursively from the root, replace every node n that has $a_C^n \geq |\delta| \cdot 2^{|Q|^2}$, with the appropriate subtree of C_n^+. Clearly the resulting tree is an S-computation of M, since we only replaced identical node configurations. Moreover, this S-computation is accepting because all nodes that are replaced are accepting node configurations, and possibly one of them is an initial node configuaration. ▮

Now we are ready to prove the following:

LEMMA 7. *A PTA M has an accepting S-computation C iff it has an accepting S-computation C', such that for very node $n \in C'$, we have $a_C^n < |\delta| \cdot 2^{|Q|^2}$.*

Proof. By Claim 6 it is enough to prove that for every node configuration n, if n is in some accepting S-computation C' of M then n is in some accepting S-computation C of M in which $a_C^n < |\delta| \cdot 2^{|Q|^2}$. Assume that the above is not true and choose a counter example m of minimal depth. Among all accepting S-computations in which m appears choose one with a minimal x_C^m and call this computation C. We can assume that every node m' in the path from m to some member of A_C^m has $d(m') > d(m)$, because otherwise we could choose the node configuration of m' to be our counter example. (Recall that A_C^m is finite, and $d(m') \geq d(m)$ by the minimality of $d(m)$.) We can also assume that any two nodes m_1 and m_2 on the path from m to any m' in A_C^m have different node configurations, because combining identical nodes inside N_F^n does not affect the S-computation.

Now, since the number of different sets S_n is at most $2^{|Q|^2}$, the number of different rules in δ is at most $|\delta|$, and $a_C^m > |\delta| \cdot 2^{|Q|^2}$, we have that for any node m' in A_C^m there exists a nice pair $n_0, m_0 \in \Pi_m^{m'}$. In particular, if we choose m' such that $P_m^{m'}$ is minimal among the nodes of A_C^m, we get by Claim 5 that $x_{C_{U_{n_0,m_0}}}^m < x_C^m$, which is a contradiction to the minimality of x_C^m. ▮

CLAIM 8. *Let C be an accepting DC_l of M such that for every node $n \in C'$ we have $a_C^n < |\delta| \cdot 2^{|Q|^2}$ and let n and m be two connected nodes in C such that $d(m) - d(n) \geq (|\delta| \cdot 2^{|Q|^2})^2$. Then there exists a nice pair $n', m' \in \Pi_n^m$ that signals F.*

Proof. Let $j = |\delta| \cdot 2^{|Q|^2}$. Assume that C is a DC_l of M that satisfies the properties of the claim, and let n, m be nodes in C with $d(m) - d(n) \geq j^2$. Define $\{n_i\}_{i=0}^j$ by:

- $n_0 = n$.
- n_{i+1} is the first node n' in Π_n^m (i.e., the one with minimal depth) such that n_i and n' form a pair that signals F.

Since $a_C^{n_i} < j$, we get that $d(n_{i+1}) - d(n_i) < j$; hence $\{n_i\}_{i=0}^{j}$ is well defined. Since all the n_i are in Π_n^m, and there are j of them, there are two nodes among them that satisfy the desired properties (i.e., these node form a nice pair that signals F). ∎

We next prove that the above conditions are not only necessary but also sufficient. To that end, we need to extend the operator $P_{n,m}$. Let c be a subtree of some DC_l of M, and let C be a (different) DC_l of M. The following is defined for $u \in [k]^*$ by recursion on the structure of the tree $[k]^*$:

- $\mathscr{R}_c(u) = T(u, t_1, ..., t_k)$, where for $1 \leqslant i \leqslant k$ we have

$$t_i = \begin{cases} c & \text{if } S_i(u) = c(\lambda) \\ ST_j(u) & \text{otherwise.} \end{cases}$$

Intuitively, $\mathscr{R}_c$ searches a given DC_l for some fixed node configuration. When a node with such a node configuraton is found, $\mathscr{R}_c$ replaces it with the subtree c. Note that $\mathscr{R}_c$ is defined by recursion on the structure of the tree, hence the search will continue on the nodes of c.

