

C-TREE SYSTOLIC AUTOMATA *

E. FACHINI and M. NAPOLI

Dipartimento Informatica e Applicazioni, Università di Salerno, 84100 Salerno, Italy

Communicated by A. Salomaa

Received September 1986

Revised June 1987

Abstract. A new type of systolic automaton is introduced, its structural properties, such as homogeneity and stability, are investigated and the class of languages accepted by these automata is studied. This class of languages, in the nondeterministic case, contains a large subclass of the Lindenmayer EPTOL languages. A characterization of the defined model is also given in terms of sequential machines.

Introduction

Recently, different types of systolic automata have been introduced as models of highly parallel synchronous computational devices (see [2-7]). They essentially differ in their communication structures: arrays, trees and trellises have been considered and the classes of languages accepted by the resulting systolic automata have been studied. In particular, the class of languages accepted by systolic binary tree acceptors has nice properties from the point of view of formal language theory: it is a boolean algebra and the emptiness, and hence the equivalence problem, is decidable. It is a subfamily of the family of EOL languages, (see [3]) and it contains languages which stand quite high up in the standard language-theoretic hierarchies even if it does not contain some quite simple languages such as $\{a^n b^n \mid n \geq 1\}$. Moreover, the class of languages accepted by nondeterministic systolic binary tree automata coincides with the class of languages accepted by deterministic systolic binary tree automata and, finally, an input word w such that $2^{n-1} < |w| \leq 2^n$ is processed in n steps.

Here we define a new type of systolic automata and we study their structural properties, as well as the class of accepted languages. The communication structure considered here consists of a graph obtained from an infinite binary labeled tree (satisfying a suitable regularity condition) without leaves, in which some nodes in the same level have been connected. To be more precise, if $N(i, j)$ is the i th node (in the order from left to right) in the j th level, then each node $N(2i, j)$ is connected to the node $N(2i+1, j)$ for $1 \leq i \leq 2^{j-1} - 1, j > 1$, and vice versa. Hence, there are $2(2^{j-1} - 1)$ new connections in the j th level. Let us call the graph so defined a C-tree.

* This work has been supported by a Research Grant from the Ministero della Pubblica Istruzione of Italy and by the Italian National Council of Research.

An input word is processed in the same way as in systolic tree automata, but a 'horizontal' step is executed between any two consecutive 'bottom-up' steps. In fact, consider the first level of the C-tree containing a number of nodes greater than or equal to the length of an input word w ; then each processor of this level receives a letter of w as input: the first processor receives the first letter, the second one receives the second letter and so on. Now the information flows horizontally through the new connections, next it flows bottom-up through the father-son connections and, from this point on, the flow of information goes on towards the root following the sequence just described. The output of the root decides whether the word is accepted or not. Thus we obtain a systolic C-tree automaton, briefly SCTA. An input word w such that $2^{n-1} < |w| \leq 2^n$ is processed in $2n - 1$ steps. Hence, a word w of length n is accepted by an SCTA in $O(\log n)$ steps as in the case of systolic binary tree automata. However, the recognizing power of the former is considerably greater than that of the latter. In fact, it is shown here that in the nondeterministic case, the class of languages accepted by an SCTA contains the class of growing ETOL languages, that is, the class of languages generated by a Lindenmayer ETOL system such that the right-hand side of all the productions has length greater than or equal to two.

The model satisfies some important structural properties such as homogeneity and stability. As to homogeneity, for every (nondeterministic) SCTA whose underlying graph is any C-tree there exists an equivalent (nondeterministic) SCTA whose underlying graph is a C-tree labeled by only two different letters; one for the nodes $N(1, j)$ and $N(2^j, j)$ and one for the nodes $N(i, j)$, $1 < i < 2^j$, $j \geq 0$. This means using only two different types of processors. As to stability, there is no loss of generality in assuming that the input goes into an arbitrary level (rather than into the first possible level as done according to the definition), both in the deterministic and in the nondeterministic case. This means the possibility of giving a smaller input than one previously given before the analysis of the latter is ended. A consequence of this is that all the processors may be active simultaneously.

The class of languages accepted by deterministic systolic C-tree automata, briefly $\mathcal{L}(\text{DSCTA})$, is a boolean algebra and it holds that the class of languages accepted by nondeterministic systolic C-tree automata, briefly $\mathcal{L}(\text{SCTA})$, is closed under union and intersection.

In this paper we also give a characterization of the systolic C-tree automaton in terms of sequential machines which can be easily programmed and analysed. For this purpose we introduce a simple Turing machine to characterize both the deterministic and nondeterministic systolic C-tree automata. As a consequence, we obtain that $\mathcal{L}(\text{SCTA})$ is contained in $\text{NTIME}(n^2)$ and $\mathcal{L}(\text{DSCTA})$ is contained in $\text{DTIME}(n^2)$. The given characterizations also allow us to state results concerning $\mathcal{L}(\text{SCTA})$. In particular, we prove that $\mathcal{L}(\text{SCTA})$ is closed under concatenation on the right with regular languages.

This paper is organized as follows: in Section 1 the definition of SCTA is given and some properties of the model and of the class of accepted languages are proved.

In Section 2 the characterization of systolic C-tree automata in term of Turing machine models is given. Section 3 contains the proof that the class of growing ETOL languages is contained in $\mathcal{L}(\text{SCTA})$.

1. The model: discussion and formal definition

We start with an informal description of the model of systolic C-tree automaton we introduce in this paper.

Consider an infinite binary tree T , without leaves, and a regular labeling of T , that is, a labeling such that T has only finitely many labeled nonisomorphic subtrees. Moreover, suppose that the labels of the nodes in the rightmost path and in the leftmost one are different from the labels of all the other nodes in the tree. Let $\text{LEVEL}(j)$ denote the set of all labeled nodes of the tree whose distance to the root is j for $j = 0, 1, 2, \dots$. Clearly, there is a natural ordering from left to right of nodes in $\text{LEVEL}(j)$. The i th node in $\text{LEVEL}(j)$, $1 \leq i \leq 2^j$, is denoted by $N(i, j)$ and its label by $L(i, j)$. Now, let us connect node $N(2i, j)$ to node $N(2i+1, j)$ and vice versa for $1 \leq i \leq 2^{j-1} - 1, j > 1$.

An example of the resulting infinite labeled directed graph is partially shown in Fig. 1.

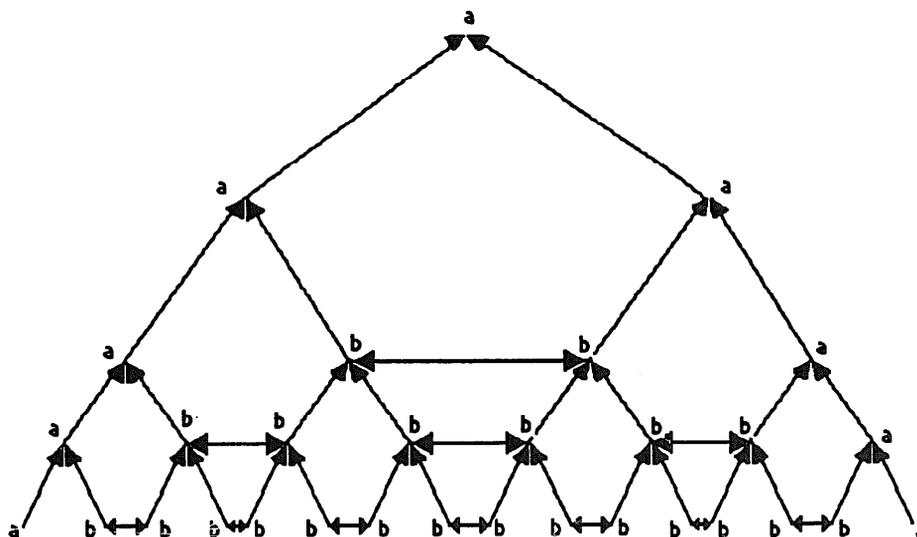


Fig. 1.

Let \mathbb{T}_Σ be the set of all the infinite labeled directed graphs obtained in this way over the alphabet Σ , and let \mathbb{T} be the union of all \mathbb{T}_Σ . For the elements of \mathbb{T} we will use all the terminology derived from trees, as if we ignored the new connections.

Let $\Sigma_p = \Sigma_1 \cup \Sigma_2$, with $\Sigma_1 \cap \Sigma_2 = \emptyset$, be the alphabet of labels of an element t in \mathbb{T} , where Σ_1 is the set of labels for the nodes in the leftmost and in the rightmost path in t and Σ_2 is the set of labels for all the other nodes in t . A node labeled X

in t will be interpreted as an X -processor and Σ_p will be called the alphabet of processor names. All X -processors for X in Σ_p have one external input pin. All X -processors for X in Σ_1 have two internal input pins and one internal output pin; the Y -processors for Y in Σ_2 have three internal input pins and two internal output pins. The automaton processes a word as follows: a word $a_1 \dots a_n$ is letter by letter put in the order from left to right into the first level j such that $2^j \geq n$, as in the case of tree systolic automata. Moreover, as usual, the nodes $N(n+1, j), \dots, N(2^j, j)$ receive a special symbol $\$$ as input if $n < 2^j$. A Y -processor, Y in Σ_2 , sends (and receives) in one time unit the output produced from the input received from its sons—or from the external pins at the beginning—to (from) the Y -processor with which it is ‘horizontally’ connected. During this time the X -processors, X in Σ_1 , in the same level are idle. Now all the processors generate an output which will be sent to their father in one unit of time.

The output produced at the root will say whether the word is accepted or not. A word w such that $2^{n-1} < |w| \leq 2^n$ is processed in $2n - 1$ steps, n of which are bottom-up steps and the others are ‘horizontal’ ones.

Definition 1.1. Given a $t \in \mathcal{T}$, a *nondeterministic systolic \mathcal{C} -tree automaton*, SCTA in short, is a 10-tuple $A = (\Sigma, \Sigma_1, \Sigma_2, L, \$, Q, B, H, I, F)$ where $\Sigma_1 \cup \Sigma_2$ is the alphabet of processor names, $\Sigma_1 \cap \Sigma_2 = \emptyset$; L is the labeling function of t with the property that Σ_1 is the set of labels of the nodes in the leftmost and the rightmost paths of t and Σ_2 is the set of labels of all the other nodes in t ; Σ and Q are the finite input alphabet and the finite set of states respectively; $\$$ is a special symbol not in Σ ; $F \subset Q$ is the set of final states; B, H and I are sets of functions defined as follows:

- $B = \{f_X : Q^2 \rightarrow P(Q) \mid X \in \Sigma_1 \cup \Sigma_2\}$ is the set of bottom-up transition functions,
- $H = \{g_{X,Y} : Q^2 \rightarrow P(Q)^2 \mid X, Y \in \Sigma_2\}$ is the set of horizontal transition functions, and
- $I = \{in_X : \Sigma \cup \{\$\} \rightarrow P(Q) \mid X \in \Sigma_1 \cup \Sigma_2\}$ is the set of input functions.

Definition 1.2. A *deterministic \mathcal{C} -tree systolic automaton*, DSCTA in short, is a 10-tuple $A = (\Sigma, \Sigma_1, \Sigma_2, L, \$, Q, B, H, I, F)$ where $\Sigma, \Sigma_1, \Sigma_2, \$, Q, F$ are as in Definition 1.1; $B = \{f_X : Q^2 \rightarrow Q \mid X \in \Sigma_1 \cup \Sigma_2\}$; $H = \{g_{X,Y} : Q^2 \rightarrow Q^2 \mid X, Y \in \Sigma_2\}$ and $I = \{in_X : \Sigma \cup \{\$\} \rightarrow Q \mid X \in \Sigma_1 \cup \Sigma_2\}$.

Generally, the processors of a systolic system are simpler than our X -processor for X in Σ_2 , which are able to compute two functions. But one can think of the model also as of a model in which each processor can compute only one function and where the information flow is unidirectional. Each connection can be replaced by a processor and by two connections as shown in Fig. 2. An example of the resulting graph is partially shown in Fig. 3. The new processors labeled c compute here the function $g_{b,b}$ and transmit to the left (right) father the first (second) component of the pair of states obtained as result of the computation.

Example 1.3. Let $A = (\{a, b, c\}, \{u, r\}, \{s\}, L, \$, \{a, b, c, \$, bc, ab, d, e, e', f, R\},$

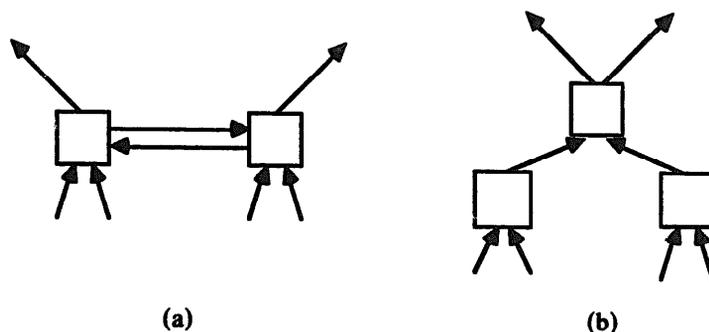


Fig. 2.

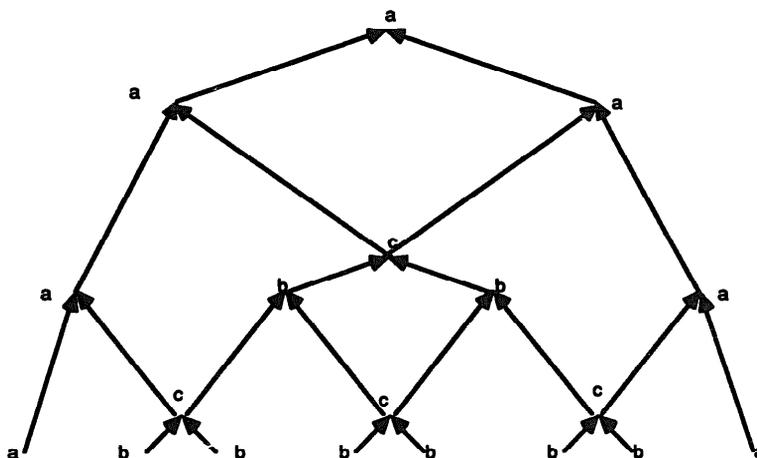


Fig. 3.

$B, H, I, \{f\}$) be a DSCTA where L is such that all the nodes in the leftmost path are labeled by u , all the nodes in the rightmost path except the root are labeled by r and the rest is labeled by s . Let $I = \{in\}$, $H = \{g\}$, $B = \{f_u, f_r, f_s\}$ ($in_u = in_r = in_s = in$, $g_{s,s} = g$), where $in(x) = x$ for every $x \in \{a, b, c, \S\}$. g and the f 's are described as follows:

$$\begin{aligned}
 g(a, b) &= (e, ab), & g(b, c) &= (bc, e), & g(c, a) &= (c, a), \\
 g(d, d) &= (d, d), & g(d, e) &= (e, d), & g(e, d) &= (d, e), \\
 g(x, \S) &= (x, \S) \quad \text{for every } x \in \{c, d, e, \S\}; \\
 f_u(a, bc) &= f_u(d, d) = d, & f_u(d, e') &= f, \\
 f_r(e, \S) &= e', & f_r(\S, \S) &= \S, \\
 f_s(a, bc) &= f_s(ab, c) = f_s(d, d) = d, & f_s(x, x) &= x \quad \text{for } x \in \{e, \S\}.
 \end{aligned}$$

In all the other cases we have $g(x, y) = (Rj, Rj)$ and $f_v(x, y) = Rj$, for $v \in \{u, r, s\}$.

