LFCS

Laboratory for Foundations of Computer Science
Department of Computer Science - The University of Edinburgh

Classes of Systolic Y-Tree Automata and a comparison with Systolic Trellis Automata

by

E. Fachini
A. Maggiolo-Schettini
D. Sangiorgi

LFCS Report Series ECS-LFCS-91-166

LFCS June 1991
Department of Computer Science
University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ

Classes of Systolic Y-Tree Automata and a Comparison with Systolic Trellis Automata+

by

E. Fachini*
A. Maggiolo-Schettini**
D. Sangiorgi***

+Research partially supported by Ministero per l'Università e la Ricerca Scientifica, Progetto 40% "Modelli e specifiche per sistemi concorrenti"

*Dipartimento di Ingegneria Elettronica, II Università di Roma "Tor Vergata", Via O. Raimondo 8, 00173 Roma, Italy
**Dipartimento di Informatica , Università di Pisa, 56100 Pisa, Italy
***Department of Computer Science, University of Edinburgh, EH93JZ Edinburgh, U.K.

Abstract

In this paper we study Systolic Y-tree Automata (SYTA), a class of systolic automata where the communication structure is obtained by adding new edges, and therefore new sons, called adoptive sons, to the nodes of the underlying tree according to some regularity condition. We study SYTA in the more specific case where the tree is t-ary or a tree with base. We show that for each $s \geq 0$ the set of classes of languages accepted by SYTA whose underlying tree is a tree with base with s leaves has a maximum, called ***LsSYTA***. We study when ***LsSYTA*** is reached depending on number and position of the adoptive sons. We prove that if s and t are powers of the same base, then ***LsSYTA=LtSYTA***. We give also a simulation of SYTA on regular and modular systolic trellis automata, strengthening a previous result on simulation of systolic tree automata on systolic trellis automata.

1. Introduction

Systolic automata have been introduced as abstract models to study systolic systems (cf. [12],[13] for survey papers on the subject). Systolic systems (cf. [14]), are parallel systems composed of a large number of (a few types of) simple processing elements interconnected in a regular pattern. At each time unit, some processing elements are active and transmit simultaneously their results to the ones connected to them. These processors become the new set of active processing elements, and the process is repeated until an output is produced. Systolic automata are (infinite) networks of (a few type of memoryless) finite automata, in which the way to input a word over a given alphabet, the direction of control flow, the transition rules of the processing elements and the output node are specified.

A systolic automaton is called *regular* if processors are distributed in the network in such a way that the label of any node is uniquely determined by the labels of its control fathers (fig.1 shows a regular systolic tree automaton and a regular systolic trellis automaton, respectively; the flow of control is represented by arrows). In this paper only regular systolic automata will be considered.

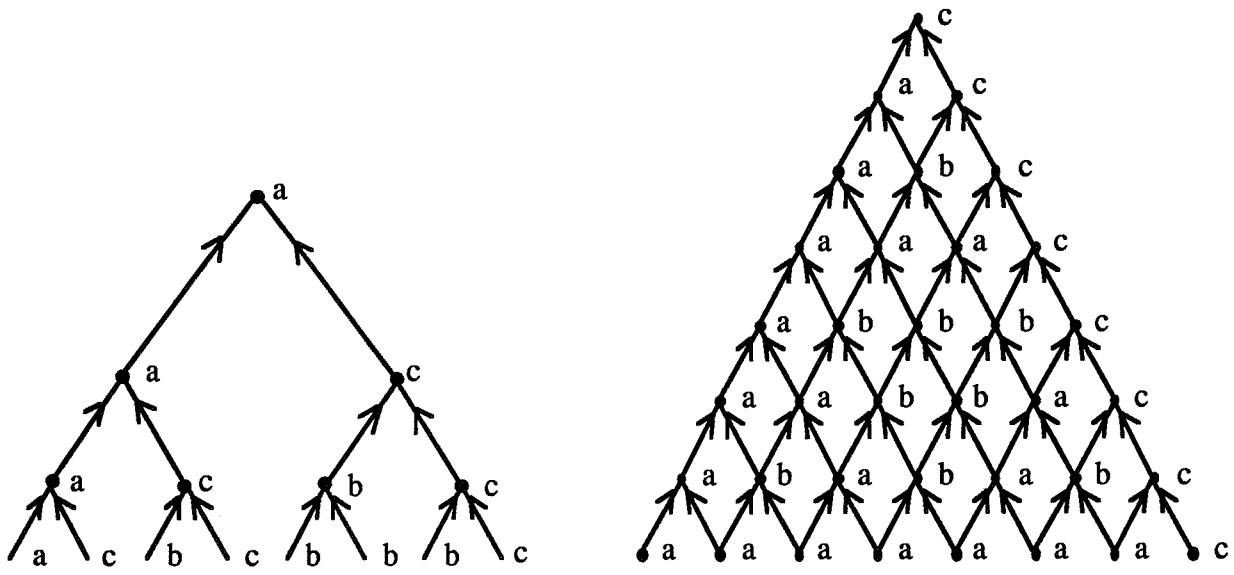

Figure 1

Systolic automata whose communication structure is an array, a tree or a trellis have been introduced and results about the power of different input/output modes, the power of various types of homogeneity, etc., have been obtained. The logarithmic time complexity of tree-like communication structures makes systolic

tree automata, STA, a very basic model; their formal investigation has been started by Culik II, Salomaa and Wood in [5] and has been continued in [1],[2],[7],[9],[16].

To keep logarithmic time complexity and to obtain greater computational power, one can add new connections as in the case of X-trees. In ([6],[8], [11]) systolic automata are studied where the connection structure is obtained from a tree by adding new connections between nodes: for a node N the only incoming edges which may be added are those with origin in the first node on the left of the leftmost son of N and in the first node on the right of the rightmost son of N. In the present paper a more general case is considered: there is no restriction on the number and origin of the added incoming edges; the only - natural - remaining conditions are that edges must connect nodes in consecutive levels and that to nodes with the same label incoming edges are added in the same way. We shall call *adoptive* the new sons assigned to a node, while the graphs and the automata themselves will be called Y-trees and systolic Y-tree automata (SYTA), respectively. In figure 2 we show a Y-tree obtained from a binary tree, in which a node labelled a has one right adoptive son and a node labeled b has one right and one left adoptive son.

The first question which arises naturally is whether and how the number or the position of these adoptive sons affects the computational power of the new model. In [8] it is proved that, with respect to binary trees, greater computational power is achieved by taking a binary tree and assigning two adoptive sons (one on the left and one on the right) only to the nodes in the leftmost path of the right subtree of the root and to the ones in the rightmost path of the left subtree of the root. Then, an infinite hierarchy of automata can be obtained by applying the above construction to subtrees rooted in nodes at successive levels. Here we prove a quite surprising result, namely that if the underlying tree is a t-ary infinite balanced tree (i.e. an infinite balanced tree with each node having exactly t sons) we get the same computational power either by assigning to any node as adoptive son the node immediately to the left (to the right) of the first (the last) son or by assigning as adoptive sons the m consecutive nodes immediately to the left of the first son and the n consecutive nodes immediately to the right of the last son, with m+n>0. This proves that the class of languages accepted by SYTA on a Y-tree whose underlying tree is a t-ary tree, has a maximum that we will call $\mathcal{LtSYTA}$. As regards the class of languages accepted if one varies the position of the adoptive sons, we prove that $\mathcal{LtSYTA}$ is also the class of languages accepted by SYTA whose underlying tree is a t-ary balanced infinite

tree and with the m-th node to the left (right) of the first (last) son of a node as its adoptive son for some m≡1(mod t).

A subclass of STA with interesting modularity properties has been introduced in [9] and furtherly studied in [16], the class of systolic automata with base (T(b)-STA). Roughly speaking, the underlying tree of a T(b)-STA, called a T(b)-tree, is obtained starting from a finite balanced tree b, called the base, and by iteratively substituting b for all the leaves of the tree already obtained. We shall call T(b)-SYTA the class of SYTA whose underlying tree is a T(b)-tree. If $\mathbf{C}$ is a class of systolic automata, we shall call $\mathcal{L}(\mathbf{C})$ the class of languages accepted by such automata.

In [16] the classes of T(b)-STA obtained by varying b are compared. In particular it is proved that if we call B^t the set of bases having t leaves, then $\mathcal{L}$(T(b)-STA), with b∈B^s, coincides with $\mathcal{L}$(T(c)-STA), with c∈B^t, if and only if s and t are powers of the same base and the number of nodes in each level of b and c satisfies a rather particular condition on the internal structure of the base. Here we tackle the same problem in the case of automata on Y-trees. We prove that for every b∈B^t, $\mathcal{L}$(T(b)-SYTA)=$\mathcal{LtSYTA}$. It follows that in order to compare classes of $\mathcal{L}$(T(b)-SYTA) where b∈B^t with varying t, it is enough to consider $\mathcal{LtSYTA}$. We show that if s and t are powers of the same number, then $\mathcal{LsSYTA} = \mathcal{LtSYTA}$. We conjecture that this condition is also necessary; that is, in the case of SYTA, the equivalence between $\mathcal{L}$(T(b)-STA) and $\mathcal{L}$(T(c)-STA) would not depend on the internal structure of the bases b and c.

Notice that we have not specified whether the automata we consider are deterministic or not. Actually, every mentioned result is proved in the paper for the deterministic case, but it holds also in the nondeterministic version. Moreover let us just mention that it would be easy to generalize the arguments given to characterize the class of languages accepted by a nondeterministic SYTA on a binary Y-tree in which every node has one adoptive son, (cf. [8]), and prove that in the nondeterministic case $\mathcal{L2SYTA} = \mathcal{LtSYTA}$, for any t>2.

In [7], it has been shown that for any systolic tree automaton on a t-ary balanced infinite tree a modular and regular systolic trellis automaton can be found which accepts the same language. A systolic trellis automaton is a systolic automaton whose communication structure is a labeled triangularly shaped infinite square grid (cf.[3],[4]). A systolic automaton is called modular if the trellis network of processors can be seen as built from simple blocks (of rhombic shape) of processors using simple recurrent rules. The modularity condition can be viewed as to imply that the corresponding chips can be designed from simple modules using recurrent design rules.

