# On The Power Of Distributed Bottom-up Tree Automata

**Kamala Krithivasan**[1] **and Ajeesh Ramanujan**[1]
[1]Department of Computer Science and Engineering
Indian Institute of Technology Madras, Chennai - 36
kamala@iitm.ac.in, ajeeshramanujan@yahoo.com

**Abstract**— *Tree automata have been defined to accept trees. Different types of acceptance like bottom-up, top-down, tree walking have been considered in the literature. In this paper, we consider bottom-up tree automata and discuss the sequential distributed version of this model. Generally, this type of distribution is called cooperative distributed automata or the blackboard model. We define the traditional five modes of cooperation, viz. $*$-mode, $t$-mode, $= k$, $\geq k$, $\leq k$ $(k \geq 1)$ modes on bottom-up tree automata. We discuss the accepting power of cooperative distributed tree automata under these modes of cooperation. We find that the $*$-mode does not increase the power, whereas the other modes increase the power. We discuss a few results comparing the acceptance power under different modes of cooperation.*

*Keywords:* Tree Automata, ranked alphabet, distributed nondeterministic tree automata, modes of cooperation

## 1. Introduction

Finite tree automata are generalizations of word automata. While a word automaton accepts a word, a tree automaton accepts a tree . The theory of tree automata arises as a straight forward extension of the theory of finite automata [6]. Tree automata were introduced in [4], [5] and [12] to solve certain decision problems in logic. Since then they were successfully applied to many other decision problems in logic and term rewriting, see e.g. [1]. Even though the two models are used in different settings they are closely related to each other since a finite automaton can be seen as a special case of a finite tree automaton. Trees appear in many areas of computer science and engineering and tree automata are used in applications such as XML manipulation, natural language processing, and formal verification and logic design.

According to the manner in which the automaton runs on the input tree, finite tree automata can be either bottom-up or top-down. A top-down tree automaton starts its computation at the root of the tree and then simultaneously works down the paths of the tree level by level. The tree automaton accepts the tree if such a run can be defined. A bottom-up tree automaton starts its computation in the leaves of the input tree and works its way up towards the root.

A finite tree automaton can be either deterministic or nondeterministic. This is an important issue since deterministic top-down automata are strictly less expressive than nondeterministic top-down automata. For the bottom-up case, deterministic bottom-up tree automata are just as powerful, from the point of view of language equivalence, as nondeterministic bottom-up tree automata. Non-deterministic top-down tree automata are equivalent to non-deterministic bottom-up tree automata [1].

In the last few years distributed and parallel computing has played an important role in Computer Science. Modelling these concepts using formal models has given rise to the concept of grammar systems and distributed automata. Grammar systems can be sequential or parallel. A co-operating distributed (CD) grammar system is sequential. Here, all grammars work on one sentential form. At any instant only one grammar is active. This is called a blackboard model. Suppose a problem is to be solved in a class. The teacher asks one student to start working on the problem on the blackboard. The student writes a few steps, then goes back. Another student comes and continues working on the problem. On his return, a third student comes and continues. The process continues till the problem is solved. Now, the question arises: at what time does one student return and the next one starts? There may be several ways for defining this. Correspondingly, in the CD grammar system, there are different modes of co-operation. The student may return when he is not able to proceed further (terminating mode); he may return at any time ($*$-mode); he may return after doing $k$-steps ($= k$-mode); he may return after doing $k$ or less steps ($\leq$-mode); he may return after doing $k$ or more steps ($\geq$-mode).

In this paper, we consider bottom-up tree automata and discuss the sequential distributed version of this model. We define the traditional five modes of cooperation, viz. $*$-mode, $t$-mode, $= k$, $\geq k$, $\leq k$ $(k \geq 1)$ modes on bottom-up tree automata. We discuss the accepting power of cooperative distributed tree automata under these modes of cooperation. We find that the $*$-mode does not increase the power, whereas the other modes increase the power. We discuss a few results comparing the acceptance power under different modes of cooperation.

In the next section we give basic definitions needed for the paper. Section 3 contains the definition of cooperative distributed tree automata and some results about their accepting power. The paper concludes with a note in section 4.