Let n, m be a nice pair in a DC_l C of M, and denote by $C_{P^*_{n,m}}$ the following tree over $N \cup \{\perp\}$:

$$C_{P^*_{n,m}} = \mathscr{R}_{P_{n,m}(n)}(C_{P_{n,m}}).$$

An accepting DC_l is termed *compacted* if every two connected nodes in it with the same node configuration, signal F. Note that since N_F^n is finite for every node n in an accepting DC_l, one can get a compacted accepting DC_l from any given accepting DC_l, simply by replacing the appropriate subtrees inside every N_F^n.

CLAIM 9. *Let C be an accepting DC_l of M, and let n' be a node of depth l in C, with $P = P_\lambda^{n'}$, such that there exists a nice pair $n, m \in \Pi_\lambda^{n'}$ that signals F. If for each $m' \in C$ such that m' is a descendant of m with $C(m) = C(m')$ we hav that the pair m, m' signals F, then $C_{P^*_{n,m}}$ is an accepting DC_l of M, and in it all nodes n'' with $P_\lambda^{n''} = P$ do not have any $\perp$ successors.*

Proof. It is not difficult to see that $C_{P^*_{n,m}}$ is a DC_l of M, since all new node configurations added by the operator $P^*_{n,m}$, have a larger stack depth than the corresponding nodes of C. The condition on the descendants of m is exactly the one that ensures us that $C_{P^*_{n,m}}$ is an accepting DC_l of M. Note that in $P_{n,m}(n)$ a node m with $d(m) = l$, such that every node m' on the path from n to m has $d(m') > d(n)$, cannot have a $\perp$ successor, since this would imply the existent node of depth $j < l$ with a $\perp$ successor in C. Also

note that any node m' in C with $P_\lambda^{m'} = P$ must have a node $n_{m'}$ on $\Pi_\lambda^{m'}$ with $C(n) = C(n_{m'})$. Hence this node will not be in $C_{P^*_{n,m}}$. ∎

LEMMA 10. *If there exists an accepting DC_l C of a PTA M with $l \geqslant (|\delta| \cdot 2^{|Q|^2})^2$ such that for every node $n \in C$ we have $a_C^n < |\delta| \cdot 2^{|Q|^2}$, then M has an accepting S-computation.*

Proof. Let C be a compact accepting DC_l of M that satisfies the conditions of the lemma. Let n' be a node of depth l in C, with $P = P_\lambda^{n'}$, and $n(n')$, $m(n')$ be the nice pair existing by the Claim 8. By Claim 9, $C_{P^*_{n(n'),m(n')}}$ is a DC_l of M in which no nodes n'' with $P_\lambda^{n''} = P$ have any $\perp$ successors. Since the number of different P_λ^n in C is finite, and since for any node in $C_{P^*_{n(n'),m(n')}}$ that has a $\perp$ successor the path $\Pi_\lambda^{n''}$ contains only nodes from C, after a finite number of applications of the operator $C_{P^*_{n,m}}$ we obtain a DC_l of M, called C_1, with the following properties: (1) C_1 is an accepting DC_l of M in which none of the nodes of depth l have a $\perp$ successor, and hence C_1 is actually an accepting DC_{l+1} of M; (2) for every node n' of *depth* $l+1$ in C_1, there is a nice pair $n(n'), m(n') \in \Pi_\lambda^{n'}$, such that $d(n(n')) \geqslant 2$, that signals F. (This is true because $m(n'), u$ is a nice pair in $C_{P^*_{n(n'),m(n')}}$, where u is the node that corresponds to $m(n')$ in $C_{P^*_{n(n'),m(n')}}$ in the same way that $m(n')$ corresponds to $n(n')$ in C.) By applying the $P^*_{m,n}$ operator to C_1 yet another finite number of times, one obtains C_2, and so on.