In this paper we prove that for every t, $\mathcal{LtSYTA}$ is contained in the class of languages accepted by regular and modular systolic trellis automata. Since SYTA on t-ary balanced infinite trees are strictly more powerful than the corresponding STA, this result strengthens that of [7]. Notice that the same result could have not been obtained for homogeneous (i.e. all the processors are identical) systolic trellis automata, which are known to be incomparable with systolic tree automata in terms of computational power ([3]). Our results allow to bring some more light in the understanding of how powerful the notions of regularity and modularity are for trellises: a number of different communication structures become provably representable on such trellises.

Simulation results like the ones mentioned above, can be helpful in the development of a VLSI algorithm; some VLSI algorithms are conceptually simpler to be formulated on a certain communication structure even though this structure may turn out to be more difficult or more expensive to be implemented than some others.

The paper is organized as follows: in section 2 we give some preliminaries; in section 3 we prove the main results of comparison among classes of SYTA, and in section 4 we study the construction of regular and modular trellises accepting $\mathcal{LtSYTA}$.

A preliminary version of sections 2 and 3 has been presented in [10].

2. Preliminary definitions and properties.

Given a infinite labeled tree T, let $level_T(i)$ denote the set of all the nodes of T whose distance to the root is i, for i=0,1,2, ... Let us consider the natural ordering from left to right of nodes in $level_T(i)$ and let us call $N_T(i,j)$ and $L_T(i,j)$ the j-th node in $level_T(i)$ and its label, respectively, for $1 \le j \le |level_T(i)|$. Given a node $N_T(i,j)$, which is the first (last) son of a node in $level_T(i-1)$, let us call n-th predecessor (successor) of $N_T(i,j)$ the node $N_T(i,j-n)$, $1 \le n < j$,($N_T(i,j+n)$, $j < n+j \le |level_T(i)|$).

Consider a finite alphabet P and an infinite leafless tree T, labeled by elements in P. We assume that the following two conditions are satisfied: the *arity condition*, namely nodes with the same label have the same number of sons, the *exponential growth condition*, namely there exists a real number $\alpha>1$ such that for every $k \ge 1$, $|level_T(k)| > \alpha^k$. Let $P_f(N^+)$ be the set of finite subsets of the set $N^+$of positive natural numbers and h be a function from P to $P_f(N^+) \times P_f(N^+)$;

a *Y-tree* $\bar{T}=(T,P,h)$ is obtained from T as follows: if h(a) = $(\{i_1, ...,i_m\}, \{j_1, ...,j_n\})$ then for each node which is labeled a add an incoming edge from the i_u-th predecessor of its first son, $1\leq u\leq m$, and an incoming edge from the j_v-th successor of its last son, $1\leq v\leq n$. When the endpoints are not defined, dummy sons are added (this does not change the power of the obtained structure, but makes the treatment easier). All the new sons of a node created by the above procedure will be called *adoptive sons*. The *degree* of a node labeled a, d(a), is the total number of its sons - adoptive sons included. A Y-tree $\bar{T}$ is *regular* if the label of a node uniquely identifies the labels of its nondummy sons. If N is a node of the Y-tree $\bar{T}=(T,P,h)$, then we will call *subtree rooted in N* the subgraph of $\bar{T}$ given by the subtree of T whose root is node N.

In fig.2 we show a regular Y-tree $\bar{T}=(T,P,h)$ where T is a binary tree with labels in P={a,b,c}, h(a)=h(c)=(∅,{1}), h(b)=({1},{1}).

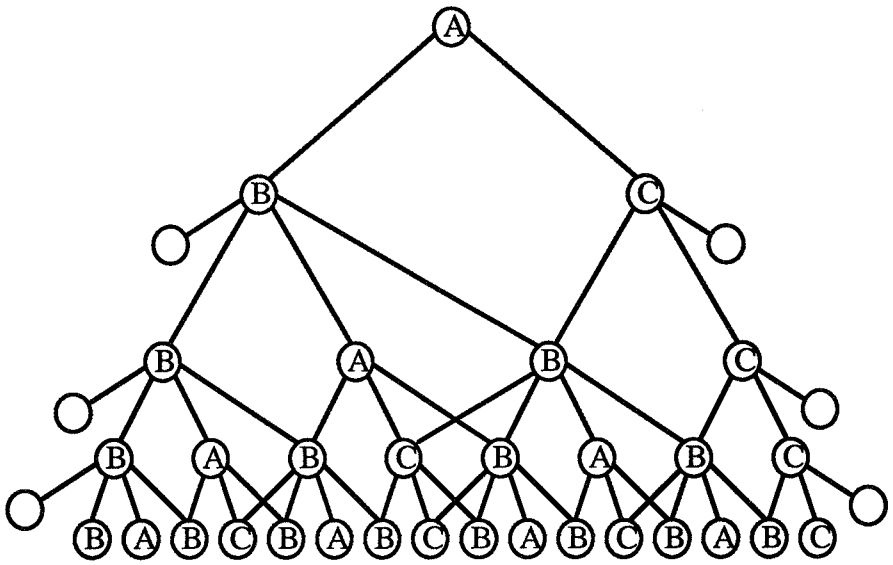

Figure 2

Definition 1. A *Systolic Automaton on a Regular Y-tree* $\bar{T}=(T,P,h)$, briefly SYTA, is a 6-tuple $A=(\bar{T},I,Q,Q',G,F)$, where:

i) $\bar{T}$ is the underlying Y-tree,

ii) I is the input alphabet, which contains the special symbol #,
iii) Q is a finite set of states which contains the special state #,
iv) $Q' \subseteq Q-\{\#\}$ is the set of final states,
v) $G = \{g_a : I \rightarrow Q-\{\#\}, a \in P\}$ is the set of input functions,
vi) $F = \{f_a : Q^{d(a)} \rightarrow Q, a \in P\}$ is the set of transition functions. ♦

If A is an SYTA and T its underlying tree, we will also denote the node $N_T(i,j)$ by A(i,j) and the nodes in $level_T(i)$ by $level_A(i)$.

To define formally the language accepted by the SYTA A, we introduce some notions. Firstly, we suppose that a dummy node is always in the particular quiescent state #. Given a word w over the alphabet I-{#} with |w|=t, let m be the smallest integer such that $|level_A(m)|=u \geq t$. Let $a_1,...,a_u$ be the labels of the nondummy nodes (from left to right) in the m-th level. We define $O_A(m,w)$ the word x_m of length u over the alphabet Q such that the j-th letter of x_m is equal to the result of applying g_{a_j} to the j-th letter of $w\#^{u-t}$. Assume that we have already defined $O_A(i,w)=x_i$, for some i with $1 \leq i \leq m$. Then $O_A(i-1,w)$ is defined as follows. Let $|level_A(i-1)|=r$ (clearly, $r \leq |x_i|$ and $|x_i|=|level_A(i)|$); consider the projections y_j, $1 \leq j \leq r$, of x_i on the nodes in the i-th level that are sons of the node A(i-1,j) and let the latter node be labelled by b_j. Let m_1 (m_2) be the number of dummy sons of A(i-1,j) on the left (right) of its nondummy sons and let $\bar{y}_j = \#^{m_1} y_j \#^{m_2}$. Then $O_A(i-1,w)$ is the word of length r over Q whose j-th letter, for j=1,...,r, is the result of applying f_{b_j} to the letters of $\bar{y}_j$, in the order. The arity condition guarantees that f_{b_j} is an s-argument function, where s is the length of $\bar{y}_j$. Let $O_A(i,j,w)$ be the j-th letter of $O_A(i,w)$, for $1 \leq i \leq m$. Sometimes we shall write $O_A(k)$ to denote the state entered by the node k.

The word w is accepted by A if and only if $O_A(0,w)$ is in Q'. The *language accepted* by a SYTA is defined as $\mathcal{L}(A)=\{w \in (I-\{\#\})^* \mid O_A(0,w) \in Q'\}$. Given a Y-tree $\bar{T}$, the class of (languages accepted by the) SYTA over $\bar{T}$ will be called $\bar{T}$-SYTA ($\mathcal{L}(\bar{T}\text{-SYTA})$). Note that, when $h(a)=(\varnothing,\varnothing)$ for every label A of T, $\bar{T}$-SYTA is the class of systolic tree automata introduced by Culik II, Salomaa and Wood in [5].

Definition 2. A SYTA $A=(\bar{T},I,Q,Q',G,F)$ is in *normal form* if: i) each input function is the identity function, ii) $f_a(x_1, ...,x_{d(a)}) = \# \Leftrightarrow x_1 = ...= x_{d(a)} = \#$, for every $a \in P$. A SYTA $A=(\bar{T},I,Q,Q',G,F)$ is *homogeneous* if it holds that $d(a)=d(b)$ implies $a=b$ for all a,b labels of $\bar{T}$. ♦

The proof of the following theorem can be easily obtained by generalizing the one for the same properties given in [6] for DSBCTA, which is the class of SYTA on $\bar{T}=(T,P,h)$ whose underlying tree T is complete binary, i.e. such that every node has arity 2 and $h(P)=\{(\{1\},\{1\})\}$.

Theorem 1. For every SYTA there exists an equivalent and homogeneous one which is in normal form. ♦

In the following, otherwise explicitly said, we shall consider only SYTA in normal form; therefore, we shall always omit specifying the input function. Moreover in the successive simulation results, Theorem 1 will allow us to assume that the simulated automata are homogeneous (simulating automata will not be homogeneous in general).

Now we shall consider a particular subclass of regular Y-trees, the class of *Y-trees with base.*

Definition 3. A *base b* is a balanced labeled finite tree. A T(b)-tree is obtained in the following way :

step 1. take a base b,

step 2. to each leaf of the so far constructed tree attach a copy of the base b in such a way that the root of the base substitutes the leaf,

step 3. repeat step 2.

