Acta Cybernetica **15** (2001) 75-100.

Logical definability of Y-tree and trellis systolic ω-languages*

Monti Angelo† and Peron Adriano‡

Abstract

In this paper we investigate the correspondence (in the style of the well known Büchi Theorem) between ω-languages accepted by systolic automata and suitable (proper) extensions of the Monadic Second Order theory of one successor ($MSO[<]$). To this purpose we extend Y-tree and trellis systolic automata to deal with ω-words and we study the expressiveness, closure and decidability properties of the two classes of ω-languages accepted by Y-tree and trellis automata, respectively. We define, then, two extensions of $MSO[<]$, $MSO[<, adj]$ and $MSO[<, 2x]$, which allow to express Y-tree ω-languages and trellis ω-languages, respectively.

1 Introduction

The subject of automata accepting infinite sequences was established in the sixties by Büchi, McNaughton and Rabin (for a survey, see [16, 17]). One motivation for considering automata on infinite sequences (Büchi automata) was the analysis of the sequential calculus ($MSO[<]$), a system of monadic second order logic for the formalisation of properties of sequences. Büchi showed that any condition on sequences that it is written in this calculus can be reformulated as a statement about acceptance of sequences by a Büchi automaton (Büchi Theorem). The resulting theory is fundamental for those areas in computer science where non-terminating computations are studied, for instance modal logics of programs and specification and verification of concurrent systems (e.g. see [15]). This paper is an attempt to set a correspondence between ω-languages accepted by systolic automata and suitable (proper) extensions of $MSO[<]$. Systolic automata (see [8] for a survey) are synchronous networks of (memoryless) processors working in discrete time. These automata have been the main theoretical models used to study various basic problems concerning systolic architectures, systems and computations. In this paper we

*Research partially supported by MURST Progetto Cofinanziato TOSCA.

†Dipartimento Scienze dell'Informazione, Università di Roma (La Sapienza), 00198, Via Salaria 113, Italy, e-mail: monti@dsi.uniroma1.it

‡Dipartimento di Matematica e Informatica, Università di Udine, 33100, Via Zanon 6, Italy, e-mail: peron@dimi.uniud.it

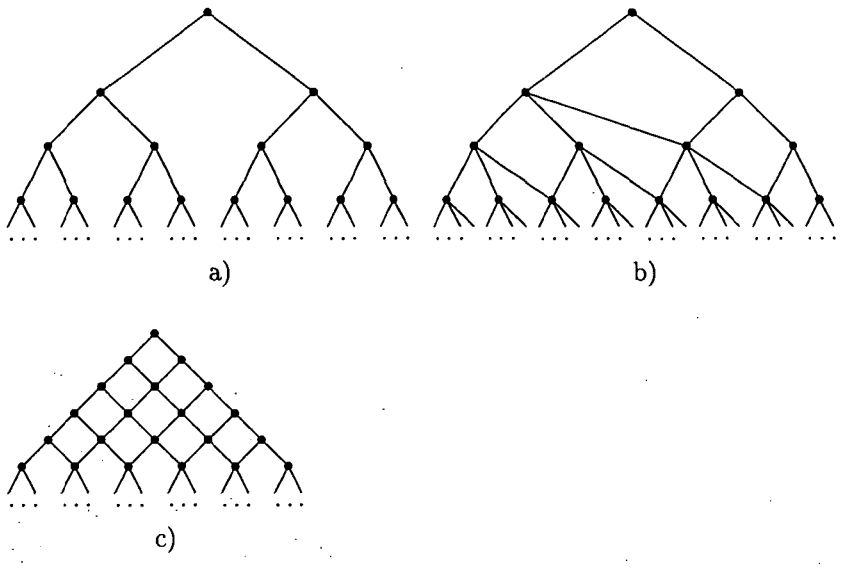

Figure 1: Interconnection structures for systolic automata: binary tree (a), Y-tree (b) and trellis (c).

consider systolic automata enforcing three of the most widely studied types of interconnection structures, i.e. leafless binary tree, leafless binary Y-tree and downward unbounded trellis (see figure 1). These automata have been mainly considered as acceptors of finitary languages. Only recently, binary tree systolic automata have been considered as acceptors of ω-languages in [12], showing that the class of *binary tree ω-languages* properly extends the class of ω-languages accepted by Büchi automata though preserving the same closure and decidability properties. In this paper we introduce also the class of *Y-tree ω-languages* (i.e. the class of ω-languages accepted by Y-tree), which extends the class of binary tree ω-languages, and the class of *trellis ω-languages* which, in its turn, extends the class of Y-tree ω-languages. The class of Y-tree ω-languages is closed under union and intersection, it is not closed under complement and the emptiness problem is undecidable. The class of trellis ω-languages is closed under union and intersection. The emptiness problem is undecidable. As well as in the finitary case (see [9]), the closure under complementation remains an open question. The logical characterisation of binary tree ω-languages has been established in [13], where it has been shown that such a class of languages corresponds (in the sense of Büchi theorem) to a proper extension of $MSO[<]$ by a suitable function $flip$ (the extension is denoted by $MSO[<, flip]$).

In this paper we propose two suitable extensions of $MSO[<]$, denoted by $EMSO[<, adj]$ (the fragment of $MSO[<, adj]$ allowing only existential quantification over predicates) and $MSO[<, 2x]$, which allow to express Y-tree and trellis ω-languages, respectively. Actually, we believe that $EMSO[<, adj]$ characterises the class of Y-tree ω-languages. The question, whether $MSO[<, 2x]$ is a char-

acterisation of ω-languages accepted by trellis automata, is an (hard) problem directly related to the open problem of the closure under complement of both finitary and ω-languages. The hierarchy of extensions of regular ω-languages accepted by Büchi automata and the corresponding hierarchy of extensions of $MSO[<]$ are summarised in figure 2.

We believe that investigating logical characterisations of systolic languages is meaningful for both theoretical and practical reasons. The results of the characterisation of binary tree ω-languages have been fruitfully applied in [10] for studying the decidability properties of logics for time granularity interpreted over ω-layered metric temporal structures (the traditional basic engine of $MSO[<]$ was, in that case, not powerful enough). We believe that the results presented in this paper could be fruitfully applied for formally studying and comparing the expressive power of meaningful extensions of the temporal logics for time granularity considered in [10]. Moreover, we expect that our investigation could contribute in an original way to the research oriented to developing systematic and sufficiently automatisable methods to synthesise systolic systems from high level specifications.

2 Preliminaries

Throughout this paper Σ denotes an alphabet; $\Sigma^\star$ denotes the set of (finite length) words on Σ, and Σ^ω denotes the set $\omega - words$ on Σ. Finite words are indicated by $u, v, w, \ldots$, ϵ is the empty word and letters $\alpha, \beta, \ldots$ are used for ω-words. The symbol $\cdot$ denotes concatenation on strings. For an ω-word α, $\alpha(i)$, with $i \in \mathbb{N}$, denotes the i-th element of α; $\alpha(m,n)$ denotes the subword $\alpha(m) \cdot \ldots \cdot \alpha(n)$ of α, and $\alpha(n,\omega)$ denotes the suffix $\alpha(n) \cdot \alpha(n+1) \cdot \ldots$ of α. For $V \subseteq \Sigma^\star$, V^ω is the set of ω-words having the form $v_0 \cdot v_1 \cdot v_2 \cdot \ldots$ with $v_i \in V$, for $i \in \mathbb{N}$.

2.1 Logical definability

Properties of words will be described by monadic second order logic formulas. Since logical formulas will be interpreted over words, it is more convenient to identify a word $\alpha \in \Sigma^\omega$ with the structure $\underline{\alpha} = \langle \mathbb{N}, <, (Q_a)_{a \in \Sigma} \rangle$, where $<$ is the usual ordering of natural numbers and $Q_a = \{i : \alpha(i) = a\}$.

In the sequel we shall consider a second order language over a set of (interpreted) binary relation symbols $B_1, \ldots, B_n$, denoted as $MSO[B_1, \ldots, B_n]$ (for notation we follow [17]), whose syntax is as follows.

Given an alphabet of *individual variables* $x, y, z, \ldots$, *predicates* $X, Y, Z, \ldots$ and predicate symbols Q_a (with $a \in \Sigma$),
Atomic formulas have the form $x = y$, xB_iy, $X(x)$ and $Q_a(x)$;
Formulas are built up from atomic formulas by means of the boolean connectives $\neg$, $\wedge$, $\vee$, $\Rightarrow$ and the quantifiers $\forall$ and $\exists$ ranging over both individual variables and predicates.

$$\begin{array}{ccc} \mathcal{L}_\omega(Trellis\ A.) & \subseteq & MSO[<, 2x] \\ \bigcup & & \bigcup | \\ \mathcal{L}_\omega(Y-Tree\ A.) & \subseteq & EMSO[<, adj] \\ \bigcup & & \bigcup \\ \mathcal{L}_\omega(B-Tree\ A.) & = & EMSO[<, flip] = MSO[<, flip] \\ \bigcup & & \bigcup \\ \mathcal{L}_\omega(Buchi\ A.) & = & EMSO[<] = MSO[<] \end{array}$$

Figure 2: The hierarchies of systolic ω-languages and of the related $MSO[<]$ extensions.

