ALTERNATING TREE AUTOMATA

Giora Slutzki[†]
Department of Computer Science
University of Kansas
Lawrence, Kansas, 66045

1. Introduction

In [CS, K] the concept of alternation has been introduced as a natural extension of the concept of nondeterminism. Intuitively, once we can think of a nondeterministic machine as an automaton all of whose configurations are existential, it is natural to generalize by distinguishing between existential and universal configurations. A configuration α is universal if all computations starting from α are accepting computations. The concept of existential configurations is the standard one. The effect of alternation was studied in the context of computational complexity, finite automata and (one-way and two-way) pushdown (and stack) automata [CS, K, CKS, LLS, S]. Some applications of alternation have been considered in algebra [B, K1], in analysis of propositional dynamic logic [FL] and combinatorial games [SC, KAI].

In this paper we discuss the effect of alternation on several varieties of tree automata. With respect to each class of automata we distinguish the following four subclasses:

(i) alternating automata--allowing the full power of alternation

(ii) nondeterministic automata--all configurations are existential

(iii) universal automata--all configurations are universal

(iv) deterministic automata--the transition function is a partial function.

The paper consists of seven sections of which this is the first. In the next section we recall some standard notation from tree language theory and in Section 3 we discuss alternating (one-way) top-down tree automata. For these automata we prove the equivalence of universality and determinism. Alternation and nondeterminism are also equivalent, but this we will prove in Section 5. In Section 4 we consider alternating two-way finite tree automata. In this case universality and nondeterminism are incomparable. In Section 5 and 6 we discuss two types of alternating two-way

[†]Supported in part by University of Kansas General Research Allocation #413-3232-20-0038.

pushdown tree automata. In Section 5 we study the synchronized type [ERS, KS] and in Section 6 we study the backtracking type [Ka]. In both cases we show that alternation is equivalent to determinism. In Section 7 we summarize.

2. Preliminaries.

An alphabet Σ is ranked if $\Sigma = \cup_k \Sigma_k$ where each Σ_k is a finite set and only for a finite number of k's, $\Sigma_k \neq \emptyset$. Elements of Σ_k are said to be of rank k. Given a ranked alphabet Σ, the set of trees over Σ, denoted by T_Σ, can be considered as a language over the alphabet $\Sigma \cup \{(,)\}$ defined recursively as follows: (i) $\Sigma_0 \subseteq T_\Sigma$. (ii) if $k \geq 1$, $\sigma \in \Sigma_k$ and $t_1,\ldots,t_k \in T_\Sigma$, then $\sigma(t_1 \ldots t_k) \in T_\Sigma$. Let Σ be a ranked alphabet and let S be a set of symbols or trees. The set of trees over Σ indexed by S, denoted by $T_\Sigma[S]$, is defined recursively: (i) $S \cup \Sigma_0 \subseteq T_\Sigma[S]$. (ii) if $k \geq 1$, $\sigma \in \Sigma_k$ and $t_1,\ldots,t_k \in T_\Sigma[S]$, then $\sigma(t_1 \ldots t_k) \in T_\Sigma[S]$. Let $X = \{x_1, x_2, \ldots\}$ be an infinite set of variables. These are used in the production rules of top-down tree automata. For any set S, $P(S)$ ($P_{fin}(S)$) denotes the set of all (finite) subsets of S.

3. Alternating One-way Top-down Tree Automata.

Parallel (one-way) top-down tree automata, nondeterministic and deterministic, are well known [D, E, El, MM, R, T, TW]. The relations between these are also well known: the nondeterministic automata characterize a class of tree languages called RECOG and they are more powerful than the deterministic automata. Here we generalize by introducing alternation (and universality). It turns out that alternating and nondeterministic top-down tree automata are equipotent; the same holds for universal and deterministic top-down automata. We proceed to the definitions.

Definition 3.1. An alternating (one-way) top-down tree automaton (atta) is a construct $M = (Q, U, q_0, \Sigma, R)$ where

Q is a finite nonempty set of states

$U \subseteq Q$ is the set of universal states; states in Q-U are called existential states

$q_0 \in Q$ is the initial state

Σ is a ranked input alphabet

R is a finite set of rules of the form

$$q(\sigma(x_1 \ldots x_k)) \to \sigma(q_1(x_1) \ldots q_k(x_k))$$

where $k \geq 0$, $\sigma \in \Sigma_k$ and $q, q_1, \ldots, q_k \in Q$.

