International Journal of Foundations of Computer Science

A NEW TYPE OF PUSHDOWN AUTOMATA ON INFINITE TREES

WUXU PENG
Department of Computer Science
Southwest Texas State University
San Marcos, Texas 78666, U.S.A.
e-mail: peng@pirates.cs.swt.edu

and

S. PURUSHOTHAMAN IYER*
Department of Computer Science
North Carolina State University
Raleigh, North Carolina 27695, U.S.A.

Received 9 October 1994
Revised 13 March 1995
Communicated by O. Ibarra

ABSTRACT

In this paper we consider pushdown automata on infinite trees with empty stack as the accepting condition (ω-EPDTA). We provide the following regarding ω-EPDTA (a) its a relationship to other Pushdown automata on infinite trees, (b) a Kleene-Closure theorem and (c) a single exponential time algorithm for checking emptiness. We demonstrate the usefulness of ω-EPDTA through two example applications: defining the temporal *uniform inevitability* property and specifying a context-free process with unbounded state space, both of which cannot be defined and/or specified by the classical finite state automata on infinite trees . We also discuss the relevance of the results presented here to model-checking.

Keywords: Finite automata, finite automata on infinite trees, context-free tree grammars, modal checking.

1. Introduction

Although hardware advances have made concurrent systems feasible, succinctly specifying and efficiently verifying them remain a major challenge. The recent development of automated verification techniques for finite-state systems is one of the most exciting development in verifying concurrent systems.[3,4]

A number of existing automated verification techniques employ the *model checking* method. Model checking involves expressing the properties of a system to be verified using some logic (such as propositional versions of *temporal logic* or *mu-calculus*) and then invoking a *decision procedure* to test whether the *start state* of

*Supported in part by NSF CCR-9004121.

the system satisfies the properties. All the existing techniques interpret the logic formulae over *finite Kripke* structures, which can only represent finite-state systems. If a finite Kripke structure is unwound, we get a *finite-state automaton* on infinite trees. Formulae in logics such as mu-calculus can also be represented by FSA on infinite trees (ω-FTA's). Therefore the model checking problem for such systems is equivalent to testing whether the intersection of two ω-FTA's on ω-trees is empty.[21]

It is clear that finite Kripke structures can model the infinite behavior of finite-state systems. It is important to distinguish between finite-state systems and infinite behavior of such systems. Systems like S_1 characterized by regular expression $r = (ab)^\omega$ is finite-state, as we only need a finite number of states to represent the system. The fact that the length of each word in r is infinite reflects the fact that S_1 is *nonterminating*, or has infinite behavior.

There are systems that have unbounded number of states and also exhibit infinite behavior. Consider the system S_2 whose behavior is characterized by the context-free language L^ω where $L = \{a^i b^i | i \geq 0\}$. Like S_1, system S_2 is also nonterminating. In contrast to S_1, however, S_2 cannot be represented by a finite Kripke structure.

Despite some recent success in verifying *finite-state* systems,[4] little progress has been made for unbounded systems. Bradfield and Stirling recently proposed a proof system for properties of unbounded concurrent systems expressible in μ-calculus.[2] They showed that their proof system is sound and complete in the sense that any valid property (expressible in μ-calculus) can be verified and that any property proved valid is correct. Clearly, due to the infinite state space, any algorithm implementing their deduction system is not guaranteed to terminate when it is provided an invalid property.

In this paper we propose a new type of automata – *empty stack pushdown automata on infinite trees* (ω-EPDTA for short) which captures a class of unbounded state systems. We show that the class of ω-EPDTA's properly contains the class of ω-FTA's. As a result, EPDTA's can model all systems with finite Kripke structures.

Our notion of ω-EPDTA's is innovative. We show that the emptiness problem of ω-EPTDA's can be decided in single exponential time, which we believe is nearly optimal. Our algorithm for deciding the emptiness problem of ω-EPDTA extends Vardi and Wolper's good subtree techniques for deciding the emptiness problem of ω-FTA's.[20] The basic idea behind our algorithm is a two step reduction – reducing the emptiness problem of ω-EPDTA's to that of a collection of finite EPDTA's, and reducing the latter to the emptiness problem of indexed languages. The first reduction is possible due to the Kleene closure property enjoyed by ω-EPDTA's, where by Kleene closure property we mean that the set of trees accepted by an ω-EPDTA is expressible as a ω-Kleene closure of context-free sets of finite trees.

Our definition of finite EPDTA is of independent interest. Finite pushdown tree automata and context-free tree grammars were traditionally studied from an algebraic point view.[9,14] Our definition is purely automata-theoretic. Yet, as we show later in this paper, our definition is consistent with the traditional algebraic definitions.

The remainder of this paper is organized as follows. In Section 2, we present

the necessary basic notations about k-ary finite trees and k-ary ω-trees. Section 3 contains definitions for empty stack pushdown tree automata (EPDTA's), empty stack pushdown automata on ω-trees (ω-EPDTA's), pushdown automata on ω-trees (ω-PDTA's), and finite automata on ω-trees (ω-FTA's). In Section 4, we investigate the relationships among the families of trees accepted by these automata. We show in Section 5 that ω-EPDTA's have the Kleene closure property. Based on the Kleene closure property, in Section 6 we give an exponential algorithm for the emptiness problem of ω-EPDTA's. This result is in sharp contrast with the triple-exponential emptiness algorithm for ω-PDTA's by Saoudi.[17] Section 7 contains two example applications of ω-EPDTA: defining the temporal *uniform inevitability* property and specifying a context-free program with unbounded state space. Concluding remarks are given in Section 8.

2. Preliminaries

Let N denote the set of non-negative integers, $k > 0$ be some fixed integer constant, and Σ be a finite alphabet. Symbols from Σ will be denoted by lower case letters $a, b, \cdots$. For any set S, $|S|$ denotes the cardinality of S. For a symbol $a \in \Sigma$, the *arity* (or the *rank*) of a is the exact number of subtrees that any node labeled by a can have. In this paper, all symbols in Σ are assumed to be of arity k.

Let $[k]$ denote the index set $\{1, 2, \cdots, k\}$. *A k-ary tree t on Σ* is a mapping $dom(t) \longrightarrow \Sigma$ such that $dom(t)$ is a prefix-closed subset of $[k]^*$. Note that

(1) The root of t is λ;

(2) For a node u, its k children are $u1, u2, \cdots, uk$.

A tree t is an ω-tree if every branch of t is infinite. t is a finite tree if every branch of t is finite. For a node u in t, we use $t(u) \in \Sigma$ to denote its label.

$T_k(\Sigma)$ ($T_k^\omega(\Sigma)$, resp.) denotes the set of all finite k-ary trees (the set of all k-ary ω-trees, resp.). In this paper, we always assume that the only 0-ary symbol is $\perp$ which denotes an empty tree. Furthermore, we always restrict, without loss of generality, the nodes of the tree to either have all of its k children be nonempty trees or all of the children to be empty trees. Thus, we identify a node to be a *leaf node* provided *all of its children are* $\perp$. Note that $\perp$ plays the same role that fr^+ nodes play in the definitions used by Thomas and as such they are not part of the tree.[19]

For a set of variables $X = \{x_1, \cdots, x_n\}$, $T_k(\Sigma, X)$ denotes the set of all finite k-ary trees where some leaf nodes of each tree in it may be labeled by variables from X. For a tree $t \in T_k(\Sigma, X)$, we use $t \setminus X$ to denote the tree $t' \in T_k(\Sigma)$ obtained from t by removing all leaf nodes labeled by variables from X. We use the standard notation $u(t_1, \cdots, t_k)$ to denote a tree rooted at node u with k children $t_1, \cdots, t_k$. For a node $u = k_1 k_2 \cdots k_p$, any term $v = k_1 k_2 \cdots k_q$, where $q \leq p$, is called a *left factor* of u.

Let t, t' be trees on Σ and u be a node in t then $t(t'/u)$ denotes the tree obtained by substituting t' for the node u in t and is defined by:

(1) $t(t'/u)(uv) = t'(v)$ for all nodes v in t';

(2) $t(t'/u)(v) = t(v)$ if u is not a left factor of v.

The yield of a finite tree t is a word defined inductively by the homomorphism $yield : T_k(\Sigma) \longrightarrow \Sigma^*$:

(1) $yield(t) = t(u)$, if t is a tree with a single node u;

(2) $yield(u(t_1, \cdots, t_k)) = yield(t_1) \cdots yield(t_k)$.