We call hgt(b) the height of a given base, and B^t the set of bases with t leaves. We call T(b)-Y-tree a Y-tree whose underlying tree is a T(b)-tree and T(b)-SYTA a systolic automaton on a T(b)-Y-tree. We call t-Y-tree a Y-tree whose underlying tree has nodes all of arity t, and t-SYTA the class of systolic automata on a t-Y-tree. Note also that if $\bar{T}=(T,P,h)$ with $h(P)=\{(\varnothing,\varnothing)\}$, $\bar{T}$-SYTA is the class of systolic tree automata with base defined in [9].

Let **k** be the set {1,...,k}. In the case where in the Y-tree $\bar{T}$ = (T,P,h) we have h(P)={(**m**,**n**)} or h(P)={({m},{n})}, the obtained Y-trees (resp. T(b)-Y-trees) will be called (**m**,**n**)t-Y-trees and (m,n)t-Y-trees (resp.(**m**,**n**)T(b)-Y-trees and (m,n)T(b)-Y-trees). The corresponding automata will be called (**m**,**n**)t-SYTA and (m,n)t-SYTA (resp. (**m**,**n**)T(b)-SYTA and (m,n)T(b)-SYTA).
In the SYTA in the figures 3-9, the nodes are marked with the output (or part of the output) that they produce. In the simulated SYTA letters all different from each other have been chosen in order to better show how the simulation takes place.

3. Comparison among classes of SYTA.

In this section, after giving a general lemma on SYTA, we study subclasses of SYTA on t-ary trees and trees with base with particular patterns of adoptive sons. Firstly, we compare the classes of (**m**,**n**)t-SYTA with varying m and n, and we show that if m+n>0 all the respective classes of accepted languages coincide in one class. We shall call $\mathcal{L}t\mathcal{SYTA}$ this unique class. We consider the classes of languages accepted by t-SYTA in which every node has one only adoptive son and we show under which conditions the class of accepted languages is $\mathcal{L}t\mathcal{SYTA}$. Next we see whether and how these comparison results can be extended to systolic automata on Y-trees with base. We shall show that it is possible to construct a (**m**,**n**)T(c)-SYTA, with m+n>0 and $c \in B^s$, equivalent to a (**p**,**q**)T(b)-SYTA, with p+q>0 and $b \in B^t$, if t and s are powers of the same number. This result confirms the usefulness of adoptive sons when compared to its analogous in the case of systolic automata on trees with base where an additional and rather involved condition on the internal structure of the base is necessary (cf. [9]); it shows also that even in the case of systolic automata on T(b)-Y-trees the presence of more than one adoptive son does not increase the computational power.

Given sets W,X,Y,Z we write $(W,X) \subseteq (Y,Z)$ if $W \subseteq X$ and $Y \subseteq Z$.

Lemma 1. $\mathcal{L}(\bar{T}\text{-SYTA}) \subseteq \mathcal{L}(\bar{T}'\text{-SYTA})$ whenever $\bar{T}=(T,P,h)$, $\bar{T}'=(T,P,h')$ and $h(a) \subseteq h'(a)$ for any $a \in P$.

Proof. Immediate. ♦

Theorem 2. Given t>1, all the classes $\mathcal{L}$((**m**,**n**)t-SYTA), with $m,n \geq 0$ and m+n>0, coincide.

Proof. We shall show, by induction on m, that $\mathcal{L}$((**m**,**n**)t-SYTA)= $\mathcal{L}$((**m+1**,**n**)t-SYTA) for a given n>0 and every m≥0. The proof that $\mathcal{L}$((**m**,**n**)t-

SYTA)=$\mathcal{L}$((**m**,**n**+**1**)t-SYTA) for a given m>0 and every n ≥ 0 is completely analogous.

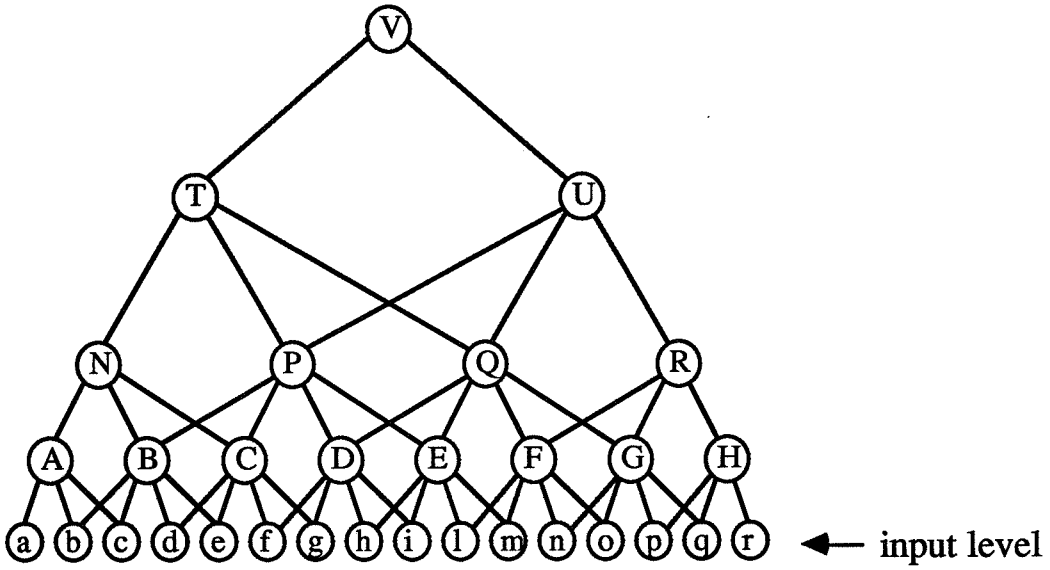

Figure 3.a

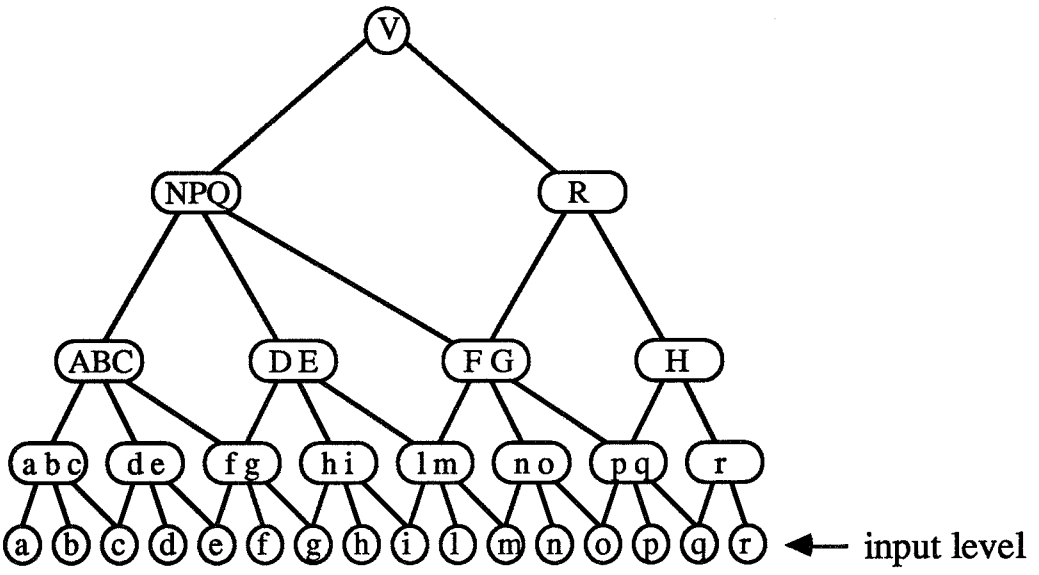

Figure 3.b

By lemma 1 we have that $\mathcal{L}$((**m**,**n**)t-SYTA) ⊆ $\mathcal{L}$((**m**+**1**,**n**)t-SYTA). To prove $\mathcal{L}$((**1**,**n**)t-SYTA)⊆$\mathcal{L}$((**0**,**n**)t-SYTA) we have to show that for every (**1**,**n**)t-SYTA A there exists an equivalent (**0**,**n**)t-SYTA A'.

The simulation of a node A(i,j) cannot be done in A' by a node in the level i because of the lack in A' of some of the edges of A. Instead such simulation is possible at the level i-1. In particular, the node A'(i-1,1) can do the same computations done by the nodes A(i,1), ..., A(i,n+t), the node A'(i-1,j), for $1<j<t^{i-1}$, can do the same computations done by the nodes A(i,(j-1)·t+n+1), ..., A(i,j·t+n), and finally the node A'(i-1,t^{i-1}) can do the same computations done by A(i,t^i+n-t+1), ...,A(i,t^i). Moreover the root processor A'(1,0) can perform the computations of A(1,1), ...,A(1,t) and, on the results of these, the computation of A(0,1), the whole in one only step. Now it is not difficult to construct an automaton A' whose behaviour is the one described above.

The proof that $\mathcal{L}$((**m+1**,**n**)t-SYTA)$\subseteq\mathcal{L}$((**m**,**n**)t-SYTA) for m>0 is analogous; (actually in this case the condition m>0 allows a simpler simulation where the node A'(i,j), i>0, performs the same computations of the nodes A(i+1,(j-1)·t+1), ..., A(i+1,j·t), i.e. the nodes in A in the same positions of its non-adoptive sons). ♦

In fig.3 we show a simulation of a (1,1)2-SYTA by a (0,1)2-SYTA following the proof of theorem 2. Dummy sons are not represented.

We call $\mathcal{LtSYTA}$ the unique class of languages obtained by theorem 2.

Since from lemma 1 and theorem 2 it can be deduced that $\mathcal{L}$((0,m)t-SYTA) $\subseteq\mathcal{L}$((**0**,**m**)t-SYTA)$\subseteq\mathcal{L}$((0,1)t-SYTA) it is interesting to know whether the inverse inclusion holds true, possibly under some conditions.

Lemma 2. If t≥2, then $\mathcal{L}$((0,1)t-SYTA) $\subseteq\mathcal{L}$((0,m)t-SYTA) if m≡1 (mod t).