Individual variables are interpreted as elements of $\mathbb{N}$ (i.e., positions of an ω-word) and predicates as subsets of $\mathbb{N}$ (i.e., sets of positions of an ω-word). Atomic formulas $x = y$ and $X(x)$ are interpreted as equality between x and y, and $x \in X$. For a fixed interpretation of binary relational symbols B_i (as subset of $\mathbb{N} \times \mathbb{N}$), atomic formulas xB_iy, are interpreted as $\langle x, y\rangle \in B_i$. Given an ω-word α, $Q_a(x)$ is interpreted as $x \in \{i : \alpha(i) = a\}$. Boolean connectives and quantification operators are endowed with the standard semantics.

If $\phi(X_1, \ldots, X_n)$ is a formula with at most the predicates $X_1, \ldots, X_n$ occurring free in ϕ, α is a ω-word and $P_1, \ldots, P_n$ are subsets of $\mathbb{N}$, then we write $\langle \underline{\alpha}, P_1, \ldots, P_n\rangle \models \phi(X_1, \ldots, X_n)$ if α satisfies ϕ under the above mentioned interpretation taking P_i as interpretation of X_i. A formula without free variables and predicates is said to be a *sentence*. If ϕ is a sentence, for sake of simplicity, we write $\underline{\alpha} \models \phi$. The language $L(\phi)$ defined by a sentence ϕ is the set of all words $\alpha \in \Sigma^\omega$ such that $\underline{\alpha} \models \phi$. A language L is $MSO[B_1, \ldots, B_n]$-definable if there is a formula ϕ of $MSO[B_1, \ldots, B_n]$ such that $L = L(\phi)$; L is $EMSO[B_1, \ldots, B_n]$-definable if there is a sentence ϕ having the form $\exists X_1 \ldots \exists X_n \psi(X_1, \ldots, X_n)$ of $MSO[B_1, \ldots, B_n]$ such that $L = L(\phi)$ and ψ contains only first order quantifiers (i.e. quantifiers occurring in ψ range only over individual variables). In the following the binary relational symbol $<$ is interpreted as the standard ordering of natural numbers. In this setting Büchi Theorem can be formulated as follows.

Büchi Theorem An ω-language is regular iff it is $MSO[<]$- definable.

In the sequel, for simplicity, atomic formulas over relational symbols interpreted as functions will be written in functional form (i.e. $flip(x) = y$, $adj(x) = y$, $2x = y$ instead of $x\ flip\ y$, $x\ adj\ y$, $x\ 2\ y$).

3 Systolic computations

Let us recall how systolic automata process a word in the finitary case. Systolic automata are networks of (memoryless) processors (called also nodes in the following) working in discrete time. As far as systolic binary tree are concerned, the

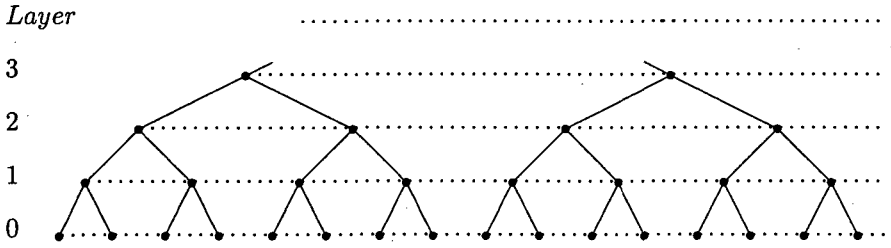

Figure 3: The upward unbounded binary tree structure.

interconnection structure of a binary tree automaton is an (infinite) leafless perfectly balanced binary tree (see figure 1.a). In order to process a word w, whose length $|w|$ is equal to 2^i, the i-th level of the tree is chosen (the level of the root is 0). Now, the automaton is fed in such a way that adjacent processors at level i-th are fed with adjacent symbols of w, and that the leftmost processor is fed with the first symbol of w. All the processors at level i-th synchronously output a symbol belonging to the *state alphabet* Q, according to the *input relation* of the automaton. Each processor at level $(i-1)$-th receives the two states output by its two sons (placed at level i) and it synchronously (with respect to processors at the same level) outputs a symbol belonging to Q according to the *transition relation* of the automaton. Therefore, information flows bottom-up, in parallel and synchronously, level by level. The result of the computation is collected at the root of the tree and the word w is accepted if the state associated with the root is a *final state*.

Let us consider now the case of ω-languages. Though the local topology of the interconnections of processors remains unchanged, the whole structure is generalised. In the finitary case the structure is a leafless binary tree, namely a downward unbounded multilevel structure (provided with infinitely many layers) having a bounded number of processors at each layer. On the contrary, the structures we shall considering are upward unbounded multilevel structures where each layer is a structure isomorphic to natural numbers (with the usual order). In this respect, an upward unbounded structure (see figure 3) can be represented as a grid of processors which is unbounded both in the horizontal dimension (processors in a layer) and vertical dimension (number of layers). In the following we shall mention the i-th column of the structure intending the set of i-th elements of each layer.

Nodes of a layer (with the exception of the 0-th layer) are connected to nodes of the previous layer drawing a regular topology: either binary tree like as in figure 3, or Y-tree like as in figure 6 or trellis like as in figure 13. In such a framework, the ω-word to be processed is associated with the 0-th layer of the upward unbounded structure, adjacent symbols being associated with adjacent nodes. The information flows up, in parallel and synchronously, level by level exactly as in the finitary case. Therefore, a systolic computation is a labelling (over the alphabet of states of the automaton) of the upward unbounded structure compatible with the transition

relation of the automaton. In order to accept/discard a computation on an ω-word, we impose a Büchi criterion on the labels of the 0-th column, namely the computation is successful if an infinite number of final states label that column. In the following we shall see how a computation on a ω-word α, defined here as a labelling of the whole structure, can be indeed defined incrementally step by step as the result of composing the results of finite computations on segments of finite length of α. Notice that the interconnection structure we are proposing allows one to easily simulate the systolic computation defined in the literature for the finitary case.

In the remaining part of this section we recall (from [13]) the definitions and results concerning systolic automata operating over an upward unbounded binary tree structure. The cases of Y-tree and trellis interconnection structures are introduced and discussed in the two next sections.

Definition 1 *A* binary tree automaton *is a tuple* $\mathcal{A} = \langle \Sigma, Q, Q_0, g, f \rangle$, *where*

- Q *is the finite set of* states *and* $Q_0 \subseteq Q$ *is the set of* final states;
- $g \subseteq \Sigma \times Q$ *is the* input relation;
- $f \subseteq Q \times Q \times Q$ *is the* transition relation.

Let us introduce now a notion of step-wise binary tree computation on ω-words. Consider an ω-word α. At each computation step, the automaton process a segment of α whose length is a power of two and doubles step by step. The computation of a systolic automaton $\mathcal{A}$ is defined for a word $w \in \Sigma^*$ only if $|w| = 2^k$ (with $k \geq 0$) and it is given by the relation $O_{\mathcal{A}} \subseteq \Sigma^* \times Q$ defined recursively on k as follows:

- if $|w| = 2^0$, then $\langle w, q \rangle \in O_{\mathcal{A}}$ if and onlyif $\langle w, q \rangle \in g$;
- if $|w| = 2^k$, with $k > 0$, then $\langle w, q \rangle \in O_{\mathcal{A}}$ if and only if $\langle q_1, q_2, q \rangle \in f$, with $\langle w_1, q_1 \rangle, \langle w_2, q_2 \rangle \in O_{\mathcal{A}}$, $|w_1| = |w_2| = 2^{k-1}$ and $w_1 \cdot w_2 = w$.

The step-wise computation of an automaton $\mathcal{A}$ over an ω-word α is recorded into an ω-sequence of states called *B-run*. Such a sequence stores at the i-th position the state q resulting from processing the prefix $\alpha(0, 2^i - 1)$ of α. The state resulting from processing the next prefix $\alpha(0, 2^{i+1} - 1)$ is obtained from q and from the states q', output by the systolic automaton fed with $\alpha(2^i, 2^{i+1} - 1)$, according to the transition relation f. The structure of the step-wise computation is depicted in figure

4. Notice that the B-run is the sequence of states labelling the 0-th column of the structure.

More formally, a *B-run* of $\mathcal{A}$ on $\alpha \in \Sigma^\omega$ is a map $\psi : \mathbb{N} \mapsto Q$ s.t.

- $\langle \alpha(0), \psi(0) \rangle \in g$;
- $\langle \psi(i-1), q, \psi(i) \rangle \in f$, with $\langle \alpha(2^{i-1}, 2^i - 1), q \rangle \in O_{\mathcal{A}}$, for $i > 0$.