Note that since for each i the depth of $n(n')$ in the i th stage is at least i, nodes of C_i with tree depth less than i do not change in C_j, for $j \geqslant i$. Hence the limit $D = \lim_{j \to \infty} (C_j)$ is well defined. It is not difficult to see that D is indeed an accepting S-computation of M, since for a given path in D, if the stack depth on this path is bounded, then this path is a path in some accepting DC_j of M; if it is not bounded, then every pair of nodes on that path with $d(m) - d(n) > l$ must signal F. Hence the path contains infinitely many states in F. ∎

From the above lemma together with Lemma 7 we have:

COROLLARY 11. *A PTA M has an accepting S-computation iff it has an accepting $DC_{(|\delta| \cdot 2^{|Q|^2})^2}$ C such that for any node $n \in C$ we have $a_C^n < |\delta| \cdot 2^{|Q|^2}$.*

CLAIM 12. *The question of whether a PTA M has a DC_l that satisfies the conditions of Corollary* 11 *is decidable in time $O(k^{c \cdot |\Gamma|^l \cdot |Q|})$.*

Proof. In a DC_l there are at most $|Q| \cdot |\Gamma|^l + 1$ different node configurations. Hence it is enough to check all the finite trees of depth $|Q| \cdot |\Gamma|^l + 1$. (This is the depth of the tree, and is not to be confused with stack depth.) The number of such trees is $k^{|Q| \cdot |\Gamma|^l + 1}$, and checking the satisfiability of the condition for each tree can be carried out in time polynomial in the size of

the tree. Note that we can assume that in the DC_l every pair of connected nodes with the same node configuration signals F. Thus, although a DC_l may be infinite, we only have to check finite trees. ▮

We may now conclude:

PROPOSITION 13. *The emptiness problem for PTA's is decidable in triple-exponential time.*

4. Reduction to Pushdown Automata

We now turn to reduce the emptiness problem for stack automata on infinite trees to that of pushdown automata. The technical result is:

PROPOSITION 14. *Let M be an STA. Then there is an effectively constructible PTA M', of size at most doubly exponential in the size of M, such that M has an accepting computation iff M' has one.*

Proof. We first prove the result for the case when $k=1$; i.e., for ω-machines.

Let $M=\langle Q, \Sigma, \Gamma, q_0, z_0, \delta, F\rangle$ be a simple ω-stack automaton. Define $M'=\langle Q, \Sigma, \Gamma, q_0, z_0, \delta, F\rangle$, where

- $\Gamma'=\{(z, L) \mid z\in\Gamma$ and L is a transitively closed subset of $Q^2\}$.
- $z_0'=(z_0, L_0)$, where $(p, q)\in L_0$ iff there is a path in M from (p, z_0) to (q, z_0), that passes only through node configurations whose stack is precisely (z_0).
- The transition function δ' is defined as follows. For any q, a, z, and L, take $\delta'(q, a, (z, L))$ to be the least set satisfying:

1. Whenever $(p, push(z_1))\in\delta(q, a, z)$, we have $(p, push(z_1, L'))\in\delta'(q, a, (z, L))$, where $(r, u)\in L'$ iff at least one of the following conditions holds:
   - $(p_1, md)\in\delta(t, a, z_1)$, $(u, mu)\in\delta(p_2, a, z)$ and $(p_1, p_2)\in L$.
   - $(u, sp)\in\delta(t, a, z)$.
   - there exists $r\in Q$, such that $(t, r)\in L$ and $(r, u)\in L$.
2. Whenever $(q, pop)\in\delta(p, a, z)$, we have $(q, pop)\in\delta'(p, a, (z, L))$ for all $(z, L)\in\Gamma'$.
3. Whenever $(p, q)\in L$, we have $(q, sp)\in\delta'(p, a, (z, L))$ for all $z\in\Gamma$.

First, note that $|M'|$ is at most exponential in $|M|$. Now, the idea behind this construction is that, given a stack and a state in M, one can determine which states can be reached via *md*, *mu*, and *sp* moves. This information is

pushed onto the stack of M', making the simulation possible. We now prove the technical claim that captures this intuition.

Define $L_M : \Gamma^+ \to 2^{Q^2}$ as follows:

$$L_M(s) = \{(p, q) \in Q^2 \mid (q, s\uparrow) \text{ is connected to } (p, s\uparrow) \text{ in } M \text{ using only } md, sp, \text{ or } mu \text{ moves}\}$$

Define $\mathscr{L}_M : \Gamma^+ \to \Gamma'^+$ by

$$\mathscr{L}_M(z_0 z_1 \cdots z_i) = (z_0, L_M(z_0))(z_1, L_M(z_0 z_1)) \cdots (z_i, L_M(z_0 \cdots z_i)).$$

CLAIM 15. *$(q, s\uparrow)$ is connected to $(q', s'\uparrow)$ in M iff $(q, \mathscr{L}_M(s)\uparrow)$ is connected to $(q', \mathscr{L}_M(s')\uparrow)$ in M'.*

Proof. Let C be an S-computation of M, in which $(q, s\uparrow)$ is connected to $(q', s'\uparrow)$, and let π be the path from $(q, s\uparrow)$ to $(q', s'\uparrow)$ in C. Let $j = \max_{r \in \pi} d(r)$. The proof proceeds by induction on j.