## 2. Basic Definitions

Let $N$ be the set of positive integers. Then the set of finite strings over $N$ is denoted by $N^*$. The empty string is denoted by $\epsilon$. A *ranked alphabet* $\Sigma$ is a finite set of symbols together with a function $Rank : \Sigma \to N$. For $f \in \Sigma$, the value $Rank(f)$ is called the rank of $f$. For any $n \geq 0$, we denote by $\Sigma_n$ the set of all symbols of rank $n$. Elements of rank $0, 1, \cdots, n$ are respectively called constants, unary, $\cdots$, $n$-ary symbols.

A *tree* $t$ over an alphabet $\Sigma$ is a partial mapping $t : N^* \to \Sigma$ that satisfies the following conditions:

- $dom(t)$ is a finite, prefix-closed subset of $N^*$, and
- for each $p \in dom(t)$, if $Rank(t(p)) = n > 0$, then $\{i | pi \in dom(t)\} = \{1, 2, \cdots, n\}$.

Each $p \in dom(t)$ is called a *node* of $t$. The node with domain element $\epsilon$ is the *root*. For a node $p$ , we define the $i^{th}$ *child* of $p$ to be the node $pi$, and we define the $i^{th}$ *subtree* of $p$ to be the tree $t^{'}$ such that $t^{'}(p^{'}) = t(pip^{'})$ for all $p^{'} \in dom(t^{'})$. A *leaf* of $t$ is a node $p$ which does not have any children, i.e. there is no $i \in N$ with $pi \in dom(t)$. We denote by $T(\Sigma)$ the set of all trees over the alphabet $\Sigma$. The *size* of a tree $t$ is the number of elements in $dom(t)$. The *height* of a tree $t$ is $\max\{|w| : w \in dom(t)\}$. Given a finite tree $t$, the *frontier* of $t$ is the set $\{p \in dom(t)|$ for all $n \in N$,$pn \notin dom(t)\}$. A tree with root $a$ and subtrees $t_1, t_2, \cdots, t_r$ is represented by $a(t_1, t_2, \cdots, t_r)$.

*Example 1:* Let $\Sigma = \{a, b, c, g, f\}$,$f \in \Sigma_2$,$g \in \Sigma_1$, $a, b \in \Sigma_0$. A tree over $\Sigma$ and its diagrammatic representation is shown in Figure 1

Let $t$ be the tree $f(g(a)f(bc))$.
$dom(t) = \{\epsilon, 1, 11, 2, 21, 22\}$.
$size(t) = 6$.
$height(t) = 2$.
$frontier(t) = \{11, 21, 22\}$.

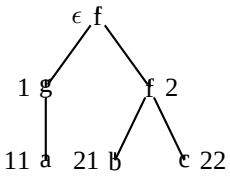


Fig. 1: A tree and its diagrammatic representation

A *nondeterministic finite tree automata* (NFTA) over an alphabet $\Sigma$ is a tuple $A = (Q, \Sigma, Q_f, \Delta)$ where,

- $Q$ is a finite set of states,
- $\Sigma$ is a ranked input alphabet,
- $Q_f \subseteq Q$ is a set of final states,
- $\Delta$ is a finite set of transition rules.

Each transition rule is a triple of the form $((q_1, q_2, \cdots, q_n), f, q)$ where $q_1, q_2, \cdots, q_n, q \in Q, f \in \Sigma_n$, i.e. $Rank(f) = n$. We use $f(q_1, q_2, \cdots, q_n) \to q$ to denote that $((q_1, q_2, \cdots, q_n), f, q) \in \Delta$. If $Rank(f) = 0$, i.e. $f$ is a constant, then we use rules of the form $f \to q$. The epsilon rules are denoted by rules of the form $q_i \to q_j$. A *run* of $A$ over a tree $t \in T(\Sigma)$ is a mapping $r : dom(t) \to Q$ such that for each node $p \in dom(t)$ where $q = r(p)$, we have that if $q_i = r(pi)$ for $1 \leq i \leq n$ then $\Delta$ has the rule $t(p)(q_1, q_2, \cdots, q_n) \to q$. A set $B = \{q_1, q_2, \cdots, q_n\} \subseteq Q, n \geq 1$ with respect to a tree $t^{'} \in T(\Sigma \cup Q)$ is said to be an active state set if every $q_i = r(pi), i \geq 0$ for some $p \in dom(t)$ and $t(p) \in \Sigma$.

An *instantaneous description(ID)* of a NFTA is a pair $(B, t)$, where $t \in T(\Sigma \cup Q)$ and $B$ is a set of active state set with respect to $t$.