The language accepted by A is $\{(abc)^{2^n} \mid n \geq 0\}$.

Figure 4 pictures the derivation steps of A for the input word $w = abcabc$.

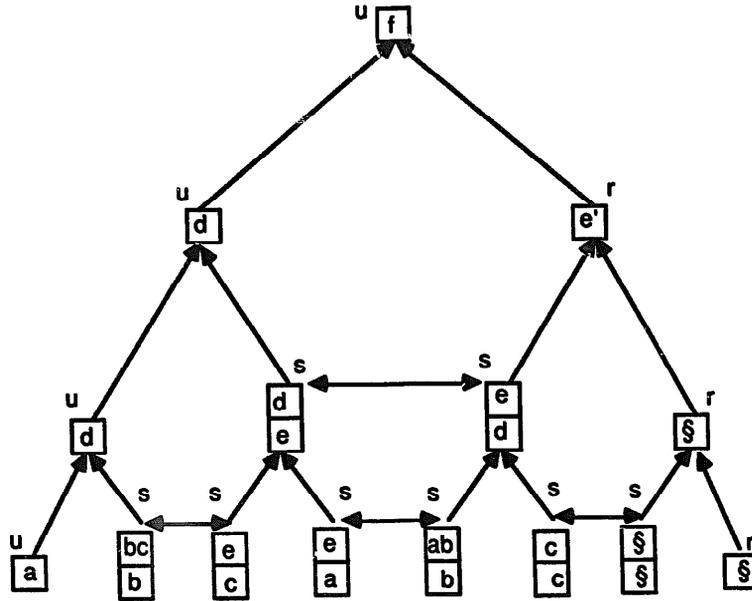


Fig. 4.

Given a (D)SCTA, let $Q^0 = \bigcup Q^{2^n}$ for $n \geq 0$, where $Q^{2^n} = \{x \in Q^* \mid |x| = 2^n\}$. We define two binary relations \rightarrow_{bu} and \rightarrow_h over Q^0 as follows:

$$q_1, \dots, q_{2^n} \rightarrow_{bu} p_1, \dots, p_{2^m}$$

iff $m = n - 1$ and

$$p_1 \in f_X(q_1, q_2), p_{2^{n-1}} \in f_Z(q_{2^{n-1}}, q_{2^n}) \text{ and } p_i \in f_{Y_i}(q_{2i-1}, q_{2i})$$

($p_1 = f_X(q_1, q_2)$, $p_{2^{n-1}} = f_Z(q_{2^{n-1}}, q_{2^n})$ and $p_i = f_{Y_i}(q_{2i-1}, q_{2i})$ for the deterministic case). Here $X = L(1, m)$ and $Z = L(2^m, m)$ belong to Σ_1 , and $Y_i = L(i, m)$ belongs to Σ_2 for $2 \leq i \leq 2^m - 1$. The second relation is defined as follows:

$$q_1, \dots, q_{2^n} \rightarrow_h p_1, \dots, p_{2^n}$$

$$\text{iff } q_1 = p_1, q_{2^n} = p_{2^n} \text{ and } (p_i, p_{i+1}) \in g_{X_i, X_{i+1}}(q_i, q_{i+1})$$

($q_1 = p_1, q_{2^n} = p_{2^n}$ and $(p_i, p_{i+1}) = g_{X_i, X_{i+1}}(q_i, q_{i+1})$ for the deterministic case). Here $L(i, n) = X_i$ belongs to Σ_2 for $2 \leq i \leq 2^n - 2$.

We say that a word $z \in Q^0$ produces, via A , the word $y \in Q^0$ in x steps if $x = 2s$ and there exist words $y_1, \dots, y_{2s} \in Q^0$ such that

$$z \rightarrow_h y_1 \rightarrow_{bu} y_2 \rightarrow_h y_3 \rightarrow_{bu} \dots \rightarrow_h y_{2s-1} \rightarrow_{bu} y_{2s} \text{ and } y_{2s} = y$$

or $x = 2s + 1$ and there exist words $y_1, \dots, y_{2s+1} \in Q^0$ such that

$$z \rightarrow_h y_1 \rightarrow_{bu} y_2 \rightarrow_h y_3 \rightarrow_{bu} \dots \rightarrow_h y_{2s-1} \rightarrow_{bu} y_{2s} \rightarrow_{bu} y_{2s+1} \text{ and } y_{2s+1} = y.$$

Given a word $w = a_1 \dots a_n \in \Sigma^*$, suppose that $2^{r-1} < n \leq 2^r$ and that $L(i, r) = X_i$ for $1 \leq i \leq 2^r$; we say that w produces, via A , a $q \in Q$ if $\text{in}_{X_1}(a_1) \dots \text{in}_{X_n}(a_n) \text{in}_{X_{n+1}}(\$) \dots \text{in}_{X_{2^r}}(\$)$ produces, via A , q in $2r - 1$ steps.

Furthermore, let E be equal to the set $\{\epsilon\}$ (where ϵ is the empty word of Σ^*) if $\text{in}_X(\$) \in F$ and $X = L(1, 0)$, and let E be equal to the empty set otherwise.

Then the set $\mathcal{L}(A)$ of words accepted by a (D)SCTA A is defined as follows:

$$\begin{aligned} \mathcal{L}(A) = & E \cup \{a \in \Sigma \mid \text{in}_X(a) \in F \text{ and } X = L(1, 0)\} \\ & \cup \{ab \mid a, b \in \Sigma, \text{in}_X(a)\text{in}_Y(b) \rightarrow_{bu} q \in F \text{ and } XY = L(1, 1)L(2, 1)\} \\ & \cup \{w \in \Sigma^* \mid w \text{ produces } q \in F\}. \end{aligned}$$

We will now prove an homogeneity result, that is, we will prove that the recognizing power of the model is not increased by allowing different labels for the nodes in the leftmost or in the rightmost path or for the nodes in the internal paths. We need only two letters, one to name the processors with only two internal input pins and another to name the processors with the three internal input pins.

In order to achieve this homogeneity result, we need a new regularity condition, that is, we have to prove that there exists only a finite number of nonisomorphic subgraphs of a certain type. Namely, those obtained by considering a given node, all its descendants and all the nodes linked to them by horizontal connections. This fact guarantees that only a finite number of different computations must be carried out to assign a state to each node. For each node, in fact, the computation depends only on the subgraph which is defined by starting from that node, going down along the father-son connections and following the horizontal links.

Let $N(t) = \{N(i, j) \mid j \geq 0 \text{ and } 1 \leq i \leq 2^j\}$ be the set of the nodes of a $t \in \mathbb{T}$. For every $v \in N(t)$, let $SG(v)$ be the infinite labeled subgraph of t , whose vertices are nodes in a set $S(v) \subset N(t)$, defined as follows:

- $v \in S(v)$;
- if $N(i, j) \in S(v)$, then $N(2i-1, j+1) \in S(v)$ and $N(2i, j+1) \in S(v)$ for $j \geq 0$ and $1 \leq i \leq 2^j$;
- if $N(2i, j) \in S(v)$, then $N(2i+1, j) \in S(v)$ for $j > 1$ and $1 \leq i \leq 2^{j-1} - 1$.

Note that $SG(N(2i, j)) = SG(N(2i+1, j))$ for every $j > 1$, $1 \leq i \leq 2^{j-1} - 1$.

Let $S(t) = \{SG(v) \mid v \in N(t)\}$, $S_1(t) = \{SG(v) \in S(t) \mid v = N(1, j), \text{ with } j \geq 0\}$, $S_r(t) = \{SG(v) \in S(t) \mid v = N(2^j, j), \text{ with } j > 0\}$ and $S_i(t) = S(t) - (S_1(t) \cup S_r(t))$.

In the following, the nodes $N(1, j)$ with $j \geq 0$ or $N(2^j, j)$ with $j > 0$ will be called external nodes while the others will be called internal nodes.

Theorem 1.4. *Let $t \in \mathbb{T}$. $S(t)$ contains only a finite number of nonisomorphic labeled graphs.*

Proof. Let t' be the underlying tree of t , that is, the tree obtained from t by ignoring the horizontal connections. For every $v = N(1, j)$ with $j > 0$, let us consider the least subtree of t' which contains all the nodes in $S(v)$, that is, the subtree rooted in $N(1, j-1)$.

It is easy to see that, for every $j, k > 0$ and $j \neq k$, if $v = N(1, j)$, $w = N(1, k)$ and $SG(v)$ is not isomorphic to $SG(w)$, then the subtrees, defined as above for v and w , are not isomorphic. From the regularity of t' it follows that $S_1(t)$ contains only a finite number of nonisomorphic graphs. Analogously, considering a subtree rooted

in $N(2^{j-1}, j-1)$ for $v = N(2^j, j)$ with $j > 0$ we can prove that $S_r(t)$ contains only a finite number of nonisomorphic graphs.

If $SG(v) \in S_i(t)$ and $v = N(2i, j)$ for $j > 1$ and $1 \leq i \leq 2^{j-1} - 1$, then we can consider the pair of subtrees of t' , t_v and t'_v rooted in $N(i, j-1)$ and $N(i+1, j-1)$ such that the nodes in $S(N(2i, j))$ are contained in the union of the sets of nodes of t_v and t'_v .

It is easy to see that if $v = N(2i, j)$ and $w = N(2h, k)$ for $j, k > 1$, $1 \leq i \leq 2^{j-1} - 1$ and $1 \leq h \leq 2^{k-1} - 1$, and if $SG(v)$ and $SG(w)$ are not isomorphic, then t_v is not isomorphic to t_w nor is t'_v isomorphic to t'_w .

Since t' satisfies the regularity condition, that is, it has only a finite number of nonisomorphic subtrees, the thesis follows also for $S_i(t)$. \square

As one can argue, it is possible to introduce an additional node labeling of a graph $t \in \mathbb{T}$ in such a way that the new label, associated (in a top-down way) to each node t uniquely identifies the subgraph defined by that node. In fact, suppose that we number the different subtrees of the underlying tree t' of t ; next, depending on that numbering, we rename the labels of t by numbers $1, \dots, m + 2n$ for suitable $m \geq 1$ and $n \geq 0$ in such a way that the numbers associated with each node determine both the numbers of its sons and the numbers of the nodes which are horizontally connected to it. Moreover, the number attached to a node determines uniquely its 'old' label in the original labeling of t . Finally, the introduced labeling is such that the root is numbered by 1, the other external nodes are numbered by $2, \dots, m$, and internal node $v = N(2i, j)$ is numbered by $m + 1, \dots, m + n$, and if $v = N(2i, j)$ is numbered $m + k$, then the node $N(2i + 1, j)$ will be numbered with $m + n + k$.

Clearly, if two nodes are labeled with the same number, they had the same label in the original labeling and determine identical subgraphs. This is the basic idea which is behind the proof of the following theorem.

Let us call the above described labeling a *canonical labeling* of the graph $t \in \mathbb{T}$.

Definition 1.5. A (D)SCTA $A = (\Sigma, \Sigma_1, \Sigma_2, L, \S, Q, B, H, I, F)$ is said to be *homogeneous* if $\text{card}(\Sigma_1) = \text{card}(\Sigma_2) = 1$.

Theorem 1.6. For every (D)SCTA there exists an equivalent homogeneous (D)SCTA.

Proof. Given a DSCTA $A = (\Sigma, \Sigma_1, \Sigma_2, L, \S, Q, B, H, I, F)$, suppose that the underlying graph t has m different subgraphs rooted in external nodes and n different subgraphs rooted in internal ones, Let L_c be the canonical labeling of t . We can define $A' = (\Sigma, \{a\}, \{b\}, \S, L_c, Q', B', H', I', F')$ such that $\mathcal{L}(A) = \mathcal{L}(A')$ as follows: Let $2n + m = s$, $Q' = (Q \cup \{E\})^s$ and, for $1 \leq h \leq s$, let $A(h) \in \Sigma_1 \cup \Sigma_2$ be the original label of the node numbered with h by the canonical labeling.

The states in Q' are tuples of states belonging to Q which correspond to computations carried out simultaneously in the different subgraphs: in fact, the processors named "a" carry out the computations of all the processors in the leftmost and in the rightmost path in the underlying graph of the original DSCTA. The processors

named "b" do the same for the computations of all the processors in the internal paths.

Formally, for every $x \in \Sigma$, $\text{in}_a(x) = (\text{in}_{A(1)}(x), \dots, \text{in}_{A(m)}(x), E^{2n})$, and $\text{in}_b(x) = (E^m, \text{in}_{A(m+1)}(x), \dots, \text{in}_{A(s)}(x))$. Moreover, for every $(q_1, \dots, q_s), (p_1, \dots, p_s) \in Q'$, if $q_j, p_j \in Q$ for $m+1 \leq j \leq s$, then

$$\begin{aligned} & \text{pr}_1(g_{bb}((q_1, \dots, q_s), (p_1, \dots, p_s))) \\ &= (E^m, \text{pr}_1(g_{A(m+1)A(m+n+1)}(q_{m+1}, p_{m+n+1})), \dots, \\ & \qquad \qquad \qquad \text{pr}_1(g_{A(m+n)A(s)}(q_{m+n}, p_s)), E^n), \end{aligned}$$

and

$$\begin{aligned} & \text{pr}_2(g_{bb}((q_1, \dots, q_s), (p_1, \dots, p_s))) \\ &= (E^{m+n}, \text{pr}_2(g_{A(m+1)A(m+n+1)}(q_{m+1}, p_{m+n+1})), \dots, \\ & \qquad \qquad \qquad \text{pr}_2(g_{A(m+n)A(s)}(q_{m+n}, p_s))) \end{aligned}$$

else $g_{bb}((q_1, \dots, q_s), (p_1, \dots, p_s)) = (E^s, E^s)$.

Let $l(i)$ and $r(i)$ represent the numbers, in the labeling introduced above, associated to the left son and right son respectively, of the node whose number is i for $1 \leq i \leq s$.

If $q_j \in Q$ for $1 \leq j \leq m$ and $p_j \in Q$ for $m+1 \leq j \leq m+n$, then

$$\begin{aligned} & f_a((q_1, \dots, q_s), (p_1, \dots, p_s)) \\ &= (f_{A(1)}((q_{l(1)}, p_{r(1)}), \dots, f_{A(m)}((q_{l(m)}, p_{r(m)}), E^{2n})) \end{aligned}$$

else $f_a((q_1, \dots, q_s), (p_1, \dots, p_s)) = E^s$.

If $q_j \in Q$ for $m+n+1 \leq j \leq s$ and $p_j \in Q$ for $m+1 \leq j \leq m+n$, then

$$\begin{aligned} & f_b((q_1, \dots, q_s), (p_1, \dots, p_s)) \\ &= (E^m, f_{A(m+1)}((q_{l(m+1)}, p_{r(m+1)}), \dots, f_{A(s)}((q_{l(s)}, p_{r(s)}))) \end{aligned}$$

else $f_b((q_1, \dots, q_s), (p_1, \dots, p_s)) = E^s$.