Proof. Given an (0,1)t-SYTA A we outline how to construct an equivalent nonhomogenous (0,m)t-SYTA A' when m≡1(mod t). It is sufficient to take three labels, namely a for the leftmost nonadoptive son of a node, b for the other sons, c for the root. Take n=min{u∈N | t^u>m}. Suppose that a word is given as input to the nodes in $level_T$(k). A node A'(i,j) is required to perform the two following computations at the same time:

i) if k-n≤i<k, A'(i,j) concatenates the inputs it receives from its nonadoptive sons, otherwise, for 0<i<k-n, it performs the computations of the nodes A(i+n,t^n·(j-1)+1),..., A(i+n,t^n), which are the t^n nodes in the subtree rooted in A(i,j) and whose distance to A(i,j) is n.

ii) if 0<i<k, A'(i,j) performs also the computation of A(i+1, t·(j-1)+1), which is the leftmost nonadoptive son of A(i,j).

Exploiting the computation (i) of its nonadoptive sons, A'(i,j) can execute (ii); the result of such computation is sent in one step to A'(i-1,l(j-m)/t^nl) and from here it goes to A'(i-n,l(j-1)/t^nl) following n-1 edges of the tree; A'(i-n,l(j-1)/t^nl) uses such information, together with the computation (i) of its non-adoptive sons, to execute (i).

Eventually, the root processor performs the last 2n-2 computation steps of A in one step. ♦

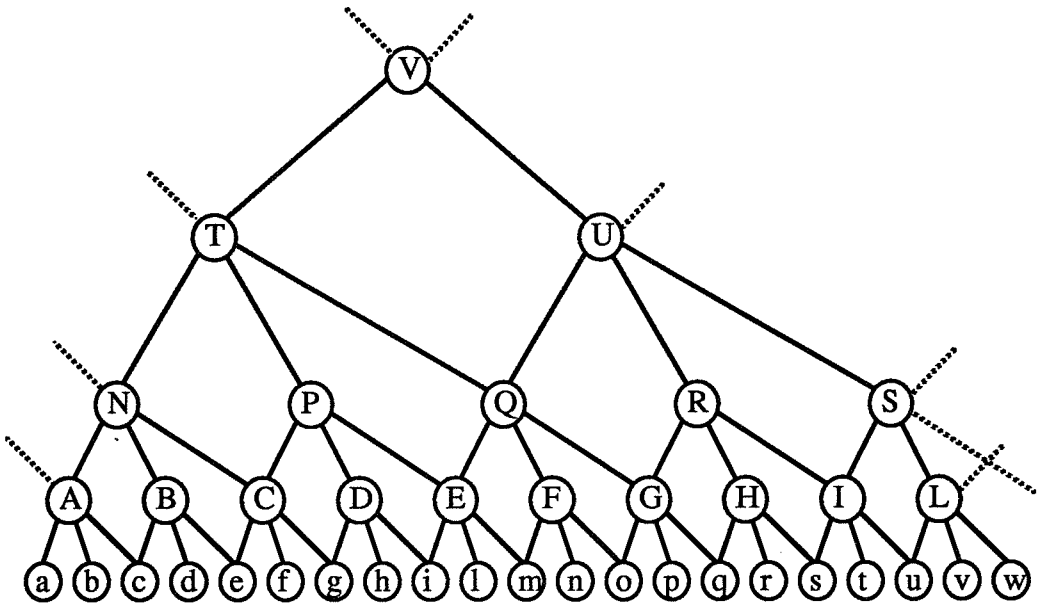

Figure 4.a

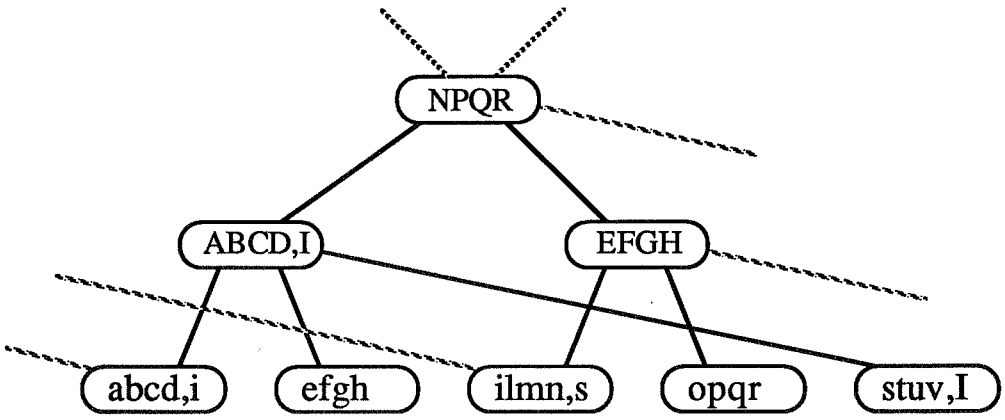

Figure 4.b

In fig.4 we represent the simulation required for the proof of lemma 2 in the case t=2, m=3, and therefore n=2. In fig.4a we show the computation of a piece of a (0,1)2-SYTA between the input level k and the level k-4. In fig.4b we represent the nodes of a (0,3)2-SYTA simulating the (0,1)2-SYTA which are in the same positions of the nodes producing as output V,T,U,N,P,Q,R,S. Inside the circles we have written only the portion of the output of the nodes which is exploited by the node on the top to simulate the nodes N,P,Q,R. In the level k-2 the letter I obtained from the fifth node is an example of computation (ii) in the proof of lemma 2.

The next lemma shows that if t>3, the condition in lemma 2 is also necessary. This does not seem to be true in the case t=2; for instance it seems that $\mathcal{L}$((0,1)2-SYTA) =$\mathcal{L}$((0,2)2-SYTA). Anyway, it is not clear yet how in general the computational power of (0,m)t-SYTA varies when m is not congruent to 1 modulo t.

Lemma 3. If $m \in N$ and $m \not\equiv 1 \pmod t$, $t>2$, then there exists a language $L \in \mathcal{L}$((0,1)t-SYTA)–$\mathcal{L}$((0,m)t-SYTA).

Proof. We consider first the case $m \not\equiv 0 \pmod t$.

Let us take the automaton $A=(\bar{T},I,Q,Q',f)$ where $\bar{T}$ is the (0,1)t-Y-tree, I={a,b,#}, Q={a,b,#,@}, Q'={a,b} and f is defined as follows:

$f(x_1,...,x_{t+1})=x_t$ if $(x_2=x_{t+1} \vee x_{t+1}=\#) \wedge (x_1,...,x_{t+1}=@$,

$f(x_1,...,x_{t+1})=@$ otherwise.

Roughly speaking, the output of a node K is the input symbol given to the rightmost node of the subtree rooted in K in the input level iff the states entered by the second and the (t+1)-th sons of K, K' and K" are equal (that is the input symbols in the rightmost node of the subtrees rooted in K' and K" are equal) and the same conditions hold for all the sons of K.

We prove that $\mathcal{L}(A) \notin \mathcal{L}$((0,m)t-SYTA). Let us suppose that $A' \in \mathcal{L}$((0,m)t-SYTA) exists with $\mathcal{L}(A)=\mathcal{L}(A')$. Let q be the number of its states. We consider an input $w \in \mathcal{L}(A)$ with $|w|=t^{\lambda}$ and $2^{\lambda-1}>q$. We write $w=w_1 \dots w_t$ for $|w_1|= \dots =|w_t|=t^{\lambda-1}$. The letter which in w_2 is in the position t^i, $0 \le i \le \lambda-2$, must be equal to the letter which in w_1 is in position $t^{\lambda-1}-t^i\cdot(t-2)$.

We define w_1^v (resp. w_2^v) the word obtained from w_1 (resp. w_2) by replacing the letter $t^{\lambda-1}-t^i$ of w_1 (resp. the letter t^i of w_2), $0 \le i \le \lambda-2$, by the i-th letter of v. We have $2^{\lambda-1}$ different choices for v. Since $2^{\lambda-1}>q$, there are at least two

different choices, namely v' and v", such that the inputs $w'=w_1^{v'} w_1^{v''} w_3 w_t$ and $w''=w_1^{v''} w_2^{v''} w_3 w_t$ produce the same output S on A'(1,1). But the value computed by A'(1,1) never depends on the letters of w_2 in position t^i, $0 \le i \le \lambda - 2$, because such letters enter nodes of A' which are not descendant of A'(1,1). Hence S is given as output by A'(1,1) also for the input $\bar{w}= w_1^{v'} w_2^{v''} w_3 w_t$, which means that $\bar{w}$ is accepted by A'. But $\bar{w} \notin \mathcal{L}(A)$, against the hypothesis on A'.

In the case $m \equiv 0 \pmod t$, it is not true that the letters of w_2 in position t^i, $0 \le i \le \lambda-2$, enter nodes which are not descendant of A'(1,1); anyway we get the same result interchanging the roles of 2 and t in the definition of the function f and in the subsequent proof. ♦

From lemmas 2 and 3 we have the following theorem.

Theorem 3. For $t>2$ $\mathcal{L}$((0,m)t-SYTA) $=\mathcal{LtSYTA}$ iff $m \equiv 1 \pmod t$.

Symmetrically, we can prove also the following theorem.

Theorem 4. For $t>2$ $\mathcal{L}$((m,0)t-SYTA)$=\mathcal{LtSYTA}$ iff $m \equiv 1 \pmod t$.

From [9], [16] we know that for any $b \in B^t$ and for any t-STA there exists an equivalent T(b)-STA equivalent to the t-STA. The following lemma allows to generalize the result to SYTA.

Lemma 4 For every $t \ge 2$ and $b \in B^t$ it holds $\mathcal{L}$((0,1)t-SYTA) $\subseteq \mathcal{L}$((0,1)T(b)-SYTA).