For two ID's $(B, t), (B^{'}, t^{'})$ we write $(B, t) \vdash (B^{'}, t^{'})$ if there is a rule of the form $a(q_1, q_2, \cdots, q_n) \to q^{'} \in \Delta$ such that $t^{'}$ is obtained from $t$ by replacing a subtree of $t$ of the form $a(t_1, t_2, \cdots, t_n)$ by $q^{'}(t_1, t_2, \cdots, t_n)$, where $a \in \Sigma_n, n \geq 0, t_1, t_2, \cdots, t_n \in T(Q)$,$r(root(t_1)) = q_1$, $r(root(t_2)) = q_2, \cdots, r(root(t_n)) = q_n, q_1, q_2, \cdots, q_n \in B$ and $B^{'}$ is the set of active state set after performing the transition.

The initial ID is $(\phi, t), t \in T(\Sigma)$ and the final ID is $(\{q_f\}, t^{'})$ for some $q_f \in Q_f, t^{'} \in T(Q)$.The reflexive and transitive closure of $\vdash$ is denoted by $\vdash^*$.

A run represents the effect of a sequence of ID's from the initial ID to a final ID.

For a NFTA $A$, $L(A) = \{t \in T(\Sigma) | (\phi, t) \vdash^* (\{q_f\}, t^{'}), q_f \in Q_f, t^{'} \in T(Q)\}$.

A set $L$ of tree languages over $\Sigma$ is *recognizable* if $L = L(A)$ for some NFTA $A$. Two NFTA are said to be *equivalent* if they recognize the same tree language.

We give an example to show that certain tree languages are not recognizable.

*Example 2:* Let $\Sigma = \{f, g, a\}$, where $Rank(f) = 2$, $Rank(g) = 1, Rank(a) = 0$. Consider the tree language $L = \{f(g^i(a), g^i(a)) | i > 0\}$. Let us suppose that $L$ is recognizable by an automaton $A$ having $k$ states. Consider the tree $t = f(g^k(a), g^k(a))$. $t$ belongs to $L$, therefore there is a successful run of $A$ on $t$. As $k$ is the cardinality of the state set, there are two distinct positions along the first branch of the tree labeled with the same state. Therefore, one could cut the first branch between these two positions leading to a term $t' = f(g^j(a), g^k(a))$ with $j < k$ such that a successful run of $A$ can be defined on $t'$ . This leads to a contradiction with $L(A) = L$.

The proof can be generalized into a theorem, similar to pumping lemma for recognizable string languages, to recognizable tree languages [1].

## 3. Distributed Nondeterministic Tree Automata (DNTA)

In this section we define distributed nondeterministic tree automata(DNTA), the different modes of acceptance

of DNTA and discuss the power of different modes of acceptance.

*Definition 1:* A DNTA is a 4-tuple $D = (K, \Sigma, F, \Delta)$ where,

- $K$ is an $n$-tuple $(K_1, K_2, \cdots, K_n)$ where each $K_i$ is a set of states of the $i^{th}$ component;
- $\Sigma$ is a finite set of ranked alphabet;
- $F \subseteq \bigcup_i K_i$ is the set of final states;
- $\Delta$ is a $n$-tuple $(\delta_1, \delta_2, \cdots, \delta_n)$ of state transistion function where each $\delta_i$ is a set of transition rules of the $i^{th}$ component having the form $f(q_1, q_2, \cdots, q_n) \rightarrow q$, $f \in \Sigma_n, q_1, q_2, \cdots, q_n \in K_i, q \in \bigcup_i K_i$ or $q_i \rightarrow q_j$.

In the case of DNTA, we can consider many modes of acceptance depending upon the number of steps the system has to go through in each of the $n$ components. The different modes of acceptance are $*$-mode, $t$-mode, $\leq k$-mode, $\geq k$-mode, and $= k$-mode, where $k$ is a positive integer. Description of each of the above modes of acceptance is as follows:

*$t$-mode acceptance :*An automaton that has a leaf transition rule begins processing the input tree. Suppose that the system starts from the component $i$. The control stays in component $i$ as long as it can follow the transition rules in component $i$. Otherwise, it transfers the control to some other component $j$, $j \neq i$ which has the transition function to proceed. If more than one component succeeds, then the selection of $j$ is done nondeterministically. The process is repeated and we accept the tree if the system reaches any one of the final states. It does not matter which component the system is in while accepting.