Finally, F' is defined to be the subset of Q' consisting of the tuples whose first element belongs to F .

By the definition of A' , it follows that $\mathcal{L}(A) = \mathcal{L}(A')$. In fact, it is quite evident that A' simulates all possible computations corresponding to different labels of t and, at the end, the states in F' check that the root produces a final state in A . \square

From now on, we will refer to any homogeneous (D)SCTA as to a 9-tuple $A = (\Sigma, \S, Q, f_a, f_b, g, \text{in}_a, \text{in}_b, F)$, by taking $\Sigma_1 = \{a\}$, $\Sigma_2 = \{b\}$ and by simply calling g the unique function g_{bb} .

As to the closure properties of $\mathcal{L}(\text{SCTA})$ and $\mathcal{L}(\text{DSCTA})$, we can prove the following theorem.

Theorem 1.7. $\mathcal{L}(\text{SCTA})$ is closed under union and intersection and $\mathcal{L}(\text{DSCTA})$ is a boolean algebra.

Proof. Let $A^i = (\Sigma^i, \S, Q_i, f_a^i, f_b^i, g^i, \text{in}_a^i, \text{in}_b^i, F^i)$ for $i = 1, 2$, be two given DSCTA's. It is easy to construct a DSCTA A' and a DSCTA A'' such that $\mathcal{L}(A') = \mathcal{L}(A^1) \cap \mathcal{L}(A^2)$ and $\mathcal{L}(A'') = (\Sigma^1)^* - \mathcal{L}(A^1)$.

Without loss of generality, we can suppose $\Sigma^1 = \Sigma^2 = \Sigma$, $Q_1 \cap Q_2 = \emptyset$.

Let $A' = (\Sigma, \$, Q', f'_a, f'_b, g', \text{in}'_a, \text{in}'_b, F')$ where

- $Q' = Q_1 \times Q_2$;
- $I' = \{\text{in}'_X : \Sigma \rightarrow Q' \mid \text{in}'_X(z) = (\text{in}_X^1(z), \text{in}_X^2(z)), X \in \{a, b\} \text{ for every } z \in \Sigma\}$;
- $B' = \{f'_X : Q'^2 \rightarrow Q' \mid f'_X((p, q)(r, s)) = (f_X^1(p, r), f_X^2(q, s)), X \in \{a, b\} \text{ for every } (p, q), (r, s) \in Q'\}$;
- $H' = \{g' : Q'^2 \rightarrow Q'^2 \mid g'((p, q), (r, s)) = ((\text{pr}_1(g^1(p, r)), \text{pr}_1(g^2(q, s))), (\text{pr}_2(g^1(p, r)), \text{pr}_2(g^2(q, s)))) \text{ for every } (p, q), (r, s) \in Q'\}$;
- $F' = F_1 \times F_2$.

It is easy to see that $\mathcal{L}(A') = \mathcal{L}(A^1) \cap \mathcal{L}(A^2)$.

Now, let $A'' = (\Sigma, \$, Q_1, f_a^1, f_b^1, g^1, \text{in}_a^1, \text{in}_b^1, F'')$ where $F'' = Q_1 - F^1$. Also in this case it is easy to verify that $\mathcal{L}(A'') = (\Sigma^1)^* - \mathcal{L}(A^1)$.

For the nondeterministic case the proof is analogous. \square

The homogeneity result allows us to give an alternative version of (D)SCTA's. In fact, the function "g" of an homogeneous (D)SCTA does not depend on the labels of the horizontally connected nodes; so the computation carried out by "g" can be made by f_b . Thus, it is legitimate to think of each processor named "b" as being able to execute the following operations:

- (1) it computes a function of two arguments, each argument being a pair of states;
- (2) it makes a copy of its own state and transmits it to the node horizontally connected with it;
- (3) it receives a copy of the state of the node horizontally connected with itself;
- (4) it sends to its father the pair of states consisting of its original state plus the state newly received in step (3).

A processor named "a" executes the following operations:

- (5) it computes a function of two arguments, one being the output of its son named "a" and one being either a pair of states sent by a son named "b" or a state sent by a son named "a";
- (6) it sends the resulting state to its father.

We suppose that a processor named "b" and a processor named "a" carry out the operations (1), (2), (3), (4) and (5), (6) respectively in one time unit, leaving the time spent to input a word out of account. Then a word of length $2^{n-1} \leq |w| \leq 2^n$ is processed in n steps.

In order to formalize all that, it is useful to distinguish between 3-tuples consisting of a state followed by a pair of state, and 3-tuples consisting of a pair of states followed by a state.

Let $t(a, b)$ be the underlying graph of an homogeneous (D)SCTA; consider the systolic automaton on $t(a, b)$ defined as follows.

Definition 1.8. A systolic BC-tree automaton, SBCTA in short, is a 5-tuple $S = (\Sigma, \$, Q, \{f_x, \text{in}_x \mid x \in \{a, b\}\}, F)$ where Σ and Q are the finite sets of input symbols and states; $\$ \notin \Sigma$ is a special symbol; $F \subset Q$ is the set of final states, $f_a : Q \times Q \cup$

$Q^2 \times Q \cup Q \times Q^2 \rightarrow P(Q)$ and $f_b: Q^2 \times Q^2 \rightarrow P(Q)$ are the transition functions and $\text{in}_x: \Sigma \rightarrow P(Q)$ for $x \in \{a, b\}$ is the input function.

A deterministic systolic BC-tree automaton, \mathcal{D} SBCTA in short, can be analogously defined.

A derivation step of a (D)SBCTA S is defined by a binary relation over $Q^0 = \bigcup Q^{2^n}$ for $n \geq 0$, as follows: $q_1, \dots, q_{2^n} \rightarrow_S p_1, \dots, p_{2^m}$ iff $m = n - 1$ and one of the following conditions hold:

- (1) $n = 1$ and $p_1 = f_a(q_1, q_2)$ or
- (2) $n > 1$ and $p_1 = f_a(q_1, (q_2, q_3))$, $p_{2^m} = f_a((q_{2^n-2}, q_{2^n-1}), q_{2^n})$ and $p_i = f_b((q_{2i-2}, q_{2i-1}), (q_{2i}, q_{2i+1}))$ for $2 \leq i \leq 2^m - 1$.

Given a word $w = s_1 \dots s_n \in \Sigma^+$, let r be the smallest integer such that $n \leq 2^r$. We say that w is accepted by S if $n = 1$ and $\text{in}_a(s_1) \in F$, or $1 < n < 2^r$ and

$$\text{in}_a(s_1) \text{in}_b(s_2) \dots \text{in}_b(s_n) \underbrace{\text{in}_b(\$) \dots \text{in}_b(\$)}_{(2^r - n - 1) \text{ times}} \text{in}_a(\$) \rightarrow_S^+ q \in F$$

or $n = 2^r$ and $\text{in}_a(s_1) \text{in}_b(s_2) \dots \text{in}_b(s_{n-1}) \text{in}_a(s_n) \rightarrow_S^+ q \in F$, where \rightarrow_S^+ is the transitive closure of \rightarrow_S .

The empty word $\epsilon \in \Sigma^*$ is accepted by S if $\text{in}_a(\$) \in F$.

Let $\mathcal{L}(S) = \{w \in \Sigma^* \mid w \text{ is accepted by } S\}$ be the language accepted by S and $\mathcal{L}((\mathcal{D})\text{SBCTA})$ be the class of languages accepted by any (D)SBCTA.

We will prove that $\mathcal{L}((\mathcal{D})\text{SCTA}) = \mathcal{L}((\mathcal{D})\text{SBCTA})$. In order to achieve this result, it is useful to give the following lemma which states that, for every SCTA, it is possible to define an equivalent SCTA such that the input function and the bottom-up transition function are nondeterministic while the horizontal function is deterministic. For simplicity we give the proof in the case of an homogeneous SCTA.

Lemma 1.9. *For every SCTA $A = (\Sigma, \$, Q, f_a, f_b, g, \text{in}_a, \text{in}_b, F)$ there exists an equivalent SCTA $A' = (\Sigma, \$, Q', f'_a, f'_b, g', \text{in}'_a, \text{in}'_b, F)$ such that $g': Q^2 \rightarrow Q^2$.*

Proof. Let $k = \max\{\text{card}(g(q_1, q_2)) \mid q_j \in Q, 1 \leq j \leq 2\}$ and $Q' = Q \cup \{[p, i] \mid p \in Q \text{ and } 1 \leq i \leq k\} \cup \{R\}$. The states $[p, i]$ for $1 \leq i \leq k$ represent copies of the state p and the functions f'_b and in'_b produce such copies as result. The function g' deterministically maps the i th copy of p and the i th copy of q to the i th element in $g(p, q)$, in some order.

For every $x \in \Sigma$ we define

$$\text{in}'_b(x) = \{[p, i] \mid p \in \text{in}_b(x) \text{ and } 1 \leq i \leq k\} \quad \text{and} \quad \text{in}'_a(x) = \text{in}_a(x).$$

For every $q_1, q_2 \in Q$, $f'_b(q_1, q_2) = \{[p, i] \mid p \in f_b(q_1, q_2) \text{ and } 1 \leq i \leq k\}$ and $f'_a(q_1, q_2) = f_a(q_1, q_2)$. If $g(p, q) = \{(p_1, q_1) \dots (p_s, q_s)\}$, then

$$g'([p, i], [q, j]) = \begin{cases} (p_i, q_i) & \text{if } i = j \text{ and } 1 \leq i \leq s, \\ (R, R) & \text{otherwise.} \end{cases}$$

From the definition of A' it follows that $\mathcal{L}(A) = \mathcal{L}(A')$. In fact, let us show that any bottom-up step followed by an horizontal step produces the same result in A as in A' . For every $p, q \in Q$, let us denote with $g_{(j)}(p, q)$, for $1 \leq j \leq \text{card}(g(p, q))$, the j th element in $g(p, q)$, considered in an arbitrary order.

Suppose that

$$p_1 \dots p_{2^n} \rightarrow_h p_1 g_{(k_1)}(p_2, p_3) \dots g_{(k_{2^{n-1}-1})}(p_{2^{n-2}}, p_{2^{n-1}}) p_{2^n}$$

is in A for $n \geq 1$ and $1 \leq k_j \leq \text{card}(g(p_{2^j}, p_{2^{j+1}}))$, $1 \leq j \leq 2^{n-1} - 1$ and that also

$$q_1 \dots q_{2^{n+1}} \rightarrow_{bu} p_1 \dots p_{2^n}$$

is in A . From the definition of A' , we have that

$$q_1 \dots q_{2^{n+1}} \rightarrow_{bu} p_1 [p_2, i_2] \dots [p_{2^{n-1}}, i_{2^{n-1}}] p_{2^n}$$

is in A' for every $1 \leq i_j \leq k$ with $2 \leq j \leq 2^n - 1$.

The next horizontal step of A' is defined if $1 \leq i_{2^j} \leq \text{card}(g(p_{2^j}, p_{2^{j+1}}))$ and $i_{2^j} = i_{2^{j+1}}$ for every $1 \leq j \leq 2^{n-1} - 1$. Hence, in A' we have that

$$p_1 [p_2, k_1] [p_3, k_1] \dots [p_{2^{n-2}}, k_{2^{n-1}-1}] [p_{2^{n-1}}, k_{2^{n-1}-1}] p_{2^n} \\ \rightarrow_h p_1 g_{(k_1)}(p_2, p_3) \dots g_{(k_{2^{n-1}-1})}(p_{2^{n-2}}, p_{2^{n-1}}) p_{2^n}$$

for every $1 \leq k_j \leq \text{card}(g(p_{2^j}, p_{2^{j+1}}))$ with $1 \leq j \leq 2^{n-1} - 1$.

The inverse proof can be analogously given. Moreover, one can prove that after the application of the input function and the first horizontal step, we have the same results in A as in A' . Since the last bottom-up step, given by $f_a = f'_a$, is obviously equal in both automata, and A and A' have the same set of final states, we have the thesis. \square

Theorem 1.10. $\mathcal{L}((D)SCTA) = \mathcal{L}((D)SBCTA)$.

Proof. Let $L \in \mathcal{L}((D)SCTA)$ and $A = (\Sigma, \$, Q, f_a, f_b, g, in_a, in_b, F)$ be the homogeneous (D)SCTA accepting L . Without loss of generality we can suppose that $g: Q^2 \rightarrow Q^2$ and

$$f_a(Q^2) \cap (f_b(Q^2) \cup \text{pr}_1(g(Q^2)) \cup \text{pr}_2(g(Q^2))) = \emptyset.$$

We can consider $S = (\Sigma, \$, Q \cup \{R\}, \{f'_x, in'_x \mid x \in \{a, b\}\}, F)$, where R is a symbol not belonging to Q and $in'_d(x) = in_d(x)$ for every $x \in \Sigma \cup \{\$\}$ and $d \in \{a, b\}$. Further, $f'_b(p, q) = f_b(\text{pr}_2(g(p)), \text{pr}_1(g(q)))$ for every $p, q \in Q^2$;

$$f'_a(p, q) = \begin{cases} f_a(p, \text{pr}_1(g(q))) & \text{if } p \in Q \text{ and } q \in Q^2, \\ f_a(\text{pr}_2(g(p)), q) & \text{if } p \in Q^2 \text{ and } q \in Q, \\ f_a(p, q) & \text{if } p, q \in Q. \end{cases}$$

It is quite evident that $\mathcal{L}(A) = \mathcal{L}(S)$.

Vice versa, given a (D)SBCTA $S = (\Sigma, \S, Q, \{f_x, \text{in}_x \mid x \in \{a, b\}\}, F)$ accepting $L \in \mathcal{L}((\text{D})\text{SBCTA})$, consider the (D)SCTA $A = (\Sigma, \S, Q', f'_a, f'_b, g, \text{in}'_a, \text{in}'_b, F)$, where $Q' = Q \cup Q^2 \cup \{R\}$ for a new symbol R , not belonging to Q . Further, we define $\text{in}'_d(x) = \text{in}_d(x)$ for every $x \in \Sigma \cup \{\S\}$ and $d \in \{a, b\}$; $g(p, q) = ((p, q), (p, q))$ for every $p, q \in Q$. Moreover, for every p and $q \in Q'$, we have

$$f'_a(p, q) = \begin{cases} f_a(p, q) & \text{if } (p, q) \in Q \times Q \cup Q^2 \times Q \cup Q \times Q^2, \\ R & \text{otherwise;} \end{cases}$$

$$f'_b(p, q) = \begin{cases} f_b(p, q) & \text{if } (p, q) \in Q^2 \times Q^2, \\ R & \text{otherwise.} \end{cases}$$

From the definition of S , it immediately follows that $\mathcal{L}(A) = \mathcal{L}(S)$. \square

From now on, of the two given equivalent versions, we will use the one which fits better.

We can extend the notion of stability introduced for systolic tree automata (see [2]) to (D)SCTA's. In general, given a (D)SCTA A , if we drop the condition that a word must be put into the first level j such that the length of the word is greater than 2^j , then the language accepted by the obtained automaton will be different. So we call *stable* the (D)SCTA's in which that does not occur.