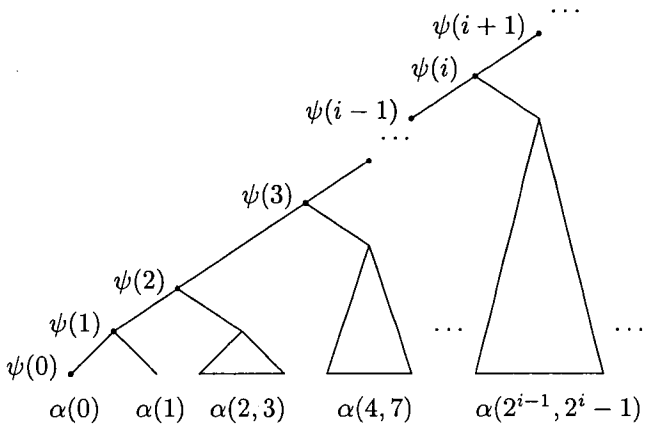

Figure 4: A B-run ψ on α.

A B-run is *successful* iff there are infinitely many i such that $\psi(i) \in Q_0$. An ω-word α is *accepted by* $\mathcal{A}$ iff there exists a successful run of $\mathcal{A}$ on α. The set of ω-words (i.e. *the language*) *accepted by* $\mathcal{A}$ is denoted by $\mathcal{L}_\omega(\mathcal{A})$.

In [13] we proved that the logical counterpart of ω-languages accepted by binary tree automata is obtained by extending $MSO[<]$ with a function *flip* given as follows.

Definition 2 *The function* $flip : \mathbb{N}^+ \to \mathbb{N}$ *is such that*

$$\text{if } x = 2^{k_n} + 2^{k_{n-1}} + \ldots + 2^{k_0} \text{ with } k_n > \ldots > k_1 > k_0, \text{ then } flip(x) = x - 2^{k_0}.$$

The logical definability of ω-languages accepted by binary tree automata is proved by showing that the function $flip$ allows to impose on natural numbers the upward unbound binary tree structure depicted in figure 5. Such a structure is obtained by associating with each even number $z = 2^{k_n} + \ldots + 2^{k_1} + 2^{k_0}$ (with $k_n > \ldots > k_1 > k_0 > 0$), the left and right son, respectively,

$$x = 2^{k_n} + \ldots + 2^{k_1} + 2^{k_0 - 1} \text{ and} \tag{1}$$

$$y = 2^{k_n} + \ldots + 2^{k_1} + 2^{k_0} + 2^{k_0 - 1}. \tag{2}$$

Such a childhood relationship can be easily defined into $MSO[<, flip]$ since

$$x = max\{s : s < z, flip(s) = flip(z)\} \text{ and} \tag{3}$$

$$y = max\{s : flip(s) = z\}. \tag{4}$$

Notice that the elements of the i-th layer (with $i \geq 0$) are

$$\{2^i + k2^{i+1} : k \geq 0\} \tag{5}$$

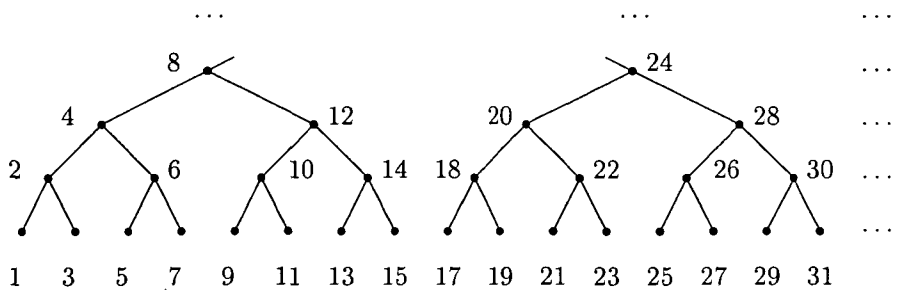

Figure 5: The upward unbounded binary tree structure on $\mathbb{N}$.

and the elements of the i-th column (with $i \geq 0$) are

$$\{(2i+1)2^k : k \geq 0\}. \tag{6}$$

Remark. Notice that the 0-th layer of the upward unbounded binary structure on $\mathbb{N}$ is isomorphic but not equal to $\mathbb{N}$ (even numbers are missing). As a consequence, when such a structure is exploited for simulating a systolic computation on an input word α, α cannot be associated directly with the 0-th layer (i.e. $\alpha(0)$ with 1, $\alpha(1)$ with 3 and so on) since, in this case, the set of numbers $\mathbb{N}$ would not be in correspondence with the index set of α. The correspondence between $\mathbb{N}$ and the index set of α is recovered by associating with the 0-th layer, instead of α, the sequences of states of the systolic automaton $q_1, q_3, q_5, \ldots, q_{2n+1}, \ldots$, with $q_i \in f(g(\alpha(i-1)), g(\alpha(i)))$ (a preprocessing of α).

Theorem 1 *(cf.[13]) An ω-language is accepted by a binary tree automaton iff it is $MSO[<, flip]$-definable (or equivalently, iff it is $EMSO[<, flip]$ -definable).*

4 Y-Tree ω-Languages

The interconnection structure of a Y-tree automaton is a leafless perfectly balanced binary tree augmented with links from a node to the immediate neighbour (if any) of its right son called the *adoptive* son (see figure 1.b). The expressive power and properties of Y-tree automata as acceptors of finitary languages have been studied in [5, 7]. In this section we introduce and study Y-tree automata as acceptors of ω-languages. In this respect, a Y-tree automaton processes a word exactly as a binary tree automaton does with the only exception that the computation is performed over an upward unbounded Y-tree structure (see figure 6).

Definition 3 *A* Y-tree automaton *is a tuple $\mathcal{Y} = \langle \Sigma, Q, Q_0, g, f \rangle$, where*

- *Q is the finite set of* states *and $Q_0 \subseteq Q$ is the set of* final states*;*
- *$g \subseteq \Sigma \times Q$ is the* input relation*;*

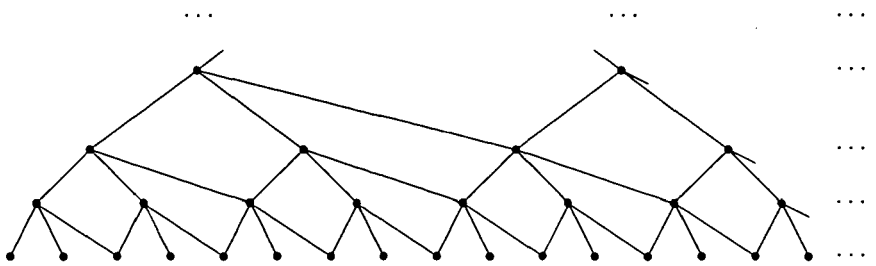

Figure 6: The upward unbounded Y-tree structure.

- $f \subseteq Q \times Q \times Q \times Q$ *is the* transition relation.

A computation of a Y-tree automaton is then a labelling of an upward unbounded Y-tree structure compatible with the input and transition relations of the automaton. As in the case of binary tree automata, we define now the step-wise version of such a computation. Given an ω-word α, at each computation step, the automaton processes a segment of α whose length is a power of two and which doubles step by step. In this way, at the i-th computation step, the prefix $\alpha(0, 2^{i+1}-2)$ of α turns out to be processed, and the result of such a computation is stored into a sequence $q_0 \ldots . q_i$ of states of Q. The sequence $q_0 \ldots . q_i$ is precisely the sequence of states labelling the path of adoptive edges from the i-th element of the 0-th column of the upward unbounded Y-tree structure leading to the 0-th layer (see, for instance, the path labelled by $v(0)v(1)v(2)$ in figure 7 which codifies the result of the 2-th step of a computation). Now, the $i+1$-th step transforms the sequence $q_0 \ldots . q_i$, encoding the computation at the i-th step, into a sequence of states $q'_0 \ldots . q'_i, q'_{i+1}$ as a result of composing $q_0 \ldots . q_i$ with the result of processing the segment $\alpha(2^{i+1}-1, 2^{i+2}-2)$ of α. Notice that, in order to store the information about the computation on the prefix of a ω-word, a sequence (of unbounded length) of states is required (in the case of binary tree automata a single state suffices). For a Y-tree automaton $\mathcal{Y}\langle \Sigma, Q, Q_0, g, f \rangle$, the transformation above, called *run step*, is given by the relation

$$O_{\mathcal{Y}} \subseteq Q^\star \times \Sigma^\star \times Q^\star$$

which satisfies the property that $\langle v, w, u \rangle \in O_{\mathcal{Y}}$ implies $|v| = k$, $|w|2^k$ and $|u| = k+1$, for some $k \geq 0$, and is defined recursively on k as follows (see figure 7 for a graphical hint):

- if $k = 0$, then $\langle \epsilon, w, q \rangle \in O_{\mathcal{Y}}$, with $\langle w, q \rangle \in g$;
- if $k > 0$ and $w_1 \cdot w_2 = w$ with $|w_1| = |w_2| = 2^{k-1}$, then $\langle v, w, q \cdot u_2 \rangle \in O_{\mathcal{Y}}$, where
  - $\langle v(1, k-1), w_1, u_1 \rangle \in O_{\mathcal{Y}}$, $\langle u_1(1, k-1), w_2, u_2 \rangle \in O_{\mathcal{Y}}$, $\langle v(0), u_1(0), u_2(0), q \rangle \in f$.