Proof. Take a $t \ge 2$ and a base $b \in B^t$.Consider a new label a not appearing among the labels of b and let b' be the base obtained from b by replacing with a the label of the rightmost node in the level hgt(b)-1. Consider now the T(b')-Y-tree T' where the function h assigning the adoptive sons is defined by h(a) $=(\varnothing,\{1\})$ and $h(a')=(\varnothing,\varnothing)$ for every a' different from a. We will prove that for every homogeneous (0,1)t-SYTA $A=(\bar{T},I,Q,Q',f)$ there exists an equivalent SYTA A' whose underlying Y-tree is T'. This would be enough for proving the thesis because from lemma 1 and theorem 1 we have $\mathcal{L}(A') \subseteq \mathcal{L}$((0,1)T(b)-SYTA).

The output of a node K in A' is a sequence of states of A obtained as follows:

i. if K has no adoptive sons, it concatenates the outputs given by its sons; otherwise (i.e. if the label of K is a) it concatenates the outputs of its non-adoptive sons together with the first symbol of the sequence of states given by its adoptive son. In both cases, let us call w the sequence of states obtained;

ii. if $w=w_1 \dots w_{d\cdot t+1}$, where w_i is a state of A, for $1\le i \le d\cdot t+1$, $d \ge 1$, and K is in a level $c\cdot h(b)$, $c \ge 0$, then K performs a simulation step of A. In particular, K gives as output $q_1 \dots q_d$, where $q_p=f(w_{(p-1)\cdot t+1} \dots w_{p\cdot t+1})$, $1\le p \le d$.

It is easy to verify that if $K=A'(hgt(b)\cdot i,j)$ and the input level is $c\cdot hgt(b)+c'$, $0\le c'<hgt(b)$, then it holds that:

i. d is the number of nodes of the level c' of the base b,

ii. if c'=0, K gives as output $O_A(i,j)$,

iii. otherwise it holds $q_p= O_A(i+1,(j-1)\cdot d+p)$, $1\le p \le d$.

Therefore the output z of the root of A' is a final state iff

i. either $|z|=1$ and z is a final state of A, or

ii. $|z|=r$, $1< r < t$, and $f(z\#^{t-r})$ is a final state of A. ♦

The following lemma can be proved analogously.

Lemma 5. For every $t\ge 2$ and $b\in B^t$ it holds $\mathcal{L}((1,0)t\text{-SYTA}) \subseteq \mathcal{L}((1,0)T(b)\text{-SYTA})$. ♦

While $\mathcal{L}(T(b)\text{-STA}) \subseteq \mathcal{L}(t\text{-STA})$, $b\in B^t$, in general is not true ([9], [16]) in the Y-trees the corresponding inclusion does hold. The foolowing lemma is crucial in the proof of this (Theorem 7).

Lemma 6. For every T(b)-SYTA on a $(\mathbf{m},\mathbf{n})$T(b)-Y-trees, $b\in B^t$, there exist $p,q\ge 0$ such that $\mathcal{L}(T(b)\text{-SYTA}) \subseteq \mathcal{L}((\mathbf{p},\mathbf{q})t\text{-SYTA})$.

Proof. Let us concentrate our attention on the right adoptive sons, forgetting the left ones in order to simplify the proof. The argument we use could be used for the case of the left adoptive sons, and the arguments for the two cases could be easily combined. Given $b\in B^t$ and a $(\mathbf{0},\mathbf{n})$T(b)-SYTA A, a nonhomogeneous $(\mathbf{0},\mathbf{q})$t-SYTA A' equivalent to A can be constructed along the following line. Let $n_{c'}$ be the number of nodes in $level_b(c')$ for $0\le c'<hgt(b)$. Now we consider the nodes in $level_A(hgt(b)\cdot i)$ as collected in t^{i-1} ordered groups of t consecutive nodes. Let G(i,j) be the j-th group of nodes in $level_A(hgt(b)\cdot i)$, that is the group consisting of the nodes $A(hgt(b)\cdot i,t\cdot(j-1)+1),\dots, A(hgt(b)\cdot i,t\cdot j)$.

Correspondingly, we consider the first $n_{c'} \cdot t^{i-1}$ nodes in $level_{A'}(i)$ as collected in t^{i-1} ordered groups of $n_{c'}$ consecutive nodes. Let H(i,j) be the j-th group of nodes in $level_{A'}(i)$, that is the group consisting of the nodes A'(i,(j-1·$n_{c'}$+1), ..., A'(i,j·$n_{c'}$). We call $G_k(i,j)$ ($H_k(i,j)$) the k-th element in G(i,j) (H(i,j)), for $1 \leq k \leq t$ ($1 \leq k \leq n_{c'}$). Now suppose that a word w is given as input to A and let hgt(b)·i+c', $0 \leq c' \leq hgt(b)$, $i \geq 0$, be the level of the nodes receiving such input. Then the word w is given as input to A' in the level i+1. Any processor of A' cannot decide to which level among hgt(b)·i, ..., hgt(b)·i+hgt(b)-1, w is given as input to A. Then the computations corresponding to all the possible cases are performed by A' simultaneously. The processors in H(i,j) simulate the computations done by the processors in G(i,j) as follows:

i. $O_{A'}(H_1(i,j),w)$ contains $(O_A(G_1(i,j),w),..., O_A(G_{tn_{c'}+1}(i,j),w)$,

ii.$O_{A'}(H_k(i,j),w)$ contains $O_A(G_{t-n_{c'}+k}(i,j),w)$, for $2 \leq k \leq n_{c'}$.

If $q \geq t^2+n$ one may be sure that the above computations are possible.

The root processor, which is also different from the others, selects the right result which depends on the number s, of processors in $level_{A'}(1)$ receiving an input not in {#}*: the right input level is c' if $n_{c'-1} < s \leq n_{c'}$. ♦

Theorem 5. For any (**m**,**n**)T(b)-Y-tree, b$\in B^t$, m+n>0, $\mathcal{L}$((**m**,**n**)T(b)-SYTA) =$\mathcal{LtSYTA}$.

Proof. By theorem 2 and lemmas 4 and 1, $\mathcal{LSYtTA}$=$\mathcal{L}$((0,1)t-SYTA)$\subseteq$ $\mathcal{L}$((0,1)T(b)-SYTA) $\subseteq$ $\mathcal{L}$((**m**,**n**)T(b)-SYTA). On the other hand, by lemma 6 there exist p,q>0 such that $\mathcal{L}$((**m**,**n**)T(b)-SYTA)$\subseteq$$\mathcal{L}$((**p**,**q**)t-SYTA) and, therefore, by theorem 2, the thesis holds. ♦

By exploiting theorems 2 and 5 the following theorem can be easily proved.

Theorem 6. Given c≥2, p,q≥1, $t_1=c^p$,$t_2=c^q$, it holds that $\mathcal{LsSYTA}$= $\mathcal{LtSYTA}$.

Theorem 7. For every T(b)-Y-tree $\overline{T(b)}$, $\mathcal{L}$($\overline{T(b)}$ -SYTA)$\subseteq$$\mathcal{LtSYTA}$.

Proof. Given a T(b)-Y-tree $\overline{T(b)}$=(T(b),P,h), by lemma 1 $\mathcal{L}$($\overline{T(b)}$-SYTA)$\subseteq$$\mathcal{L}$((**m**,**n**)T(b)-SYTA), where m is the maximum number occurring in h(a), for every a$\in$P. Then the result follows from theorem 5. ♦

4. Simulation of SYTA by systolic trellis automata.

Let us call SRMTA the class of systolic trellis automata which are both regular and modular. In this section - which is very much based on [7] - we show that $\mathcal{L}t\mathcal{SYTA} \subseteq \mathcal{L}$(SRMTA). Actually we will only outline the proof for the case t=2. Both a more detailed proof of this and its generalization to any t can easily be obtained adapting the schema of the proof of $\mathcal{L}$(t-STA)$\subseteq\mathcal{L}$(SRMTA) in [7].

An infinite *trellis* (trellis for the rest of the paper) is an infinite directed graph which satisfies the following conditions:

1) there is exactly one node (called the root) without outgoing edges,

2) every node K, a father, has three ingoing edges from three nodes, the left son L(K), the middle son M(K) and the right son R(K), such that R(L(K))= M(K)=L(R(K)).

A *labeled trellis* T is a trellis whose nodes are labeled by symbols from a finite alphabet Δ - called the label alphabet.

If T is a labeled trellis and $1\leq j\leq i$, then $N_T(i,j)$ will denote the j-th node (the enumeration is from left to right starting with one) in the i-th level of nodes (the enumeration is from the root starting with one) and $\lambda_T(i,j)$ its label. For any $1\leq j\leq i$, the sequence of nodes $N_T(i,j)$, $N_T(i-1,j)$, $N_T(i-2,j)$, ..., $N_T(j,j)$ is called the left-to-right diagonal (starting at the node $N_T(i,j)$) and the sequence $N_T(i,j)$, $N_T(i-1,j-1)$, $N_T(i-2,j-2)$, ..., $N_T(i-j+1,1)$ is called the right-to-left diagonal (starting at the node $N_T(i,j)$).

A trellis T is said to be *regular* if the label of any node is uniquely determined by the labels of its fathers (sometimes in the literature the trellises with such property are called *top-down deterministic* and a regular trellis is then defined as a trellis obtained from a top-down deterministic trellis T by means of a relabeling c, i.e. a function from the label alphabet Δ of T to some "new" label alphabet Δ'; anyway such two notions of regularity are equivalent in term of computational power ([3]).

A labeled trellis T is said to be *strictly (p,q)-modular* if there is a morphism ϕ that maps Δ into two-dimensional rectangular words of size p×q (they can be viewed as mappings of $\{1,2, ..., p\} \times \{1,2, ..., q\}$ into Δ) such that $\phi(\lambda_T(1,1))(1,1) = \lambda_T(1,1)$ and $T= \lim_{i\to\infty}\phi^i\lambda_T(1,1))$, where the symbol ϕ is used also to denote a natural extension of ϕ to map arbitrary two-dimensional words into two dimensional words. The trellis T is said to be *(p,q)-modular* if T=c(T')

for some strictly (p,q)-modular trellis T' and a coding c. A trellis is said to be *modular* if it is (p,q)-modular for some p,q.