Definition 1.11. A (D)SCTA $A = (\Sigma, \Sigma_1, \Sigma_2, L, \S, Q, B, H, I, F)$ is said to be *stable* if for every $w = a_1 \dots a_n \in \Sigma^*$, w produces a $q \in F$ iff $\text{in}_{x_1}(a_1) \dots \text{in}_{x_n}(a_n) \text{in}_{x_{n+1}}(\S) \dots \text{in}_{x_{2r}}(\S)$ produces, via A , a final state in $2r - 1$ steps for every r such that $2^r \geq n$ and $X_i = L(i, r)$ for $1 \leq i \leq 2^r$.

The following lemma will be useful in the proof of the existence of a stable (D)SCTA equivalent to a given one.

Lemma 1.12. For every (D)SCTA $A = (\Sigma, \S, Q, f_a, f_b, g, \text{in}_a, \text{in}_b, F)$ there exists an equivalent (D)SCTA $A' = (\Sigma, \S, Q', f'_a, f'_b, g', \text{in}'_a, \text{in}'_b, F')$ with the properties:

- (i) $\text{in}'_z(x) \notin f'_a(Q'^2) \cup f'_b(Q'^2) \cup \text{pr}_1(g'(Q'^2)) \cup \text{pr}_2(g'(Q'^2))$ for every $z \in \{a, b\}$ and $x \in \Sigma \cup \{\S\}$,
- (ii) $\text{in}'_a(x) = \text{in}'_b(x)$ for every $x \in \Sigma \cup \{\S\}$, and
- (iii) $\text{in}'_z(\S) \cap \text{in}'_z(x) = \emptyset$, for every $z \in \{a, b\}$ and $x \in \Sigma$.

Proof. We just give the proof for a DSCTA since the nondeterministic case is very closely similar. Without loss of generality, we can assume that A is homogeneous and that $Q \cap (\Sigma \cup \{\S\}) = \emptyset$. Further, $Q' = Q \cup \Sigma \cup \{\S\}$; $\text{in}'_z(x) = x$ for $z \in \{a, b\}$ and $x \in \Sigma \cup \{\S\}$. Moreover, let the functions f'_a, f'_b and g' be defined as follows for every

$p, q \in Q$ and $x, y \in \Sigma \cup \{\$\}$:

$$\begin{aligned}
 g'(x, y) &= g(\text{in}_b(x), \text{in}_b(y)), & g'(p, q) &= g(p, q), \\
 f'_a(x, q) &= f_a(\text{in}_a(x), q), & f'_a(p, x) &= f_a(p, \text{in}_a(x)), \\
 f'_z(p, q) &= f_z(p, q) \quad \text{for } z \in \{a, b\}.
 \end{aligned}$$

Furthermore let $F' = F \cup \{x \mid x \in \Sigma \cup \{\$\} \text{ and } \text{in}_a(x) \in F\}$.

It is obvious that A' satisfies (i), (ii) and (iii) and that $\mathcal{L}(A) = \mathcal{L}(A')$. \square

In the following, we will refer to a (D)SCTA which satisfies properties (i), (ii) and (iii) with an 8-tuple $A = (\Sigma, \$, Q, f_a, f_b, g, \text{in}, F)$.

Theorem 1.13. *For every (D)SCTA A , there exists an equivalent stable (D)SCTA A' .*

Proof. Let us give the proof in the deterministic case; the nondeterministic case can be proved analogously. Let $A = (\Sigma, \$, Q, f_a, f_b, g, \text{in}, F)$ be an homogeneous DSCTA satisfying the properties of Lemma 1.12. Suppose also that $\text{pr}_i(g(\$, \$))$, $i = 1, 2$, does not belong to the ranges of f_a, f_b and does not occur in the pairs belonging to $g((Q - \{\$\})^2)$. Consider the graph $t' \in \mathbb{T}$, shown in Fig. 5, whose labeling is defined as follows: for every $j \geq 0$, $L(1, j) = L(2^j, j) = a$, and for every $j > 1$, $i \neq 1$ and $i \neq 2^j$

$$L(i, j) = \begin{cases} c & \text{if } i = 2^k \text{ for } 1 \leq k < j, \\ b & \text{otherwise.} \end{cases}$$

Consider the DSTCA $A' = (\Sigma, \{a\}, \{b, c\}, L, \$, Q', B', H', I', F')$ defined as follows: $Q' = Q \cup \{r, s\} \cup Q\{\&\} \cup Q\{!\} \cup Q^2$, where $r, s, \&$ and $!$ are new symbols not in Q ; further, $F' = F \cup F\{\&\} \cup \{s, r\}$. The states $q\&$ will signal the end of the input word, the states $q!$ will signal the end of the simulation of the given DSCTA over

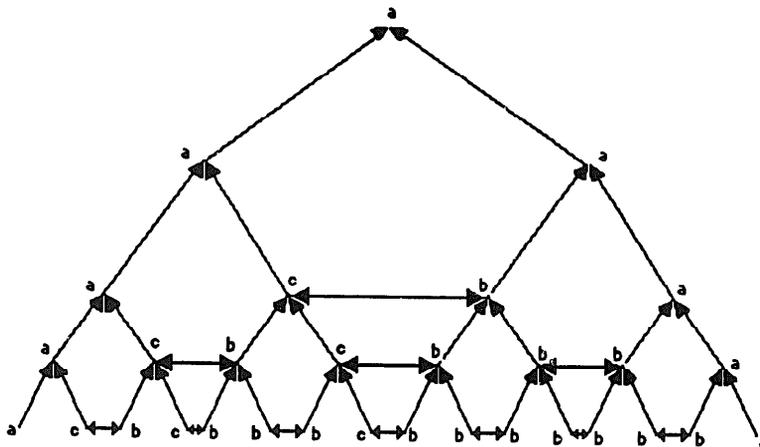


Fig. 5.

an input word fed onto a level different from the first possible one. Moreover, a c -processor produces a pair of states because it stands for an a -processor or a b -processor. Hence, the states computed by both the a -processor and the b -processor in A are carried out by the c -processor in A' until it is possible to decide which computation is the correct one.

In our definition for A' we further have $I' = \{in'_x : \Sigma \cup \{\$, \&\} \rightarrow Q', x \in \{a, b, c\}\}$ where for every $u \in \Sigma$, $in'_x(u) = in(u)$, $in'_x(\$) = \$$ for $x \in \{a, b\}$; and $in'_c(u) = (in(u), in(u))$, and $in'_c(\$) = (\$, \$)$. Finally, $H' = \{g'_{c,b}, g'_{b,b} : Q'^2 \rightarrow Q'^2\}$ and $B' = \{f'_x : Q'^2 \rightarrow Q', x \in \{a, b, c\}\}$ are given by the rules below, in which p, p', q, q' are intended to belong to Q .

The first rules allow the new automaton to simulate the behavior of A and to carry out the computations which correspond to considering a node labeled c as a node labeled a or b .

- (1) $g'_{b,b}(p, q) = g(p, q)$,
- (2) $g'_{b,b}(\$, \$) = g(\$, \$)$,
- (3) $g'_{c,b}((\$, \$), \$) = ((\$, pr_1(g(\$, \$))), pr_2(g(\$, \$)))$,
- (4) $g'_{c,b}((p, q), p') = ((p, pr_1(g(q, p'))), pr_2(g(q, p')))$,
- (5) $f'_x(p, q) = f_x(p, q)$, $x \in \{a, b\}$,
- (6) $f'_a(p, \$) = f_a(p, \$)$,
- (7) $f'_a(p, (p', q)) = f_a(p, q)$,
- (8) $f'_c(p, (p', q)) = (f_a(p, p'), f_b(p, q))$.

The following rules recognize the end of the input word, in all the possible cases. The first two rules simply notify this fact by the symbol $\&$, the other ones point it out by using the symbol $!$ and signal that the node labeled by c has been found with the property that it must be considered as if it were labeled by a and such that all the computations on the subtrees on the right-hand side must not have influence.

- (9) $g'_{b,b}(p, \$) = (pr_1(g(p, \$))\&, pr_2(g(p, \$)))$,
- (10) $f'_b(p, pr_1(g(\$, \$))) = f_b(p, pr_1(g(\$, \$)))\&$,
- (11) $g'_{c,b}((p, q), \$) = (p!, \$)$,
- (12) $f'_c(p, (\$, q)) = f_a(p, \$)!$.

The following rules simply transfer the end-of-word signal $\&$:

- (13) $g'_{c,b}((p, q), p'\&) = (pr_1(g(q, p'))\&, pr_2(g(q, p'))\&)$,
- (14) $g'_{b,b}(p\&, q) = ((pr_1(g(p, q)))\&, pr_2(g(p, q)))$,
- (15) $g'_{b,b}(p, q\&) = (pr_1(g(p, q))\&, pr_2(g(p, q))\&)$,
- (16) $f'_b(p\&, q) = f'_b(p, q\&) = f_b(p, q)\&$,
- (17) $f'_a(p\&, q) = f_a(p, q)\&$.

The following rules do the same for $!$ and carry out the correct computation.

- (18) $g'_{c,b}(p!, q) = (p!, q)$,
- (19) $f'_c(p, q!) = f_a(p, q)!$.

Next we have

- (20) $f'_c(p\&, (p', q)) = f_a(p, q)!$.

This rule has a similar effect as (11) and (12), but only after some computation steps by which the end-of-word signal $\&$ has been transferred. In fact, it recognizes

that a node labeled by c has to be considered as labeled by a and that all the computation on the subtrees on the right-hand side must not have influence. By the rule

(21) $f'_a(p, q!) = s$ if $f_a(p, q) \in F$, a new final state s is introduced, when the computation of the original DSCTA comes to a final state. The state s is transferred towards the root by the rules

$$(22) f'_a(s, p) = s,$$

$$(23) f'_a(s, (p, q)) = s.$$

Finally, the following rules treat the case of the empty word and of an input word of length 1.

$$(24) f'_a(p, (p', q')) = r \text{ if } p \in (\text{in}(\Sigma) \cup \{\$\}) \cap F,$$

$$(25) f'_a(r, (p, q)) = r,$$

$$(26) f'_a(r, p) = r.$$

The automaton A' behaves as A if the input word is put into the first level containing a number of nodes greater than the length of the word. Anyway, if this is not the case, the automaton A' will accept only the words accepted by A . In fact, suppose that a word $w = u_1 \dots u_m$, $1 \leq 2^{r-1} < m \leq 2^r$ (the cases $0 \leq m \leq 1$ should be treated separately), is fed onto the n th level with $n > r$. Let p_1, \dots, p_{2^n} and p'_1, \dots, p'_{2^n} be the sequences of states such that $p_i = \text{in}(u_i)$ for $1 \leq i \leq m$ and $p_i = \text{in}(\$)$ for $m < i \leq 2^r$; for $1 \leq i \leq m$,

$$p'_i = \begin{cases} \text{in}'_c(u_i) & \text{if } i \text{ is a power of } 2, \\ \text{in}'_b(u_i) & \text{otherwise;} \end{cases}$$

for $m < i < 2^n$,

$$p'_i = \begin{cases} \text{in}'_c(\$) & \text{if } i \text{ is a power of } 2, \\ \text{in}'_b(\$) & \text{otherwise.} \end{cases}$$

Finally, $p'_{2^n} = \text{in}'_a(\$)$ and $p'_1 = \text{in}'_a(u_1)$.

It is straightforward but tedious to show that $p_1 \dots p_{2^r}$ produces, via A , pq in $2r-2$ steps iff p'_1, \dots, p'_{2^n} produces, via A' , $pq!q_3 \dots q_{2^{n-r+1}}$ in $2r-2$ steps. Now, $pq!q_3 \dots q_{2^{n-r+1}}$ produces s , via A' , in $2n-2r+1$ steps iff $f'_a(p, q)$ is a final state. \square

2. Characterization of DSCTA and SCTA in terms of sequential devices

We are going to introduce sequential machines that will characterize (D)SCTA's. As in the case of the characterization of binary systolic automata given in [6], this tool turns out to be useful in proving properties of (D)SCTA's. First we will give an informal definition of the model.

A C-Turing machine is a model obtained by the composition of two particular kinds of Turing machines. The first one, called initializing Turing machine, has the task of preparing the input data: it translates a word $a_1 \dots a_n$ in a word $b_1 \dots b_n \xi^i$,

where $i = \min\{j \geq 0 \mid n + j \text{ is a power of } 2\}$. Then, $b_1 \dots b_n \S^i$ is given as input to a 2-tape restricted Turing machine, DRTM, which simulates both the bottom-up and the horizontal transition functions of a (D)SCTA.

A deterministic initializing Turing machine (DITM) is a Turing machine with two semi-infinite tapes, a counter tape (ct), a working tape (wt), and a read-only input tape. Initially, all the cells on both ct and wt contain blanks (λ 's), except the boundary cells, which contain @. The counter tape is used for binary counting. Moreover, an input word is presented to the machine with an endmarker \$. At every step the DITM reads an input symbol, writes a symbol on the working tape and adds 1 to the content of the counter tape; hence, before reaching the input endmarker, the DITM presents on the ct a binary number which is the length of the portion of the input which has already been read. When the input endmarker is met, the DITM keeps on writing \S 's on the working tape and increasing the counter tape until it contains the least power of two which is greater than or equal to the length of the input word. The DITM needs two states, in and in', which allow knowing whether the counter tape contains a binary number which is a power of two or not: in fact, the DITM stops when it reads the input endmarker with the counter tape containing a power of 2. The output $b_1 \dots b_n \S^j$, where $j = \min\{i \mid n + i \text{ is a power of } 2\}$, is given as input to a DRTM. It first produces a word $b'_1 \dots b'_m$ for $m = (n + i)/2$, where b'_1 is computed depending on b_1, b_2 and b_3 ; b'_2 is computed depending on b_2, b_3, b_4 and b_5 , and so on, exactly as a DSCTA does. The DRTM keeps on producing words whose lengths are reduced to one half until it produces a word of length 1. This will be its output.

With more details, a DITM is a 8-tuple $(\Sigma, \Gamma, Q, \delta, Q_s, q_0, \$, \S)$ where Σ and Γ are the input and output alphabets respectively; \S is a new symbol; Q is the set of states of the DITM; $Q_s \subset Q$ is the set of special states of the DITM. The set Q_s contains the final state q_0 , the states in and in' (the only ones which allow to read an input symbol) and the states q_f and q'_f for terminal moves. $Q - Q_s$ is the set of states which the DITM can enter after reading an input symbol. The behavior of the transition function δ is informally described as follows.

The DITM, in state in or in' while reading @ on the ct and an input symbol different from \$, enters a state of $Q - Q_s$ and moves the counter head (ch) to the left. While moving to the left, ch replaces all the 1's it meets with 0, until a symbol 0 (or λ) is read on the ct: then, the DITM writes 1 on the ct so that the content of the counter tape turns out to be incremented by 1. Moreover, the DITM writes an output symbol on the wt and enters the state in (or in' in the case that λ has been read, which means that the length of the portion of the input already consumed is a power of 2). Then the working head (wh) moves one step to right and the ch moves rightwards to the boundary cell containing the symbol @. The previous operations are iterated until the input endmarker is read. Then, if the machine is in the state in, it enters the state q_f , moves the ch to the left and replaces all the 1's it meets with 0, until it reads a symbol 0 (or λ) on the ct: then, the DITM writes 1 on the ct and \S on the wt, and enters the state in (or in'), while the wh

moves one step to right and the **ch** moves to the boundary cell. These operations are iterated, until the DITM is, for the first time, in state **in'** while reading **\$** in input, that is, until **ct** contains the least power of two greater than or equal to the length of the input. Then the machine enters the state q'_i , moves the **wh** to the first symbol written on **wt**, enters the state q_0 and stops.