Notice that by above definition, $yield(\perp)$ is undefined. We say that a tree t_1 is a *prefix* of a tree t_2, notated as $t_1 \leq t_2$, if $dom(t_1) \subseteq dom(t_2)$ and for each node $u \in dom(t_1)$ $t_1(u) = t_2(u)$. Tree t_1 is a *proper prefix* of tree t_2, notated as $t_1 < t_2$, if $dom(t_1) \subset dom(t_2)$ and for each node $u \in dom(t_1)$ $t_1(u) = t_2(u)$. For a set T of finite trees, the *Rabin limit* (cf. Ref. (17)) of T is the set of ω-trees defined by

$$Rlim(T) \ = \ \{t \mid \exists (t_i)_{0 \leq i \leq \omega} s.t. \forall i \geq 0 : t_i \in T, t_i < t_{i+1}, \text{ and } t_i < t\}.$$

Let $T_i \subseteq T_k(\Sigma, X)$, $0 \leq i \leq n$, be sets of finite trees. We use $T_0.\langle T_1, \cdots, T_n \rangle$ to denote the set of finite trees defined inductively as follows:

(i) For a tree $t = a \in T_0$, $t.\langle T_1, \cdots, T_n \rangle = \{t\}$;

(ii) If $t = x_i \in T_0$, then $t.\langle T_1, \cdots, T_n \rangle = T_i$;

(iii) If $t = a(t_1, \cdots, t_k) \in T_0$, then $t.\langle T_1, \cdots, T_n \rangle = \{a(s_1, \cdots, s_k) \mid s_j \in t_j.\langle T_1, \cdots, T_n \rangle, 1 \leq j \leq k\}$;

(iv) $T_0.\langle T_1, \cdots, T_n \rangle = \{t' \mid t' \in t.\langle T_1, \cdots, T_n \rangle$, for some $t \in T_0\}$.

Define $T_0.\langle T_1, \cdots, T_n \rangle^m = (T_0.\langle T_1, \cdots, T_n \rangle^{m-1}).\langle T_1, \cdots, T_n \rangle$, where $T_0.\langle T_1, \cdots, T_n \rangle^0 = T_0$. We further define $T_0.\langle T_1, \cdots, T_n \rangle^\omega = \{t \in T_k^\omega(\Sigma) \mid \exists (t_i)_{0 \leq i \leq \omega}$ s.t. $t_0 \in T_0$, $t_i \in t_{i-1}.\langle T_1, \cdots, T_n \rangle$, and $t \in Rlim(\{t_i \setminus X : 0 \leq i \leq \omega\})\}$. We say that a set T of ω-trees is the *ω-Kleene closure* of a collection of finite tree sets $T_0, T_1, \cdots, T_n$ if $T = T_0.\langle T_1, \cdots, T_n \rangle^\omega$.

3. EPDTA's and ω-EPDTA's

Definition 1 *(Empty Stack Pushdown Tree Automaton) An empty stack pushdown tree automaton (EPDTA for short) is a 6-tuple $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$, where Q is a finite set of states, Σ is the input alphabet, Γ is the stack alphabet, $q_0 \in Q$ is the initial state, $Z_0 \in \Gamma$ is the initial stack symbol. Without loss of generality, we assume that any stack symbol (including the initial stack symbol Z_0) that can appear at the bottom of the stack can never appear on top of any other stack symbol in the pushdown store. The transition function δ is a mapping:*

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \longrightarrow 2^{(Q \times \Gamma^*)^k} \cup 2^{Q \times \Gamma^*}$$

A transition rule of the form $(q, \gamma) \in \delta(q', \varepsilon, Z)$ is called an ε-rule, while a transition of the form $(q_1, \gamma_1) \cdots (q_k, \gamma_k) \in \delta(q, a, Z)$ is called a read rule. A configuration (at a particular node of a tree) is a tuple $\langle q, s \rangle$, where $q \in Q$ is the current state and $s = Zs' \in \Gamma^*$ is the stack contents with Z as the top stack symbol. An ε-path e of a configuration $\langle q_1, s_1 \rangle$ is a sequence of ε moves starting at $\langle q_1, s_1 \rangle$, i.e. $e = (\langle q_1, s_1 \rangle, \cdots, \langle q_l, s_l \rangle)$ where for $2 \leq i \leq l$, if $s_{i-1} = Zs'$, then $(q_i, \gamma) \in \delta(q_{i-1}, \varepsilon, Z)$, and $s_i = \gamma s'$. $E(q, s)$ denotes the set of all ε-paths starting at $\langle q, s \rangle$.

For a k-ary finite tree $t : [k]^* \longrightarrow \Sigma$, a run *of M on t is a tree $c : [k]^* \longrightarrow (Q \times \Gamma^*)^+$ inductively defined by*

(1) $c(\lambda) = (\langle q_0, Z_0\rangle, \langle q_1, s_1\rangle, \cdots, \langle q_l, s_l\rangle) \in E(q_0, Z_0)$;

(2) If $c(u) = (\langle q_1, s_1\rangle, \cdots, \langle q_l, s_l\rangle) \in E(q_1, s_1)$, *where* $s_l = z\gamma$, *and* $(q_1^1, s_1^1)\cdots(q_1^k, s_1^k) \in \delta(q_l, t(u), z)$, *and node* u *has children, then for* $1 \le i \le k$,

$$c(ui) = (\langle q_1^i, s_1^i\gamma\rangle, \cdots, \langle q_{n_i}^i, s_{n_i}^i\gamma\rangle) \in E(q_1^i, s_1^i\gamma).$$

A run c *is* accepting *if each leaf node* u *of* c *is labeled by an* ε*-path* $(\langle q_1, Z\rangle, \cdots, \langle q_l, Z\rangle)$ *where* Z_0 *is a symbol that can appear only at the bottom of the stack, and there exists* $(q^1, \varepsilon)\cdots(q^k, \varepsilon) \in \delta(q_l, t(u), Z_0)$.

An EPDTA M accepts a tree t if M has an accepting run on t. $T(M)$, the set of finite k-ary trees accepted by M, is defined by

$$T(M) \;=\; \{t \in T_k(\Sigma) | M \text{ has an accepting run on } t\}$$

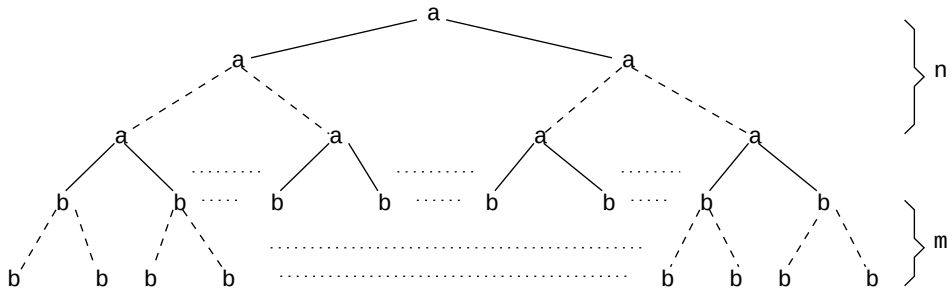


Fig. 1. 2-ary trees accepted by EPDTA M in Example 1

Example 1. Consider a 2-ary EPDTA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$ where $Q = \{q_0, q_1\}$, $\Sigma = \{a, b, c\}$, $\Gamma = \{Z_0, A, B\}$, and the transition function δ is defined by:

(1) $\delta(q_0, \varepsilon, Z_0) = \{(q_0, BZ_0), (q_0, AZ_0)\}$; (2) $\delta(q_0, \varepsilon, B) = \{(q_0, BB), (q_0, AB)\}$;
(3) $\delta(q_0, \varepsilon, A) = \{(q_0, AA), (q_1, A)\}$; (4) $\delta(q_1, a, A) = \{(q_1, \varepsilon)(q_1, \varepsilon)\}$;
(5) $\delta(q_1, b, B) = \{(q_1, \varepsilon)(q_1, \varepsilon)\}$; (6) $\delta(q_1, b, Z_0) = \{(q_1, \varepsilon)(q_1, \varepsilon)\}$.

Then $T(M)$ is the set of trees shown in Fig. 1, where $n, m > 0$. □

Definition 2 *(Empty Stack* ω*-Pushdown Tree Automata) An empty stack* ω*-pushdown* k*-ary tree automaton (*ω*-EPDTA for short) is a 6-tuple* $M = (Q, \Sigma, \Gamma, q_0, Y, \delta)$, *where* Q, Σ, Γ, q_0, *and* δ *are defined the same as EPDTA's, and* $Y \in \Gamma$ *is a special initial stack symbol that can only be at the bottom of the stack.*

For an ω*-tree* $t : [k]^* \longrightarrow \Sigma$, *a* run *of* M *on* t *is an* ω*-tree* $c : [k]^* \longrightarrow (Q \times \Gamma^*)^+$ *and is defined similar to the run on a finite tree, except that now a computation is on an infinite tree.*

Each infinite branch starting at the root λ *in a run* c *is called an* ω*-path. A run* $c : [k]^* \longrightarrow (Q \times \Gamma^*)^+$ *is* accepting *if the stack symbol* Y *becomes the top stack symbol infinitely many times along each* ω*-path.*

An ω-EPDTA M accepts an ω-tree t if M has an accepting run on t. $T(M)$ is the set of ω-trees on each of which M has an accepting run.