An <u>instantaneous description</u> (ID) of M on a tree t in T_Σ is a tree in the set $Q(T_\Sigma) \cup T_\Sigma[Q(T_\Sigma)] \cup T_\Sigma$, where $Q(T_\Sigma)$ is the set of trees $\{q(t) \mid q \in Q,\ t \in T_\Sigma\}$ (here we view states as having rank 1). $q_0(t)$ is the <u>initial</u> ID of M on t, and trees in T are the <u>accepting</u> ID's. For two ID's s and r we write $s \vdash_M r$ if there is a rule $q(\sigma(x_1 \ldots x_k)) \to \sigma(q_1(x_1) \ldots q_k(x_k))$ in R such that r is obtained from s by replacing a subtree of s of the form $q(\sigma(t_1 \ldots t_k))$ for certain $t_1, \ldots, t_k \in T_\Sigma$ by $\sigma(q_1(t_1) \ldots q_k(t_k))$. Given $\vdash_M$, $\overset{*}{\vdash}_M$ is the reflexive-transitive closure of $\vdash_M$. A computation tree of M on t is a finite, nonempty tree labeled by ID's of M (on t) and satisfying the following properties.

(i) The root of the tree is labeled by the initial ID of M on t: $q_0(t)$.

(ii) Let n be an internal node of the tree labeled by an ID s and let $q(\sigma(t_1 \ldots t_k))$ be a subtree of s (at node π) for some $\sigma \in \Sigma_k$ and $t_1, \ldots t_k \in T_\Sigma$. Let $r_1, \ldots, r_m$ be all the rules in R that have $q(\sigma(x_1 \ldots x_k))$ as a left-hand side and denote by s_i the ID obtained from s by application of rule r_i at the node π of s.

(a) if q is a universal state (in U) then n will have m sons $n_1, \ldots, n_m$ labeled respectively by the ID's $s_1, \ldots, s_m$.

(b) if q is an existential state (in Q-U) then n will have a single son n' labeled by one of the ID's s_i, $1 \leq i \leq m$.

An <u>accepting computation tree</u> of M on t is a computation tree of M on t whose leaves are all labeled by accepting ID's. M accepts the tree t if there exists an accepting computation tree of M on t. The tree language defined by M is $L(M) = \{t \in T_\Sigma \mid M \text{ accepts } t\}$. An atta is <u>universal</u> (utta) if U=Q and it is <u>nondeterministic</u> (ntta) if $U=\emptyset$. An atta is <u>deterministic</u> (dtta) if for all $k \geq 0$, $\sigma \in \Sigma_k$ and $q \in Q$, there is at most one rule with left-hand side $q(\sigma(x_1 \ldots x_k))$. Names for the various classes of automata defined above are obtained by capitalizing all the letters, e.g. the class of all universal top-down tree automata is denoted by UTTA. The names of the families of tree languages defined by these classes of automata are obtained by changing the last letter 'A' of the automata-class name to 'L'; for example, ATTL is the family of tree languages recognizable by automata in ATTA. We shall keep those notational conventions throughout the paper.

<u>Definition 3.2</u>. The class of tree languages NTTL will be denoted by RECOG. A tree language L in RECOG is said to be <u>recognizable</u>.

We now show that the features of universality and determinism are equivalent for ATTA's.

<u>Theorem 3.3</u>. DTTL = UTTL.

Proof. The proof is by a simple subset construction. $\boxtimes$

The recognizable tree language $\left\{ \begin{smallmatrix} & S & \\ a & & b \end{smallmatrix} \; , \; \begin{smallmatrix} & S & \\ b & & a \end{smallmatrix} \right\}$ is known not to be in DTTL. Thus for ATTL's we obtain the inclusion diagram of Figure 1. A broken line means an inclusion not yet proved to be either proper or equality. In the case of ATTL we shall prove equality in Section 5.

$$\begin{tikzcd}
\text{ATTL} \arrow[d, dash, dashed] \\
\text{NTTL} = \text{RECOG} \arrow[d, dash] \\
\text{UTTL} = \text{DTTL}
\end{tikzcd}$$

Figure 1.

Remark. Although it is not clear how to define alternating bottom-up automata we may define the universal class in a natural way. By an easy subset construction we can show that universal bottom-up tree automata are equivalent to the deterministic version and hence recognize exactly RECOG.

4. Alternating Two-way Finite Tree Automata.

Nondeterministic and universal two-way finite tree automata have been introduced and studied in [KS, Ka]. In this section we generalize these concepts by introducing alternation.