A nondeterministic model (ITM) can be achieved by allowing nondeterminism in the choice of the state of $Q - Q_s$ which the machine enters after reading an input symbol different from **\$**.

If $M = (\Sigma, \Gamma, Q, \delta, Q_s, q_0, \$, \$)$ is a DITM or a ITM, we say that $a_1 \dots a_n \Rightarrow_M q_0 : b_1 \dots b_n \S^j$, with $a_i \in \Sigma$, $b_i \in \Gamma$, if, when an input $a_1 \dots a_n \$$ is presented to M , the machine, starting in state **in'**, eventually gets to state q_0 with $b_1 \dots b_n \S^j$ on the working tape.

A deterministic restricted 2-tape Turing machine (DRTM) is a 6-tuple $(\Sigma, \$, \Sigma_r, Q_0, q_0, \delta)$ where Σ is a finite alphabet of input symbols; $\$ \in \Sigma$; $\Sigma_r \subset \Sigma$ is the set of accepting symbols; $Q_0 = \{q_0, q_0^1, q_1, q_2, q_3, q_4, q_a, q_b, q_c\}$ is the set of states and

$$\delta: Q_0 \times (\Sigma \cup \{\lambda\})^2 \rightarrow Q_0 \times (\Sigma \cup \{\lambda\})^2 \times \{S, R, L\}^2,$$

is the transition function. The interpretation of a transition step is that if $\delta(q, a, b) = (q', a', b', x, y)$, then the machine in state q , while reading symbol a on the first tape and symbol b on the second tape, enters state q' , writes a' on the first tape and b' on the second tape, moves, or not, the head of the first (second) tape according to x (y) (S indicates no moves, L and R indicate a leftward and rightward shift respectively). We will often write $(q, a, b) \rightarrow (q', a', b', x, y)$ instead of $\delta(q, a, b) = (q', a', b', x, y)$. The transition steps have the following form: For every $x, y \in \Sigma$,

- (1) $(q_0, x, \lambda) \rightarrow (q_0^1, \lambda, x, R, R)$,
- (2) $(q_0^1, x, \lambda) \rightarrow (q_2, \lambda, x, R, S)$,
- (3) $(q_3, x, y) \rightarrow (q_4, x, y, S, L)$,
- (4) $(q_4, x, y) \rightarrow (q_b, x, y, S, R)$,
- (5) $(q_4, x, \lambda) \rightarrow (q_a, x, \lambda, S, R)$,
- (6) $(q_1, \lambda, x) \rightarrow (q_1, \lambda, x, R, R)$,
- (7) $(q_1, x, \lambda) \rightarrow (q_2, \lambda, x, R, S)$,
- (8) $(q_2, \lambda, x) \rightarrow (q_a, x, \lambda, S, L)$,
- (9) $(q_1, \lambda, \lambda) \rightarrow (q_c, \lambda, \lambda, S, L)$,
- (10) $(q_c, \lambda, x) \rightarrow (q_c, x, \lambda, L, L)$,
- (11) $(q_c, \lambda, \lambda) \rightarrow (q_0, \lambda, \lambda, R, S)$.

For every $x, y \in \Sigma$, there exist $z, t \in \Sigma$ such that

- (12) $(q_2, x, y) \rightarrow (q_3, z, t, S, L)$.

For every $x, y \in \Sigma$, there exists $z \in \Sigma$ such that

- (13) $(q_a, x, y) \rightarrow (q_1, \lambda, z, S, R)$,
- (14) $(q_b, x, y) \rightarrow (q_1, \lambda, z, S, R)$.

There exists an $x \in \Sigma$ such that

- (15) $(q_0, \lambda, \lambda) \rightarrow (q_0^1, \lambda, x, S, R)$.

Note that only the rules from (12) to (15) have to be specified in order to give a complete definition of a DRTM: it corresponds to specifying the functions f_a, f_b and g in the definition of a DSCTA. Moreover, the last rule determines exclusively the acceptance of the empty word.

The meaning of the previous rules is the following: Let us suppose that an input word $a_1 \dots a_n$ is presented to the DRTM on the first tape, where n is equal to a power of 2. If $n = 1$, rule (1) copies the input symbol to the second tape and the machine halts because $\delta(q_0^1, \lambda, \lambda)$ is not defined. In the other cases, rules (1) and (2) copy the first two input symbols to the second tape. Then a_2 , which is on the second tape, and a_3 on the first tape are updated (rule (12)). The symbol written on the first tape will be used to modify a_1 : the second tape head shifts to the left to check whether the preceding symbol is the first one of the input string. Rule (5) signals that it is the case and then a_1 , on the second tape, is updated (rule (13)). Then rule (10) copies the input symbol a_4 to the second tape and rule (3) again changes this symbol and a_5 . So the DRTM keeps on processing the word in the same way as a DSCTA does. Rule (14) will apply instead of rule (13) when the symbol on the second tape is different from both a_1 and a_n . Rule (8) recognizes when the symbol on the second tape is a_n and activates rule (13). Rule (9) recognizes that the whole input word has been processed and allows to copy the content of the second tape onto the first one (rule (10)–(11)). Hence the process starts again: note that the first tape now contains a word of length $n/2$. The process will stop when the first tape contains a word of length 1.

A nondeterministic version (RTM) can be defined in an obvious way, by allowing more than one right-hand side in rules of the type (12)–(15).

If $M = (\Sigma, \$, \Sigma_f, Q_0, q_0, \delta)$ is a DRTM or a RTM, a word $w \in \Sigma^*$ is said to be accepted by M if, when M starts in the state q_0 with its first tape containing w and the second one containing λ 's, M stops in the state q_0^1 with its first tape containing λ 's and the second one containing just a symbol $a \in \Sigma_f$. $\mathcal{L}(M)$ will denote the language accepted by M .

Definition 2.1. Let $M1 = (\Sigma, \Gamma, Q, \delta, Q_s, q_0, \$, \$)$ be a (D)ITM and $M2 = (\Gamma', \$, \Gamma_f, Q_0, q_0, \delta')$ be a (D)RTM, with $\Gamma \subset \Gamma'$. A (deterministic) C-Turing machine, briefly (D)CTM, $M = (M1, M2)$ can be defined as the machine obtained from the composition of the (D)ITM $M1$ with the (D)RTM $M2$ by unifying the working tape of $M1$ with the first tape of $M2$.

Definition 2.2. A word $w = a_1 \dots a_n \in \Sigma^*$ is said to be accepted by a (D)CTM $M = (M1, M2)$ with $M1 = (\Sigma, \Gamma, Q, \delta, Q_s, q_0, \$, \$)$ and $M2 = (\Gamma', \$, \Gamma_f, Q_0, q_0, \delta')$ if $n = 0$ and the empty word in Γ'^* belongs to $\mathcal{L}(M2)$, or $n > 0$ and there exist $b_1, \dots, b_n \in \Gamma$ such that $a_1 \dots a_n \Rightarrow_{M1} q_0: b_1 \dots b_n \i , where $i = \min\{j \geq 0 \mid n+j \text{ is a power of } 2\}$, and $b_1 \dots b_n \$^i \in \mathcal{L}(M2)$. $\mathcal{L}(M)$ will denote the language accepted by M ; $\mathcal{L}(\text{DCTM})$ and $\mathcal{L}(\text{CTM})$ will denote the class of languages accepted by DCTM's and CTM's respectively.

We will now show that the classes of languages accepted by DCTM's and CTM's coincide with the classes of languages accepted by DSCTA's and SCTA's respectively.

Theorem 2.3. $\mathcal{L}((D)SCTA) = \mathcal{L}((D)CTM)$.

Proof. For every (D)SCTA $A = (\Sigma, Q, \$, f_a, f_b, g, \text{in}, F)$ which satisfies the properties of Lemmas 1.12, we will define a (D)CTM $M = (M1, M2)$, with $M1 = (\Sigma, \Gamma, Q', \delta, Q_s, q_0, \$, \S)$ and $M2 = (\Gamma', \$, \Gamma_f, Q_0, q_0, \delta')$, such that

- (1) $a_1 \dots a_n \Rightarrow_{M1} q_0: \text{in}(a_1) \dots \text{in}(a_n) \S^i$, where $i = \min\{j \geq 0 \mid n+j \text{ is a power of } 2\}$;
- (2) $\forall b_1, \dots, b_n \in \Gamma$ and $\forall i$ such that $n+i$ is a power of 2, $b_1 \dots b_n \S^i \in \mathcal{L}(M2)$ iff $b_1 \dots b_n \S^i$ produces $q \in F$, via A , in $2 \log(n+i) - 1$ steps;
- (3) the empty word belongs to $\mathcal{L}(A)$ if and only if it belongs to $\mathcal{L}(M)$.

In the following we will give the definition of $M1$ and $M2$ in the deterministic case, the nondeterministic case being analogous: $\Gamma = \{\text{in}(x) \mid x \in \Sigma\}$; $Q' - Q_s = \Sigma$.

The moves of $M1$ are the following: When $M1$ reads an input symbol $x \in \Sigma$, it enters a state x and when $M1$ is in a state x and reads 0 or λ on the counter tape, it writes the symbol $\text{in}(x)$ on the working tape. So $M1$ behaves exactly as property (1) requires.

As regards the definition of $M2$, let $\Gamma' = Q$; $\Gamma_f = F$. The transition function δ is given by specifying the rules of the type (12)-(15). By the following rules $M2$ calculates the functions f and g in order to simulate the behavior of A : For every $x, y \in Q$,

$$(q_2, x, y) \rightarrow (q_3, \text{pr}_1(g(y, x)), \text{pr}_2(g(y, x)), S, L),$$

$$(q_a, x, y) \rightarrow (q_1, \lambda, f_a(y, x), S, R),$$

$$(q_b, x, y) \rightarrow (q_1, \lambda, f_b(y, x), S, R).$$

Moreover, the following rule takes into account the empty word:

$$(q_0, \lambda, \lambda) \rightarrow (q_0^1, \lambda, \$, S, R).$$

In fact, the empty word belongs to $\mathcal{L}(A)$ if and only if $\$ \in F$.

From the definition of M , it follows that $\mathcal{L}(M) = \mathcal{L}(A)$.

Vice versa, we prove that, for every (D)CTM $M = (M1, M2)$ with $M1 = (\Sigma, \Gamma, Q', \delta, Q_s, q_0, \$, \S)$ and $M2 = (\Gamma', \$, \Gamma_f, Q_0, q_0, \delta')$, there exists an (D)SCTA $A = (\Sigma, \Sigma_1, \Sigma_2, L, \$, Q, B, H, I, F)$ such that

- (1) if $a_1 \dots a_n \Rightarrow_{M1} q_0: b_1 \dots b_n \S^i$, then $i = \min\{j \geq 0 \mid n+j \text{ is a power of } 2\}$ and, for every $1 \leq j \leq n$, $\text{in}_{L_j}(a_j) = b_j$, where L_j is the label associated, by the labeling function L , to j th node of the level containing $n+i$ nodes;
- (2) $\forall b_1, \dots, b_n \in \Gamma$ and $\forall i$ such that $n+i$ is a power of 2, $b_1 \dots b_n \S^i \in \mathcal{L}(M2) \Leftrightarrow b_1 \dots b_n \S^i$ produces a $q \in F$, via A , in $2 \log(n+i) - 1$ steps;
- (3) the empty word belongs to $\mathcal{L}(A)$ if and only if it belongs to $\mathcal{L}(M)$.

We will give the definition of A for the deterministic case: $\Sigma_1 = \{a\}$; $\Sigma_2 = \{b, c\}$; the labeling function L is such that, for every $j \geq 0$, $L(1, j) = L(2^j, j) = a$, and for every $j > 1$

$$L(i, j) = \begin{cases} c & \text{if } i = 2^k \text{ for } 1 \leq k < j, \\ b & \text{otherwise.} \end{cases}$$

Further, we have $Q = \Gamma'$. For every $x \in \Sigma$, if $M1$, after reading the input symbol x , in the state \underline{in}' enters a state in which it writes the symbol y on the working tape, then $in_a(x) = in_c(x) = y$; if $M1$, after reading the input symbol x , in the state \underline{in} enters a state in which it writes the symbol y on the working tape, then $in_b(x) = y$. Moreover, $in_s(\$) = \$$ for every $s \in \{a, b, c\}$. For every $x, y, z, t \in \Gamma'$,

- if $(q_2, x, y) \rightarrow (q_3, z, t, S, L)$ is in δ' , then $g_{s,s'}(y, x) = (z, t)$, for $s, s' \in \{b, c\}$,
- if $(q_a, x, y) \rightarrow (q_1, \lambda, z, S, R)$ is in δ' , then $f_a(y, x) = z$,
- if $(q_b, x, y) \rightarrow (q_1, \lambda, z, S, R)$ is in δ' , then $f_b(y, x) = f_c(y, x) = z$.

Finally, let x_0 be such that $(q_0, \lambda, \lambda) \rightarrow (q_0^1, \lambda, x_0, S, R)$, then

$$F = \begin{cases} \Gamma_f \cup \{\$\} & \text{if } x_0 \in \Gamma_f, \\ \Gamma_f & \text{otherwise.} \end{cases}$$

It is easy to see that $\mathcal{L}(M) = \mathcal{L}(A)$. \square

The preceding theorem supplies also an upper bound to the time complexity of a (D)SCTA; in fact, it leads to $\mathcal{L}(\text{SCTA}) \subset \text{NTIME}(n^2)$ and $\mathcal{L}(\text{DSCTA}) \subset \text{DTIME}(n^2)$.

Using the characterizations of a (D)SCTA in terms of sequential machines we are able to prove new results, as well as to give simpler proofs of known results. In fact, the characterizations are such that we can easily convert sequential programs for (deterministic) C-Turing machines into parallel programs for (D)SCTA's, and vice versa. In the following, we will show that the class of languages accepted by SCTA's is closed with respect to right concatenation with regular languages. A modified version of CTM's will be more useful for that purpose. We will show that, for CTM's, we can remove, without modifying the accepting power of the model, the restriction that ITM's can enter only states \underline{in} or \underline{in}' while moving the counter head to the right before reading the next input symbol. That is, instead of the states \underline{in} and \underline{in}' , the machine may have two sets of states, say $I1$ and $I2$, and, after writing a symbol on the working tape, it can enter any state in $I1$ or $I2$ (according to the last symbol read on the counter tape, that is, according to the length of the portion of input already consumed). This means that $M1$ can retain the memory of the already read input symbols and then it behaves also as a finite-state automaton. Moreover, we can provide this machine with a rejection state.