A *regular and modular systolic trellis automaton* (SRMTA) is a construct $H=(T,\Delta, \Sigma,\Gamma,\Gamma_o,g,h)$ where T is a regular and modular trellis labeled by symbols from Δ, and Σ, Γ, $\Gamma_o \subseteq \Gamma$ are finite alphabets of terminal, operating and accepting symbols, $g:\Delta\times\Sigma\to\Gamma$ is the input function, $h:\Delta\times\Gamma\times\Gamma\times\Gamma\to\Gamma$ is the transition function (actually g and h specify a certain collection of input and transition functions on Σ and $\Gamma\times\Gamma\times\Gamma$, respectively, one for each $\delta\in\Delta$).

In order to define a flow of computation on H, we extend the functions h and g as follows. If $w=w_1...w_n$ for $w_i\in\Sigma$ and $\lambda_1, ..., \lambda_n$ are the labels of the nodes, from left to right, at the n-th level of T, then $g(w)=g(\lambda_1,w_1)g(\lambda_2,w_2)...g(\lambda_n,w_n)$. If $w=w_1...w_n$ and $z=z_1...z_{n+1}$, for $w_i,z_i\in\Sigma$ and $\lambda_1, ..., \lambda_n$ are the labels of the nodes at the (n-1)-th level of T, then $h(w,z)=h(\lambda_1,w_1,z_2,w_2)h(\lambda_2,w_2,z_3,w_3)...h(\lambda_{n-1},w_{n-1},z_n,w_n)$.

If $w\in\Sigma^+$, then $k_0(w)=g(w)$, $k_1(w)=h(k_0(w),\#^{(|w|+1)})$, $k_2(w)=h(k_1(w),k_0(w))$, $k_3(w)=h(k_2(w),k_1(w))$, ..., $k_{|w|}(w)=h(k_{|w|-1}(w),k_{|w|-2}(w))$ is the sequence of output words, level by level, when w is processed on H ; $\mathcal{L}(H)=\{w|w\in\Sigma^+, k_{|w|}(w)\in\Gamma_o\}$ is the *language accepted* by H.

Theorem 8. $\mathcal{L}((0,1)2\text{-SYTA}) \subseteq \mathcal{L}(\text{SRTA})$.

Proof. Consider the top-down deterministic trellis R' defined as follows:

i. the label alphabet of R' is {a,b},

ii. the nodes of the rightmost and leftmost paths of R' are labeled by a,

iii. a node R'(i,j), $1\le i$, $1<j<i$, is labeled by b if its fathers on the level i-1 have the same label, otherwise it is labeled by a.

The constructed trellis has the property that the nodes in a level i are all labeled by a iff $i=2^j$ for some $j\ge 0$.

The regular and modular trellis R we need here is obtained from R' by relabeling with a new letter c the nodes of the rightmost path of R'.

Let A be a (0,1)t-SYTA; we will outline the construction of a SRMTA A' equivalent to A.

We shall divide the rest of the proof into two cases. Let w be the input word to be processed. Firstly, we show how A' can simulate the computation of A if $w=2^j$. Then we shall consider the remaining possible lengths for w.

For a node M in A we shall call A'(M) the node in A' which simulates M. We will denote w(i,j) the substring of w which enters the subtree rooted in A(i,j), $1\le i\le l$. We will call a-nodes, b-nodes, ... nodes labeled by a, b, ...

Case 1: There exists $\lambda\ge 0$ s.t. $|w|=2^\lambda$.

The computation of A' implements the "strategy" given by the following three rules:

1) A'(A(i,j)) is located as follows: A'(A(i,j)) is in the same left to right diagonal of the node of A' which receives as input the first letter of w(i,j); the level of A'(A(i,j)) is $\lambda-2^{\lambda-i+1}+2$ if $j<i$, $\lambda-2^{\lambda-i}+1$ if $j=i$.

2) A node K=A'(A(i,j)) sends $O_A(i,j)$:

- along the left to right diagonal (except when K is a c-node);
- along the right to left diagonal if K is a c-node or if K is an a-node with a b-node as left father (therefore in the latter case $O_A(i,j)$ is sent along both the diagonals).

Now $O_A(i,j)$ follows this (these) diagonal(s) until the condition for the application of rule 3 below is satisfied.

3) When a state S of Q propagating along the left to right diagonal and a t-uple $\bar{S}$ of states of A, t=1,2 propagating along the right to left diagonal, reach the same node K:

- if t=1, K collects S and $\bar{S}$ into a pair and sends it along the right to left diagonal;
- if t=2, K computes $f(S,\bar{S}_1,\bar{S}_2)$, where $\bar{S}_i$, i=1,2 is the i-th component of $\bar{S}$.

With reference to the SYTA of fig. 5.a, fig.5.b shows an example of an "ideal" computation for A' according to these rules (in practice the implementation of such strategy would introduce some more complications). The input is abcdefgh and has length 2^3. In the trellis the nodes performing the simulation steps and the paths followed by the states of the (0,1)2-SYTA so obtained, have been evidenced.

Case 2: There is no $\lambda \geq 0$ s.t. $|w|=2^\lambda$.

We write $w=w_1 \ldots w_r$, $|w_i|=2t^{n_i}$, $n_i > n_{i+1}$, $1 \leq i \leq r$.

First, we introduce the equilateral triangles T_i, B_i and Q_i. The sequence of equilateral triangles T_i, $1 \leq i \leq r$, each of side $|w_i|$, cover, in the given order, the first |w| nodes of the rightmost path of T (see fig. 6). The nodes on the sides of these triangles are labeled by a and c. The triangle B_i, $1 \leq i < r$, shares the base with T_i, has all the remaining nodes labeled b and its vertex is in a level greater or equal with respect to the input level. The base of the triangle Q_i, $1 \leq i < r$, is determined by the input nodes not belonging to B_i and receiving letters of w_i: its vertex is the left son of the leftmost node in the base of T_i .