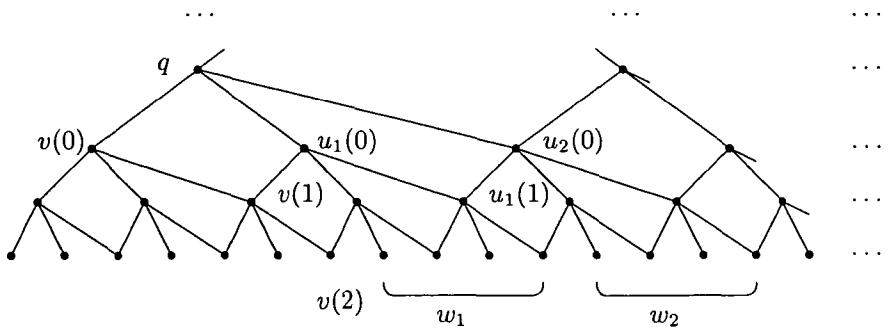

Figure 7: A Y-tree step.

A *Y-run* for an ω-word α stores the result of each computation step and it is defined as a map $\psi : \mathbb{N} \mapsto Q^\star$ such that:

- $\langle \alpha(0), \psi(0) \rangle \in g$;
- $\langle \psi(i), \alpha(2^{i+1} - 1, 2^{i+2} - 2), \psi(i+1) \rangle \in O_{\mathcal{Y}}$, for $i \geq 0$.

A Y-run ψ over $\alpha \in \Sigma^\omega$ is *successful* iff there are infinitely many $i \in \mathbb{N}$ s.t. $\psi(i) = v_i$ and $v_i(0) \in Q_0$. The structure of a Y-run is shown in figure 8. The definition of a ω-language accepted by Y-tree automaton is exactly as the one for binary tree automata.

Example 1 *An instance of Y-tree ω-language is:*

$$L = \{\alpha \in \{0,1\}^\omega : \alpha(2^s + k2^{s+1}) = 1 \text{ for some } s \geq 0, \text{ for all } k \geq 0\}. \quad (7)$$

Notice that $\alpha \in L$ iff the positions where α is set to 1 include at least the set of natural numbers belonging to a layer of the upward unbounded binary tree structure on $\mathbb{N}$ (see figure 5 and Eq.5).
Before giving a formal definition of the automaton accepting the language L, we sketch the idea behind the recognising process. The automaton propagates the symbols $\{0,1\}$, received at its input layer, till a layer which is supposed to be uniformly labelled by symbol 1 is reached. At this point, all of the processors in the active layer check the uniformity of the labelling of the previous layer, by verifying that all of their left and adoptive sons are uniformly labelled, and propagates the successful/unsuccessful result of the test. A symbol is propagated along the adoptive edge at the first step and along the right edge in all the remaining steps. At each step a processor chooses non-deterministically whether to propagate the signal it is receiving from its right son or to perform the uniformity check. The check can be successful only if all of the processors of a layer choose simultaneously and non-deterministically to perform the uniformity check. The automaton $\mathcal{Y} = \langle \Sigma, Q, Q_0, g, f \rangle$ accepting L is given by the following components:

- $Q = \{0, 1, \overline{0}, \overline{1}, ok\}$, $Q_0 = \{ok\}$, $g = \{\langle x, \overline{x} \rangle : x \in \{0,1\}\}$;

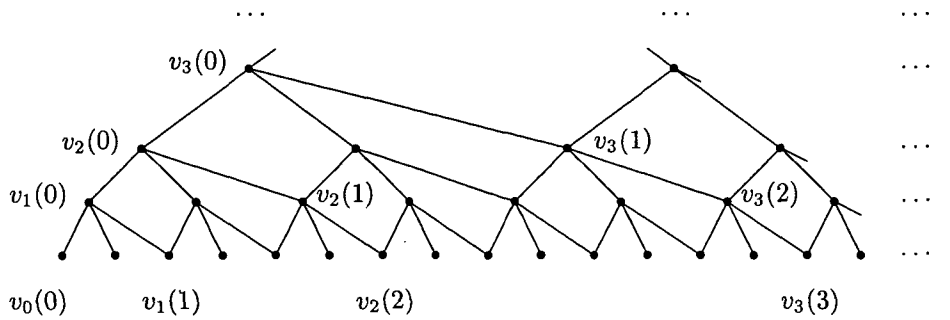

Figure 8: The structure of a Y-run ($v_i = \psi(i)$).

- $f = \{\langle \overline{x}, \overline{1}, \overline{y}, ok\rangle, \langle \overline{x}, \overline{y}, \overline{z}, z\rangle, \langle 1, x, 1, ok\rangle, \langle x, y, z, y\rangle, \langle ok, ok, ok, ok\rangle :$ *with* $x, y, z \in \{0, 1\}\}$.

Over-lining of states is used to distinguish the first step of the computation from the others. The state ok is output after a successful uniformity check. The first and second schemas of tuples manage check and transmission, respectively, at the first step. The third and fourth schemas of tuples manage check and transmission, respectively, in all the other steps.

As far as expressivity is concerned, the class of Y-tree ω-languages includes the class of binary tree ω-languages as an immediate consequence of the fact that the upward unbounded Y-tree structure is a upward unbounded binary tree structure enriched with adoptive edges. In fact, as in the finitary case, Y-tree automata are more expressive than binary tree ones.

Theorem 2 *The class of Y-tree ω-languages strictly includes the class of binary tree ω-languages.*

Proof. We show that the Y-tree language L of Example 1 is not accepted by any systolic binary tree automaton. The proof is by contradiction. Let us assume that there exists a systolic binary tree automaton $\mathcal{A}$ having n states and accepting L. We take now the $n+1$ (different) ω-words $\alpha_0, \ldots, \alpha_n$ of L such that $\alpha_i(j) = 1$ iff $j = 2^i + k2^{i+1}$ (for all $k \geq 0$). The positions set to 1 in α_i are exactly those associated with the i-th layer of the upward unbounded structure. Let ψ_i be a successful B-run of $\mathcal{A}$ for α_i (with $0 \leq i \leq n$). Since the number of states is n, there must be two runs ψ_j and ψ_k with $j \neq k$ such that $\psi_j(n+1) = \psi_k(n+1)$. This immediately implies, by the definition of B-run, that the ω-word

$$\alpha' = \alpha_j(0, 2^{n+1} - 1) \cdot \alpha_k(2^{n+1}, \omega)$$

has a succesful run (e.g. $\psi_j(0, n+1) \cdot \psi_k(n+2, \omega)$). This leads to a contradiction since $\alpha' \notin L$. □

By exploiting standard techniques it is easy to prove the closure properties of Y-tree ω-languages stated in the following proposition.

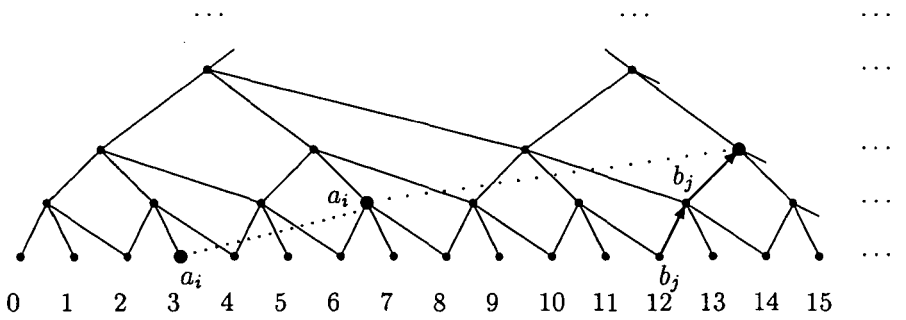

Figure 9: Trasmitting a_i along the third column and b_j along left edges.

Proposition 3 *The class of Y-tree ω-languages is closed under union, intersection and projection.*

As in the finitary case (see [7]), the class of languages accepted by Y-tree automata is not closed under complementation. We show in the following example a Y-tree language whose complement is not accepted by any Y-tree automaton.