Example 2. Consider a 2-ary ω-EPDTA $M_1 = (Q, \Sigma, \Gamma, q_0, Y, \delta)$ where $Q = \{q_0, q_1\}$, $\Sigma = \{a, b, c\}$, $\Gamma = \{Y, Z_0, A, B\}$, and the transition function δ is defined by:

(1) $\delta(q_0, \varepsilon, Y) = \{(q_0, Z_0Y)\}$; (2) $\delta(q_0, \varepsilon, Z_0) = \{(q_0, BZ_0), (q_0, AZ_0)\}$;
(3) $\delta(q_0, \varepsilon, B) = \{(q_0, BB), (q_0, AB)\}$; (4) $\delta(q_0, \varepsilon, A) = \{(q_0, AA), (q_1, A)\}$;
(5) $\delta(q_1, a, A) = \{(q_1, \varepsilon)(q_1, \varepsilon)\}$; (6) $\delta(q_1, b, B) = \{(q_1, \varepsilon)(q_1, \varepsilon)\}$;
(7) $\delta(q_1, \varepsilon, Y) = \{(q_0, Y)\}$.

Call a finite 2-ary tree t an (n, m)-tree if t is accepted by M in Example 1 with $n > 0$ levels of a's and $m > 0$ levels of b's. $T(M_1)$ is the set of 2-ary ω-trees t such that there is an (n, m)-tree t_1 that is a prefix of t. At every leaf node of tree t_1, instead of having to terminate as in the EPDTA M, M_1 starts another round of actions to recognize another (n_1, m_1)-tree. This action repeats recursively to infinity. □

Definition 3 *(ω-Pushdown Tree Automata) An ω-pushdown k-ary tree automaton (ω-PDTA for short) is a 7-tuple $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$, where Q, Σ, Γ, q_0, and δ are defined as in ω-EPDTA's, Z_0 is the initial stack symbol, and $F \subseteq Q$ is the designated set of final states.*

For an ω-path $r = \Pi_{i=0}^{\omega} u_i$, $c(r) = \Pi_{i=0}^{\omega} c(u_i)$ denotes the infinite sequence obtained by concatenating ε-paths found on each node along r. Define the state trace *τ as a mapping: $[k]^* \longrightarrow Q^*$. For each node u in a run c, $\tau(u) = q_1 \cdots q_l$, where $(\langle q_1, s_1\rangle, \cdots, \langle q_l, s_l\rangle) = c(u)$. We extend τ as a mapping: $([k]^*)^\omega \longrightarrow (Q^*)^\omega$ such that for an ω-path $r = \Pi_{i=0}^{\omega} u_i$ in c, $\tau(r) = \Pi_{i=0}^{\omega} \tau(u_i)$.*

Let r be an ω-path such that $\tau(r) = q_0 q_1 \ldots q_i \ldots$. Path r induces a mapping $f_r : N \longrightarrow Q$ where $f_r(i) = q_i$. Define $inf(r) = \{q \mid |f_r^{-1}(q)| = \omega\}$. A run $c : [k]^* \longrightarrow (Q \times \Gamma^*)^+$ is accepting *iff $F \cap \inf(r) \neq \emptyset$ for each ω-path r.*

$T(M)$, the set of ω-trees accepted by an ω-PDTA M is defined similarly. An EPDTA (or ω-EPDTA, or ω-PDTA) M is *deterministic* if (1) $|\delta(q, a, z)| \leq 1$, where $a \in \Sigma \cup \{\varepsilon\}$; and (2) if $|\delta(q, \varepsilon, z)| = 1$ then $|\delta(q, a, z)| = 0$ for all $a \in \Sigma$. If the stack is removed from ω-PDTA's definition, we obtain the finite k-ary ω-tree automata (ω-FTA for short, cf. Ref. (21)). If $k = 1$ then we get pushdown automata on infinite strings (ω-PDA).[5]

Definition 4 *(Context-Free Tree Grammar[9]) An k-ary context-free tree grammar (CFTG) G is a 4-tuple (V_N, Σ, P, S_0), where (1) $V_N = \{S_0, S_1, \cdots, S_{m-1}\}$ is the set of finite ranked nonterminal symbols. For $S \in V_N$, $r(S)$ denotes its rank; (2) Σ is the set of finite ranked terminal symbols; (3) $S_0 \in V_N$ is the initial nonterminal of rank 0; (4) P is the set of productions of the form $S_i(x_1, \cdots, x_{r(S_i)}) \rightarrow t_i^j$, where for $i = 0, \cdots, m-1$, $j = 1, \cdots, n_i$, $t_i^j \in T_k(V_N \cup \Sigma, \{x_1, \cdots, x_{r(S_i)}\})$.*

A tree t derives *t', written as $t \Rightarrow t'$, iff there exists a prefix $\underline{t}$ of t, a production $S_i(x_1, \cdots, x_s) \rightarrow t_i^j \in P$, some node u of t labeled by S_i, subtrees $t_1, \cdots, t_s$ of t such that $t = \underline{t}(S_i(t_1, \cdots, t_s)/u)$ and $t' = \underline{t}(t_i^j(t_1, \cdots, t_s)/u)$. $t \Rightarrow t'$ means that, a node u in tree t labeled by S_i, which has children nodes $t_1, \cdots, t_s$, is replaced by*

$t_i^j(t_1, \cdots, t_s)$.

The transitive closure $\stackrel{*}{\Rightarrow}$ of one-step derivation $\Rightarrow$ is defined in the usual way. $T(G)$, the set of trees generated by G, is defined by

$$T(G) \;=\; \{t \mid t \in T_k(\Sigma) \;\&\; S \stackrel{*}{\Rightarrow} t\}.$$

Example 3. Consider the CFTG $G = (V_N, \Sigma, P, S_0)$, where $V_N = \{S_0, A_0, A_1, B_0, B_1, R\}, \Sigma = \{a, b, c\}, P$ contains productions: (1) $S_0 \rightarrow B_0(R) \mid A_0(R)$; (2) $B_0(x) \rightarrow B_0(B_1(x))$; (3) $B_0(x) \rightarrow A_0(B_1(x))$; (4) $A_0(x) \rightarrow A_0(A_1(x))$; (5) $A_0(x) \rightarrow A_1(x)$; (6) $A_1(x) \rightarrow a(x, x)$; (7) $B_1(x) \rightarrow b(x, x)$; (8) $R \rightarrow b$.

$T(G) = T(M)$, where M is the EPDTA in Example 1 in Section 2. For instance, the tree $a(b(b, b), b(b, b))$ is generated by the derivations: $S_0 \Rightarrow B_0(R) \Rightarrow A_0(B_1(R)) \Rightarrow A_1(B_1(R)) \Rightarrow a(B_1(R), B_1(R)) \Rightarrow a(b(R, R), B_1(R)) \Rightarrow a(b(b, R), B_1(R)) \Rightarrow a(b(b, b), B_1(R)) \Rightarrow a(b(b, b), b(R, R)) \Rightarrow a(b(b, b), b(b, R)) \Rightarrow a(b(b, b), b(b, b))$ □

A CFTG G is regular if for all $S \in V_N$, $r(S) = 0$, i.e. all nonterminals are parameterless in the productions.

We will use $\mathcal{FT}$ to denote the family of trees accepted by a Buchi tree automaton, or equivalently generated by regular tree grammars. The family of trees accepted by (or generated by, as appropriate) ω-PDA's, ω-FTA's, ω-PDTA's, ω-EPDTA's, EPDTA's, and CFTG's are denoted by $\mathcal{PD}_\omega$, $\mathcal{FT}_\omega$, $\mathcal{PDT}_\omega$, $\mathcal{EPDT}_\omega$, $\mathcal{EPDT}$, and $\mathcal{CFT}$ respectively.

4. $\mathcal{FT}_\omega \subset \mathcal{EPDT}_\omega \subset \mathcal{PDT}_\omega$

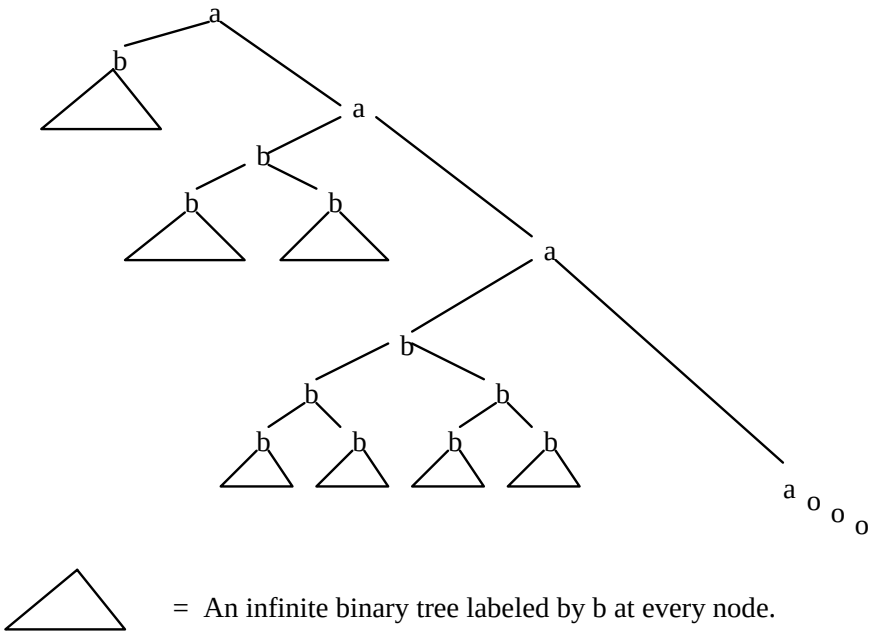
In this section we investigate the relationships among $\mathcal{PDT}_\omega$, $\mathcal{EPDT}_\omega$, and $\mathcal{FT}_\omega$. We show that $\mathcal{FT}_\omega$, the class of trees accepted by Büchi automata, is properly contained in $\mathcal{EPDT}_\omega$ and $\mathcal{EPDT}_\omega$ in turn is properly contained in $\mathcal{PDT}_\omega$.