A modified nondeterministic initializing Turing machine (MITM) is an 11-tuple $(\Sigma, \Gamma, Q, \delta, I1, I2, Q_p, q_0, q_i, \$, \$)$ where $\Sigma, \Gamma, Q, \delta, q_0, \$$ and $\$$ are the same as in the definition of an ITM; $Q_p \subset Q$ is the set of special states $\{q_0, q_r, q_r', r\}$; $I1 \subset Q$ and $I2 \subset Q$ are nonempty sets of states which allow to read an input symbol and $q_i \in I2$ is the initial state. The MITM enters a state $q \in I1$ (or $q \in I2$) exactly under

the same condition in which an ITM enters the state \underline{in} (or \underline{in}'); that is, according to the fact that the portion of the input which has already been read has length 2^j for some j or not. Moreover, when the MITM reaches the input endmarker, if it is in a state $q \in I1$, then it enters either the state q_r or r , while if it is in a state $q \in I2$, then it enters either the state q'_r or r . In the other cases, in a state $q \in I1$ ($q \in I2$), the MITM behaves exactly as an ITM behaves in the state \underline{in} (\underline{in}'). The same moves as in the definition of the ITM are defined for states q_r and q'_r . In state r no move is allowed.

Let $M1 = (\Sigma, \Gamma, Q, \delta, I1, I2, Q_p, q_0, q_i, \$, \S)$ be a MITM and $M2 = (\Gamma', \S, \Gamma_r, Q_a, Q_b, Q_0, q_0, \delta')$ be an RTM. If $\Gamma \subset \Gamma'$, a modified nondeterministic C-Turing machine (MCTM) $M = (M1, M2)$ can be defined as the machine obtained from the composition of $M1$ with $M2$ by unifying the working tape of $M1$ with the first tape of $M2$.

The language accepted by an MCTM M , denoted by $\mathcal{L}(M)$, and the class of accepted languages, denoted by $\mathcal{L}(\text{MCTM})$, can be defined analogously to the case of CTM's.

Lemma 2.4. $\mathcal{L}(\text{MCTM}) = \mathcal{L}(\text{CTM})$.

Proof. Given an MCTM $M = (M1, M2)$ with $M1 = (\Sigma, \Gamma, Q, \delta, I1, I2, Q_p, q_0, q_i, \$, \S)$ and $M2 = (\Gamma', \S, \Gamma_r, Q_a, Q_b, Q_0, q_0, \delta')$, we can define a CTM $M' = (M1', M2')$, with $M1' = (\Sigma, \Delta, Q', \mu, Q_s, q_0, \$, \S)$ and $M2' = (\Delta', \S, \Delta_r, Q_0, q_0, \delta')$ in such a way that

(1) $a_1 \dots a_n \Rightarrow_{M1} q_0 \cdot b_1 \dots b_n \S^i$ if and only if there exist $p_1, \dots, p_n \in I1 \cup I2$ such that

$$a_1 \dots a_n \Rightarrow_{M1'} q_0 \cdot [q_i, b_1, p_1][p_1, b_2, p_2] \dots [p_{n-1}, b_n, p_n] \S^i$$

(indeed, the symbol $[p_j, b_{j+1}, p_{j+1}]$ written by $M1'$ means that $M1$, in state $p_j \in I1 \cup I2$ while reading the input symbol a_{j+1} , enters a state in which it writes b_{j+1} on the working tape and enters state p_{j+1});

(2) if $[p_1, b_1, p'_1][p_2, b_2, p'_2] \dots [p_n, b_n, p'_n] \S^i \in \mathcal{L}(M2')$, then $p_1 = q_i, p'_h = p_{h+1}$ for every $1 \leq h < n-1$, $p'_n \notin \{q \in I1 \cup I2 \mid M1 \text{ in the state } q, \text{ while reading the input endmarker } \$, \text{ enters state } r\}$ and $b_1 \dots b_n \S^i \in \mathcal{L}(M2)$;

(3) if $b_1 \dots b_n \S^i \in \mathcal{L}(M2)$, then there exist $p_1, \dots, p_n \in I1 \cup I2$ such that $p_n \notin \{q \in I1 \cup I2 \mid M1 \text{ in state } q, \text{ while reading the input endmarker } \$, \text{ enters the state } r\}$ and $[q_i, b_1, p_1][p_1, b_2, p_2] \dots [p_{n-1}, b_n, p_n] \S^i \in \mathcal{L}(M2')$.

In fact, $M2'$ will check whether the input word has the form $[p, b_1, p_1] \cdot [p_1, b_2, p_2] \dots [p_{n-1}, b_n, p_n] \S^i$. Moreover, $M2'$ simulates the behavior of $M2$ in such a way that if $M2$, starting with an input $b_1 \dots b_n \S^i$, stops in state q_0^1 with a symbol b on the second tape, then $M2'$ starting with input $[p, b_1, p_1] \cdot [p_1, b_2, p_2] \dots [p_{n-1}, b_n, p_n] \S^i$ stops in state q_0^1 with $[p, b, p_n]$ on the second tape. Furthermore, $[p, b, p_n]$ is an accepting symbol for $M2'$ if and only if b is an accepting symbol for $M2$, $p = q_i$ and p_n does not yield to the rejection state r in $M1$ (i.e., $M1$ in state p_n , while reading the input symbol $\$,$ does not enter state r).

We omit the constructions of $M1'$ and $M2'$ since they are quite simple but tedious. \square

In the following we will prove that the class of language accepted by SCTA's is closed with respect to right concatenation with regular languages.

Theorem 2.5. $\mathcal{L}(\text{SCTA})$ is closed with respect to right concatenation with regular languages.

Proof. Let us consider a stable SCTA $A_1 = (\Sigma, \$, Q_1, f_a, f_b, g, \text{in}_a, \text{in}_b, F_1)$ accepting a language $L \in \mathcal{L}(\text{SCTA})$ and a regular language R recognized by a finite automaton $A_2 = (\Sigma, \delta_2, Q_2, p_0, F_2)$ (without loss of generality, we can assume that L and R are languages on the same alphabet Σ).

Let $M = (M1, M2)$ be the CTM such that $L(M) = L(A_1) = L$, where $M1 = (\Sigma, \Gamma, Q, \delta, Q_a, q_0, \$, \$)$ and $M2 = (\Gamma', \$, \Gamma_r, Q_a, Q_b, Q_0, q_0, \delta')$. Note that, from the stability of A_1 , it can be derived that $b_1 \dots b_n \$^i \in \mathcal{L}(M_2)$ for every $i \geq 0$ such that $n + i$ is a power of 2 if and only if $b_1 \dots b_n \$^i \in \mathcal{L}(M_2)$ for $i = \min\{h \geq 0 \mid n + h \text{ is a power of } 2\}$.

We will define an MITM $M1' = (\Sigma, \Gamma \cup \{\$, \}, Q', \delta'', I1, I2, Q_p, q_0, q_i, \$, \$)$ such that the MCTM $M' = (M1', M2)$ recognizes LR , as follows:

- $I1 = \{\underline{\text{in}}\} \cup Q_2$;
- $I2 = \{\underline{\text{in}}'\} \cup \{q' \mid q \in Q_2\}$;
- $Q' = Q_p \cup I1 \cup I2 \cup (Q - Q_a) \cup Q_2$.

The transition function δ'' contains the same rules as δ , but in the state $\underline{\text{in}}$ (or $\underline{\text{in}}'$), while reading the input endmarker, the machine enters the state q_r (or q_r') if and only if p_0 is in F_2 , and otherwise it enters state r : that is, if the empty word belongs to R , then $M1'$ behaves as $M1$. Besides, δ'' contains moves which allow to process any prefix of the input as $M1$ does, and to check whether the remainder of the input word belongs to R . In fact, in every state $p \in Q - Q_a$, after writing 1 on the counter tape and a symbol from Γ on the working tape, $M1'$ enters the state p_0 (besides the state $\underline{\text{in}}$) or the state p'_0 (besides the state $\underline{\text{in}}'$). This allows to start the simulation of A_2 after processing any prefix of the input word. During the simulation of A_2 , only the symbol $\$$ is written on the second tape. Hence, in a state q (or q'), with $q \in Q_2$, while reading an input symbol $a \in \Sigma$, $M1'$ enters the state $\delta_2(q, a)$. In a state $q \in Q_2$, while reading 0 (or λ) on the counter tape, $M1'$ writes the symbol $\$$ on the working tape and remains in the state q (or enters the state q' , if λ has been read). If $M1'$ reaches the input endmarker in a state q (or q'), with $q \in Q_2$, then it enters the state q_r (or q_r') if $q \in F_2$, and enters state r otherwise.

Finally, in state q_i while reading an input symbol $a \in \Sigma$, $M1'$ enters the state p_0 if and only if the empty word belongs to L , i.e., if and only if there exists an $x \in \Gamma_r$ such that $(q_0, \lambda, \lambda) \rightarrow (q_0^1, \lambda, x, S, R)$ is in δ' .

It holds that

$$\begin{aligned}
 a_1 \dots a_n \in LR &\Leftrightarrow \exists j, 0 \leq j \leq n, a_1 \dots a_j \in L \text{ and } a_{j+1} \dots a_n \in R \\
 &\Leftrightarrow a_1 \dots a_j \Rightarrow_{M_1} q_0 : b_1 \dots b_j \S^i, i = \min\{h | j+h \\
 &\quad \text{is a power of 2}\}, b_1 \dots b_j \S^i \in \mathcal{L}(M_2) \text{ and } a_{j+1} \dots a_n \in R \\
 &\Leftrightarrow a_1 \dots a_n \Rightarrow_{M_1} q_0 : b_1 \dots b_j \S^{n-j+l}, l = \min\{h | n+h \\
 &\quad \text{is a power of 2}\}, b_1 \dots b_j \S^{n-j+l} \in \mathcal{L}(M_2') \\
 &\Leftrightarrow a_1 \dots a_n \in \mathcal{L}(M'). \quad \square
 \end{aligned}$$

3. SCTA and L-SYSTEMS

In this section we will show that the class of growing EPTOL systems generates a class of languages which is contained in $\mathcal{L}(\text{SCTA})$.

We briefly recall the definition of E(P)TOL system (see [8], for example).

Definition 3.1. An ETOL system is a 4-tuple $S = (\Sigma, \mathbb{H}, \omega, \Delta)$ where Σ is a finite alphabet; $\Delta \subset \Sigma$ is the alphabet of terminal symbols; $\omega \in \Sigma^+$ is the axiom; \mathbb{H} is a finite set of tables. Each table h in \mathbb{H} is a finite substitution over Σ . Given $a \in \Sigma$, as usual, if $w \in h(a)$, we write $a \rightarrow w$ and $a \rightarrow w$ is called production. Let \Rightarrow_S be the usual yield relation for ETOL systems defined as follows: for every $w, w' \in \Sigma^*$, $w \Rightarrow_S w'$ if there exists an $h \in \mathbb{H}$ such that $w' \in h(w)$. The language generated by S , denoted by $\mathcal{L}(S)$, is defined to be $\{v \in \Delta^* | \omega \Rightarrow_S^* v\}$, where \Rightarrow_S^* is the standard reflexive-transitive closure of \Rightarrow_S .

Let $S = (\Sigma, \mathbb{H}, \omega, \Delta)$ be an ETOL system. S is called an EOL system if \mathbb{H} is a singleton; S is a propagating E(T)OL system, briefly: EP(T)OL system if, for every $h \in \mathbb{H}$ and $a \in \Sigma$, $\varepsilon \notin h(a)$, where ε is the empty word of Σ^* .

Definition 3.2. An EPTOL system is termed growing, shortly g-EPTOL system, if the length of the right-hand side of all the productions is greater than or equal to 2. Let $\mathcal{L}(\text{g-EPTOL})$ be the class of languages generated by g-EPTOL systems.

Example 3.3. A growing EPTOL system is given by $S = (\{S, S_1, S_2, F, c, d, \phi\}, \{h_1, h_2\}, S, \{c, d, \phi\})$ where $h_1(S) = \{S_1 S_2\}$, $h_1(S_1) = \{S_1 S_1\}$, $h_1(S_2) = \{S_1 S_2, S_1 S_1 \phi S_1\}$, $h_1(\phi) = \{FF\}$, $h_2(S) = \{FF\}$, $h_2(S_1) = \{S_1 S_1, cc, cd, dc, dd\}$, $h_2(\phi) = \{\phi\phi\}$, $h_2(S_2) = \{FF\}$, $h_i(x) = \{FF\}$ for $x \in \{c, d, F\}$ and $i \in \{1, 2\}$. It is easy to see that $\mathcal{L}(S) = \{v \phi^{2^m} w | v, w \in \{c, d\}^*, |v| = 2^n \text{ for } n \geq 3, |w| = 2^m \text{ and } 1 \leq m \leq n - 2\}$.

Theorem 3.4. *The class of languages $\mathcal{L}(\text{g-EPTOL})$ is contained in $\mathcal{L}(\text{SCTA})$.*

Proof. Given a g-EPTOL system $S = (\Sigma, \mathbb{H}, \omega_0, \Delta)$ with n tables, we will define an

SCTA which is able to back up through a derivation of a word in S . Let $k = \max\{h_i(a) \mid a \in \Sigma \text{ and } 1 \leq i \leq n\}$, $n = \{1, \dots, n\}$, $\Sigma^r = \{x \in \Sigma^* \mid |x| \leq r\}$ for $r \geq 0$, and let new symbols be $R, j, q, \$$; then, for every $w \in \Sigma^*$, consider the following definitions, which allow to back up through a derivation step of a word in S . Let $A(w) = \{(x, y, i) \in \Sigma^+ \times \Sigma^{k-1} \times n \mid h_i(x)y = w\}$.

$$\text{DEC}[\mathbb{H}](w) = \begin{cases} A(w) & \text{if } A(w) \text{ is not empty and } |w| \geq k, \\ \{(q, w, i) \mid 1 \leq i < n\} & \text{if } |w| < k, \\ \{R\} & \text{otherwise.} \end{cases}$$

The set $\text{DEC}[\mathbb{H}](w)$ contains all the information needed to back up through one step of derivation having w or a proper prefix of w as result. In fact, $\text{DEC}[\mathbb{H}](w)$ will be useful when w is a prefix of the whole word to be recognized. Take the g-EPTOL system of Example 3.3 and the words $w_1 = dcdcdc$, $w_2 = cdcd\phi\phi\phi$ and $w_3 = c$. We have that

$$\begin{aligned} \text{DEC}[\{h_1, h_2\}](w_1) &= \{(S_1^3, \epsilon, 2), (S_1^2, dc, 2)\}, \\ \text{DEC}[\{h_1, h_2\}](w_2) &= \{(S_1^2\phi, \phi, 2), (S_1^2, \phi^3, 2)\}, \\ \text{DEC}[\{h_1, h_2\}](w_3) &= \{(q, w_3, 1), (q, w_3, 2)\}. \end{aligned}$$

It means that w_1 may be derived from S_1^3 by applying the first or the second table of $\{h_1, h_2\}$ and that a prefix of w_1 with a rest dc , may be derived from S_1^2 . The word w_2 cannot be derived, but some prefixes of it may be derived respectively from $S_1^2\phi$, with rest ϕ and table 2, and from S_1^2 , with rest ϕ^3 and table 2. Note that we are interested only in rests whose lengths are less than k .