Definition 4.1. An alternating two-way finite tree automaton (2ata) is a construct $M = (Q,U,\Sigma,\delta,q_0,F)$ where

- Q is a finite nonempty set of states
- $U \subseteq Q$ is the set of universal states; states in $Q-U$ are called existential states
- Σ is a ranked input alphabet
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of accepting states
- δ is the transition function

  $\delta : Q \times \Sigma \to \mathcal{P}(Q \times D)$

  where $D = \{-1,0,1,2,\dots,m\}$ with m being the maximal i such that $\Sigma_i \neq \emptyset$.

An instantaneous description (ID) of M on a tree t in T_Σ is a triple of the form (q,n,t) where $q \in Q$, $t \in T_\Sigma$ and n is a node of t or ω. A universal (existential) ID is an ID (q,n,t) with $q \in U$ $(q \in Q-U)$. An accepting (rejecting) ID is one in which $q \in F$ $(q \in Q-F)$ and $n=\omega$. The initial ID is (q_0,r,t) with r being the root of t. We next define the computation relation between ID's. $(q,n,t) \vdash_{M} (p,n',t)$ if $(p,i) \in \delta(q,a)$ where a is a label of n and n' is given by the following self-explanatory code:

```
n' := if i = -1 then if n = root-of(t) then ω else father(n)
                 else if i = 0 then n else i-th-son-of(n).
```

The reflexive-transitive closure $\vdash_{M}$ is denoted by $\vdash^{*}_{M}$.

A computation tree of M on t is a nonempty (not necessarily finite) tree labeled by ID's of M (on t) and satisfying the following properties.

(i) the root of the tree is labeled by the initial ID of M on t.

(ii) if n is an internal node of the tree and label[n] (the ID labeling the node n) is an existential ID, then n has a single son n' and its label must satisfy label[n] $\vdash_{M}$ label[n'].

(iii) if n is an internal node of the tree, label[n] is a universal ID and $\{I \mid \text{label}[n] \vdash_{M} I\} = \{I_1,\dots,I_k\}$, then n has k sons $n_1,\dots,n_k$ such that for each $1 \le i \le k$, label[n] $\vdash_{M} I_i = \text{label}[n_i]$.

An accepting (rejecting) computation tree of M on t is a finite computation tree of M on t whose leaves are (not) all labeled by accepting ID's. The automaton M accepts the tree t if there exists an accepting computation tree of M on t. The tree language accepted by M is $L(M) = \{t \in T_{\Sigma} \mid M \text{ accepts } t\}$. A 2ata is universal two-way tree automaton (2uta) if U=Q and it is nondeterministic two-way tree automaton (2nta) if $U=\emptyset$; these two classes of automata were defined in [Ka] and [KS] respectively. Deterministic two-way tree automata (2dta) are obtained from 2ata's by requiring that the transition function is a partial function. Names for the various classes of automata defined above are obtained by capitalizing the letters and the names of the families of tree languages characterized by these varieties of tree automata are obtained by changing the last letter of automata-class name from 'A' to 'L', exactly as in Section 3.

Example 4.2. Let $\Sigma=\Sigma_0 \cup \Sigma_2$ where $\Sigma_0=\{a,b\}$ and $\Sigma_2=\{A\}$. Define a tree language $L = \{t \in T_{\Sigma} \mid \text{all leaves of } t \text{ are labeled by } a\}$.

(i) the 2uta $M = (\{q,d,p\},\{q,d,p\},\Sigma,\delta,q,\{p\})$ where $\delta(q,A)=\{(q,1),(q,2)\}$; $\delta(q,a)=\{(p,-1)\}$; $\delta(q,b)=\{(d,-1)\}$; $\delta(p,A)=\{(p,-1)\}$; $\delta(d,A)=\{(d,-1)\}$ accepts exactly L.

(ii) the 2nta $N = (\{q,d,p\},\emptyset,\Sigma,\delta,q,\{d\})$ accepts exactly $\bar{L}$, the complement of L. ☒

In [KS] it was shown that $\bar{L}$ is in 2NTL but not in 2DTL and that L is not in 2NTL. In [Ka] it is shown that $\bar{L}$ is not in 2UTL. Since by example 4.2 L is in

2UTL and $\bar{L}$ is in 2NTL, it follows that 2NTL and 2UTL are incomparable and so the inclusion diagrm of Figure 2 is correct.

The proofs of the following lemma, theorem and a corollary are left as exercises to the reader.