*Definition 2:* The instantaneous description(ID) of a DNTA $D = (K, \Sigma, F, \Delta)$ working in $t$-mode is given by a triple $(B, t, i)$ where $B \subseteq \bigcup_i K_i$ and it denotes the current active state set of the whole system, $t \in T(\Sigma \cup \bigcup_i K_i)$ and $i, 1 \leq i \leq n$ the index of the component in which the system is currently in.

The transition between the ID's is defined as follows:

i) $(B, t, i) \vdash_t (B^{'}, t^{'}, i)$ if there is a rule of the form $a(q_1, q_2, \cdots, q_n) \rightarrow q^{'} \in \delta_i$ such that $t^{'}$ is obtained from $t$ by replacing a subtree of $t$ of the form $a(t_1, t_2, \cdots, t_n)$ by $q^{'}(t_1, t_2, \cdots, t_n)$, where $a \in \Sigma_n, n \geq 0, t_1, t_2, \cdots, t_n \in T(\bigcup_i K_i)$, $r(root(t_1)) = q_1, r(root(t_2)) = q_2, \cdots,$ $r(root(t_n)) = q_n$, $q_1, q_2, \cdots, q_n \in B$ and $B^{'}$ is the set of active state set after performing the transition.

ii) $(B, t, i) \vdash_t (B, t, j)$ iff component $i$ does not have a transition to proceed and component $j$ has a transition to proceed.

The reflexive and transitive closure of $\vdash_t$ is denoted by $\vdash_t^*$.

*Definition 3:* The language accepted by a DNTA $D = (K, \Sigma, F, \Delta)$ working in $t$-mode is defined as follows: $L_t(D) = \{t \in T(\Sigma) | (\phi, t, i) \vdash_t^* (\{q_f\}, t^{'}, j), t^{'} \in T(\bigcup_i K_i),$ for some $q_f \in F, 1 \leq i, j \leq n\}$.

We now give an example of a distributed bottom up tree automata working in $t$-mode.

*Example 3:* Consider the language $L_1 = \{a(bd(g^j d)^i(f), ce(h^k e)^l(f)), i, j, k, l \geq 1, |i - l| \leq 1\}$ over $\Sigma = \{a, b, c, d, e, f, g, h\}$, $a \in \Sigma_2$,$b, c, d, e, g, h \in \Sigma_1$,$f \in \Sigma_0$.

We define a distributed tree automaton $D_1 = (K, \Sigma, \{q_a\}, \Delta)$ working in $t$-mode as follows.

The components are defined as follows

- Component 1
  - $K_1 = \{q_f, q_g, q_1, q_2\}$,
  - $\delta_1 = \{d(q_f) \rightarrow q_g, g(q_g) \rightarrow q_1, g(q_1) \rightarrow q_1, q_2 \rightarrow q_f\}$
- Component 2
  - $K_2 = \{q_f, q_e, q_1, q_2\}$,
  - $\delta_2 = \{e(q_f) \rightarrow q_e, h(q_e) \rightarrow q_2, h(q_2) \rightarrow q_2, q_1 \rightarrow q_f\}$
- Component 3
  - $K_3 = \{q_f, q_a, q_b, q_c, q_d, q_e, q_1, q_2\}$,
  - $\delta_3 = \{f \rightarrow q_f, b(q_g) \rightarrow q_b, c(q_e) \rightarrow q_c, a(q_b, q_c) \rightarrow q_a\}$

The processing starts in component 3 , with the two leaves using the rule $f \rightarrow q_f$. As further processing is not possible in component 3, processing continues with 2 or 1. Then it alternates between 1 and 2 processing $d$'s, $g$'s, $e$'s and $f$'s. Finally when the labels are $b$ and $c$, processing takes the tree to $q_b$ and $q_c$ and in component 3 state $q_a$ is reached by the root.

*Theorem 1:* There exists a language accepted by a DNTA working in $t$-mode which is not recognizable.

*Proof:* Consider the tree language $L_1$. Let us suppose that $L_1$ is recognizable by an automaton $A$ having $k$ states. Consider the tree $t = a(bd(gd)^k(f), ce(he)^k(f)), k > 0$. $t$ belongs to $L_1$, therefore there is a successful run of $A$ on $t$. As $k$ is the cardinality of the state set, there are two distinct positions along the first branch of the tree labeled with the same state. Therefore, one could cut the first branch between these two positions leading to a term $t' = a(bd(gd)^j(f), ce(he)^k(f))$ with $j < k$ such that a successful run of $A$ can be defined on $t'$ . This leads to a contradiction with $L(A) = L_1$. So $L_1$ is not recognizable. ■

*$*$-mode acceptance :* An automaton that has a leaf transition rule begins processing the input tree. Suppose that the system starts from the component $i$. Unlike the termination mode, the automaton can transfer the control to any of the components at any time i.e., if there is some $j, j \neq i$ such that the next move is possible then the system can transfer the control to the component $j$. The selection of $j$ is done nondeterministically if there is more than one $j$.