Let $B(w) = \{(z, u, v, i) \in \Sigma^{k-1} \times \Sigma^* \times \Sigma^* \times n \mid h_i(u)v = zw\}$.

$$G[\mathbb{H}](w) = \begin{cases} B(w) & \text{if } B(w) \text{ is not empty} \\ \{(z, q, zw, i) \mid zw \in \Sigma^{k-1}, 1 \leq i < n\} & \text{if } |w| < k, \\ \{R\} & \text{otherwise.} \end{cases} \quad \text{and } |w| \geq k,$$

The set $G[\mathbb{H}](w)$ contains the same kind of information as $\text{DEC}[\mathbb{H}](w)$ regarding a derivation step having zw or a proper prefix of zw as result for every $z \in \Sigma^{k-1}$. This information is saved in 4-tuples of the kind (z, x, y, i) , which mean that a prefix of zw may be derived from x with a rest y by using the i th table in \mathbb{H} . The set $G[\mathbb{H}](w)$ is useful when the whole word to be analysed has the form $w' = xwy$; in this case we have to guess the rest z which can be obtained by analysing the prefix x of w' .

Consider, as an example, the g-EPTOL system previously given and the word $w = \phi d d c d$. We have that

$$\begin{aligned} G[\{h_1, h_2\}](w) &= \{(\phi, \phi S_1^2, \epsilon, 2), (\phi, \phi S_1, cd, 2), (\phi^3, \phi^2, S_1^2, \epsilon, 2), (\phi^3, \phi^2 S_1, cd, 2) \\ &\quad (x\phi, S_1\phi S_1^2, \epsilon, 2), (x\phi, S_1\phi S_1, cd, 2) \text{ for } x \in \{cc, cd, dc, dd\}\}. \end{aligned}$$

Finally, let $C(w) = \{(z, u\$, i) \in \Sigma^{k-1} \times \Sigma^*\{\$\} \times n \mid h_i(u) = zw\}$.

$$\text{EG}[\mathbf{H}](w) = \begin{cases} C(w) & \text{if } C(w) \text{ is not empty,} \\ \{R\} & \text{otherwise.} \end{cases}$$

The set $\text{EG}[\mathbf{H}](w)$ contains all the information needed to take a derivation step backwards, considering w as a suffix of the whole word to be recognized. Hence, just the case in which the rest must not occur is considered. The symbol $\$$ is put in to signal the end of the input word. As an example, take the previously considered g -EPTOL system and $w = \phi\phi dc$; we have that

$$\text{EG}[\{h_1, h_2\}](w) = \{(\varepsilon, \phi S_1 \$, 2), (\phi^2, \phi^2 S_1 \$, 2), (x, S_1 \phi S_1 \$, 2) \\ \text{for } x \in \{cc, cd, dc, dd\}\}.$$

Since the right-hand side of each rule in the table of the EPTOL system S may have length k , the first time we try to take a derivation step backwards, we need to know a subword of length at least k . Let s be the greatest integer such that $k > 2^s$ ($k \geq 2$ under the hypothesis we used). The first s steps of the SBCTA that we construct store in each state a subword of the input word w of length $r = 2^s$. Each successive step corresponds to a derivation step of S , hence, we can back up through at most $n - s$ derivation steps for an input word of length l , $2^{n-1} \leq l \leq 2^s$. Then, generally, the axiom cannot be reached. We just need a way to recognize a word derived in S from the axiom with $s + 1$ steps. Note that only one bottom-up step of the SBCTA will be necessary to do that. To this aim, the following definitions are useful.

Let $\text{Ax} = \{h_{i_1}(\dots h_{i_{s+1}}(\omega_0) \dots) \mid 1 \leq i_k \leq n, 1 \leq k \leq s + 1\}$ be the set of words derived in S from the axiom with $s + 1$ steps and let $m = \max\{|\omega| \mid \omega \in \text{Ax}\}$. Further we have: $\text{Pre}(\omega) = \{w \mid \exists z \in \Sigma^m, wz = \omega\}$;

$$\text{PAX}(w) = \begin{cases} \{w@\} & \text{if } w \in \text{Ax}, \\ \{(\omega@, w) \mid \omega \in \text{Ax} \text{ and } w \in \text{Pre}(\omega)\} & \text{if } \exists \omega \in \text{Ax} \text{ and } w \in \text{Pre}(\omega), \\ \{R_j\} & \text{otherwise;} \end{cases}$$

$$\text{IAX}(w) = \begin{cases} \{(z, \omega@, zw) \mid z \in \Sigma^m, \omega \in \text{Ax}, zw \in \text{Pre}(\omega)\} & \text{if } \exists z \in \Sigma^m, \exists \omega \in \text{Ax} \\ & \text{and } zw \in \text{Pre}(\omega), \\ \{R_j\} & \text{otherwise;} \end{cases}$$

$$\text{SAX}(w) = \begin{cases} \{(z, \omega@ \mid z \in \Sigma^m, \omega \in \text{Ax} \text{ and } zw = \omega\} & \text{if } \exists z \in \Sigma^m, \exists \omega \in \text{Ax} \text{ and } zw = \omega, \\ \{R_j\} & \text{otherwise.} \end{cases}$$

We define an SBCTA A such that $\mathcal{L}(A) = \{h_{i_1}(\dots h_{i_j}(x) \dots) \mid x \in \text{Ax}, 1 \leq i_u \leq n, 1 \leq u \leq j\}$. An SCTA accepting $\mathcal{L}(S)$ is obtained from A and an SCTA accepting $\{h_{i_1}(\dots h_{i_j}(\omega_0) \dots) \mid 1 \leq i_u \leq n, 1 \leq u \leq j, 0 \leq j \leq s + 1\}$.

Informally, the SBCTA A works as follows. Suppose, for simplicity, that a word w of length 2^n , $n > s$, is put into level n . The first s bottom-up steps of A give, at level $h = n - s$, states q_1, \dots, q_l , $l = 2^h$ each one storing a word $w_i \in \Sigma^*$ of length r for $0 < i < l$ and a word $w_i\$ \in \Sigma^*\{\$, |w_i| = r$, such that $w = w_1 \dots w_l$. After the successive step, the state at node $N(1, h - 1)$ stores an element of $\text{DEC}[H](w_1 w_2) \cup \text{PAX}(w_1 w_2)$; the state at node $N(i, h - 1)$ stores an element of $G[H](w_{2i-1}, w_{2i}) \cup \text{IAX}(w_{2i-1}, w_{2i})$ for $2 \leq i \leq (l-2)/2$; and the state at node $N(l/2, h - 1)$ stores an element of $\text{EG}[H]w_{l-1}, w_l\$ \cup \text{SAX}(w_{l-1}, w_l\$)$. The following step checks whether the rest stored in the state at node $N(i, h - 1)$ is equal to the guessed rest stored in the state at node $N(i + 1, h - 1)$ for $1 < i < l/2 - 1$, and tries to back up again through a derivation step in the same way as before. Moreover, it tries nondeterministically to recognize a word belonging to Ax . If the length of a word stored in a state is less than k , then the word is considered as a rest. This process is repeated until the root is reached. If the word is accepted, a final state stores a word of the set Ax .

Let $A = (\Delta \cup \{\$, \$, Q, \{in_a, in_b, f_a, f_b\}, F)$, where the set of states can be deduced by the definitions of the functions in_a, in_b, f_a, f_b ; $F = \{\omega@ \mid \omega \in Ax\}$; $in_z(x) = x$ for every $x \in \Delta \cup \{\$, z \in \{a, b\}$ and the functions f_a and f_b are described below.

To obtain the first s steps, let $x, y, z, t \in \Sigma^r \cup \Sigma^{r/2}\{\$ \cup \{\$, \}$. We have that

$$f_b(t, x, y, z) = \begin{cases} xy\$ & \text{if } r \neq 1, x, y, t \in \Sigma \text{ and } z = \$, \\ x\$ & \text{if } r \neq 1, x, t \in \Sigma \text{ and } y = z = \$, \\ xy & \text{if } r \neq 1 \text{ and } t, x, y \in \Sigma^{r/2} \cup \Sigma^{r/2}\{\$, \\ x & \text{if } r \neq 1, x \in \Sigma^{r/2}\{\$ \cup \{\$, y = z = \$, \\ G[H](xy) \cup \text{IAX}(xy) & \text{if } x, y, t \in \Sigma^r \text{ and } |x| = |y| = r \text{ and } z \neq \$, \\ \text{EG}[H](xy') \cup \text{SAX}(xy') & \text{if } r \neq 1, y = y'\$, x, y', t \in \Sigma^r, |x| = r \text{ and } z = \$ \\ & \text{or } r = 1, y = y' \text{ and } x, y, t \in \Sigma \text{ and } z = \$, \\ \text{EG}[H](x') \cup \text{SAX}(x') & \text{if } r \neq 1, x = x'\$, x', t \in \Sigma^r \text{ and } y = z = \$ \\ & \text{or } r = 1, x = x', x, t \in \Sigma \text{ and } y = z = \$, \\ \{R_j\} & \text{otherwise;} \end{cases}$$

$$f_a((t, x), y) = \begin{cases} xy\$ & \text{if } r \neq 1, t, x, y \in \Sigma, \\ x\$ & \text{if } r \neq 1, x \in \Sigma \text{ and } y = \$, \\ x & \text{if } x \in \Sigma^{r/2}\{\$ \cup \{\$, y = \$, \\ xy & \text{if } t, x \in \Sigma^{r/2} \text{ and } y \in \Sigma^{r/2}\{\$, \\ \text{EG}[H](xy') \cup \text{SAX}(xy') & \text{if } r \neq 1, y = y'\$, t, x, y' \in \Sigma^r, |x| = r \\ & \text{or } r = 1, y = y' \text{ and } t, x, y \in \Sigma, \\ \text{EG}[H](x') \cup \text{SAX}(x') & \text{if } r \neq 1, x = x'\$, t, x' \in \Sigma^r \text{ and } y = \$ \\ & \text{or } r = 1, x = x', t, x \in \Sigma \text{ and } y = \$, \\ \{R_j\} & \text{otherwise;} \end{cases}$$

$$f_a(x, (y, z)) = \begin{cases} xy\$ & \text{if } r \neq 1, x, y \in \Sigma \text{ and } z = \$, \\ x\$ & \text{if } r \neq 1, x \in \Sigma \text{ and } y = z = \$, \\ xy & \text{if } r \neq 1 \text{ and } x, y \in \Sigma^{r/2} \cup \Sigma^{r/2}\{\$, \\ x & \text{if } r \neq 1, x \in \Sigma^{r/2}\{\$ \} \cup \{\$, y = z = \$, \\ \text{DEC}[\text{H}](xy) \cup \text{PAX}(xy) & \text{if } x, y \in \Sigma^r, |x| = |y| = r \text{ and } z \neq \$, \\ \text{EG}[\text{H}](xy) \cup \text{SAX}(xy) & \text{if } x \in \Sigma^r, |x| = r \text{ and } z = \$, \\ \{Rj\} & \text{otherwise.} \end{cases}$$

In the successive steps, the arguments of the functions f_a and f_b belong to

$$\text{DEC}[\text{H}](w) \cup G[\text{H}](w) \cup \text{EG}[\text{H}](w) \cup \text{PAX}(w) \cup \text{IAX}(w) \cup \text{SAX}(w).$$

Let $y_1, y_2, y_3, y_4, y_5, y'_1, y'_2, y'_3, y'_4 \in \Sigma^{k-1}$; $x, y, z, t \in \Sigma^r$; $i \in n$ and let $Y = \{(y', q, y'', j) \mid y', y'' \in \Sigma^{k-1} \text{ and } j \in n\}$.

$$f_b(\alpha, \beta, \gamma, \delta) = \begin{cases} G[\text{H}](xy) \cup \text{IAX}(xy) & \text{if } \alpha = (y_1, t, y_2, i), \beta = (y_2, x, y_3, i), \\ & \gamma = (y_3, y, y_4, i) \\ & \text{and } \delta = (y_4, z, y_5, i) \text{ or } \delta = (y_4, z\$, i) \\ & \text{or if } \gamma = (y_3, xy, y_4, i), \delta = (y_4, z, y_5, i) \text{ or} \\ & \delta = (y_4, z\$, i) \text{ and} \\ & y_3 = y_2 y'_2, \alpha = (y_1, t, y_2, i), \\ & \beta = (y_2, q, y_3, i) \\ & \text{or } y_2 = y_1 y'_1, y_3 = y_2 y'_2, \alpha = (y_1, q, y_2, i), \\ & \beta = (y_2, q, y_3, i) \\ & \text{or if } \alpha(y_1, t, y_2, i) \text{ and} \\ & \beta = (y_2, xy, y_3, i) \text{ and} \\ & y_4 = y_3 y'_3, \gamma = (y_3, q, y_4, i), \\ & \delta = (y_4, z, y_5, i) \text{ or} \\ & \delta = (y_4, z\$, i) \\ & \text{or } y_4 = y_3 y'_3, y_5 = y_4 y'_4, \gamma = (y_3, q, y_4, i), \\ & \delta = (y_4, q, y_5, i); \\ \\ \text{EG}[\text{H}](xy) \cup \text{SAX}(xy) & \text{if } \alpha = (y_1, t, y_2, i), \beta = (y_2, x, y_3, i), \\ & \gamma = (y_3, y\$, i) \\ & \text{and } \delta = \$ \\ & \text{or if } \alpha = (y_1, t, y_2, i), \\ & \beta = (y_2, xy\$, i) \text{ and } \gamma = \delta = \$; \\ \\ Y & \text{if } y_3 = y_2 y'_2 \text{ and } \alpha = (y_1, t, y_2, i) \text{ and} \\ & \beta = (y_2, q, y_3, i) \\ & \text{or } y_2 = y_1 y'_1, y_3 = y_2 y'_2, \alpha = (y_1, q, y_2, i), \\ & \beta = (y_2, q, y_3, i) \end{cases}$$

$$f_a(\alpha, (\beta, \gamma)) = \left\{ \begin{array}{l} \omega@ \\ \{Rj\} \\ \text{DEC}[H](xy) \cup \text{PAX}(x, y) \text{ if } \alpha = (x, y_1, i), \beta = (y_1, y, y_2, i), \\ \quad \gamma = (y_2, z, y_3, i) \\ \quad \text{or } \gamma = (y_2, z\$, i) \text{ or } \gamma = (y_2, q, y_2y'_2, i) \\ \text{or if } \alpha = (q, y_1, i), \beta = (y_1, xy, y_2, i), \\ \quad \gamma = (y_2, z, y_3, i) \\ \quad \text{or } \gamma = (y_2, z\$, i) \text{ or } \gamma = (y_2, q, y_2y'_2, i) \\ \text{or if } \alpha = (xy, y_1, i), y_2 = y_1y'_1, \\ \quad \beta = (y_1, q, y_2, i), \\ \quad \gamma = (y_2, z, y_3, i) \text{ or } \gamma = (y_2, z\$, i) \\ \text{or if } \alpha = (xy, y_1, i), y_2 = y_1y'_1, y_3 = y_2y'_2, \\ \quad \beta = (y_1, q, y_2, i), \gamma = (y_2, q, y_3, i); \\ \omega@ \\ \{Rj\} \\ \text{if } \alpha = (q, y_1, i), y_2 = y_1y'_1, \\ \quad \beta = (y_1, q, y_2, i) \text{ and} \\ \quad \gamma = (y_2, z, y_3, i) \text{ or} \\ \quad \gamma = (y_2, z\$, i) \text{ or } y_3 = y_2y'_2 \\ \quad \text{and } \gamma = (y_2, q, y_3, i); \\ \text{if } \alpha = (q, y_1, i) \text{ or } \alpha = (\omega@, y_1, i), \\ \quad \beta = (y_1, \omega@, y_2) \text{ and} \\ \quad \gamma = (y_2, \omega@, y_3) \text{ or } \gamma = (y_2, \omega@) \\ \text{or if } \alpha = \beta = \gamma = \omega@; \\ \text{otherwise.} \end{array} \right.$$

and $y_4 = y_3y'_3, \gamma = (y_3, q, y_4, i),$
 $\delta = (y_4, z, y_5, i)$
 or $\delta = (y_4, z\$, i)$
 or $y_4 = y_3y'_3, y_5 = y_4y'_4, \gamma = (y_3, q, y_4, i),$
 $\delta = (y_4, q, y_5, i);$

$\omega@$

if $y_3 = y_2y'_2, \alpha = (y_1, t, y_2, i)$ and
 $\beta = (y_2, q, y_3, i)$
 or $y_2 = y_1y'_1, y_3 = y_2y'_2, \alpha = (y_1, q, y_2, i),$
 $\beta = (y_2, q, y_3, i)$ and
 $\gamma = (y_3, \omega@, y_4), \delta = (y_4, \omega@, y_5)$ or
 $\delta = (y_4, \omega@)$
 or if $\alpha = (y_1, \omega@, y_2), \beta = (y_2, \omega@, y_3),$
 $\gamma = (y_3, \omega@, y_4)$ and
 $\delta = (y_4, \omega@, y_5)$ or
 $\delta = (y_4, \omega@)$
 or if $\alpha = (y_1, \omega@, y_2), \beta = (y_2, \omega@),$
 $\gamma = \delta = \omega@$
 or if $\alpha = \beta = \gamma = \delta = \omega@;$

$\{Rj\}$

otherwise.