Theorem 1 $\mathcal{FT}_\omega \subset \mathcal{EPDT}_\omega$.

Proof. Given any ω-FTA $A = (Q, \Sigma, \delta, q_0, F)$, we can construct an ω-EPDTA $M = (Q', \Sigma, \Gamma, q_0', Y, \delta')$ such that $T(A) = T(M)$ as follows. $Q' = Q \cup \{p\}$, where $p \notin Q$. $\Gamma = \{Z, Y\}$. M essentially puts ZY onto the stack and then simulates A's moves. Every time A enters a state $q \in F$, M will enter the special state p and pop Z (thus exposing symbol Y). It then pushes back Z onto the stack and return to state q to resume the simulation. It is clear to see that for any tree t, M has an accepting run on t iff A has one. Thus $\mathcal{FT}_\omega \subseteq \mathcal{EPDT}_\omega$ holds.

To see that the inclusion is proper, consider the ω-EPDTA M_1 in Example 2 in Section 3. We claim that there is no ω-FTA A such that $T(A) = T(M_1)$. To justify this, we observe that any tree t accepted by M_1 consists of a prefix t_0 of t that is an (n, m)-tree rooted at λ. Each of t_0's leaf nodes are the root of another (n, m)-tree, and so on. There are no FTA's that accept only (n, m)-trees. This can be easily proved by applying the Pumping Lemma from.[14] Hence there are no ω-FTA's that accept trees accepted by M_1. □

Theorem 2 $\mathcal{EPDT}_\omega \subset \mathcal{PDT}_\omega$.



= An infinite binary tree labeled by b at every node.

Fig. 2. An infinite tree showing $\mathcal{EPDT}_\omega \subset \mathcal{PDT}_\omega$

Proof. Let $M = (Q, \Sigma, \Gamma, q_0, Y, \delta)$ be an ω-EPDTA. We can construct an ω-PDTA $M' = (Q', \Sigma, \Gamma, q_0, Y, \delta', F)$ such that $T(M) = T(M')$ as follows. Define $\overline{Q} = \{\overline{q} | q \in Q\}$. $Q' = Q \cup \overline{Q}$; $F = \overline{Q}$; $\delta \subseteq \delta'$. In addition, δ' also contains rules $(\overline{q}, Y) \in \delta'(q, \varepsilon, Y)$ and $(q, Y) \in \delta'(\overline{q}, \varepsilon, Y)$, for each $q \in Q$. Intuitively, M' simulates M until Y is exposed in M. M' then can enter a state $\overline{q} \in F$, and come back to state q from state $\overline{q}$. It then continues the normal simulation of M. It is easy to see that $T(M) = T(M')$.

To show that the inclusion is proper, let us consider the ω-PDTA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$, where $Q = \{q_0, q_1, q_2\}$, $\Gamma = \{B, Z_0\}$, and δ is defined by (1) $\delta(q_0, a, Z_0) = \{(q_1, BZ_0)(q_2, BZ_0)\}$; (2) $\delta(q_1, b, B) = \{(q_1, \varepsilon)(q_1, \varepsilon)\}$; (3) $\delta(q_1, c, Z_0) = \{(q_1, Z_0)(q_1, Z_0)\}$; (4) $\delta(q_2, a, B) = \{(q_1, BB)(q_2, BB)\}$ (this example is an modified version of Example 1 from Ref. (17)). Let $F = \{q_1, q_2\}$. $T(M)$ contains only one 2-ary tree t where the right most branch of t contains the ω-word a^ω, and each of other branches begins with $i(> 0)$ a's, followed by i b's, and then followed by infinitely many c's (refer to Figure 2). It is easy to see that no EPDTA can accept $T(M)$, because the stack along the right most branch can never be emptied. Hence the inclusion is proper. □

5. Kleene Closure of $\mathcal{EPDT}_\omega$

$\mathcal{PD}_\omega$ and $\mathcal{FT}_\omega$ have the Kleene closure property. $\mathcal{PDT}_\omega$, on the other hand, does not have this property.[17] Kleene closure is important in that it provides a handy tool to prove the decidability of the emptiness problem.[13] In this section we show that $\mathcal{EPDT}_\omega$ also has Kleene closures. We begin by recalling one of the results

in Ref. (17).

Theorem 3 *(Nivat and Saoudi.[17]) For a set T of ω-trees, the following conditions are equivalent:*

(1) T is the ω-Kleene closure of $\mathcal{FT}$;

(2) T is accepted by an ω-FTA. □

Saoudi showed that $\mathcal{PDT}_\omega$ is not equal to the ω-Kleene closure of $\mathcal{CFT}$.[17] We show here that $\mathcal{EPDT}_\omega$ is equal to the ω-Kleene closure of $\mathcal{EPDT}$.

Let $M = (Q, \Sigma, \Gamma, q_0, Y, \delta)$ be an ω-EPDTA. Consider an ω-tree t and an accepting run c of M on t. Because c is an accepting run, Y is reached infinitely often along each ω-path of c. Therefore each branch starting from the root node λ of c will reach, infinitely often, a node u labeled by an ε-path $(\langle q_1, s_1\rangle, \cdots, \langle q_l, s_l\rangle)$, where for some i, $1 \leq i \leq l$, $s_i = Y$. Call such a node u a *periodic node*, and the configuration $\langle q_i, Y\rangle$ the *periodic configuration*. Starting from the root λ, c will reach a periodic node along each of its branches for the first time in a finite number of steps. It is easy to see that the subtree t_λ rooted at λ with all these periodic nodes as the leaf nodes is a finite k-ary tree that can be accepted by an EPDTA. Starting from any leaf node u of t_λ, M will continue to grow the stack and ultimately reach a periodic node along each of its branches for the first time in a finite number of steps. The subtree t_u rooted at u with these periodic nodes as leaf nodes is again a finite k-ary tree accepted by an EPDTA. The same arguments can be applied to the leaf nodes of the subtree t_u and so on.

On the other hand, assume that $T = T_0.\langle T_1, \cdots, T_n\rangle^\omega$, where $T_i = T(M_i)$, M_i is an EPDTA. Let $t \in T$. T_0 has a run on a prefix t_0 of t. All the leaf nodes are labeled by variables $x_j \in X$ and the stack of M_0 is emptied. From each leaf node x_j, T_j has a run on a portion of t and then eventually empties the stack at all its leaf nodes. Therefore it appears that we can construct an ω-EPDTA that accepts t. The following theorem makes these ideas precise.

Theorem 4 *For a set T of ω-trees, the following conditions are equivalent:*

(1) T is the ω-Kleene closure of $\mathcal{EPDT}$;

(2) T is accepted by an ω-EPDTA.

Proof. (1) $\Rightarrow$ (2). Assume that T is the ω-Kleene closure of the set of trees $T_0, T_1, \cdots, T_n \in \mathcal{EPDT}$, i.e. $T = T_0.\langle T_1, \cdots, T_n\rangle^\omega$. Let $M_i = (Q_i, \Sigma \cup X, \Gamma_i, q_0^i, Z_0^i, \delta_i)$, $0 \leq i \leq n$, be the EPDTA s.t. $T_i = T(M_i)$ (recall that $X = \{x_1, \cdots, x_n\}$). Assume that $Q_i \cap Q_j = \emptyset$ and $\Gamma_i \cap \Gamma_j = \emptyset$, when $i \neq j$. We construct, from M_i, $0 \leq i \leq n$, an ω-EPDTA $M = (Q, \Sigma, \Gamma, q_0, Y, \delta)$, s.t. $T(M) = T$. Specifically we have $Q = \cup_{i=0}^n Q_i \cup \{q^j : 1 \leq j \leq n\} \cup \{q_0\}$ $(q_0, q^j \notin \cup_{i=0}^n Q_i)$, $\Gamma = \cup_{i=0}^n \Gamma_i \cup \{Y\}$ $(Y \notin \cup_{i=0}^n \Gamma_i)$. The transition function δ includes all transition rules in δ_i $(0 \leq i \leq n)$, except those rules of the form $\delta_i(q, x_j, Z)$, where $x_j \in X$. In addition, δ also contains the following two types of rules: (a) $\delta(q_0, \varepsilon, Y) = \{(q_0^0, Z_0^0 Y)\}$; (b) If $(q_1^i, \varepsilon) \cdots (q_k^i, \varepsilon) \in \delta_i(q, x_j, Z)$, then $(q_0^j, \varepsilon) \in \delta(q, \varepsilon, Z)$, and $(q_0^j, Z_0^j Y) \in \delta(q_0^j, \varepsilon, Y)$.