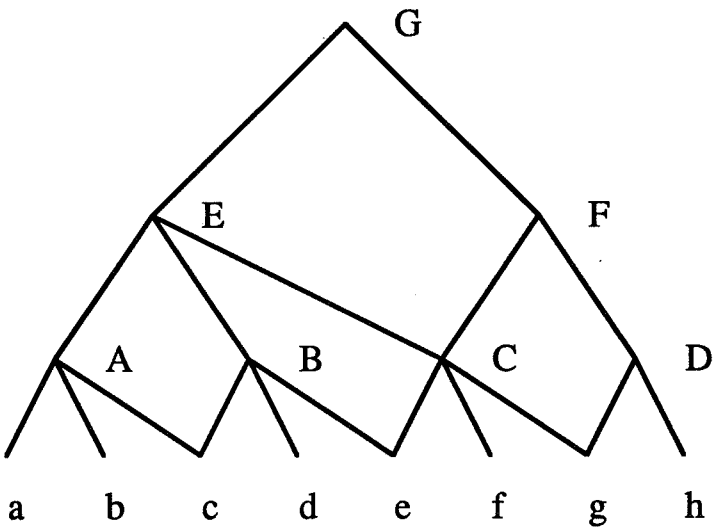

Figure 5.a

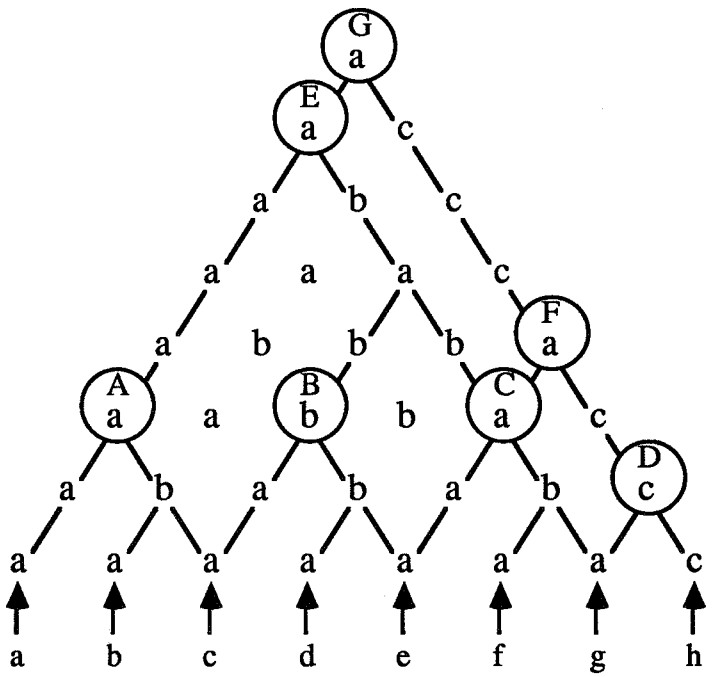

Figure 5.b

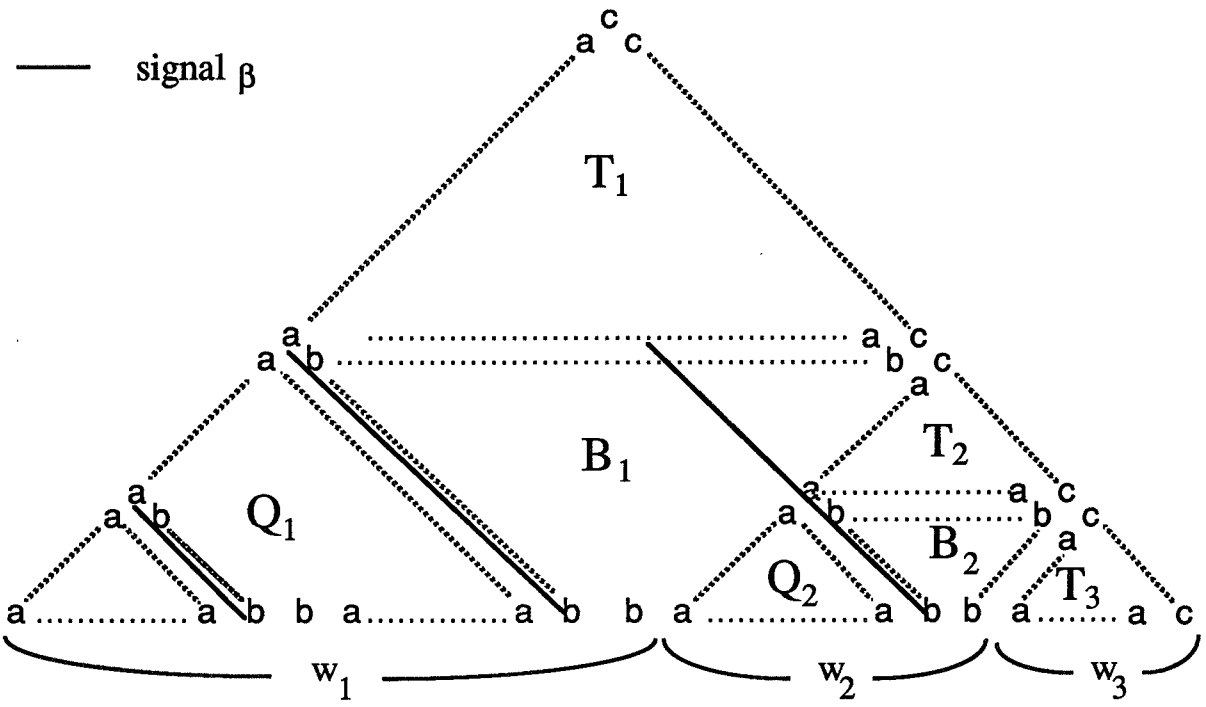

Figure 6

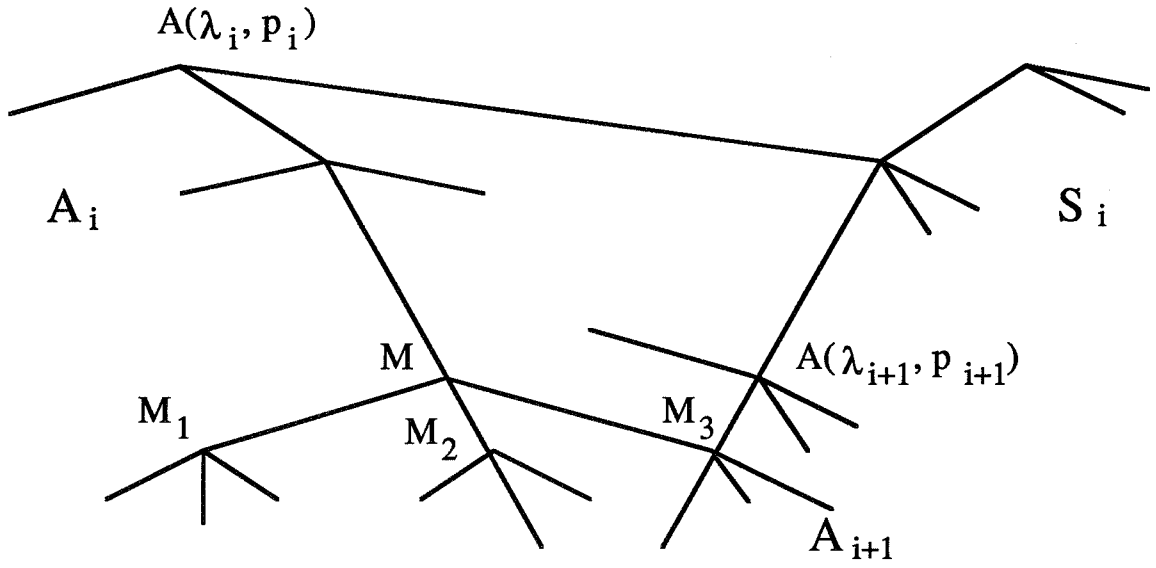

Figure 7

Let A_i be the subtree in the underlying (0,1)2-Y-tree of H which receives w_i as input and $A(\lambda_i,p_i)$ its root (note that then we have $w(\lambda_i,p_i)=w_i$).

The idea is that to bring the letters of w_i up to the base of T_i and to make the simulation of A_i on T_i.

The letters of w_i entering nodes of B_i reach the base of T_i by imposing that an input b-node just makes its input to be transmitted along the left to right diagonal through b-nodes until an a-node or a c-node is reached.

Now we consider the letters of w_i entering nodes in Q_i. During the computation of A' we cannot distinguish the nodes of T_i from the nodes of Q_i, the former having to compute the simulation of the nodes in A_i (we shall refer to this kind of computation as to simulating computation), the latter having to transmit the input letters along the left to right diagonals (we shall refer to this kind of computation as to input transmission). Therefore a node in Q_i or T_i performs both the computations at the same time.

We use the signal β to prevent the simulating computation of Q_i from affecting the nodes which do not belong to Q_i. A signal β starts in an input b-node whose brother on the left is an a-node, flows along the right to left diagonal and ends in the second a-node that meets.

The simulating computation of Q_i is stopped by the signal β running through the leftside of B_i. An input transmission passes signal β and will cause the beginning of a new simulating computation when an a-node is met.

Let us denote by S_i the subtree rooted in $A(\lambda_i,p_i+1)$ (see fig. 7) and by A_i^r the nodes of A_i on the rightmost path. Using the previous rules 1 (for the level of the base of T_i instead of λ), 2 and 3, it is possible to simulate the nodes of A_i-A_i^r in T_i. Instead, the simulation of the nodes of A_i^r will be a bit more involved because their computation depends also on the nodes of the leftmost path of S_i, which we will denote by S^λ. Anyway we can still impose that the position on T_i of the nodes simulating those in A_i^r is given by rule1 (for the level of the base of T_i instead of λ). Let us prove how this is obtained.

We need a new signal, which we call signal η (see fig. 8). A signal η starts from a node N iff all the following conditions hold:

- N is a b- node
- its left son is a b- node
- its right son is an a- node
- the right son is an input node or is reached by a signal β.

The signal η starting from N runs along the vertical links until:

- a c-node is reached
- a β signal is met.

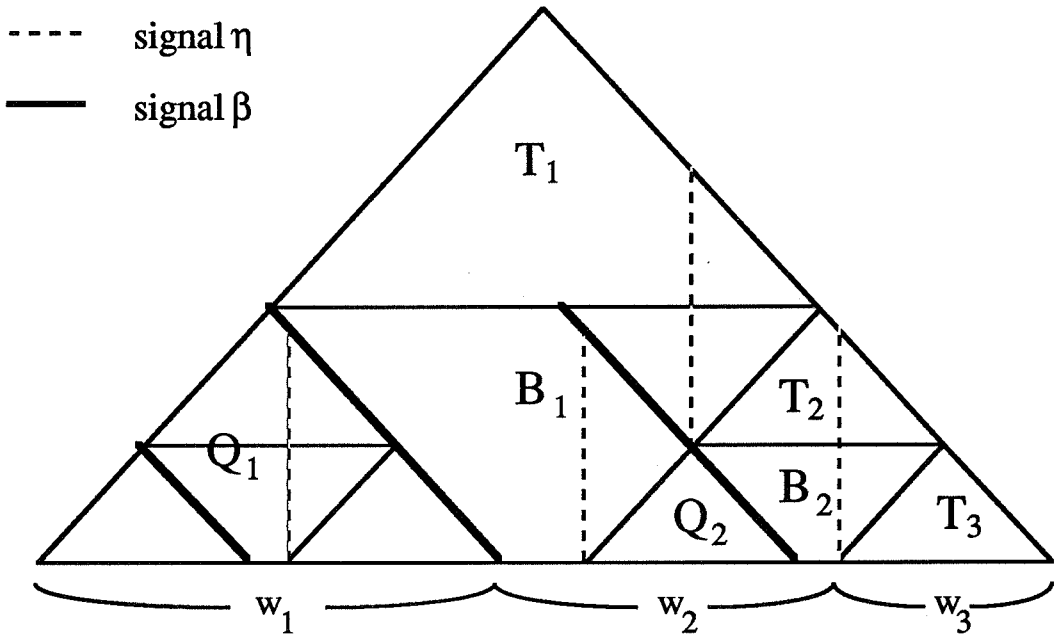

Figure 8

In fig.8 there are two η signals which are stopped by β signals, whereas two other η signals reach the rightmost path of the trellis. Let $M \in A_i^r$, and M_1, M_2, M_3 be its first, second and third son, respectively. $A'(M_1)$ is the left son of $A'(M)$; $A'(M_2)$ is a c-node descendant of $A'(M)$. Therefore it is possible to transmit $O_A(M_1)$ and $O_A(M_2)$ to $A'(M)$.