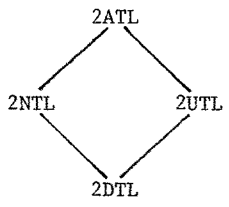

Figure 2.

Lemma 4.3. 2UTL $\cup$ 2NTL is properly included in 2ATL.

Theorem 4.4. ATTL $\subseteq$ 2ATL.

Corollary. UTTL $\subseteq$ 2UTL.

5. Alternating Two-way Synchronized Pushdown Tree Automata.

Two-way pushdown tree automata have the control structure of two-way finite tree automata; the operation of their storage, the pushdown, is synchronized with the movements of the automaton up and down the tree. We consider two different mechanisms for achieving synchronization. One, first studied in [ERS] in the context of tree transducers, see also [KS], will be the subject of this section; the other was introduced in [Ka] and it will be discussed in the next section. We study the effect of alternation and universality on these automata.

Definition 5.1. An alternating two-way synchronized pushdown tree automaton (2as-pta) is a contruct $M = (Q,U,\Sigma,\Gamma,\delta,q_0,z_0,F)$ where

- Q is a finite nonempty set of states
- $U \subseteq Q$ is the set of universal states; states in $Q-U$ are called existential states
- Σ is a ranked input alphabet
- Γ is the pushdown alphabet
- $q_0 \in Q$ is the initial state
- $z_0 \in \Gamma$ is the bottom (initial) pushdown symbol
- $F \subseteq Q$ is the set of accepting states
- δ is the transition function
  $\delta : Q\times\Sigma\times\Gamma \to P(Q\times D)$

where $D = \{-1\}\cup\{(0,\gamma) \mid \gamma\in\Gamma\}\cup\{(i,\gamma_1\gamma_2) \mid 1\le i\le m;\ \gamma_1,\gamma_2\in\Gamma\}$ with m being the maximal j with $\Sigma_j \neq \emptyset$. Intuitively, D specifies the direction of move and the pushdown instruction: -1 means 'move up on the tree and simultaneously pop the pushdown', $(0,\gamma)$ means 'stay at the same node on the tree and simultaneously replace the top of the stack by γ' and $(i, \gamma_1\gamma_2)$ means 'move down to the i-th son on the tree and simultaneously replace the top cell of the pushdown by two cells $\gamma_1\gamma_2$'; in the last case γ_1 is the new top of the pushdown. An <u>instantaneous description</u> (ID) of M on a tree t in T_Σ is a quadruple of the form (q,α,β,t) where $q\epsilon Q$, α is a simple path in t: $\alpha = (n_1,\ldots,n_k)$ with n_1 the root of t and n_k the node currently scanned, and $\beta = \gamma_1\ldots\gamma_k\epsilon\Gamma^*$ is the contents of the pushdown store $(\gamma_i\epsilon\Gamma)$. A <u>universal</u> (<u>existential</u>) ID is an ID (q,α,β,t) with $q\epsilon U$ $(q\epsilon Q-U)$; it is <u>accepting</u> (<u>rejecting</u>) if $\alpha = (\ )$, $\beta=\lambda$, and $q\epsilon F$ $(q\epsilon Q-F)$. Let $I = (q,(n_1,\ldots,n_k),\gamma\beta_1,t)$ and $J = (p,(n_1,\ldots,n_\ell),\beta_2,t)$ be two ID's with n_k labeled by σ and $\gamma\epsilon\Gamma$. Then $I \vdash_{\overline{M}} J$ if either of the following holds.

(1) $(p,(j,\gamma_1\gamma_2))\epsilon\delta(q,\sigma,\gamma)$; $\beta_2=\gamma_1\gamma_2\beta_1$, $\ell= k+1$ and n_ℓ is the j-th $(j\geq 1)$ son of n_k.

(2) $(p,(0,\gamma_1))\epsilon\delta(q,\sigma,\gamma)$; $\beta_2=\gamma_1\beta_1$, $\ell=k$ and $n_\ell=n_k$.

(3) $(p,-1)\epsilon\delta(q,\sigma,\gamma)$; $\beta_2=\beta_1$, $\ell=k-1$ and $n_\ell=n_{k-1}$ if $k>1$; if $k=1$ then $I = (q,(n_1),\gamma,t) \vdash_{\overline{M}} J = (p,(\ ),\lambda,t)$.