Example 2 *An instance of Y-tree ω-language is*

$$L = \{\alpha \in \{0,1\}^\omega : \alpha(2^s + k2^{s+1}) \neq \alpha(2^{s'} + k2^{s'+1}) \text{ for some } s', s, k, \geq 0\}. \quad (8)$$

Notice that $\alpha \in L$ iff there are two positions i and j of α such that $\alpha(i) \neq \alpha(j)$ and i and j belong to the same column of the upward unbounded binary tree structure on $\mathbb{N}$ (see Eq.6). In order to prove that $\alpha \in L$, it is sufficient to check that there exists an odd position i (i.e. the least element of a column) of α such that $\alpha(i) \neq \alpha(j)$ and i and j are elements of the same column. We sketch the algorithm used by the automaton to check that positions i and j as above belong to the same column. The idea is that of (upward) propagating the symbol a_i from position i, and b_j from position j, till they meet at a common node (see figure 9). The symbol b_j is forced to move only along left edges. The symbol a_i is propagated, by exploiting non-determinism, along the i-th column. Following this procedure, symbols a_i and b_j meet if and only if i and j belong to the same column. In fact, if i and j belong to the same column, then $j = i2^k$ (for some k). When j moves from the 0-th layer to the 1-th along a left edge, it reaches the $i2^{k-1}$-th position of that layer, and then the $i2^{k-2}$ at the next step, and so on (see figure 9). It remains to show that it is possible to move the symbol a_i along the i-th column. Let us consider the example of figure 10. The symbol a is input at position 5. At the second computation step, the node at position 5-th in the 1-th layer guesses that it is an element of the 5-th column and it enters the state A. The correctness of such a guess has to be checked. To this purpose, both the nodes labelled by a and A transmit upward a check-symbol (c_a and C_A, respectively). The guess is correct if the sequences of edge types (left/right) in the two paths from the nodes labelled by a and A to the 0-th column is the same. Since the path length could be arbitrarily long and the

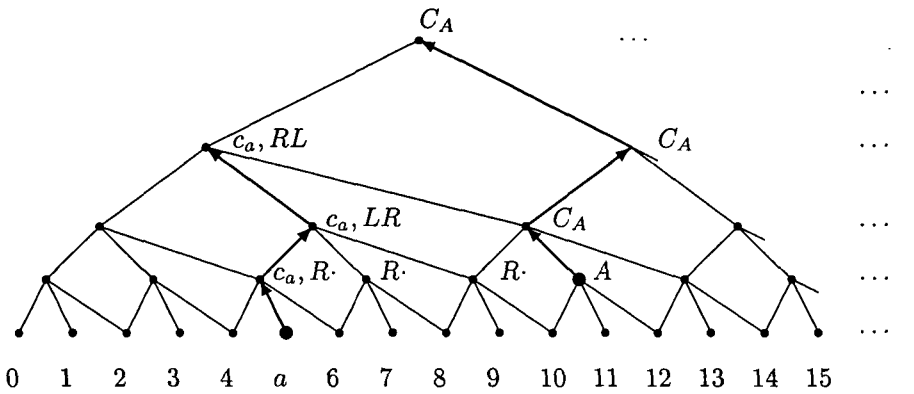

Figure 10: Transmitting a symbol along the 5-th column.

number the states is finite, this comparison cannot be performed at the completion of the path but must be performed step by step in the following way. Each sequence of nodes in the segment of the layer bounded by the path of c_a and C_A keeps trace of the last two types of edges crossed by c_a in its way. In figure 10, $R\cdot$, LR and RL stand for Right, Left – Right and Right – Left, respectively. This information is used by the nodes on the path of C_A to force the crossing of an edge of the same type as the one crossed by c_a a step before. The formal definition of the automaton is now easy and it is, for the sake of simplicity, omitted.

Theorem 4 *The class of Y-tree ω-languages is not closed under complementation.*

Proof. We show that the language

$$L = \{\alpha \in \{0,1\}^\omega : \alpha(2^s + k2^{s+1})\alpha(2^{s'} + k2^{s'+1}),\ \text{for all } s', s \geq 0 \text{ with } k \geq 0\} \quad (9)$$

is not a Y-tree ω-language. Note that such a language is the complement of the Y-tree ω-language of Example 2, since $\alpha \in L$ iff all of the positions of α belonging to the same column are labelled by the same symbol. The proof is by contradiction. Let us assume that there exists a systolic Y-tree automaton $\mathcal{Y}$ having n states and accepting L. For a suitable m, we take now $s = 2^{\frac{2^{m+1}-2}{2}} > n^{m+1}$ (different) ω-words $\alpha_1, \ldots, \alpha_s$ of L such that $\alpha_i(k) \neq \alpha_j(k')$ for odd k, k', with $1 \leq k, k' \leq 2^{m+1} - 2$, for all $1 \leq i < j \leq s$. Notice that each odd number identifies a column, and then α_i and α_j disagree at least in the labelling of a column. Let ψ_i be a succesful Y-run of $\mathcal{Y}$ for α_i (with $1 \leq i \leq s$). Since there are at most $n^{m+1} < s$ different strings on states of $\mathcal{Y}$ of length $m+1$, there must be two runs ψ_j and ψ_i with $i \neq j$ such that $\psi_i(m) = \psi_j(m)$. This immediately implies, by definition of Y-run, that the ω-word

$$\alpha' = \alpha_j(0, 2^{m+1} - 2) \cdot \alpha_i(2^{m+1} - 1, \omega)$$

has a succesful Y-run, namely ψ such that $\psi(k) = \psi_j(k)$, for $0 \leq k \leq m$ and $\psi = \psi_i$, elsewhere. This lead to a contradiction since $\alpha' \notin L$. □

Proposition 5 *The emptiness problem for Y-tree ω-languages is not decidable.*

Proof. In order to show that emptiness is undecidable, it is sufficient to realise that for a given Y-tree automaton $\mathcal{Y}$ (for finitary words on the alphabet Σ) we can construct a Y-tree automaton $\mathcal{Y}'$ for ω-words on alphabet $\Sigma \cup \{\sharp\}$ (with $\sharp$ not in Σ), such that the language accepted by $\mathcal{Y}$ is empty iff the language accepted by $\mathcal{Y}'$ is empty. Now, the undecidability of emptiness problem for the case of ω-words follows from the undecidability of the emptiness problem in the finitary case (see [6]). □

To define a calculus which is as powerful to accept Y-tree ω-languages, we extend $MSO[<]$ by a (partial) function $adj : \mathbb{N} \mapsto \mathbb{N}$, s.t.

$$\text{if } x = 2^{k_n} + 2^{k_{n-1}} + \ldots + 2^{k_0}, \text{ with } k_n > k_{n-1} > \ldots > k_0 > 0, \text{ then}$$
$$adj(x) = x + 2^{k_0} + 2^{k_0 - 1}.$$

The logical definability of ω-languages accepted by Y-tree automata can be proved by showing that the function adj allows us to impose on natural numbers the upward unbound Y-tree structure depicted in figure 11. Such a structure is obtained by enriching the binary tree structure of figure 5 with an edge between x and y iff $adj(x) = y$.
In fact, $MSO[<, adj]$ allows one to define two predicates $Right\_son(z, x)$ and $Left\_son(z, x)$, which are true iff the number x is the right and left son, respectively, of z (in the sense of Eq.2 and Eq.1).
Now, x is the left son of z, if either $x + 1 = z$ and z is at the 1-th layer (i.e. $\exists s(adj(z) = s \wedge \neg\exists t(adj(s) = t))$), or z is at the i-th layer (with $i > 1$) and

$$x = min\{s : s < z \wedge adj(s) > z)\}.$$

The two cases above can be expressed by a formula of $MSO[<, adj]$ which, for simplicity, we denote by $Left\_son(z, x)$. The formula $Right\_son(z, x)$ can be expressed in terms of adj and $Left\_son(z, x)$ as follows

$$\neg\exists s(adj(x) = s) \wedge x = z + 1 \vee$$
$$\exists s, t(adj(x) = s \wedge adj(z) = t \wedge Left\_son(t, s)).$$

Example 3 *A $EMSO[<, adj]$-formula defining the language L of Eq.7 is*

$$\exists Y(\forall x(Y(x) \Rightarrow Q_1(x)) \wedge$$
$$\forall y, z(son(z, y) \wedge Y(y) \Rightarrow \forall s(son(z, s) \Rightarrow Y(s)))),$$

where $son(z, y)$ stands for $Left\_son(z, y) \vee Right\_son(z, y) \vee adj(z) = y$.

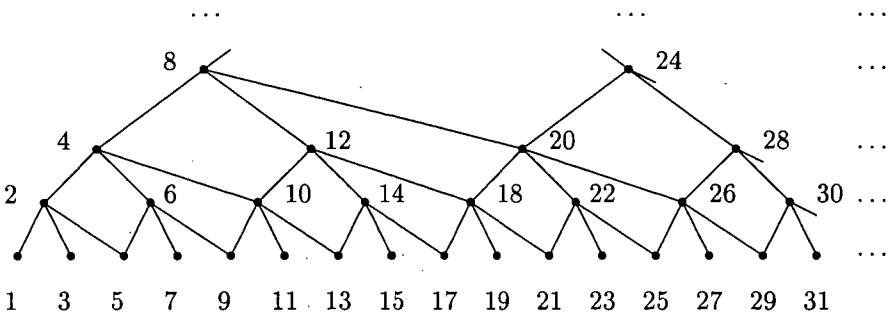

Figure 11: The upward unbounded Y-tree structure on $\mathbb{N}$.

Now, we can state the main theorem of this section, namely the logical definability of Y-tree ω-languages. As it is precisely shown in the proof of Th. 6, the upward unbounded Y-tree structure on $\mathbb{N}$ allows to simulate a Y-tree systolic computation. Also in this case, as remarked in the end of section 2, an ω-word α cannot be associated directly with the 0-th layer of the structure but it is associated to this layer only after a preprocessing step.