$j = 0$: by definition, $z'_0 = \mathscr{L}_M(z_0)$, and since clause 3 in the definition of δ' is the only rule allowing *sp* moves in δ', we are done.

We assume the claim is true for j, and we prove the claim for $j+1$. Let s be a stack configuration of depth j. It suffices to show that

1. if $(p, s\uparrow)$ is connected to $(q, sz\uparrow)$ in C then $(p, \mathscr{L}_M(s)\uparrow)$ is connected to $(q, \mathscr{L}_M(sz)\uparrow)$ in M'.

2. if $(p, sz\uparrow)$ is connected to $(q, sz\uparrow)$ in C then $(p, \mathscr{L}_M(sz)\uparrow)$ is connected to $(q, \mathscr{L}_M(sz)\uparrow)$ in M'.

3. if $(p, sz\uparrow)$ is connected to $(q, s\uparrow)$ in C then $(p, \mathscr{L}_M(sz)\uparrow)$ is connected to $(q, \mathscr{L}_M(s)\uparrow)$ in M'.

These follow directly from the inductive hypothesis applied to s and the definition of δ'.

The other direction is proved in exactly the same way by induction on the depth in the computation of M'. ∎

This can be seen to prove the Proposition for the case $k = 1$, i.e., for ω-strings. To deal with ω-trees, we employ a similar construction. The details, however, are more complicated, and we need some new notations. Denote the set $\{md, mu, sp\}$ by *in-stack moves*, and the set $\{push(\ ), pop, sp\}$ by *top-stack moves*. In the same way, node configurations in which $d(u) = h(u)$ are called *top-stack configurations*, and all others are called *in-stack configurations*.

The idea is to put more information on the stack, enabling the pushdown automaton M' to "known" what connections are possible through in-stack moves in the stack automaton M. Since accepting configurations have to be top-stack configurations, every in-stack node n in an

accepting S-computation must be the root of a finite subtree, whose leaves are top-stack configurations with the same stack as n. Hence our goal in the following construction is to create a pushdown automaton with the property that the successors of any node n are precisely these leaves.

The fact that the number of such leaves may be $2^{|Q|}$ suggests that M' should work on wider trees; speifically, we have $k' = k \cdot 2^{|Q|}$. What makes the proof rather technical is the need to generate the additional information we want to put on the stack.

Let $M = \langle Q, \Sigma, \Gamma, q_0, z_0, \delta, F \rangle$ be a simple stack k-ary ω-tree automaton, and let $k' = k \cdot 2^{|Q|}$. Define $M' = \langle Q, \Sigma, \Gamma', q_0, z_0', \delta', F \rangle$ as a pushdown k'-ary ω-tree automaton, where:

- $\Gamma' = \Gamma \times 2^{Q \times 2^Q}$.
- $z_0' = (z_0, Q \times \{\varnothing\})$.
- The transition function δ' is defined as follows. (Note that the definition uses the special set of mappings Ψ_L^z, which are defined below.) For each $q \in Q$, $z \in \Gamma$, and $L \in 2^{Q \times 2^Q}$, whenever $((q_1, b_1), ..., (q_k, b_k)) \in \delta(q, a, z)$, with $b_i \neq mu$ for each i, and such that $((q_1^i, b_1^i), ..., (q_{2^k}^i, b_{2^k}^i)) \in \Psi_L^z(q_i, b_i)$, we also have $((q_1^1, b_1^1), ..., (q_{2^k}^1, b_{2^k}^1), ..., (q_{2^k}^k, b_{2^k}^k)) \in \delta'(q, a, (z, L))$.

Given z and L, we now define the special mapping Ψ_L^z, from $Q \times B$ to $(Q \times B')^{2^k}$.