The relfexive-transitive closure of $\vdash_{\overline{M}}$ is denoted by $\overset{*}{\vdash}_{\overline{M}}$. The concepts of computation tree, accepting (rejecting) computation tree (of M on t), acceptance and the tree language recognized by a 2as-pta, are defined exactly as in the section on alternating two-way finite tree automata, except that configurations now are different. Also the four varieties of automata, their names and the classes of language they characterize are defined in an analogous fashion. For example, 2US-PTL is the family of all tree languages recognizable by 2us-pta's, universal two-way synchronized pushdown tree automata.

From theorem 4.4 it follows that NTTL = RECOG $\subseteq$ ATTL $\subseteq$ 2ATL. It is also known that NTTL=RECOG=2NS-PTL and it is obvious that 2ATL $\subseteq$ 2AS-PTL. We will now show that 2NS-PTL=2AS-PTL, implying that all these classes are equal (to RECOG). Our proof will be based on a simulation of a 2as-pta by a 2ns-pta. We will need some notation. Let $M=(Q,U,\Sigma,\Gamma,\delta,q_0,Z_0,f)$ be a 2as-pta. Recall that $Q\times D$ is the set of all instructions of M and let $D'=D-\{-1\}$; then $P(Q\times D')$ is the set of all sets of instructions, excluding "move-up" instructions. We define a new alphabet: $\Delta = \{<\phi,s,z>|\phi\epsilon P(Q\times D'),\ s\subseteq Q,\ z\epsilon\Gamma\}$ and a 2ns-pta $N=(P(Q),\emptyset,\Sigma,\Delta,\hat{\delta},\{q_0\},<\emptyset,\emptyset,Z_0>,P(F)-\{\emptyset\})$ where $\hat{\delta}$ will be defined by means of a program written in a pidgin Algol (together with some English). Symbols in the pushdown store are of the form $<\phi,s,z>$ where $\phi\epsilon P(Q\times D')$, $s\subseteq Q$ and $z\epsilon\Gamma$; we will have four variables Φ, S, Z and σ ranging respectively over $P(Q\times D')$, $P(Q)$, Γ and Σ, and they will always refer to the respective values of the three components of the topmost pushdown symbol and the label of the currently scanned node of the input tree. Suppose $\delta(q,\sigma,A) =$

$\{(p_1,(i_1,\alpha_1)),\ldots,(p_k,(i_k,\alpha_k)),(q_1,-1),\ldots,(q_\ell,-1)\}$ where $i_j \geq 0$ $(1 \leq j \leq k)$. We will use two functions $\mathrm{DOWN}(\delta(q,\sigma,A)) = \{(p_1,(i_1,\alpha_1)),\ldots,(p_k,(i_k,\alpha_k))\}$ and $\mathrm{UP}(\delta(q,\sigma,A)) = \{q_1,\ldots,q_\ell\}$. The function CHOOSE(B) nondeterministically chooses an element of the set B. The pushdown store is initialized to $<\emptyset,\emptyset,Z_0>$. N simulates M by storing in the pushdown all the moves that still have to be taken. At any node of the input tree, N first tries all the computations down the tree (using the first component of the current pushdown symbol) and only when those are exhausted, N moves up in state that is the second component of the pushdown symbol. The procedure UPDATE updates these two components taking into account the universal or existential nature of the states. We leave it to the reader to convince himself that this program correctly simulates M and can be realized by a transition function $\hat{\delta}$ of a 2ns-pta.

```
procedure UPDATE(q)
     if $q \in U$ then $\Phi \leftarrow \Phi \cup \mathrm{DOWN}(\delta(q,\sigma,Z))$;
                  $S \leftarrow S \cup \mathrm{UP}(\delta(q,\sigma,Z))$
             else // q existential //
                  $m \leftarrow \mathrm{CHOOSE}(\delta(q,\sigma,Z))$;
                  if $m \in \mathrm{UP}(\delta(q,\sigma,z))$ then $S \leftarrow S \cup \{m\}$
                       else $\Phi \leftarrow \Phi \cup \{m\}$
                  fi
     fi
end UPDATE:

// main program //
UPDATE($q_0$);
while STACK nonempty do
     case $\Phi$ of
          $=\emptyset$ $\rightarrow$ pop pushdown and simultaneously move up in state S;
               for $q \in S$ do UPDATE(q) od:
          $\neq\emptyset$ $\rightarrow$ $m=(p,(i,\alpha)) \leftarrow \mathrm{CHOOSE}(\Phi)$;
               $\Phi \leftarrow \Phi - \{m\}$;
               if i=0 then $Z \leftarrow \alpha$; UPDATE(p)
                    else let $\alpha = \gamma_1\gamma_2$; $Z \leftarrow \gamma_2$;
                         move down to the i-th son in state {p} and simultaneously
                         push $<\emptyset,\emptyset,\gamma_1>$;
                         UPDATE(p)
               fi
     esac
od
```

We have proved

Theorem 5.2. 2NS-PTL = 2AS-PTL.