Theorem 6 *Any ω-language accepted by a Y-tree automaton is $EMSO[<, adj]$-definable.*

Proof. Assume without loss of generality that the set of states of the Y-tree automaton $\mathcal{Y}$ is $Q = 1, \ldots, m$. A Y-tree computation can be obtained by labelling, compatibly with the transition relation f, the upward unbounded Y-tree structure with elements of Q. Such a labelling is defined by partitioning the set of natural numbers into m sets $Y_1, \ldots, Y_m$ and assuming that a number x belongs to Y_i iff the state i labels the node x of the upward unbounded Y-tree structure. We introduce some short notations:
$Odd(x)$ (i.e. x is an odd number) stands for $\neg\exists z(adj(x) = z)$.
$Power(x)$ (i.e. x is a power of two) stands for
$x = 1 \vee \exists z(adj(x) = z \wedge \forall y, s((y < x \wedge adj(y) = s) \Rightarrow s < z))$.
$x \xrightarrow{L} i$ (i.e. the left son of x is associated with the state i) stands for
$\exists y\,(Left\_son(x, y) \wedge Y_i(y))$.
$x \xrightarrow{R} i$ (i.e. the right son of x is associated with the state i) stands for
$\exists y\,(Right\_son(x, y) \wedge Y_i(y))$.
$x \xrightarrow{A} i$ (i.e. the adoptive son of x is associated with the state i) stands for
$\exists y\,(adj(x) = y \wedge Y_i(y))$
Given a Y-tree automaton $\mathcal{Y} = \langle\{1, \ldots, m\}, Q_0, g, f\rangle$, a formula ϕ belonging to $EMSO[<, adj]$ accepting $\mathcal{L}_\omega(\mathcal{Y})$ is the following:
$\exists Y_1, \ldots, Y_m$ (

$$\bigwedge_{i \neq j} \neg\exists y\,(Y_i(y) \wedge Y_j(y)) \wedge \tag{10}$$

$$\forall x \left(Odd(x) \Rightarrow \left(\bigwedge_{a,b,c \in \Sigma} Q_a(x-1) \wedge Q_b(x) \wedge Q_c(x+1) \right) \Rightarrow \bigvee_{\{i:\langle j,k,l,i\rangle \in f, \langle a,j\rangle, \langle b,k\rangle, \langle c,l\rangle \in g\}} Y_i(x) \right) \right) \wedge \tag{11}$$

$$\forall z \left(\bigwedge_{i,j,l \in Q} \left((z \xrightarrow{L} i \wedge z \xrightarrow{R} j \wedge z \xrightarrow{A} l) \Rightarrow \bigvee_{\{k:\langle i,j,l,k\rangle \in f\}} Y_k(z) \right) \right) \wedge \tag{12}$$

$$\bigvee_{i \in F} \forall x \exists y \, (x < y \wedge Power(y) \wedge Y_i(y)) \tag{13}$$

). □

As a consequence of Theorem 6 and the undecidability of emptiness for Y-tree ω-languages (see Proposition 5), we have the following corollary.

Corollary 7 *The theory $EMSO[<, adj]$ is undecidable.*

The theory $EMSO[<, adj]$ extends $MSO[<, flip]$ since the class of Y-tree ω-languages strictly includes the class of binary tree ω-languages (see Theorem 2) and $MSO[<, flip]$ is a characterisation of that class of languages (see Theorem 1).

Corollary 8 *$EMSO[<, adj]$ is a proper extension of $MSO[<, flip]$.*

The function *adj* we have chosen to extend $MSO[<]$ seems not to be more expressive than it is required to define Y-tree ω-languages. This is suggested by the fact that the ω-language encoding the function *adj* (as well as its complement) can be accepted by a Y-tree automaton. Given a relation $R \subseteq \mathbb{N} \times \mathbb{N}$, the ω-word on the alphabet $\{0,1,2\}$ *encoding the pair* $\langle i,j \rangle \in R$ is $\alpha_{\langle i,j \rangle} \in \{0,1,2\}^\omega$ such that $\alpha_{\langle i,j \rangle}(i) = 1$, $\alpha_{\langle i,j \rangle}(j) = 2$ and $\alpha_{\langle i,j \rangle}(k) = 0$ for all $k \neq i, j$. The ω-language *encoding the relation* R is the set $\{\alpha_{\langle i,j \rangle} : \langle i,j \rangle \in R\}$.

Proposition 9 *There exists a deterministic Y-tree automaton accepting the ω-language L encoding the adj function.*

Proof. We define two Y-tree automata $\mathcal{Y}_1$ and $\mathcal{Y}_2$ such that $L = \mathcal{L}_\omega(\mathcal{Y}_1) \cup \mathcal{L}_\omega(\mathcal{Y}_2)$. The automaton $\mathcal{Y}_1$ (resp. $\mathcal{Y}_2$) accepts ω-words having the form

$$0^{2^{k_n}+\ldots+2^{k_0}}.1.0^{2^{k_0}+2^{k_0-1}-1}.2.0^\omega$$

with $k_n > k_{n-1} > \ldots > k_0$ and $k_0 = 1$ (resp. $k_0 > 1$). The claim follows from closure under union of languages accepted by Y-trees. As concerns the automaton $\mathcal{Y}_1$, notice that the node of the Y-tree unbound structure that receives the input 1 (resp. 2) must be the left son of a right son node (resp. the right son of a left son node). Such a pattern can be detected in two steps by the following automaton:

$$\mathcal{Y}_1 = \langle \{0,1,2\}, \{0,1,2,\overline{1},\overline{2},s\}, \{s\}, g_1, f_1 \rangle, \text{ where}$$

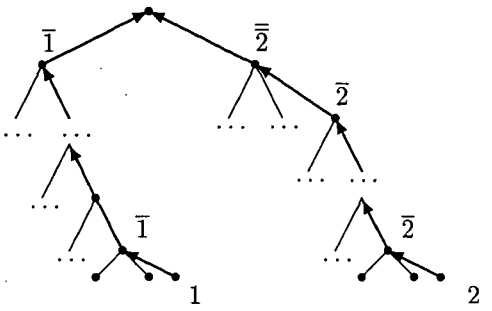

Figure 12: The pattern recognised by automaton $\mathcal{Y}_2$ of Proposition 9.

- g_1 is the identity;
- $f_1 = \{ \langle 0,0,0,0\rangle, \langle 0,0,1,\overline{1}\rangle, \langle 0,2,0,\overline{2}\rangle, \langle 1,0,0,0\rangle, \langle \overline{1},0,\overline{2},s\rangle,$
  $\langle \overline{2},0,0,0\rangle, \langle 0,0,\overline{1},0\rangle, \langle s,0,0,s\rangle, \langle 0,s,0,s\rangle, \langle 0,0,s,0\rangle\};$

The pattern detected by automaton $\mathcal{Y}_2$ is depicted in figure 12. The symbol 1, received at input level, must be propagated the first time along an adoptive edge and then along a right edge as far as possible. The symbol 2 must be propagated the first time along an adoptive edge, then along a right edge as far as possible, and, finally, again along an adoptive edge. Signals propagated in this way must be eventually collected by a node from the left and right son. The automaton $\mathcal{Y}_2$ is as follows:

$$\mathcal{Y}_2 = \langle \{0,1,2\}, \{0,1,2,\overline{1},\overline{2},\overline{\overline{2}},s\}, \{s\}, g_2, f_2\rangle, \text{ where}$$

- g_2 is the identity;
- $f_2 = \{ \langle 0,0,1,\overline{1}\rangle, \langle 0,\overline{1},0,\overline{1}\rangle, \langle 1,0,0,0\rangle, \langle 0,0,\overline{1},0\rangle,$
  $\langle 0,0,2,\overline{2}\rangle, \langle 0,\overline{2},0,\overline{2}\rangle, \langle 0,0,\overline{2},\overline{\overline{2}}\rangle, \langle 2,0,0,0\rangle, \langle \overline{2},0,0,0\rangle,$
  $\langle \overline{1},\overline{\overline{2}},0,s\rangle, \langle s,0,0,s\rangle, \langle 0,s,0,s\rangle, \langle 0,0,s,0\rangle\}.$

In general, if $\alpha(i) = 1$, then $\overline{1}$ flows up to k-th layer and labels a left-son processor iff $i = 2^k + m2^{k+1}$ with $(m \geq 0)$. If $\alpha(j) = 2$, the signal flows up to the k-th layer labelling a right-son processor iff $j = 2^{k-1} + m2k + 1$ (with $m \geq 1$). If i and j are the left and right son of the same node, then $i = 2^k + m2^{k+1}$ and $j = 2^{k-1} + (m+1)2^{k+1}$ and then $j = i + 2^{k-1} + 2^k = adj(i)$. □