$((q_1, b_1), ..., (q_{2^k}, b_{2^k})) \in \Psi_L^z(p, b)$ iff one of the following holds:

1. $b \in \{sp, pop\}$, and for all $1 \leqslant i \leqslant 2^k$ we have $(q_i, b_i) = (p, b)$.

2. $b = md$ and $(p, \{q_1, ..., q_{2^k}\}) \in L$, and for all $1 \leqslant i \leqslant 2^k$ we have $b_i = sp$.

3. $b = push(z')$, and for all $1 \leqslant i \leqslant 2^k$, we have $q_i = p$ and $b_i = push(z', L')$, where $L' \subseteq 2^{Q \times 2^Q}$ is defined by $(q', \mathbf{s}) \in L'$ iff $\mathbf{s}$ is the set of leaves of a finite tree T over Q with root q', which satisfies the following condition: For any internal node p' in T, there is a rule $((q_1', b_1') \cdots (q_k', b_k')) \in \delta(p', a, z)$, such that the set of successors of p' in T is the smallest set $\mathbf{s}_p$, for which

   - if $b_i' = sp$ then $q_i' \in \mathbf{s}_p$ is an internal node in T;
   - if $b_i' = mu$ then $q_i' \in \mathbf{s}_p$ is a leaf in T;
   - if $b_i' = md$ then there exists $(q_i', \mathbf{s}_{q'}) \in L$ such that $\mathbf{s}_{q'} \subset \mathbf{s}_p$.

The intuitive meaning of the information we add is that in any node $n = (q, (z_0, L_0) \cdots (z_l, L_l)\uparrow)$ of a computation of M', $(p, \mathbf{s}) \in L_l$ iff $(p, z_0 \cdots \uparrow z_l)$ is the root of a finite subtree of a computation of M, whose internal nodes are in-stack configurations, and whose leaves are of the form $(q', z_0 \cdots z_l\uparrow)$, where $q' \in \mathbf{s}$. In order to complete the proof of Proposition 14 we need a few more definitions.

Define $L_M : \Gamma^+ \to 2^{Q \times 2^Q}$ as follows:

$$L_M(z_0) = Q \times \{\varnothing\}$$

$$L_M(z_0 \cdots z_{i-1} z_i) = \{(p, \mathbf{s}) \in Q \times 2^Q \mid \text{there is a subtree } T \text{ of a computation } C \text{ of } M \text{ rooted at } (p, z_0 \cdots z_{i-1} \uparrow z_i), \text{ such that all the internal nodes in } T \text{ are in-stack configurations, all its leaves are all the top-stack configurations of the form } (q, z_0 \cdots z_i \uparrow), \text{ where } q \in \mathbf{s}.\}$$

Define $\mathscr{L} : S \to S'$ by

$$\mathscr{L}(z_0 z_1 \cdots z_i \uparrow) = (z_0, L_M(z_0)), (z_1, L_M(z_0 z_1)), \ldots, (z_i, L_M(z_0 \cdots z_i)) \uparrow,$$

where $\mathscr{L}$ is undefined for all in-stack configurations.

CLAIM 16. *A top-stack configuration (q, s) is the root of a finite subtree T of a computation C of M, all of whose leaves are top-stack configurations and all of whose internal nodes are in-stack configurations iff we have $((q_1^1, b_1^1), \ldots, (q_{2^k}^k, b_{2^k}^k)) \in \delta'(q, a, \mathscr{L}(s))$, and (q', s') is a leaf of T iff there exists $1 \leqslant i \leqslant 2^k$ and $1 \leqslant j \leqslant k$, such that $q' = q_i^j$ and $\mathscr{B}(\mathscr{L}(s), b_i^j) = \mathscr{L}(s')$.*

Proof. By induction on $j = d(q, s)$.

If $j = 0$, we have $(q, s) = (q, z_0 \uparrow)$, and by the definitions of z_0' and $\mathscr{L}$, we obtain $\mathscr{L}(z_0 \uparrow) = (z_0, Q \times \varnothing) \uparrow = z_0' \uparrow$. In this case, whenever $\delta(q, s) = ((q_1, b_1), \ldots, (q_k, b_k))$, b_i must be either *sp* or *push*(z'). If $b_i = sp$, then by clause (1) in the definition of Ψ we are done. If $b_i = push(z')$, then clearly $L_M(z_0 z') = L'$, as defined in clause (3) therein, and the claim follows.