Intuitively, at initial state q_0, M puts Z_0^0 onto the stack and starts to simulate M_0. If $(q_1^i, \varepsilon) \cdots (q_k^i, \varepsilon) \in \delta_i(q, x_j, Z)$, then at configuration $\langle q, ZY\rangle$ M will nondeterministically transfer to configuration $\langle q_0^j, Y\rangle$. At configuration $\langle q_0^j, Y\rangle$, M has to enter configuration $\langle q_0^j, Z_0^j Y\rangle$, hence starting to simulate M_j. If M gets into

configuration $\langle q', Z'Y\rangle$ where $(q_1^j, \varepsilon)\cdots(q_k^j, \varepsilon) \in \delta_j(q', x_l, Z')$, then M can nondeterministically transfer to configuration $\langle q_0^l, Z_0^l Y\rangle$, hence starting to simulate M_l, $1 \le j, l \le n$.

(2) $\Rightarrow$ (1). Assume that there exists an ω-EPDTA $M = (Q, \Sigma, \Gamma, q_0, Y, \delta)$, s.t. $T = T(M)$. Define $trans(M) = \{(q,\gamma) \mid (q,\gamma) \in \delta(q', \varepsilon, Z)\} \cup \{(q_i, \gamma_i) : 1 \le i \le k \mid \exists q \in Q, a \in \Sigma, Z \in \Gamma\ (q_1,\gamma_1)\cdots(q_k,\gamma_k) \in \delta(q, a, Z)\}$.

We construct, from M, a collection of EPDTA's $M_i = (Q_i, \Sigma \cup X, \Gamma_i, q_0^i, Y, \delta_i)$, $0 \le i \le n$, s.t. $T = T(M_0).\langle T(M_1), \cdots, T(M_n)\rangle^\omega$, where $n = |trans(M)|$. We assume that the tuples in $trans(M)$ is ordered and a function $\rho : trans(M) \longrightarrow \{1, \cdots, n\}$ such that for each tuple $(q,\gamma) \in trans(M)$ if $\rho(q,\gamma) = i$ then (q,γ) is the i^{th} tuple in $trans(M)$. Intuitively, each tuple (q,γ) corresponds to the machine M_j, where $\rho(q,\gamma) = j$, $1 \le j \le n$.

Let us call rules of the form $(q_1,\gamma_1 Y)\cdots(q_k,\gamma_k Y) \in \delta(q, a, Y)$ type (a) rule, and rules of the form $(q,\gamma Y) \in \delta(q', \varepsilon, Y)$ type (b) rules. Note that these rules are needed to continue after the bottom-of-the-stack symbol Y has been exposed.

M_i $(0 \le i \le n)$ is defined as follows. $Q_0 = Q, q_0^0 = q_0, Q_j = Q \cup \{q_0^j\}$ $(1 \le j \le n)$, where $q_0^j \notin Q$. Define $\Gamma_i = \Gamma \cup \{Z_{\rho(q,\gamma)} : (q,\gamma) \in trans(M)\}$. Finally, δ_i includes all the rules in δ except those rules involving $\delta(q,\theta,Y)$, where $\theta \in \Sigma \cup \{\varepsilon\}$. δ_i also contains the following two types of rules:

- For each type (a) rule $(q_1,\gamma_1 Y)\cdots(q_k,\gamma_k Y) \in \delta(q, a, Y)$ we should have

  $$(q_1, Z_{\rho(q_1,\gamma_1 Y)})\cdots(q_k, Z_{\rho(q_k,\gamma_k Y)}) \in \delta_i(q, a, Y)$$

  and $(q_i,\varepsilon)\cdots(q_i,\varepsilon) \in \delta_i(q_i, x_{\rho(q_i,\gamma_i Y)}, Z_{\rho(q_i,\gamma_i Y)})$.

- For each type (b) rule $(q,\gamma Y) \in \delta(q', \varepsilon, Y)$, we should add the transition $(q,\varepsilon)\cdots(q,\varepsilon) \in \delta_i(q', x_{\rho(q,\gamma Y)}, Y)$ to machine M_i.

In addition to all of these transitions, we need rules to start any machine M_j with the appropriate stack symbols. Consequently, we add the transition $\delta_j(q_0^j, \varepsilon, Y) = \{(q,\gamma Y)\}$, where $\rho(q,\gamma Y) = j$, to M_j.

M_0 starts with configuration $\langle q_0, Y\rangle$ (which is also M's initial configuration), while M_j $(1 \le j \le n)$ starts with configuration $\langle q_0^j, Y\rangle$ and then immediately transfers to configuration $\langle q,\gamma Y\rangle$,where $\rho(q,\gamma Y) = j$. After that, M_i $(0 \le i \le n)$ will simulate M until a type (a) or (b) rule is to be used by M. In case of type (a) rule, M_i acts according to the rule $(q_1, Z_{\rho(q_1,\gamma_1 Y)})\cdots(q_k, Z_{\rho(q_k,\gamma_k Y)}) \in \delta_i(q, a, Y)$, then follows the rule $(q_j,\varepsilon)\cdots(q_j,\varepsilon) \in \delta_i(q_j, x_{\rho(q_j,\gamma_j Y)}, Z_{\rho(q_j,\gamma_j Y)})$. Therefore every time M finds the symbol Y on top of the stack and is poised to enter the configuration $\langle q_j,\gamma_j Y\rangle$ at its j^{th} child node, M_i simply accepts $x_{\rho(q_j,\gamma_j Y)}$ and empties the stack. If $\rho(q_j,\gamma_j Y) = l$, then M_l will start to grow the tree from configuration $\langle q_j,\gamma_j Y\rangle$. □

6. Emptiness Problem of ω-EPDTA's

In this section, using the Kleene closure property of last section, we give an exponential algorithm for the emptiness problem of ω-EPDTA's. This result is in sharp contrast to the known triple-exponential algorithm for ω-PDTA's.[10]

By Theorem 4, each set $T(M)$ of ω-trees accepted by an ω-EPDTA M is the ω-Kleene closure of sets of finite trees $T(M_0), \cdots, T(M_n)$ accepted by EPDTA $M_0, \cdots, M_n$ respectively. The emptiness of $T(M)$ can be decided based on the emptiness of the $T(M_i)$'s. We first show that for each EPDTA M' there exists a CFTG G such that $T(M') = T(G)$. To test whether $T(G)$ is empty, we observe the fact that the set of yields of trees accepted by a CFTG G is an indexed language. This observation allows us to use the exponential time algorithm in Ref. (18) to test the emptiness of CFTG's.

Lemma 1 $\mathcal{EPDT} \subseteq \mathcal{CFT}$.

Proof. The construction follows that of Ref. (9) and is repeated as it is critical to the rest of the paper. Let $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$ be a EPDTA where $Q = \{q_0, q_1, \cdots, q_{m-1}\}$. Let $I = \{0, 1, \cdots, m-1\}$ and $X = \{x\}$. We construct a CFTG $G = (V_N, \Sigma, P, S_0)$ as follows. $V_N = \{Z^i | 0 \leq i \leq m-1, Z \in \Gamma\}$. Each nonterminal $Z^i \in V_N$ has rank $m \times r(Z^i)$, where $r(Z^i)$=0 if Z^i can appear at the bottom of the stack and 1 otherwise. Define $X^I = \{x^i | i \in I\}$. Define a mapping $\sigma : Q \times (\Gamma^* X \cup \Gamma^*) \longrightarrow T(V_N, X^I)$ inductively:

(1) $\sigma(q_i, s) = s^i$, if s is a variable or s is a bottom stack letter;

(2) $\sigma(q_i, \varepsilon) = \perp$;

(3) $\sigma(q_i, Zt) = Z^i(\sigma(q_0, t), \sigma(q_1, t), \cdots, \sigma(q_{m-1}, t))$.

The start nonterminal $S_0 = \sigma(q_0, Z_0) = Z_0^0$, and the set of productions P is defined by:

a) For each ε-rule $(q', \gamma) \in \delta(q, \varepsilon, Z_0)$, we have a production of the form:

$$\sigma(q, Z_0) \rightarrow \sigma(q', \gamma);$$

b) For each ε-rule $(q', \gamma) \in \delta(q, \varepsilon, Z)$, $Z \neq Z_0$, we have a production of the form:

$$\sigma(q, Zx) \rightarrow \sigma(q', \gamma x);$$

c) For each read rule $(q_1, \pi_1) \cdots (q_k, \pi_k) \in \delta(q, a, Z_0)$, we have a production of the form:

$$\sigma(q, Z_0) \rightarrow a(\sigma(q_1, \pi_1), \cdots, \sigma(q_k, \pi_k));$$

d) For each read rule $(q_1, \pi_1) \cdots (q_k, \pi_k) \in \delta(q, a, Z)$, $Z \neq Z_0$, we have a production of the form:

$$\sigma(q, Zx) \rightarrow a(\sigma(q_1, \pi_1 x), \cdots, \sigma(q_k, \pi_k x))'$$

Notice that in case (c), if $\pi_i = \varepsilon$, $1 \leq i \leq k$, then $\sigma(q_i, \pi_i) = \perp$, and hence $a(\sigma(q_1, \varepsilon), \cdots, \sigma(q_k, \varepsilon)) = a$. Following an argument analogous to that in Ref. (9), it can be easily shown that $T(M) = T(G)$. □