Corollary 5.3. 2ATL = NTTL = ATTL = 2NS-PTL = 2DS-PTL = 2US-PTL = 2AS-PTL = RECOG.
Proof. Only 2DS-PTL = 2NS-PTL was not mentioned before. It is proved in [KS].

⊠

6. Alternating Two-way Backtracking Pushdown Tree Automata.

Deterministic two-way backtracking pushdown automata (2db-pta) were introduced in [Ka]. These automata are similar to 2as-pta's with the following important difference. The pushdown has two tracks; a standard one for storing information for future reference, and an extra track for storing pointers to the nodes of the input tree. The machine can push the pushdown store while moving up or down the tree; at that time the (pointer to the) node is pushed along with a symbol of the pushdown alphabet. When popping, the automaton "backtracks", that is, continues it's operation at the node that appears at the top of the pushdown immediately after the pop.

In this section 2db-pta's are extended by adding the features of alternation, nondeterminism and universality.

Definition 6.1. An alternating two-way backtracking pushdown tree automaton (2ab-pta) is a construct $M = (Q,U,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ where

- Q is a finite, nonempty set of states
- $U \subseteq Q$ is the set of universal states; states in $Q-U$ are called existential
- Σ is a ranked input alphabet
- Γ is the pushdown alphabet
- $q_0 \in Q$ is the initial state
- $Z_0 \in \Gamma$ is the bottom (initial) pushdown symbol
- $F \subseteq Q$ is the set of accepting states
- δ is the transition function

$$\delta : Q\times\Sigma\times\Gamma \to P(Q\times D)$$

where (the instruction set) $D = \{pop\} \cup \{(i,\alpha) \mid -1 \le i \le m$, and $|\alpha|=1$ if $i=0$ and $|\alpha|=2$ otherwise$\}$. m is the maximal j such that $\Sigma_j \ne \emptyset$.

An instantaneous description (ID) of M on a tree t in T_Σ is a quadruple (q,α,β,t) where $q \in Q$, α is a (not necessarily simple) path in t: $\alpha=(n_1,\ldots,n_k)$ with

n_1 the root of t and n_k the currently scanned node, and $\beta = \gamma_1 \dots \gamma_k \in \Gamma^*$ is the contents of the pushdown store ($\gamma_i \in \Gamma$). A universal, existential, accepting and rejecting ID's are defined exactly as for 2as-pta's in the previous section. For two ID's $I = (q,(n_1,\dots,n_k),\gamma\beta_1,t)$ and $J = (p,(n_1,\dots,n_\ell),\beta_2,t)$ with n_k labeled σ and $\gamma \in \Gamma$ we define $I \vdash_{M} J$ if either of the following holds.

(1) $(p,(j,\gamma_1\gamma_2)) \in \delta(q,\sigma,\gamma)$; $\beta_2 = \gamma_1\gamma_2\beta_1$, $\ell = k+1$ and n_ℓ is the j-th son of n_k if $j \geq 1$ or the father of n_k if $j = -1$.

(2) $(p,(0,\gamma_1)) \in \delta(q,\sigma,\gamma)$; $\beta_2 = \gamma_1\beta_1$, $\ell = k$ and $n_\ell = n_k$.

(3) $(p,\text{pop}) \in \delta(q,\sigma,\gamma)$; $\beta_2 = \beta_1$, $\ell = k-1$ and $n_\ell = n_{k-1}$ if $k>1$; if $k=1$ then $I = (q,(n_1),\gamma,t) \vdash J = (p,(\,),\lambda,t)$.

The reflexive-transitive closure of $\vdash_{M}$ is denoted by $\vdash^{*}_{M}$. Computation trees, accepting (rejecting) computation trees and acceptance are defined as in Section 5 (i.e., as in Section 4). The nondeterministic (universal, deterministic) variant of 2ab-pta's are denoted by 2ub-pta (2ub-pta, 2db-pta) and the corresponding families of automata and languages are denoted using the same conventions as we have used so far. For example, 2UB-PTL is the class of all tree languages recognizable by 2ub-pta's. In [Ka] it is shown that 2DB-PTL = RECOG. In this section we show that all four classes of two-way backtracking pushdown tree automata are equipotent, i.e. they all recognize RECOG. Instead of having one very involved construction we break our argument into three parts. The proof of part three, Theorem 6.4., is an adaptation of a proof in [Ka] where it is shown that any one-state 2db-pta can be simulated by a 2uta.