As regards $O_A(M_3)$, we distinguish three cases:

i) $M_3 \in (S_i^\lambda \cap A_{i+1} - A(\lambda_{i+1}, p_{i+1}))$ (see fig.7) : $A'(M_3)$ is an a-node sending $O_A(M_3)$ also along the right to left diagonal (by the rule 2). When $O_A(M_3)$ meets an η signal (it will be the one starting from the left vertex of the base of T_{i+1}), it changes direction and reaches the right son of $A'(M)$ through the left to right diagonal.

ii) $M_3 = A(\lambda_{i+1}, p_{i+1})$: the difference is that $A'(M_3)$ is a c-node and $O_A(M_3)$, through the c- nodes, joins the η signal in $A'(M_2)$.

iii) $M_3 \in (S_i^\lambda - A_{i+1})$: let $U_i \in A_i^r$ be in the same level as $A(\lambda_{i+1}, p_{i+1})$; $A'(U_i)$ is identified as the c-node in T_i reached by an η signal. We impose that the behaviour of A' satisfies the following conditions:

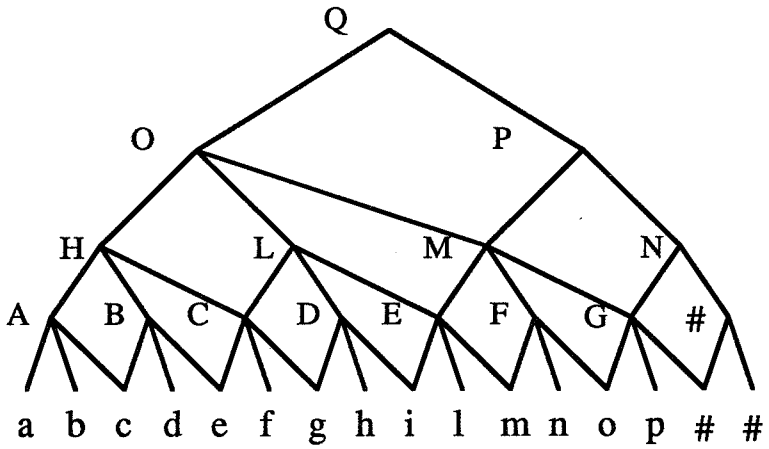

Figure 9.a

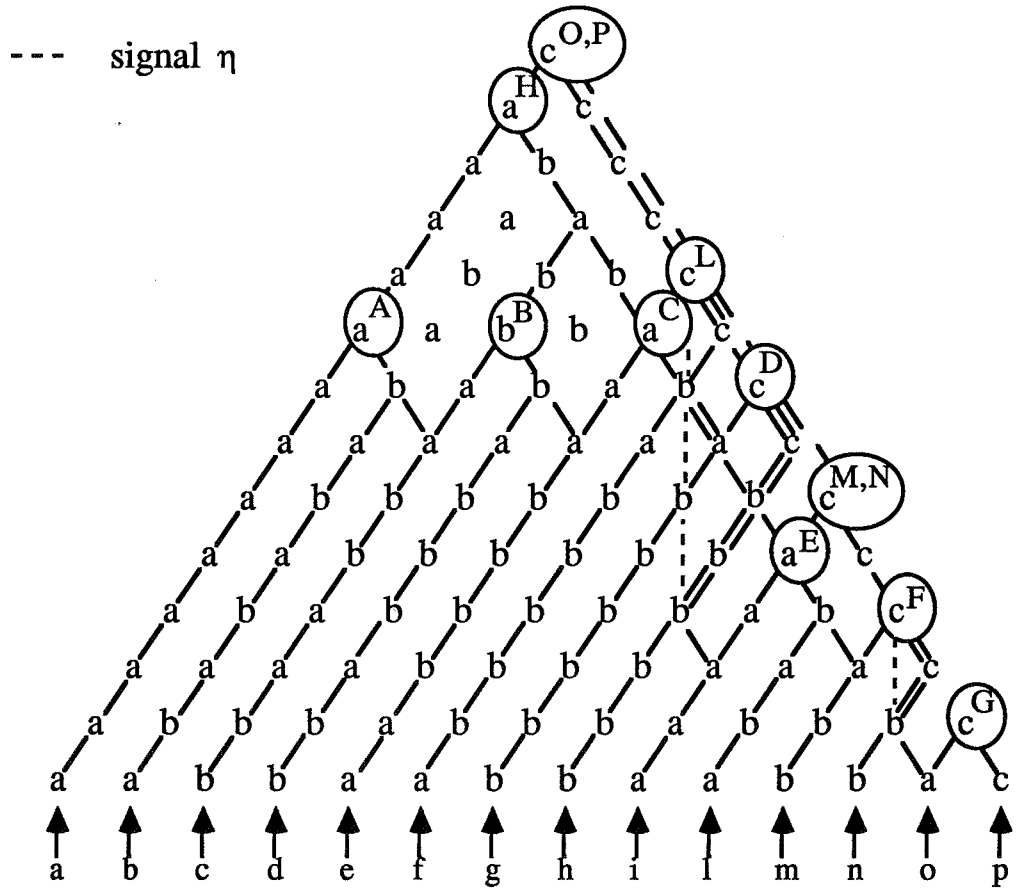

Figure 9.b

Condition 1) For $i<r$, let $V \in A_i^r$ be located between $A(\lambda_i,p_i)$ and U_i (U_i excluded). The node $A'(V)$ simulates also the node in S_i^{λ} in the same level of V;

Condition 2) For $i>1$, the states produced by the roots of A_i and S_i are transmitted along the c-nodes, from $A'(A(\lambda_i,p_i))$ up to $A'(U_{i-1})$.

Notice that Condition 2 makes Condition 1 satisfiable when V is the father of U_i (the only potentially difficult case). Condition 2 is satisfiable because the father of $A'(A(\lambda_i,p_i))$, namely Z, is identified by the left to right diagonal starting from the input level and ending in the left son of Z, which consists of nodes all labeled by b . Condition 1 obviously implies that $O_A(M_3)$ reaches A'(M).

Now we have that $O_A(0,1) \in O_{A'}(0,1)$ since $A'(A(\lambda_1,p_1))=A'(0,1)$.

Withe reference to the SYTA of fig. 9.a, fig. 9.b shows an example of a computation of A' for the case when the length of the input w is not a power of 2. If we take w_1=abcdefgh, w_2=ilmn, w_3=op, then it is the same situation of figures 6 and 8. Only the "correct" η signals (that is the ones that reach the rightmost path of the trellis) are represented. The β signals, the simulating computations of the triangles Q_1 and Q_2 and input transmissions of the triangles T_1, T_2 and T_3 are not shown (Q_1, Q_2, T_1, T_2 and T_3 are the triangles represented in figures 6 and 8). ♦

5. Conclusions.

In this paper we have studied Systolic Y-tree Automata (SYTA), a class of systolic automata where the communication structure is obtained by adding new edges, and therefore new sons, called adoptive sons, to the nodes of the underlying tree, according to some regularity condition. We have studied SYTA in the more specific case where the tree is t-ary or a tree with base. We have shown that for each $s \geq 0$ the set of classes of languages accepted by SYTA whose underlying tree is a tree with base with s leaves has a maximum, called $\mathcal{LsSYTA}$, and we have studied when $\mathcal{LsSYTA}$ is reached depending on number and position of the adoptive sons. We have proved that if s and t are powers of the same base, then $\mathcal{LsSYTA}=\mathcal{LtSYTA}$. We have also given a simulation of SYTA on regular and modular systolic trellis automata, strengthening the result of [7].

These first results of comparison among classes of SYTA suggest that both the position and the number of the adoptive sons influence the computational power of the considered automata. By fixing the underlying tree and varying position and number of adoptive sons one can obtain classes of automata which, with respect to the accepted languages, are equivalent, contained one in the other or incomparable. We think that the algebraic structure one obtains is a lattice, but we have not studied yet the problem. As regards the comparison results one may obtain by varying the underlying tree, we believe that the sufficient condition of equivalence of theorem 9 is also necessary. Most of the comparison problems of this type are still open for the classes of T(b)-SYTA languages strictly contained in the maximal one. In the maximal case we noticed that all the classes of nondeterministic systolic t-ary automata are equal, for $t \geq 2$. This may not be true for the nonmaximal cases. From this point of view the nondeterminism may offer several surprises.

Ackowledgement. We are grateful to Jozef Gruska for encouragement, suggestions and helpful discussions.

References

[1] K. Culik II, J. Gruska, A. Salomaa, *Systolic Automata for VLSI on Balanced Tree,* Acta Informatica 18, 335-344 (1983).

[2] K. Culik II, J. Gruska, A. Salomaa, *On a Family of L Languages Resulting from Systolic Tree Automata*, Theoretical Computer Science 23, 231-242 (1983).

[3] K. Culik II, J. Gruska, A. Salomaa, *Systolic Trellis Automata,* Part I and Part II, International Journal of Computer Mathematics 15, 195-212 (1984) and 16, 3-22 (1984) .

[4] K. Culik II, J. Gruska, A. Salomaa, *Systolic Trellis Automata: Stability, Decidability and Complexity*, Information and Control 71, 218-230 (1986).

[5] K. Culik II, A. Salomaa, D. Wood, *Systolic Tree Acceptors*, R.A.I.R.O. Informatique théorique 18, 53-69 (1984).

[6] E. Fachini, R. Francese, M. Napoli, D, Parente, *BC-tree Systolic Automata: Characterization and properties*, Journal of Computers and Artificial Intelligence 8, 53-82 (1989).

[7] E. Fachini, J. Gruska, A. Maggiolo-Schettini, D. Sangiorgi, *Simulation of Systolic Tree Automata on Trellis Automata,* International Journal of Foundations of Computer Science 1, 87-110 (1990).

[8] E. Fachini, J. Gruska, M. Napoli, D. Parente, *Power of Interconnection and of Nondeterminism in Regular Y-Tree Systolic Automata*, in preparation.

[9] E. Fachini, A. Maggiolo-Schettini, G.Resta, D.Sangiorgi, *Some Structural Properties of Systolic Tree Automata*, Fundamenta Informaticae XII, 571-585 (1989).

[10] E. Fachini, A. Maggiolo Schettini, D. Sangiorgi, *Comparisons among classes of Y-tree Systolic Automata.* In: "Proceedings of MCFS '90" (Lecture Notes in Computer Science, vol. 452, pp. 254-260) Berlin, Heidelberg, New York: Springer 1990.

[11] E. Fachini, M. Napoli, *C-tree systolic Automata*, Theoretical Computer Science 56, 155-186 (1988).

[12] J. Gruska, *Systolic Automata: Power, Characterizations, Nonhomogenity*, in "Proceedings of MCFS '84" (Lecture Notes in Computer Science, vol. 176, pp. 32-49) Berlin, Heidelberg, New York: Springer 1984.

[14] J. Gruska, *Synthesis, Structure and Power of Systolic Computations,* Theoretical Computer Science 71, 47-78 (1990).

[15] H. T. Kung, *Why Systolic Architecture ?*, Computer Magazine 15, 37-46 (1982).

[16] A. Monti, D. Parente, *Comparison Results for Particular Systolic Tree Automata.* In: G. Ausiello, D.P. Bovet, R. Petreschi (eds.) Proc. First Italian Conference on Algorithms and Complexity, pp.172-187, Singapore, World Scientific Publisher 1990.