Since Y-tree ω-languages are closed under union and intersection, by Proposition 3, any formula freely constructed from atomic formulas using the boolean connectives $\wedge$, $\vee$ and $\neg$ has a corresponding Y-tree automaton. If ϕ is a formula with a corresponding Y-tree automaton, also the existential quantification of ϕ (on both individual and predicates) has a Y-tree automaton as a counterpart since Y-tree ω-languages are closed under projection. The problem whether any formula $\neg\exists x\phi$,

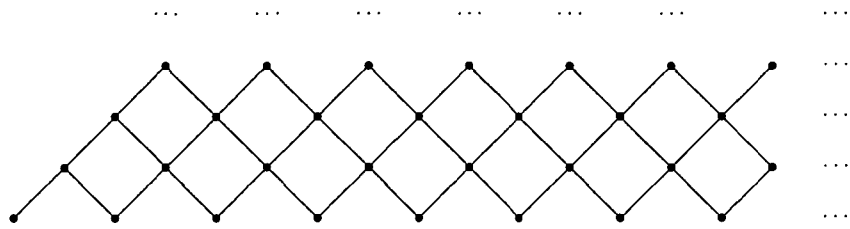

Figure 13: An upward unbounded trellis structure.

with x an individual variable and ϕ a first order formula, can have a corresponding Y-tree automaton, remains an open problem. If such a problem had positive answer, we would have that $EMSO[<, adj]$ would be a characterisation of Y-tree ω-languages, namely that an ω-language is accepted by a Y-tree automaton iff it is $EMSO[<, adj]$-definable.

5 Trellis ω-Languages

The expressive power and properties of trellis automata as acceptors of finitary languages (over the standard interconnection structure of figure 1.c) have been studied in [1, 2] for the deterministic case and in [9] for the non-deterministic case. In this section we introduce and study non-deterministic trellis automata as acceptors of ω-languages. In this respect, a trellis automaton processes a word exactly as a binary tree (or a Y-tree) automaton does with the only exception that it uses, for the computation, an upward unbounded trellis structure (see figure 13).

Definition 4 *A* trellis automaton *is a tuple* $\mathcal{T} = \langle \Sigma, Q, Q_0, g, f \rangle$, *where*

- Q *is the finite set of* states *and* $Q_0 \subseteq Q$ *is the set of* final states;
- $g \subseteq \Sigma \times Q$ *is the* input relation;
- $f \subseteq Q \times Q \times Q$ *is the* transition relation.

A computation of a trellis automaton is then a labelling of an upward unbounded trellis structure compatible with the input and transition relations. As in the case of binary tree and Y-tree automata, we define a step-wise version of such a computation. Given an ω-word α, at each computation step, the automaton processes a symbol of α. So, at the i-th step (with $i \geq 0$) the prefix $\alpha(0, i)$ of α is processed and the result of such a computation is stored into a sequence $q_0 \ldots . q_i$ of states of Q. The sequence $q_0 \ldots . q_i$ is precisely the sequence of states labelling the path of right edges, from the i-th element of the 0-th column of the upward unbounded trellis structure, leading to the 0-th layer. Now, the $i+1$-th step

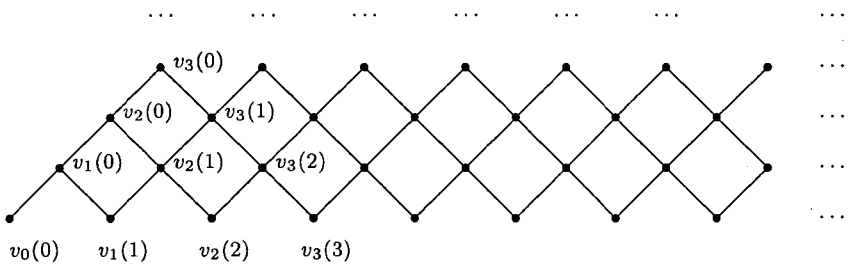

Figure 14: The structure of a T-run.

transforms the sequence $q_0 \ldots . q_i$, encoding the computation at the i-th step, into a sequence of states $q'_0 \ldots . q'_i, q'_{i+1}$ as a result of composing $q_0 \ldots . q_i$ with the result of processing the symbol $\alpha(i+1)$. For a trellis automaton $\mathcal{T} = \langle \Sigma, Q, Q_0, g, f \rangle$, the transformation above, called *run step*, is defined by the relation

$$O_{\mathcal{T}} \subseteq Q^\star \times \Sigma \times Q^\star,$$

which is such that $\langle v, a, u \rangle \in O_{\mathcal{T}}$ implies $|v| = k$ and $|u| = k+1$, for some $k \geq 0$ and is defined recursively on k as follows:

- if $k = 0$, then $\langle \epsilon, a, q \rangle \in O_{\mathcal{T}}$, with $\langle a, q \rangle \in g$;
- if $k > 0$, then $\langle v, a, u \rangle \in O_{\mathcal{T}}$, where
  - $\langle a, u(k) \rangle \in g$ and $\langle v(i), u(i+1), u(i) \rangle \in f$, for $0 \leq i < k$.

A *T-run* for an ω-word α stores the result of each computation step and it is defined as a map $\psi : \mathbb{N} \mapsto Q^\star$ such that:

$$\langle \alpha(0), \psi(0) \rangle \in g \text{ and } \langle \psi(i), \alpha(i+1), \psi(i+1) \rangle \in O_{\mathcal{T}}, \text{ for } i \geq 0.$$

A T-run ψ is *successful* iff there are infinitely many $i \in \mathbb{N}$ such that $\psi(i) = v_i$ and $v_i(0) \in Q_0$.
The structure of a T-run is depicted in figure 14.

Example 4 *An instance of trellis ω-language is:*

$$L = \{\alpha \in \{0,1\}^\omega : \alpha(i)\alpha(i+k) \textit{ for some } k \geq 1, \textit{ for all } i \geq 0\}. \qquad (14)$$

Notice that $\alpha \in L$ iff $\alpha = w^\omega$ with $w \in \{0,1\}^\star$. Before giving a formal definition of the automaton accepting the language L above, we sketch the idea of the recognising process. Assume that $\alpha = w^\omega$, with $|w| = k$. Each input symbol is propagated upward along the left and the right edges till it reaches the $k-1$-th layer. Each processor of the k-th layer checks whether the symbols, received from the left and the right sons, agree. Processors in the next layer check that all their sons had a

successful result and so on. A processor chooses non-deterministically whether to propagate the signals it is receiving or to perform the test, and the computation can be successful only if all of the processors of the k-th layer non-deterministically choose to perform the test.

The automaton $\mathcal{T} = \langle \Sigma, Q, Q_0, g, f \rangle$ *accepting* L *has the following components:*

- $Q = \{ok, x, [x,y] : x, y \in \{0,1\}\}$, $Q_0 = \{ok\}$, $g = \{\langle x, x\rangle : x \in \{0,1\}\}$;
- $f = \{\langle x, y, [y,x]\rangle, \langle [x,y], [z,t], [z,y]\rangle, \langle [x,y], [y,z], ok\rangle, \langle ok, ok, ok\rangle : x, y, z, t \in \{0,1\}\}$.

States $[x,y]$ *are used to forward properly the input symbols upward along the left and right edge, respectively. Each node of the* 1*-th layer enters the state* $[y,x]$ *after receiving symbol* x *from the left-son and symbol* y *from the right-son. Each processor, after receiving state* $[x,y]$ *from the left-son and state* $[z,t]$ *from the right-son, propagates the symbols* y *and* z *by entering the state* $[z,y]$. *An example of successful computation is shown in figure 15 for* $k = 3$.

We consider now the expressive power of trellis automata with respect to Y-tree automata.

Theorem 10 *The class of trellis* ω*-languages strictly includes the class of* Y*-tree* ω*-languages.*

Proof. We start showing that a systolic computation over an upward unbounded Y-tree structure can be simulated over an upward unbounded trellis structure.