Theorem 6.2. For any 2ab-pta there is an equivalent 2nb-pta.

Lemma 6.3. For any 2nb-pta there is an equivalent one-state 2nb-pta.

Theorem 6.4. For any one-state 2nb-pta there is an equivalent 2ata.

We omit the proof of Theorem 6.2 which is similar to the proof of Theorem 5.2 (with some small differences). The intuitive idea behind the proof of Lemma 6.3 is that N keeps the finite control information of M on its pushdown store, including a guess as to the state after the current pushdown square is popped; here we will skip the proof.

Proof of Theorem 6.4. (Outline) Let $M = (\{s\},\emptyset,\Sigma,\Gamma,\delta,s,Z_0,\{s\})$ be the given one-state 2nb-pta. We can assume without losing generality that M has no stay-instructions. For each $\delta\epsilon\Sigma$ and $Z\epsilon\Gamma$ we construct a finite visit-tree that representes all the possible subsequent visits to a node labeled σ (in the input tree) and to a pushdown square from the time it is pushed (and then it carries Z) until it is popped.

(i) The root of the visit-tree is labeled by Z.

(ii) Let a node of the visit-tree be labeled by Y (in Γ).

(a) if $(s,(i,\gamma_1\gamma_2))\epsilon\delta(s,\sigma,Y)$ then create a son of Y and label it by γ_2.

(b) if $(s,\text{pop})\epsilon\delta(s,\sigma,Y)$ then create a son of Y and label it 'pop'.

(c) for each son of Y that is not labeled by 'pop', check if its label occurs on the path from the root of the visit-tree to the node labeled by Y. If it does, relabel that node by 'loop'.

Clearly, the visit-tree corresponding to σ and Z is a finite tree whose leaves are labeled by 'pop', 'loop' or $A\epsilon\Gamma$ for which $\delta(s,\sigma,A)=\emptyset$.

Let $\Pi_{\sigma,Z}$ be the set of all root-to-pop-leaf paths through the visit tree corresponding to σ and Z. For $\pi\epsilon\Pi_{\sigma,Z}$, $\ell(\pi)$ is the sequence of labels of the nodes along π; thus $\ell(\pi)$ is of the form $Z\alpha(\text{pop})$ where $\alpha\epsilon\Gamma^*$. Now let $\ell(\pi)=ZZ_1\ldots Z_k(\text{pop})$ and let the sequence of moves "defining" π in the visit-tree be:

$(s,(i_1,Z_1'Z_1))\epsilon\delta(s,\sigma,Z)$
$(s,(i_2,Z_2'Z_2))\epsilon\delta(s,\sigma,Z_1)$
.
.
.
$(s,(i_k,Z_k'Z_k))\epsilon\delta(s,\sigma,Z_{k-1})$
$(s,\text{pop})\epsilon\delta(s,\sigma,Z_k)$

Think of (i_j,Z_j') as an instruction for the 2ata saying: go to i_j-th son if $i_j\geq 1$ and father if $i_j=-1$ in state Z_j'. Then π induces a set of instructions

$$\text{instr}(\pi)=\{(i_1,Z_1'),\ldots,(i_k,Z_j')\}$$

that should lead to successful termination (i.e., they should be and-ed). Different paths of $\Pi_{\sigma,Z}$ induce different instruction sets which should be or-ed between themselves. All these instructions should be organized by introducing new existential and universal states (different for each and $Z\epsilon\Gamma$) according to Figure 3, where $\exists(\forall)$ in a node means a new existential (universal) state. We trust the reader to fill in the details. ⊠

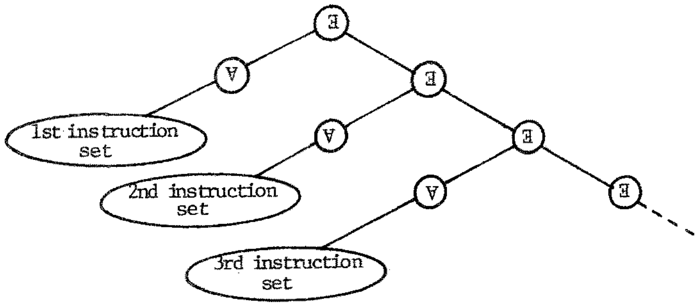

Figure 3.