Example 4. For the EPDTA M in Example 1, we can construct an equivalent CFTG $G = (V_N, \Sigma, P, Z^0)$ using Lemma 1. $V_N = \{Z_0^0, Z_0^1, A^0, A^1, B^0, B^1\}$, where $r(Z_0^0) = r(Z_0^1) = 0$, $r(A^0) = r(A^1) = r(B^0) = r(B^1) = 2$, and the productions are

$Z_0^0 \rightarrow B^0(Z_0^0, Z_0^1) \mid A^0(Z_0^0, Z_0^1)$;
$B^0(x^0, x^1) \rightarrow B^0(B^0(x^0, x^1), B^1(x^0, x^1))$;
$B^0(x^0, x^1) \rightarrow A^0(B^0(x^0, x^1), B^1(x^0, x^1))$;
$A^0(x^0, x^1) \rightarrow A^0(A^0(x^0, x^1), A^1(x^0, x^1))$;
$A^0(x^0, x^1) \rightarrow A^1(x^0, x^1)$;
$A^1(x^0, x^1) \rightarrow a(x^1, x^1)$;
$B^1(x^0, x^1) \rightarrow b(x^1, x^1)$;
$Z_0^1 \rightarrow b$;

For instance, the tree $a(b(b,b),b(b,b))$ can be derived by

$$Z_0^0 \Rightarrow B^0(Z_0^0, Z_0^1) \Rightarrow A^0(B^0(Z_0^0, Z_0^1), B^1(Z_0^0, Z_0^1)) \Rightarrow A^1(B^0(Z_0^0, Z_0^1), B^1(Z_0^0, Z_0^1)) \Rightarrow$$
$$a(B^1(Z_0^0, Z_0^1), B^1(Z_0^0, Z_0^1)) \Rightarrow a(b(Z_0^1, Z_0^1), B^1(Z_0^0, Z_0^1)) \Rightarrow a(b(b, Z_0^1), B^1(Z_0^0, Z_0^1)) \Rightarrow$$
$$a(b(b,b), B^1(Z_0^0, Z_0^1)) \Rightarrow a(b(b,b), b(Z_0^1, Z_0^1)) \Rightarrow a(b(b,b), b(b, Z_0^1)) \Rightarrow a(b(b,b), b(b,b)).$$

□

A CFTG G is in *Quasi-Chomsky normal form* (QCNF) if each production in G is in one of the following forms:

(1) $A(x_1, \cdots, x_n) \rightarrow a(C_1(x_1, \cdots, x_n), \cdots C_m(x_1, \cdots, x_n))$;
(2) $A(x_1, \cdots, x_n) \rightarrow B(C_1(x_1, \cdots, x_n), \cdots C_m(x_1, \cdots, x_n))$;
(3) $A(x_1, \cdots, x_n) \rightarrow a(x_1, \cdots, x_n)$;
(4) $A(x_1, \cdots, x_n) \rightarrow x_i$

Recall (from Theorem 4) that for an ω-EPDTA M $trans(M)$ is defined as the set of all tuples of the form (q, γ) s.t. (q, γ) appears in some transition rule of M. Let $\eta(M) = max(|\gamma| : (q, \gamma) \in trans(M))$, and $rule(M)$ be the number of transition rules in M.

Lemma 2 *Let $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$ be an EPDTA and $G = (V_N, \Sigma, P, S_0)$ be the CFTG constructed from M by Lemma 1. If $\eta(M) \leq 2$ then G is in QCNF.*

Let $(q, \gamma) \in trans(M)$, such that $|\gamma| \leq 2$. Let $q_i \in Q$ and consider three cases:

Case 1: $\gamma = \varepsilon$. By Lemma 1, $\sigma(q_i, \varepsilon) = x^i$.

Case 2: $\gamma = Z$. $\sigma(q_i, \gamma x) = \sigma(q_i, Zx) = Z^i(\sigma(q_0, x), \cdots, \sigma(q_{m-1}, x)) = Z^i(x^0, \cdots, x^{m-1})$. Therefore $\sigma(q_i, \gamma x)$ meets the requirement of QCNF regardless of whether it will appear on right hand side or left hand side of a production.

Case 3. $\gamma = Z_2 Z_1$. $\sigma(q_i, \gamma x) = \sigma(q_i, Z_2 Z_1 x) = Z_2^i(\sigma(q_0, Z_1 x), \cdots, \sigma(q_{m-1}, Z_1 x))$ $= Z_2^i(Z_1^0(x^0, \cdots, x^{m-1}), \cdots, Z_1^{m-1}(x^0, \cdots, x^{m-1}))$. Notice that here $\sigma(q_i, \gamma x)$ can only appear at the right hand side of productions. Again it is easy to see that $\sigma(q_i, \gamma x)$ meets the requirement of QCNF.

It was well known that the set of yields of trees accepted by an CFTG G, $yield(G)$, is an indexed language.[14] Furthermore, if G is already in QCNF, then testing whether $yield(G) = \emptyset$ can be done in exponential time.[18] Because $T(G) = \emptyset$ iff $yield(G) = \emptyset$, we have the following theorem regarding the emptiness problem of EPDTA's.

Theorem 5 *Given an EPDTA M with $\eta(M) \leq 2$, the problem whether $T(M)$ is empty can be solved in exponential time.*

Theorem 6 *Let M be an ω-EPDTA with $\eta(M) \leq 2$. Let H be the time needed to solve the emptiness problem of CFTG's. Then the problem of whether $T(M)$ is empty can be solved in time $O(n^2(n+H))$, where $n = |\text{trans}(M)|$.*

Proof. Let $M = (Q, \Sigma, \Gamma, q_0, Y, \delta)$ be the given ω-EPDTA. Consider the Algorithm EMPTINESS shown in Table 1, where M_i, $0 \leq i \leq n$, are defined as in Theorem 4.

Claim: $T(M) \neq \emptyset$ iff $M_0 \in C_{new}$ at the end of Algorithm EMPTINESS.

Proof of the claim: $\Rightarrow$ Assume that $T(M) \neq \emptyset$. Let $t \in T(M)$ and c be an accepting run of M on t. Let u_i, $1 \leq i \leq n_1$ be the collection of first periodic nodes from the root λ of ω-tree c and let $\langle q^i, Y\rangle$ be the periodic configuration associated with u_i. There are two cases to consider, depending upon if the periodic configuration is the last configuration in the ε-path of $c(u_i)$. We consider the case where the periodic configuration is the last one (the other case is simpler). Assume that $(q_1, \gamma_1 Y) \cdots (q_k, \gamma_k Y) \in \delta(q^i, a, Y)$. Then δ_0 has the rules $(q_1, Z_{\rho(q_1,\gamma_1 Y)}) \cdots (q_k, Z_{\rho(q_k,\gamma_k Y)}) \in \delta_0(q^i, a, Y)$ and $(q_i, \varepsilon) \cdots (q_i, \varepsilon) \in \delta_0(q_i, x_{\rho(q_i,\gamma_i Y)}, Z_{\rho(q_i,\gamma_i Y)})$. If $T(M_{\rho(q_i,\gamma_i)}) = \emptyset$, for some i, then $t \notin T(M)$, because t would have a finite branch. Therefore all $M_{\rho(q_i,\gamma_i)}$, $1 \leq i \leq n_1$, are still in C_{new} after the algorithm. Hence $T(M_0) \neq \emptyset$ at the end of the algorithm, which implies that $M_0 \in C_{new}$.

$\Leftarrow$ Assume that $M_0 \in C_{new}$ at the end of Algorithm EMPTINESS. Let $t_\lambda \in T(M_0)$ and c_λ be a run of M_0 on t_λ. Let $u_1, u_2, \cdots, u_{n_1}$ be the leaf nodes of c_λ and $\langle q_i, Z_{(q_i,\gamma_i)}\rangle$ be the last configuration of $c_\lambda(u_i)$. Each $\langle q_i, Z_{(q_i,\gamma_i)}\rangle$ corresponds to an emptying action $(q_i, \varepsilon) \cdots (q_i, \varepsilon) \in \delta_0(q_i, x_{(q_i,\gamma_i)}, Z_{(q_i,\gamma_i)})$. Because these emptying actions are still in M_0, we can conclude that $M_{\rho(q_i,\gamma_i)}$ is still in C_{new} at the end of the algorithm (cf. line 10, 11). Hence $T(M_{\rho(q_i,\gamma_i)}) \neq \emptyset$, $1 \leq i \leq n_1$. Now we can apply the same argument to $M_{\rho(q_i,\gamma_i)}$. The leaf nodes of any tree u_i accepted by $M_{\rho(q_i,\gamma_i)}$ again will refer to a collection of periodic nodes and configurations associated with emptying actions. Continuing the argument we can obtain an infinite sequence of trees $(t_i)_{0 \leq i \leq \omega}$, such that $t_i \in t_{i-1}.\langle T_1, \cdots, T_n\rangle$. Therefore there exists $t \in \mathit{Rlim}((t_i)_{0 \leq i \leq \omega})$, hence $T(M) \neq \emptyset$.