The ID and the language accepted by the system in $*$ mode, $L_*(D)$ is defined as follows.

*Definition 4:* The instantaneous description(ID) of a DNTA $D = (K, \Sigma, F, \Delta)$ working in $*$-mode is given by a triple $(B, t, i)$ where $B \subseteq \bigcup_i K_i$ and it denotes the current active state set of the whole system, $t \in T(\Sigma \cup \bigcup_i K_i)$ and $i, 1 \leq i \leq n$ the index of the component in which the system is currently in.

The transition between the ID's is defined as follows:

i) $(B, t, i) \vdash_* (B^{'}, t^{'}, i)$ if there is a rule of the form $a(q_1, q_2, \cdots, q_n) \rightarrow q^{'} \in \delta_i$ such that $t^{'}$ is obtained from $t$ by replacing a subtree of $t$ of the form $a(t_1, t_2, \cdots, t_n)$ by $q^{'}(t_1, t_2, \cdots, t_n)$, where $a \in \Sigma_n, n \geq 0, t_1, t_2, \cdots, t_n \in T(\bigcup_i K_i)$, $r(root(t_1)) = q_1, r(root(t_2)) = q_2, \cdots,$ $r(root(t_n)) = q_n, q_1, q_2, \cdots, q_n \in B$ and $B^{'}$ is the set of active state set after performing the transition.

ii) $(B, t, i) \vdash_* (B, t, j)$ iff component $j$ has a transition to proceed.

The reflexive and transitive closure of $\vdash_*$ is denoted by $\vdash_*^*$.

*Definition 5:* The language accepted by a DNTA $D = (K, \Sigma, F, \Delta)$ working in $*$-mode is defined as follows: $L_*(D) = \{t \in T(\Sigma) | (\phi, t, i) \vdash_*^* (\{q_f\}, t^{'}, j), t^{'} \in T(\bigcup_i K_i)$, for some $q_f \in F, 1 \leq i, j \leq n\}$

We give an example of a distributed bottom up tree automata working in $*$-mode.

*Example 4:* Consider the language $L_2 = \{a(b^i(d), c^j(d)), i, j \geq 1\}$ over $\Sigma = \{a, b, c, d\}$, $a \in \Sigma_2, b, c \in \Sigma_1, d \in \Sigma_0$. We define a distributed tree automaton $D_2 = (K, \Sigma, \{q_f\}, \Delta)$ as follows.

The components are defined as follows

- Component 1
  - $K_1 = \{q_b, q_d\}$
  - $\delta_1 = \{b(q_d) \rightarrow q_b, b(q_b) \rightarrow q_b\}$
- Component 2
  - $K_2 = \{q_d, q_c\}$
  - $\delta_2 = \{c(q_d) \rightarrow q_c, c(q_c) \rightarrow q_c\}$
- Component 3
  - $K_3 = \{q_f, q_b, q_c, q_d\}$
  - $\delta_3 = \{d \rightarrow q_d, a(q_b, q_c) \rightarrow q_f\}$

Processing starts in component 3, with the two leaves using the rule $d \rightarrow q_d$. As further processing is not possible in component 3, processing continues with components 1 or 2. Then it alternates between components 1 and 2 processing $b$'s and $c$'s. When all the $b$'s and $c$'s are exhausted the automaton moves to component 3 and reaches the final state by using rule $a(q_b, q_c) \rightarrow q_f$. The processing of any tree in $L_2$ uses component 3 two times, in the first and the last step.

*Theorem 2:* For any DNTA $D$ working in $*$-mode, $L_*(D)$ is recognizable.