$\text{DEC}[H](xy) \cup \text{PAX}(x, y)$ if $\alpha = (x, y_1, i), \beta = (y_1, y, y_2, i),$
 $\gamma = (y_2, z, y_3, i)$
 or $\gamma = (y_2, z\$, i)$ or $\gamma = (y_2, q, y_2y'_2, i)$
 or if $\alpha = (q, y_1, i), \beta = (y_1, xy, y_2, i),$
 $\gamma = (y_2, z, y_3, i)$
 or $\gamma = (y_2, z\$, i)$ or $\gamma = (y_2, q, y_2y'_2, i)$
 or if $\alpha = (xy, y_1, i), y_2 = y_1y'_1,$
 $\beta = (y_1, q, y_2, i),$
 $\gamma = (y_2, z, y_3, i)$ or $\gamma = (y_2, z\$, i)$
 or if $\alpha = (xy, y_1, i), y_2 = y_1y'_1, y_3 = y_2y'_2,$
 $\beta = (y_1, q, y_2, i), \gamma = (y_2, q, y_3, i);$

$f_a(\alpha, (\beta, \gamma)) = \left\{ \{(q, \varepsilon, j) | j \in n\}$

$\omega@$

$\{Rj\}$

otherwise.

$$f_a((\alpha, \beta), \gamma) = \left\{ \begin{array}{l} \text{EG[H]}(x, y) \cup \text{SAX}(xy) \text{ if } \alpha = (y_1, t, y_2, i) \text{ or} \\ \alpha = (y_1, q, y_2, i) \text{ with} \\ y_2 = y_1 y'_1, \beta = (y_2, x, y_3, i) \text{ and} \\ \gamma = (y_3, y\$, i) \\ \text{or if } \alpha = (y_1, t, y_2, i) \text{ or} \\ \alpha = (y_1, q, y_2, i) \text{ with} \\ y_2 = y_1 y'_1, \beta = (y_2, q, y_3, i) \text{ with} \\ y_3 = y_2 y'_2 \text{ and } \gamma = (y_3, xy\$, i) \\ \text{or if } \alpha = (y_1, t, y_2, i) \text{ or} \\ \alpha = (y_1, q, y_2, i) \text{ with} \\ y_2 = y_1 y'_1, \beta = (y_2, xy\$, i) \text{ and } \gamma = \$; \\ \omega@ \\ \text{if } \alpha = (y_1, t, y_2, i) \text{ or} \\ \alpha = (y_1, q, y_2, i) \text{ with} \\ y_2 = y_1 y'_1, \beta = (y_2, q, y_3, i) \text{ with} \\ y_3 = y_2 y'_2 \text{ and } \gamma = (y_3, \omega@) \\ \text{or if } \alpha = (y_1, \omega@, y_2), \\ \beta = (y_2, \omega@, y_3) \text{ and} \\ \gamma = (y_3, \omega@) \\ \text{or if } \alpha = (y_1, \omega@, y_2), \\ \beta = (y_2, \omega@) \text{ and } \gamma = \omega@ \\ \text{or } \gamma = \$ \\ \text{or if } \alpha = \beta = \gamma = \omega@; \\ \$ \\ \text{if } \beta = \gamma = \$; \\ \{Rj\} \\ \text{otherwise.} \end{array} \right.$$

$$f_a(\alpha, \beta) = \left\{ \begin{array}{l} \text{PAX}(tx) \text{ if } \alpha = (t, y_1, i) \text{ and } \beta = (y_1, x\$, i); \\ \omega@ \text{ if } \alpha = (\omega@, y_1) \text{ and } \beta = (y_1, \omega@) \text{ or } \alpha = \beta = \omega@, \\ \{Rj\} \text{ otherwise.} \end{array} \right.$$

We will prove that $\mathcal{L}(A) = \{h_{i_1} \dots h_{i_j}(x) \mid x \in Ax, 1 \leq i_k \leq n \text{ and } j \geq 0\}$ by considering only one case, the other ones being analogous.

Suppose that the sequence of states attached to the nodes in the n th level of t is the following:

$$q_1 = (x_1, r_1, j_1), \quad q_i = (a_i, x_i, r_i, j_i), \quad q_{2m} = (a_{2m}, x_{2m}\$, j_{2m})$$

with $x_1, x_i, x_{2m} \in \Sigma^r$; $(r_1, j_1), (a_i, j_i), (r_i, j_i), (a_{2m}, j_{2m}) \in \Sigma^{k-1} \times n$ for $1 < i < 2m \leq 2^n$; and $q_{2m+1} = \dots = q_{2^n} = \$$. If $j_u = j_{u+1}$ and $r_u = a_{u+1}$, for $1 \leq u < 2m$, the next step gives

a sequence of states p_1, \dots, p_{2^n-1} such that

$$p_1 \in f_a(q_1, (q_2, q_3)) = \text{DEC}[\mathbb{H}](x_1x_2) \cup \text{PAX}(x_1x_2);$$

$$p_{i+1} \in f_b(q_{2i}, q_{2i+1}, q_{2i+2}, q_{2i+3}) = G[\mathbb{H}](x_{2i+1}x_{2i+2}) \cup \text{IAX}(x_{2i+1}x_{2i+2})$$

for $1 \leq i \leq m-2$.

Moreover, if $2m < 2^n$, then $p_i = \S$ for $2m < i < 2^{n-1}$ and

$$p_m \in f_b(q_{2m-2}, q_{2m-1}, q_{2m}, q_{2m+1}) = \text{EG}[\mathbb{H}](x_{2m-1}x_{2m}) \cup \text{SAX}(x_{2m-1}x_{2m});$$

if $2m = 2^n$, then

$$p_m \in f_a((q_{2m-2}, q_{2m-1}), q_{2m}) = \text{EG}[\mathbb{H}](x_{2m-1}x_{2m}) \cup \text{SAX}(x_{2m-1}x_{2m}).$$

If there exist $1 \leq i, j \leq m$ such that

$$p_i \in \text{PAX}(x_{2i-1}x_{2i}) \cup \text{IAX}(x_{2i-1}x_{2i}) \cup \text{SAX}(x_{2i-1}x_{2i})$$

and

$$p_j \in \text{DEC}[\mathbb{H}](x_{2j-1}x_{2j}) \cup G[\mathbb{H}](x_{2j-1}x_{2j}) \cup \text{EG}[\mathbb{H}](x_{2j-1}x_{2j}),$$

then the next step gives a rejecting state.

Suppose that

$$p_1 = (y_1, s_1, k_1), \quad p_i = (b_i, y_i, s_i, k_i), \quad p_m = (b_m, y_m \$, k_m)$$

with $y_i \in \Sigma^r$; $(s_1, k_1), (b_i, k_i), (s_i, k_i), (b_m, k_m) \in \Sigma^{k-1} \times n$ for $1 < i < m$; and $p_{m+1} = \dots = p_{2^n-1} = \S$. It results that there exist

$$z_1, \dots, z_l \in \{h_i(a) \in \Sigma^k \mid a \in \Sigma \text{ and } i \in \{1, \dots, n\}\}$$

such that $x_1 \dots x_{2m} = z_1 \dots z_l$, $z_1 \dots z_{i_1} \in h_{k_1}(y_1)$, $z_{i_1+1} \dots z_{i_2} \in h_{k_2}(y_2)$, and $z_{i_{m-1}+1} \dots z_l \in h_{k_m}(y_m)$. If the next step will lead to a nonrejecting state, then $k_u = k_{u+1}$ and $s_u = b_{u+1}$ for $1 \leq u < m$.

Then we can conclude that if the next step from $p_1 = (y_1, s_1, k_1)$, $p_2 = (s_1, y_2, s_2, k_1), \dots, p_{m-1} = (s_{m-2}, y_{m-1}, s_{m-1}, k_1)$, $p_m = (s_{m-1}, y_m \$, k_1)$, $p_{m+1} = \$, \dots, p_{2^n-1} = \S$ with $y_i \in \Sigma^r$; $s_i \in \Sigma^{k-1}$ for $1 \leq i < m$; and $k_1 \in \{1, \dots, n\}$ does not lead to a rejecting state, then the sequence $q_1 = (x_1, r_1, j_1)$, $q_2 = (r_1, x_2, r_2, j_1), \dots, q_{2m-1} = (r_{2m-2}, x_{2m-1}, r_{2m-1}, j_1)$, $q_{2m} = (r_{2m-1}, x_{2m} \$, j_1)$, $q_{2m+1} = \$, \dots, q_{2^n} = \S$ with $r_i \in \Sigma^{k-1}$; $x_1, x_i, x_{2m} \in \Sigma^r$ for $1 \leq i \leq 2m-1 \leq 2^n$; and $j_1 \in n$ produces, in one step via A , p_1, \dots, p_{2^n-1} iff $x_1 \dots x_{2m} \in h_{k_1}(y_1 \dots y_m)$.

Suppose that $p_1 = (\omega_1 @, x_1x_2)$, $p_i = (b_i, \omega_i @, x_{2i-1}x_{2i})$, $p_m = (b_m, \omega_m \$)$, where $\omega_u \in \text{Ax}$ for $1 \leq u < m$; $b_j \in \Sigma^{2r}$, $1 < j \leq m$, and $x_1x_2z = \omega_1$; $z_i b_i x_{2i-1} x_{2i} s_i = \omega_i$; $z_m b_m x_{m-1} x_m = \omega_m$; and $p_{m+1} = \dots = p_{2^n-1} = \S$ for some $z, z_j, s_i \in \Sigma^*$, $1 < i < m$, $1 < j \leq m$. Then we can conclude that if the next step from $p_1 = (\omega_1 @, x_1x_2)$, $p_2 = (x_1x_2, \omega_1 @, x_3x_4)$, \dots , $p_{m-1} = (x_{2m-5}x_{2m-4}, \omega_1 @, x_{2m-3}x_{2m-2})$, $p_m = (x_{2m-3}x_{2m-2}\omega_1 @)$, $p_{m+1} = \dots = p_{2^n-1} = \S$ with $x_i x_{i+1} \in \Sigma^{2r}$ for $1 \leq i < 2m-1$ and $\omega_1 \in \text{Ax}$ does not lead to a rejecting state, then the sequence $q_1 = (x_1, r_1, j_1)$, $q_2 = (r_1, x_1, r_2, j_1), \dots, q_{2m-1} = (r_{2m-2}, x_{2m-1}, r_{2m-1}, j_1)$, $q_{2m} = (r_{2m-1}, x_{2m} \$, j_1)$, $q_{2m+1} =$

$\$, \dots, q_{2^n} = \$$ with $x_1, x_i, x_{2^m} \in \Sigma^r$; $r_i \in \Sigma^{k-1}$ for $1 \leq i \leq 2m-1 \leq 2^n$ and $j_1 \in n$ produces, in one step via A , $p_1, \dots, p_{2^{n-1}}$ iff $x_1 \dots x_{2^m} = \omega_1$.

Moreover, given a word $w \in \mathcal{L}(S)$ generated in m steps so that $2^m \leq |w|$, suppose that $2^{n-1} < |w| \leq 2^n$ and that w is given as input to the SBCTA A . Then the first s steps bring a sequence of states to the $(n-s)$ th level of the underlying graph, the next $m-(s+1)$ steps bring a sequence of states to the $(n-m+1)$ th level of t and each computation step of A corresponds to one derivation step in S . If $n > m$, then the next two steps bring a final state to the $(n-m-1)$ th level. Otherwise, just one step brings a final state to the root. The shortest word a g-EPTOL system can generate in m steps is a word of length 2^m so that in any case the numbers of available steps is sufficient to recognize such a word. \square

4. Open problems

The most important open problem is the characterization of the class $\mathcal{L}((D)SCTA)$. The class is very wide but, as pointed out in the Introduction, some quite simple languages as $\{a^n b^n \mid n \geq 0\}$ seem not to be accepted. The reason is analogous to the one for which the language $\{a^n b^n \mid n \geq 0\}$ is not accepted by a systolic automaton over a binary tree, but there is not a proof of that.

Acknowledgment

Parts of this paper were written while E. Fachini was visiting the Laboratoire d'Informatique Théorique et Programmation, University of Paris VII.

We thank J. Gruska for his comments and suggestions which have been very useful for the preparation of the final version of the paper.

We thank I. Guessarian for her careful revision and helpful criticism.

References

- [1] B. Courcelle, Fundamental properties of infinite trees, *Theoret. Comput. Sci.* **25** (1983) 95-169.
- [2] K. Culik II, J. Gruska and A. Salomaa, Systolic automata for VLSI on balanced trees, *Acta Inform.* **18** (1983) 335-344.
- [3] K. Culik II, J. Gruska and A. Salomaa, On a family of L languages resulting from systolic tree automata, *Theoret. Comput. Sci.* **23** (1983) 231-242.
- [4] K. Culik II, J. Gruska and A. Salomaa, Systolic trellis automata, Part I and Part II, *Internat. J. Comput. Math.* **15-16** (1984) 195-212 and 3-22.
- [5] K. Culik II, A. Salomaa, and D. Wood, Systolic trees acceptors, *RAIRO Inform. Théor.* **18** (1984) 53-69.
- [6] O.H. Ibarra and S.M. Kim, A characterization of systolic binary tree automata and applications, *Acta Inform.* **29** (1984) 193-207.
- [7] H.T. Kung, Why systolic architectures? *Comput. Magazine* **15**(1) (1982) 37-46.
- [8] G. Rozenberg and A. Salomaa, *The Mathematical Theory of L Systems* (Academic Press, New York, 1980).