If $j \geqslant 1$, the inductive hypothesis implies that $(q, s) = (q, z_0 \cdots z_j \uparrow)$ is a leaf in a computation of M iff $(q, \mathscr{L}(s))$ is a leaf in a computation of M'. If $b_i = md$, then by clause (2) in the definition of Ψ, and by $L_j = L_M(z_0 \cdots z_j)$, which follows from the inductive hypothesis, the claim is obtained. The case $b_i = pop$ is trivial, and the cases in which $b_i \in \{push(z'), sp\}$ are dealt with in thee same way as the base case $j = 0$. ∎

Since in any accepted computation C of M, all the accepting states must be top-stack configurations, Claim 16 actually proves that M has an accepting computation iff M' has one. Moreover, by the construction, the size of M' is at most doubly exponential in the size of M, which proves Proposition 14. ∎

THEOREM 17. *The emptiness problem for STAs is decidable in five-fold exponential time.*

5. Discussion

In Section 3 we showed that the emptiness problem for Büchi PTAs is decidable. For a discussion on PTAs with different acceptance criteria we refer the reader to [S]. We believe that our techniques can be used to establish the decidability of emptiness for PTAs with more general accepting conditions (like C_3 of [S]). The details, however, would become more complicated.

Another interesting direction involves properties of the set of trees accepted by PTAs. As far as we know, such issues have not been treated in the literature. Let us point out that it is not difficult to see that many results about closure properties of ω-CFLs can be easily applied to PTAs too. The main observation is that for any context-free ω-language L one can construct a PTA that accepts all the trees whose leftmost path is in L. Conversely, for any PTA M, one can construct an ω-PDA that accepts all leftmost paths of trees accepted by M. For results regarding ω-CFLs the reader is refered to [CG].

By defining STAs and proving the decidability of their emptiness problem, we have barely touched upon this subject. The related power and various accepting criteria of these machines is yet to be explored.

Acknowledgment

We thank G. Sénizergues for pointing out some inaccuracies in an earlier version of the paper.

Received May 1, 1991; final manuscript received May 8, 1992

References

[CG] Cohen, R. S., and Gold, A. Y. (1978), ω-Computations on deterministic pushdown machines, *J. Comput. System Sci.* **16**, 257–300.

[H] Harel, D. (1983), Dynamic logic, in "Handbook of Philosophical Logic" (Gabbay and Gunthner, Eds.), Vol. II, pp. 497–603, Reidel, Dordrecht.

[HoR] Hossley, R., and Rackoff, C. W. (1972), The emptiness problem for automata on infinite trees, *in* "13th IEEE Symposium on Switching and Automata Theory", pp. 121–124.

[HPS] Harel, D., Pnueli, A., and Stavi, J. (1983), Propositional dynamic logice of nonregular programs, *J. Comput. System Sci.* **26**, 222–243.

[HR] Harel, D., and Raz, D. (1993), Deciding properties of nonregular programs, *SIAM J. Comput.* **22**, 857–874.

[K] Kain, R. Y. (1972), "Automata Theory: Machines and Languages," pp. 195–202, McGraw–Hill, New York.

[R1] Rabin, M. O. (1969), Decidability of second order theories and automata on infinite trees, *Trans. AMS* **141**,1–35.

[R2] RABIN M. O. (1970), Weakly definable relations and special automata, *in* "Proceedings Symposium Mathematical Logic and Foundations of Set Theory" (Y. Bar-Hillel, Ed.), pp. 1–23, North-Holland, Amsterdam.

[S] SAOUDI, A. (1992), Pushdown automata on infinite trees and nondeterministic context-free programs, *Internat. J. Found. Comput. Sci.* **3**, 21–39.

[VW] VARDI, M., AND WOLPER, P. (1986), Automata-theoretic techniques for modal logics of programs, *J. Comput. System Sci.* **32**, 183–221.

[WW] WAGNER, K., AND WECHSUNG, G. (1986), "Computational Complexity," Reidel, Dordrecht.