*Proof:* Let $D = (K, \Sigma, F, \Delta)$ be a DNTA working in $*$-mode where, $\Delta = (\delta_1, \delta_2, \cdots, \delta_n)$ and the components have states $K_1, K_2, \cdots, K_n$. Define a NFTA $N = (K^{'}, \Sigma, F^{'}, \delta)$ where,

$$K^{'} = \{[q, i] | q \in \bigcup_i K_i, 1 \leq i \leq n\}$$

$$F^{'} = \{[q_f, i] | q_f \in F, 1 \leq i \leq n\}$$

$\delta$ contains the following transitions
for each $a(q_1, q_2, \cdots, q_r) \rightarrow q \in \delta_i, r \geq 0, q_1, q_2, \cdots,$
$q_r \in K_i, 1 \leq i \leq n, a \in \Sigma,$
$\{a([q_1, i_1], [q_2, i_2], \cdots, [q_r, i_r]) \rightarrow [q, j]\} \in \delta,$
$1 \leq j \leq n, q \in K_j, 1 \leq i_1, i_2, \cdots, i_r \leq n.$
If $q_s \rightarrow q_t$ is a rule in the $i^{th}$ component and $q_t \in K_i$, then add $[q_s, i] \rightarrow [q_t, j], 1 \leq j \leq n$ to $\delta$.
If a tree $t$ is accepted by a DNTA, then there is a sequence of ID's $(\phi, t) \vdash (B_1, t_1) \vdash \cdots \vdash (\{q_f\}, t_r)$ leading to acceptance. The corresponding sequence of ID's for the NFTA is as follows: $(\phi, t, i_0) \vdash (B_1, t_1, i_1) \vdash \cdots \vdash (\{q_f\}, t_r, i_r)$, $1 \leq i_j \leq n$. Similarly, if there is a sequence of ID's leading to acceptance in NFTA, then there is a corresponding sequence of ID's leading to acceptance in the DNTA. This construction of NFTA shows that $L_*(D) = L(N)$ and so $L_*(D)$ is recognizable.

■

$= k$-mode ($\leq$ $k$-mode,$\geq$ $k$-mode )*acceptance :* An automaton that has a leaf transition rule begins processing the input tree. Suppose that the system starts from the component $i$. The automaton transfers the control to another component $j, j \neq i$ only after the completion of exactly $k(k^{'}(k^{'} \leq k), (k^{'} \geq k))$ number of steps in the component $i$. The selection of $j$ is done nondeterministically if there is more than one $j$.

*Definition 6:* The instantaneous description(ID) of a DNTA $D = (K, \Sigma, F, \Delta)$ working in $= k$-mode, $\leq k$-mode, $\geq k$-mode is given by a 4-tuple $(B, t, i, j)$ where $B \subseteq \bigcup_i K_i$ and it denotes the current active state set of the whole system, $t \in T(\Sigma \cup \bigcup_i K_i)$, $i$ the index of the component in which the system is currently in, $1 \leq i \leq n$, $j \geq 0$ denotes the number of steps for which the system has been in the $i^{th}$ component.

The system accepts the tree only if the DNTA is in the final state in some component $i$ after processing the tree and provided it has completed $k$-steps in the component $i$ in the case of $= k$-mode of acceptance (it has completed some $k^{'}(k^{'} \leq k)$ steps in the component $i$ in the case of $\leq k$-mode acceptance or it has completed some $k^{'}(k^{'} \geq k)$ steps in the component $i$ in the case of $\geq k$-mode of acceptance. The language accepted by the respective modes are denoted as $L_{=k}, L_{\leq k}, L_{\geq k}$.

We give an example of a distributed bottom-up tree automata working in $= 2$-mode.

*Example 5:* Consider the language $L_4 = \{b(a(b^{2i}(d), c^{2j}(d)), i, j \geq 1, i = j$ or $i = j + 1$ or $j = i + 1\}$ over $\Sigma = \{a, b, c, d\}, a \in \Sigma_2, b, c \in \Sigma_1, d \in \Sigma_0$.

We define a distributed tree automaton
$D_4 = (K, \Sigma, \{q_f\}, \Delta)$ working in $=$ 2-mode as follows.

The components are defined as follows

- Component 1
  - $K_1 = \{q_b, q_d\}$,
  - $\delta_1 = \{b(q_d) \rightarrow q_b, b(q_b) \rightarrow q_b\}$
- Component 2
  - $K_2 = \{q_d, q_c\}$,
  - $\delta_2 = \{c(q_d) \rightarrow q_c, c(q_c) \rightarrow q_c\}$
- Component 3
  - $K_3 = \{q_f, q_a, q_b, q_c, q_d\}$,
  - $\delta_3 = \{d \rightarrow q_d, a(q_b, q_c) \rightarrow q_a, b(q_a) \rightarrow q_f\}$

Component 3 starts the processing, active for the first two steps, then the system switches between component 1 and 2 and ends the processing with component 3 for the last 2 steps. Using the technique used in example 2 we can show that $L_4$ is not recognizable.