Each iteration of the **while** loop will remove at least one EPDTA $M_{(q,\gamma)}$ from $c(new)$. Line 5 will be executed $O(n^2)$ times. Because $\eta(M) \leq 2$, each EPDTA M_i can be transformed to a CFTG G_i. By Theorem 5, testing the emptiness of each of these CFTG's takes time $O(H)$. The **for** loop at lines 9, 10, 11 takes $O(n^3)$. All together the algorithm takes time $O(n^2(n+H))$. □

Theorem 6 is correct when $\eta(M) \leq 2$. Although most ω-EPDTA's do satisfy this requirement, to be complete we should also consider the case where $\eta(M) > 2$. The following theorem says that given any ω-EPDTA M with $\eta(M) > 2$, we can create another ω-EPDTA M_1, such that $\eta(M_1) \leq 2$, with a comparable number of states and transition rules.

Theorem 7 *For each ω-EPDTA $M = (Q, \Sigma, \Gamma, q_0, Y, \delta)$ with $\eta(M) = m > 2$, we can construct another ω-EPDTA $M_1 = (Q_1, \Sigma, \Gamma, q_0, Y, \delta_1)$ equivalent to M such that $\eta(M_1) = 2$, and $|Q_1| = O(nm|Q|)$, where $n = |\text{trans}(M)|$, and* rule$(M_1) \leq$

Algorithm EMPTINESS

1. $C_{new} := \{M_0\} \cup \{M_j \mid (q,\gamma) \in trans(M)\ \&\ \rho(q,\gamma) = j\}$;
2. $C_{old} := \emptyset$;
3. **while** $(C_{new} \neq C_{old})$ **and** $(M_0 \in C_{new})$ **do**
4. $\quad C_{old} := C_{new}$;
5. $\quad$ Let $E := \{M_{\rho(q,\gamma)} \mid M_{\rho(q,\gamma)} \in C_{new}\ \&\ T(M_{\rho(q,\gamma)}) = \emptyset\}$;
6. $\quad$ **if** $E = \emptyset$ **then**
$\quad\quad$ exit;
$\quad$ **endif**;
8. $\quad C_{new} := C_{new} - E$;
9. $\quad$ **for** each $M_{\rho(q,\gamma)} \in C_{new}$ **do**
10. $\quad\quad$ Let $D := \{\delta_{\rho(q,\gamma)}(q_i, x_{\rho(q_i,\gamma_i)}, Z_{\rho(q_i,\gamma_i)}) : M_{\rho(q_i,\gamma_i)} \in E\}$;
11. $\quad\quad \delta_{\rho(q,\gamma)} = \delta_{\rho(q,\gamma)} - D$;
$\quad$ **endfor**;
endwhile;

Table 1: Algorithm EMPTINESS

$m \times \text{rule}(M)$.

Proof. Let $M = (Q, \Sigma, \Gamma, q_0, Y, \delta)$ be an ω-EPDTA such that $\eta(M) = m > 2$. We construct another ω-EPDTA $M_1 = (Q_1, \Sigma, \Gamma_1, q_0, Y, \delta_1)$ such that $T(M) = T(M_1)$ and $\eta(M_1) = 2$ as follows. $Q_1 = Q \cup Q'$, where $Q' = \{[q,\gamma] \mid (q, \gamma\gamma' ZZ')$ $\in trans(M)$ where $Z, Z' \in \Gamma$, $\gamma, \gamma' \in \Gamma^*$ and $|\gamma| > 0\}$. It is clear that $|Q'| \leq nm|Q|$ where $n = |trans(M)|$. δ_1 contains all the rules in δ except those rules that involve tuples (q,γ) where $|\gamma| > 2$.

Each ε-rule $(q,\gamma) \in \delta(q', \varepsilon, Z)$, where $\gamma = Z_{i_j} \cdots Z_{i_1}$, $j > 2$, is decomposed into $j-1$ ε-rules: $([q, Z_{i_j} \cdots Z_{i_3}], Z_{i_2} Z_{i_1}) \in \delta(q', \varepsilon, Z)$, $([q, Z_{i_j} \cdots Z_{i_{l+1}}], Z_{i_l} Z_{i_{l-1}}) \in \delta([q, Z_{i_j} \cdots Z_{i_l}], \varepsilon, Z_{i_{l-1}})$, $3 \leq l \leq j-1$, and $(q, Z_{i_j} Z_{i_{j-1}}) \in \delta([q, Z_{i_j}], \varepsilon, Z_{i_{j-1}})$.

Each read rule $(q_1,\gamma_1) \cdots (q_k,\gamma_k) \in \delta(q, a, Z)$, where some $|\gamma_i| > 2$, is decomposed into the one read rule and a collection of ε-rules as follows. If $|\gamma_i| > 2$, write $\gamma_i = \beta_i \gamma_i'$, where $|\gamma_i'| = 2$. The read rule is: $\alpha_1 \cdots \alpha_k \in \delta(q, a, Z)$, where $\alpha_i = (q_i, \gamma_i)$ if $|\gamma_i| \leq 2$, and $\alpha_i = ([q_i, \beta_i], \gamma_i')$ if $|\gamma_i| > 2$. For each $([q_i, \beta_i], \gamma_i')$, where $\beta_i = Z_{i_j} \cdots Z_{i_1}$, $j > 0$, and $\gamma_i' = Z\gamma_i''$, we have the following ε-rules, depending upon the value of j. If $j = 1$, then we have only one ε-rule: $(q_i, Z_{i_1} Z) \in \delta([q_i, Z_{i_1}], \varepsilon, Z)$. If $j > 1$, we have j corresponding ε-rules: $([q_i, Z_{i_j} \cdots Z_{i_2}], Z_{i_1} Z)$ $\in \delta(q_i, \varepsilon, Z)$, $([q_i, Z_{i_j} \cdots Z_{i_{l+1}}], Z_{i_l} Z_{i_{l-1}}) \in \delta([q_i, Z_{i_j} \cdots Z_{i_l}], \varepsilon, Z_{i_{l-1}})$, $2 \leq l \leq j-1$, $(q_i, Z_{i_j} Z_{i_{j-1}}) \in \delta([q_i, Z_{i_j}], \varepsilon, Z_{i_{j-1}})$.

It is straightforward to show that $T(M) = T(M_1)$. Moreover $rule(M_1) \leq \eta(M) \times rule(M)$. Hence the conclusion holds. □

Combining Theorem 6 and 7, we have our main theorem.

Theorem 8 *The emptiness problem for ω-EPDTAs can be solved in time exponential to their size.*

7. Applications

Automata on infinite trees play an important role in modal logics of programs. Consider non-terminating programs with finite states. Informally, the behavior of any such program R can be captured by an ω-FTA M as follows. Each node v of M represents a global state s of R. Each child node of v represents a possible successor state of s. The root node of M corresponds to the initial state of R. Every infinite path starting from the root node of M represents a possible execution history of R. Let P be a logical property of interest to the correctness of R. From the point view of automata theory, verifying if R possesses P is equivalent to checking if P is satisfied along each infinite path of M, or alternatively, if M *recognizes* the property P.

Emerson in Ref. (6) gave a more formal description of an ω-FTA recognizing a property. The fundamental correctness property of concurrent programs known as *inevitability of predicate* P is expressible in branching time temporal logic as AFP, meaning that along every future computation path r there exists a time i along r at which P is true. Given a set of atomic propositions $AP = \{P_1, \cdots, P_k\}$, let Σ be the set of 2^k strings $P_1^* \cdots P_k^*$, where each P_i^* denotes either P_i or its negation. Assume the *Muller acceptance condition*, which is defined in terms of a family $F \subseteq \text{PowerSet}(Q)$ of designated subsets of the state set Q. A sequence r meets the Muller acceptance condition iff $inf(r) \subseteq F$.[6,20] Then, ω-FTA's on Σ can be viewed in an obvious way as defining (branching time) temporal logic modalities. For example, (ordinary) inevitability of P, AFP, corresponds to the ω-FTA $M_0 = (Q, \Sigma, \delta, q_0, F)$, where

- $Q = \{q_0, q_1\}$; $\Sigma = \{P, \overline{P}\}$; $F = \{\{q_0\}\}$;
- The transition function δ:

$$\delta(q_0, P) = (q_1, q_1); \quad \delta(q_0, \overline{P}) = (q_0, q_0);$$
$$\delta(q_1, P) = (q_1, q_1); \quad \delta(q_1, \overline{P}) = (q_1, q_1).$$

Emerson showed in Ref. (6) that the correctness property *uniform inevitability of a predicate* P is not definable by any type of finite automata on infinite trees. Uniform inevitability is a property that is much stronger than the ordinary inevitability property AFP. It means that there exists a time i such that, along every computation path r, P holds at time i. He also indicated that a tree with a pushdown store can be used to define uniform inevitability.