Corollary 6.5. 2DB-PTL = 2UB-PTL = 2NB-PTL = 2AB-PTL = RECOG.

7. Summary.

In this paper we have studied the effect of alternation on serveral varieties of tree automata. We have seen that once the pushdown facility is available (either synchronized or backtracking) alternation provides no additional power. In the absence of pushdown, alternation (and universality) define new classes of tree languages. Altogether we have distinguished between five families of tree languages as shown in Figure 4. The diagram is correct (see also [Ka]) because the finite language {S(a b) , S(b a) } is in 2DTL and not in DTTL, while the language L of Example 4.2 is not in 2NTL but certainly in DTTL.

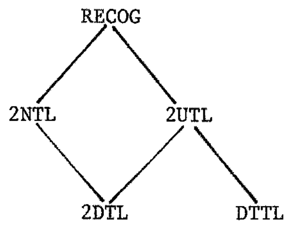

Figure 4.

Acknowledgement. I thank Tsutomu Kamimura for helpful discussions.

References

[B] Berman, L. Precise Bounds for Presburger Arithmetic and the Reals with Addition. 18th Symposium FOCS, 1977, pp. 95-99.

[CS] Chandra, A. K. and Stockmeyer, L. J. Alternation. 17th Symposium FOCS, 1976, pp. 98-108.

[CKS] Chandra, A. K., Kozen, D. C. and Stockmeyer, L. J. Alternation. JACM 28 (1981), pp. 114-133.

[D] Doner, J. Tree Acceptors and Some of Their Applications. JCSS 4 (1970), pp. 406-451.

[E] Engelfriet, J. Tree Automata and Tree Grammars. Lecture Notes DAIMI FN-10, University of Aarhus, Denmark, 1975.

[E1] Engelfriet, J. Bottom-up and Top-down Tree Transformations--A Comparison. Math. Systems Theory 9 (1975), pp. 193-231.

[ERS] Engelfriet, J. Rozenberg, G. and Slutzki, G. Tree Transducers, L-Systems, and Two-way Machines. JCSS 20 (1980), pp. 150-202.

[FL] Fischer, M. J. and Ladner, R. E. Propositional Modal Logic of Programs. 9th ACM STOC, 1977, pp. 286-294.

[K] Kozen, D. C. On Parallelism in Turing Machines. 17th Symposium FOCS, 1976, pp. 89-97.

[K1] Kozen, D. C. Complexity of Boolean Algebras. Theoretical Computer Science 70 (1980), pp. 221-247.

[Ka] Kamimura, T. Tree Automata and Attribute Grammars. TR-82-1, Dept. of Comp. Sci., Univ. of Kansas, Lawrence, Kansas, 66045, USA.

[KAI] Kasai, T., Adachi, A. and Iwata, S. Classes of Pebble Games and Complete Problems. SIAM Journal of Computing 8 (1979), pp. 574-586.

[KS] Kamimura, T. and Slutzki, G. Parallel and Two-way Automata on Directed Ordered Acyclic Graphs. Information and Control 49 (1981), pp. 10-51.

[LLS] Ladner, R. E., Lipton, R. J. and Stockmeyer, L. J. Alternating Pushdown Automata. 19th Symposium FOCS, 1978, pp. 92-106.

[MM] Magidor, M. and Moran, G. Finite Automata over Finite Trees. TR-69-30, Dept. of Math., Hebrew Univ., Jerusalem, Israel.

[R] Rabin, M. O. Matheamtical Theory of Automata. Proc. Symp. Appl. Math., Vol. 19, AMS, Providence, R.I., 1968, pp. 153-175.

[S] Slutzki, G. Alternating, Nondeterministic and Universal Pushdown Automata. TR-82-2, Dept. of Comp. Sci., Univ. of Kansas, Lawrence, Kansas, 66045, USA.

[SC] Stockmeyer, L. J. and Chandra, A. K. Provably Difficult Combinatorial Games. SIAM Journal of Computing 8 (1979), pp. 151-174.

[T] Thatcher, J. W. Tree Automata: informal survey. In "Currents in the Theory of Computing" (A. V. Aho, Ed.), Prentice-Hall, Englewood Cliffs, N. J., pp. 143-172.

[TW] Thatcher, J. W. and Wright, J. B. Generalized Finite Automata Theory with an Application to a Decision Problem of Second Order Logic. Math. Systems Theory 2 (1968), pp. 57-81.