Similarly we can find languages for $= k$-mode for $k \geq 3$.

*Theorem 3:* There exists a language accepted by a DNTA working in $= k$-mode, $k \geq 1$ which is not recognizable.

*Proof:* For $k = 2$, example 5 prove the result. For $k > 2$ consider the language
$L_5 = \{a_{k-1}a_{k-2}\cdots a_1a_0(b^{ki}(e^{k-2}(d)), c^{kj}(g)), i, j \geq 1,$
$k > 2, i = j$ or $i = j + 1$ or $j = i + 1\}$ over
$\Sigma = \{b, c, d, e, g, a_0, a_1, a_1, a_2, \cdots, a_{k-1}\}, a_0 \in \Sigma_2, b, c, e,$
$a_1, \cdots, a_{k-1} \in \Sigma_1, d, g \in \Sigma_0$.
Constructing a DNTA for $L_5$ is similar to the construction in example 5. It is not difficult to see that $L_5$ can be accepted by a DNTA working in $=$ $k$-mode with 3 components. Using the technique used in example 2 we can show that $L_5$ is not recognizable. ■

*Theorem 4:* There exists a language accepted by a DNTA working in $\geq k$-mode, $k \geq 1$ which is not recognizable.

*Proof:* Consider the language
$L_6 = \{f^n a_{k-1}a_{k-2}\cdots a_1a_0(b^{ki}(e^{k-2}(d)), c^{kj}(g)), i, j,$
$n \geq 1, k > 2, i = j$ or $i = j + 1$ or $j = i + 1\}$ over
$\Sigma = \{b, c, d, e, f, g, a_0, a_1, a_2, \cdots, a_{k-1}\}, a_0 \in \Sigma_2, b, c, e,$
$f, a_1, \cdots, a_{k-1} \in \Sigma_1, d, g \in \Sigma_0$.

Constructing a DNTA for $L_6$ is similar to the construction in example 5. It is not difficult to see that $L_6$ can be accepted by a DNTA working in $\geq k$-mode with 3 components. Using the technique used in example 2 we can show that $L_6$ is not recognizable. For $k = 2$, example similar to 5 can be provided. ■

*Theorem 5:* There exists a language accepted by a DNTA working in $\leq k$-mode, which is not recognizable.

*Proof:* Consider the language
$L_7 = \{g(a^m(e), b^n(e)), m \geq 3, \frac{m+5}{8} \leq n \leq \frac{m+3}{2}\}$ over
$\Sigma = \{g, a, b, e\}, g \in \Sigma_2, a, b \in \Sigma_1, e \in \Sigma_0$.
We define a distributed tree automaton
$D_7 = (K, \Sigma, \{q_f\}, \Delta)$ working in $\leq$ 2-mode as follows.

The components are defined as follows

- Component 1
  - $K_1 = \{q_{11}, q_{12}, q_{21}, q_{22}, q_{23}\}$,
  - $\delta_1 = \{a(q_{11}) \rightarrow q_{12}, a(q_{12}) \rightarrow q_{12}, e \rightarrow q_{11},$
    $g(q_{12}, q_{21}) \rightarrow q_f, g(q_{12}, q_{22}) \rightarrow q_f,$
    $g(q_{12}, q_{23}) \rightarrow q_f\}$
- Component 2
  - $K_2 = \{q_{11}, q_{21}, q_{22}, q_{23}, \}$,
  - $\delta_2 = \{e \rightarrow q_{11}, b(q_{11}) \rightarrow q_{21}, q_{21} \rightarrow q_{22},$
    $q_{22} \rightarrow q_{23}, q_{23} \rightarrow q_{11}\}$

Using the technique used in example 2 we can show that $L_7$ is not recognizable. ■

*Theorem 6:* For any recognizable language $L$, there is a DNTA $D$ working in $=$ 1-mode with two components.

*Proof:* Let $A = (Q, \Sigma, Q_f, \Delta)$ be a NFTA recognizing $L$. We construct a distributed tree automaton $D = (K, \Sigma, Q_f, \Delta^{'})$ working in $=$ 1-mode as follows.