Indeed, the uniform inevitability of P in branching time temporal logic is definable by ω-EPDTA's. Intuitively, an ω-EPDTA can use its pushdown store to *register* the time i at which the predicate P should hold along every computation path. The content of the pushdown store is then distributed to each of its the child nodes. A child node will first pop the pushdown store and then distribute the remaining content of the pushdown store to each of its own child node in turn. Notice that ω-FTA's have no pushdown store, hence have no such ability. Let P be an atomic proposition and $\overline{P}$ be its negation. Consider the following ω-EPDTA $M' = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$, where

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $\Sigma = \{P, \overline{P}\}$, and $\Gamma = \{Z_0, Z\}$.
- The transition function δ:

$$\begin{array}{l}
\delta(q_0, \varepsilon, Z_0) = \{(q_1, ZZ_0)\}; \\
\delta(q_0, \varepsilon, Z_0) = \{(q_1, Z_0)\}; \\
\delta(q_1, \varepsilon, Z) = \{(q_1, ZZ)\}; \\
\delta(q_1, \varepsilon, Z) = \{(q_2, Z)\}; \\
\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}; \\
\delta(q_2, P, Z) = \{(q_2, \varepsilon)(q_2, \varepsilon)\}; \\
\delta(q_2, \overline{P}, Z) = \{(q_2, \varepsilon)(q_2, \varepsilon)\}; \\
\delta(q_2, \varepsilon, Z_0) = \{(q_3, Z_0)\}; \\
\delta(q_3, P, Z_0) = \{(q_4, Z_0)(q_4, Z_0)\}; \\
\delta(q_4, P, Z_0) = \{(q_4, Z_0)(q_4, Z_0)\}; \\
\delta(q_4, \overline{P}, Z_0) = \{(q_4, Z_0)(q_4, Z_0)\}.
\end{array}$$

We claim that the above ω-EPDTA M' defines the uniform inevitability of P.

Proof sketch for the claim: In state q_0, M' will enter state q_1 by pushing either a stack symbol Z or nothing onto the pushdown store. In state q_1, M' has three types of possible actions:

- $\delta(q_1, \varepsilon, Z) = \{(q_1, Z)\}$: It pushes Z onto the pushdown store and remains in state q_1;
- $\delta(q_1, \varepsilon, Z) = \{(q_2, Z)\}$: It pushes Z onto the pushdown store and enters state q_2;
- $\delta(q_1, \varepsilon, Z) = \{(q_2, Z)\}$: With Z_0 at the top pushdown store symbol, it enters state q_2 without changing the pushdown store.

In state q_2, if Z is the top pushdown store symbol, M' will just pop the pushdown store and then transfer control to its two child nodes. If Z_0 is the top pushdown store symbol, M' will enter state q_3. In state q_3, which is a critical state, M' has popped off all the Z's on the pushdown store that were pushed onto before. It insists on reading a P before it transfers control to its two child nodes with q_4 as their states. The actions in state q_4 are unimportant to the uniform inevitability property and hence are arbitrary.

It can be easily shown by induction on i, that every ω-tree accepted by M' consists of two parts. The first part is a complete binary tree of height i rooted at the root of the ω-tree. M' is in state q_3 at every leaf node of this complete binary tree. The second part consists of all those computations that start from those leaf nodes of the complete binary tree. □

Because the class of ω-EPDTA is enclosed in the class of ω-PDTA, it appears that ω-EPDTA is the smallest class of pushdown automata on infinite trees that can define the uniform inevitability property. Notice that the above ω-EPDTA is nondeterministic.

ω-EPDTA can be used to specify a class of unbounded state space systems. For example, we can capture the behavior of a counter which counts but which should count down to zero infinitely often, by the following ω-EPDTA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$:

- $Q = \{q_0\}$, Σ=$\{$inc, dec$\}$, and $\Gamma = \{Z_0, Z\}$.
- $$\delta(q_0,\ \text{inc}, Z_0) = \{(q_0, ZZ_0)(q_0, ZZ_0)\}$$
$$\delta(q_0,\ \text{inc}, Z) = \{(q_0, ZZ)(q_0, ZZ)\}$$
$$\delta(q_0,\ \text{dec}, Z) = \{(q_0, \varepsilon)(q_0, \varepsilon)\}$$

Clearly ω-EPDTA can be used to capture systems that have a context-free behavior. There has been a lot of interest lately on context-free processes.[1,12] We conjecture that the behavior of normed context-free processes, as discussed in Ref.(12), can be captured as trees accepted by a ω-EPDTA.

8. Discussion

We have introduced the notion of ω-EPDTA that accepts an infinite tree provided the bottom-of-the-stack symbol is exposed infinitely often in some run of the tree. We have established its relation to pushdown automata on infinite trees that accept by distinguished set of final states infinitely often. More importantly, we have shown that trees accepted by ω-EPDTA are expressible as ω-Kleene closure of a finite number of context-free term languages. We have shown that the emptiness problem for ω-EPDTA is solvable in single exponential time, which is probably optimal.

We have also shown that ω-EPDTA, which is properly enclosed in the class of ω-PDTA, can define the *uniform inevitability* property. In addition, ω-EPDTA can specify a class of concurrent systems with unbounded state spaces. In contrast, the classical ω-FTA can neither define the uniform inevitability property, nor specify systems with unbounded state spaces.

The exponential complexity for deciding the emptiness of ω-EPDTA can be an obstacle in applying ω-EPDTA to model checking. However, the work in Ref.(8) indicates that polynomial time decision procedures might exist for certain class of concurrent systems that can be specified by ω-EPDTA. Further research is needed in this direction.

Acknowledgments

. The authors would like to thank the anonymous referee who pointed out an error in one definition and many other typo errors.

References

1. J. C. M. Baeten, J. A. Bergstra, J. W. Klop, "Decidability of Bisimulation Equivalence for Processes Generating Context-Free Languages," *LNCS 259*, Springer-Verlag, 1987, pp.93-114.
2. J. Bradfield and C. Stirling, "Local Model Checking for Infinite State Spaces," *Theoretical Comp. Sci.*. to appear.
3. E. M. Clarke, E. A. Emerson, A. P. Sistla, "Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications: a Practical Approach," *Proc. of the 10th ACM Symp. on Principles of Prog. Lang.*, 1983, pp.117-126.

4. R. Cleaveland, "Tableau-Based Model Checking in the Propositional Mu-Calculus," *Acta Informatica*, **27**(1990) 725-747.

5. R. S. Cohen and A. Y. Gold, "Theory of ω-Languages I: Characterizations of ω-Context-Free Languages," *J. of Comp. and Sys. Sci.*, **15**(1977) 169-184.

6. E. A. Emerson, "Uniform Inevitability Is Tree Automaton Ineffable," *Info. Processing Letter*, Vol.24(3), Jan., 1987, pp.77-79.

7. E. A. Emerson and C. S. Jutla, "The Complexity of Tree Automata and Logics of Programs," *Proc. of the 29th IEEE Symp. on Found. of Comp. Sci.*, 1988, pp.328-337.

8. E. A. Emerson, T. Sadler, and J. Srinivasan, "Efficient Temporal Satisfiability," *J. of Logic Computation*, Vol.2(2), Oxford University Press, 1992, pp.173-210.

9. I. Guessarian, "Pushdown Tree Automata," *Math. Sys. Theory*, **16**(1983) 237-263.

10. D. Harel and D. Raz, "Deciding Properties of Nonregular Programs," *Proc of IEEE 31st Symp. on Found. of Comp. Sci.*, 1990, pp.652-661.

11. J. E. Hopcroft and J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1979.

12. H. Hüttel and C. Stirling, "Actions Speak Louder Than Words: Proving Bisimularity for Context-Free Processes," *Report ECS-LFCS-91-145*, Dept. of Comp. Sci., Univ. of Edinburgh, April 1991. Also in *Proc. of 6th Symp. on Logic in Comp. Sci.*, 1991.

13. M. Linna, "On ω-Sets Associated with Context-Free Languages," *Information and Control*, **31**(1976) 272-293.

14. T. S. E. Maibaum, "Pumping Lemmas for Term Languages," *J. of Comp. and Sys. Sci.*, **17**(1978) 319-330.

15. R. Milner, Communication and Concurrency, Prentice-Hall, 1989.

16. W. Peng and S. Purushothaman, "Empty Stack Pushdown ω-Tree Automata," *Proc. of 17th Colloquium on Trees in Algebra and Programming*, Rennes, France, Feb. 1992. LNCS 581, pp.248-264.

17. A. Saoudi, "Pushdown Automata on Infinite Trees and Omega-Kleene Closure of Context-Free Tree Sets," *Proc. of Math. Found. of Comp. Sci., LNCS 379*, Springer-Verlag, 1989.

18. S. Tanaka and T. Kasai, "The Emptiness Problem for Indexed Language is Exponential-Time Complete," *Sys. and Comp. in Japan*, Vol. 17(9), 1986, pp.29-37.

19. W. Thomas. "Automata on Infinite Objects," In *Handbook of Theoretical Computer Science, Vol B*, (ed) J. van Leeuwen, MIT Press, 1990, pp.133-192.

20. M. Y. Vardi and P. Wolper, "Automata-Theoretic Techniques for Modal Logics of Programs," *J. of Comp. and Sys. Sci.*, **32**(1986) 183-221.

21. M. Y. Vardi and P. Wolper, "An Automata-Theoretic Approach to Automatic Program Verification," *Proc. of 1st Symp. on Logic in Comp. Sci.*, 1986, pp.332-344.