The components are defined as follows

- Component 1
  - $K_1 = Q$,
  - $\delta_1 = \Delta$
- Component 2
  - $K_2 = Q$,
  - $\delta_2 = \Delta$

The construction shows that any recognizable language can recognized by by a DNTA working in $=$ 1-mode with two components. ■

*Theorem 7:* For any recognizable language $L$, there is a DNTA $D$ working in $t$-mode with two components.

*Proof:* Let $A = (Q, \Sigma, Q_f, \Delta)$ be a NFTA recognizing $L$. We construct a distributed tree automaton $D = (K, \Sigma, Q_f, \Delta^{'})$ working in $t$-mode as follows.
The components are defined as follows

- Component 1
  - $K_1 = Q \cup \{q^{'} | q \in Q\}$
  - $\delta_1$ contains the following transitions
    for each $a(q_1, q_2, \cdots, q_n) \rightarrow q \in \Delta, n \geq 0,$
    $q_1, q_2, \cdots, q_n \in K_1, a \in \Sigma \cup \{\epsilon\}$
    $a(q_1, q_2, \cdots, q_n) \rightarrow q^{'} \in \delta_1, q^{'} \in K_1$.
- Component 2
  - $K_2 = Q \cup \{q^{'} | q \in Q\}$
  - $\delta_2$ contains the following transitions
    $\forall q^{'} \in K_2, q^{'} \rightarrow q \in \delta_2, q \in K_2$.

The construction shows that any recognizable language can recognized by by a DNTA working in $t$-mode with two components. ■

*Theorem 8:* For any DNTA working in $*$-mode, there is a DNTA working in $=$ 1-mode with two components.

*Proof:* From theorem 2 we know that any DNTA working in $*$-mode is recognizable. The theorem follows from the result of theorem 6. ■

We conjecture the following.

*Conjecture 1:* Any DNTA D working in $= k$ mode with $2$ components is recognizable.

*Conjecture 2:* For any DNTA working in $= k$-mode, there is a DNTA working in $= 1$-mode.

## 4. Conclusion

In this paper we have defined cooperative distributed tree automata and the languages accepted under $*, t, = k, \leq k, \geq k$ (where $k$ is an integer $\geq 1$) modes. We showed that the power of tree automata is not increased by the $*$ mode of cooperation, whereas under the other modes, the power is increased. We have proved some results comparing their acceptance power. Other comparisons and decidability issues are being pursued. We are also looking into other application areas like representation of XML schemas and in syntactic pattern recognition.

The application of variable arity trees in representing XML schemas is considered in Murata [9]. The inference of such tree grammars is considered in [10]. Whether distributed tree automata (may be for variable arity trees) will be a better model for representing of XML schemas in an application which can be explored. Distributed version of automata for variable arity trees and other models of tree automata like top-down acceptance and tree walking automata may be more helpful in the above process.

## References


[1] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Draft book; available electronically on http://www.grappa.univ-lille3.fr/tata, 2008.

[2] E. Csuha j-Varju, J.Dassow, J.Kleeman and Gh. Paun. Grammar Systems: A Grammatical Approach to Distribution and cooperation. Gordon and Breach, London, 1994.

[3] J. Dassow, G. Paun and G. Rozenberg, Grammar Systems chapter in Handbook of Formal Languages Vol2. edited by G. Rozenberg and A. Salomaa., Springer, 1997.

[4] J. E. Doner. Decidability of the week-second order theory of two successors. *Notices of the American Mathematical Society*, 12:365-468, 1965.

[5] J. E. Doner. Tree acceptors and some of their applications. *J. Comput. Syst. Sci.*, 4(5):406-451,1970.

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

[7] K. Krithivasan, M.Sakthi Balan and P.Harsha. Distributed Processing in Automata, *International Journal of Foundations of Computer Science*, 10(4): 443-464, 1999.

[8] K. Krithivasan and R. Rama. Introduction to Formal Languages, Automata Theory and Computation. Pearson, 2009.

[9] M. Murata, D. Lee, M. Mani and K. Kawaguchi. Taxonomy of XML Schema Languages using Formal Language Theory, *ACM Trans. Inter. Tech.*, 5(4):660-704, 2005.

[10] Neetha Sebastian and K. Krithivasan. Learning Algorithms for Grammars of Variable Arity Trees, *International Conference of Machine Learning and Applications*, 98-103, 2007.

[11] G. Paun. Grammar Systems: A Grammatical Approach to Distribution and Computation, *Lecture Notes in Computer Science*, 944:429-443, 1995.

[12] J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic, *Mathematical Systems Theory*, 2:57-82, 1968.