Let $\mathcal{Y}$ be a Y-tree automaton, we outline the construction of a trellis automaton $\mathcal{T}$ accepting the same ω-language of $\mathcal{Y}$. The behaviour of the node of $\mathcal{Y}$ placed in the i-th layer and j-th column of the upward unbounded Y-tree structure is simulated by the behaviour of the node of the automaton $\mathcal{T}$ placed in the $j(3^i - 3^{i-1})$-th column and the $3^i - 3^{i-1}$-th layer of the upward unbounded trellis structure (for $i \geq 1$, $j \geq 0$). With reference to figure 16, simulating nodes are labelled by symbol x. The left, right and adoptive edges connecting a node with its children in the upward unbounded Y-tree structure are simulated by the left, right and adoptive paths (in bold in the figure). Notice that the states coming from a left and right x-labelled node are collected by a y-labelled node and then forwarded to the x-labelled father node. Let us consider now the complete labelling of the figure 16 given in figure 17. States t_0 an t_1 label nodes in the left and right path, respectively, from a x-labelled node to a y-labelled node. State t_1 labels nodes in the path between a y-labelled and a x-labelled state. State t_2 labels nodes in the adoptive path between two x-labelled nodes. The nodes in the region bounded by a t_1-labelled and t_2-labelled path (i.e. the region where a left and right son are collected) are labelled by symbol b. The region of nodes where an adoptive son is transmitted (bounded by a t_2-labelled path) are labelled by symbol a. All the remaining nodes are labelled by symbol c. Now, it is not difficult to define an automaton whose successful computations are exactly those producing the above described labelling

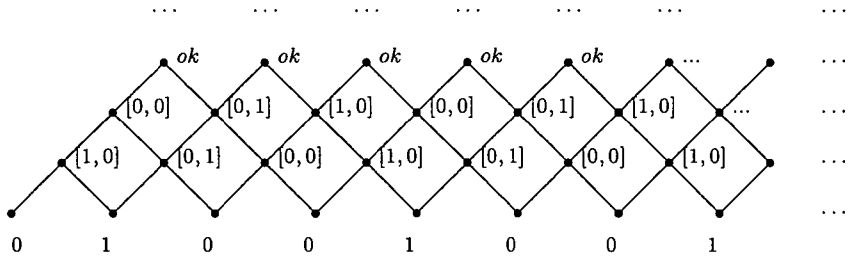

Figure 15: A successful computation for the ω-word $(010)^\omega$.

of the upward unbounded trellis structure. Such an automaton can then be easily adapted to simulate any given Y-tree automaton.

We prove now that the inclusion is proper, by showing that the trellis ω-language L of Example 4 is not a Y-tree ω-language. The proof is by contradiction. Let us assume that there exists a Y-tree automaton $\mathcal{Y}$ having n states and accepting L. For a suitable m, we take now $s = 2^{2^{m+1}-1} > n^{m+1}$ (different) ω-words $\alpha_1, \ldots, \alpha_s$ of L such that $\alpha_i = w_i{}^\omega$, $|w_i| = 2^{m+1} - 1$, $w_i \neq w_j$, $1 \leq i \neq j \leq s$. Let ψ_i be a successful Y-run of $\mathcal{Y}$ for α_i (with $1 \leq i \leq s$). Since there are at most $n^{m+1} < s$ different strings of states of length $m+1$, there must be two runs ψ_j and ψ_i with $i \neq j$ such that $\psi_i(m) = \psi_j(m)$. This immediately implies, by definition of Y-run, that the ω-word

$$\alpha' = \alpha_j(0, 2^{m+1} - 2) \cdot \alpha_i(2^{m+1} - 1, \omega) = w_j \cdot w_i{}^\omega$$

has a successful Y-run, namely the run ψ such that $\psi(k) = \psi_j(k)$, for $0 \leq k \leq m$ and $\psi = \psi_j$, elsewhere. This leads to a contradiction since $\alpha' \notin L$. □

Since the emptiness problem is undecidable for the class of Y-tree ω-languages we have the following corollary.

Corollary 11 *The emptiness problem for the class of trellis ω-languages is not decidable.*

By applying standard techniques, the following closure properties can be proved.

Proposition 12 *The class of trellis ω-languages is closed under union, intersection and projection.*

Moreover, it remains an open question whether the class of trellis ω-languages is closed under complementation. Also in the case of finitary languages accepted by trellis automata the closure under complementation remains still an hard open question. In [9] it is suggested that the question has probably negative answer since, otherwise, a positive answer would imply the closure under complementation of the well known complexity class NP.

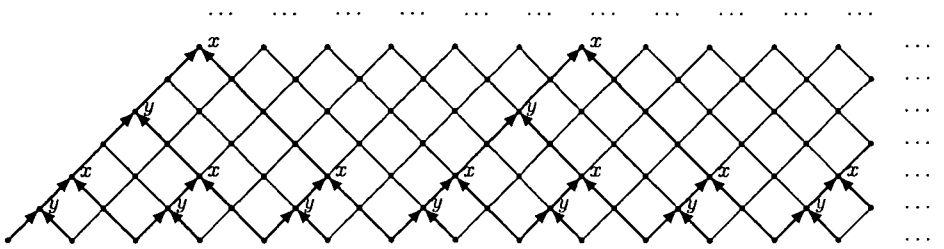

Figure 16: Embedding of the upward unbounded Y-tree structure in the upward unbounded trellis structure.

Let us consider now the logical definability of trellis ω-languages. In order to capture the class of ω-languages accepted by trellis automata we extend $MSO[<]$ with the function $2x$ giving, for a number x, its double.

Example 5 *We show that the language encoding the function adj can be defined in* $MSO[<, 2x]$. *With reference to the upward unbound* Y*-tree structure, it is easy to see that if* x *belongs to the column* c *which includes the odd number* k, *then* $adj(x)$ *belongs to the column* c' *which includes the odd number* $2k+3$. *Notice also that if* c *is the* i*-th column (with* $i > 0$*), then* $adj(x)$ *is the least element in* c' *greater than* x. *Otherwise, if* c *is the* 0*-th column, then* $adj(x)$ *is the double of the least element in* c' *greater than* x.
The formula $Column(x, y)$ *(i.e.* y *belongs to the the column containing the odd number* x*) stands for*

$$Odd(x) \wedge \exists X(X(x) \wedge X(y) \wedge$$
$$\forall z, s(X(z) \Rightarrow X(2z) \wedge Odd(z) \Rightarrow z = x \wedge (X(s) \wedge 2z = s) \Rightarrow X(z)),$$

where $Odd(x)$ *stands for* $\neg\exists y(x = 2y)$.
Now, the formula defining the function adj is as follows:

$$\begin{array}{c}\exists x, y(x < y \wedge Q_1(x) \wedge Q_2(y) \wedge \forall z(z \neq x \wedge z \neq y \Rightarrow Q_0(z)) \wedge \\ \exists z, s(Column(z, x) \wedge Column(s, y) \wedge s = 2z + 3 \wedge \\ (z = 1 \Rightarrow \exists h(Column(s, h) \wedge x < h \wedge y = 2h \wedge \\ \forall r(Column(s, r) \wedge x < r \Rightarrow h \leq r))) \wedge \\ (z > 1 \Rightarrow \forall h(Column(s, h) \wedge x < h \Rightarrow h \leq y))),\end{array}$$

with $z = 1$ *and* $z > 1$ *obvious shorthands.*

As an immediate consequence of Example 5 we have the following expressiveness property.

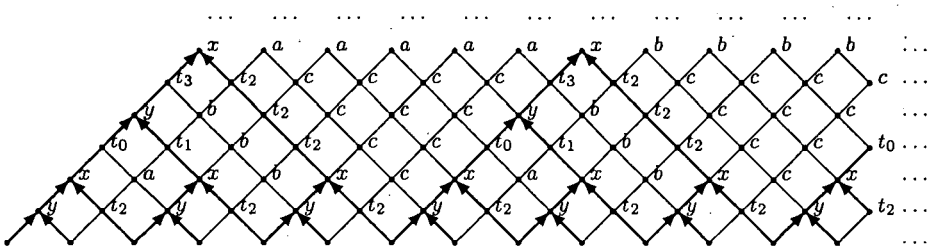

Figure 17: A Y-tree labelling of an upward unbounded trellis structure

Proposition 13 *$MSO[<, 2x]$ is at least as expressive as $EMSO[<, adj]$.*

Notice that if $EMSO[<, adj]$ were a characterization of Y-tree ω-languages, then $MSO[<, 2x]$ would be a proper extension of $EMSO[<, adj]$. The double function allows impose on the set of natural numbers an upward unbounded trellis structure (see figure 18). The sets identifying the i-th layer and column of the structure are defined exactly as in Eq.5 and Eq.6, respectively. The left son of a number x is the half part of x. As concerns the right son, we proceed as follows. If x belongs to the 0-th column, then the right son of x is

$$y = min\{z : z \text{ belongs to the 1-th column}, z > x\}.$$

If x belongs to the i-th column (with $i > 0$), then the right son is

$$y = max\{z : z \text{ belongs to the } i + 1 - \text{th column}, z < x\}.$$

Theorem 14 *Any ω-language accepted by a trellis automaton is $MSO[<, 2x]$-definable.*

Proof. Assume without loss of generality that the set of states of $\mathcal{T}$ is $Q = 1, \ldots, m$. The idea of the proof is similar to that of Theorem 6. In particular, we shall consider predicates $Y_1, \ldots, Y_m$ assuming that a number x belongs to Y_i iff the state i labels the node x of the unbounded trellis structure. We introduce some short notations:
$x \xrightarrow{L} i$ (i.e. the left son of x is associated with the state i) stands for
$\exists y(2y = x \wedge Y_i(y))$.
$Right - son(x, y)$ (i.e. y is the right son of x) stands for

$$\exists r(Column(r, x) \wedge Column(r + 2, y) \wedge$$
$$(r = 1 \Rightarrow y > x \wedge \forall s(Column(v + 2, s) \wedge s > x \Rightarrow y \leq s)) \wedge$$
$$(r > 1 \Rightarrow y < x \wedge \forall s(Column(r + 2, s) \wedge s < x \Rightarrow y \geq s))).$$

$x \xrightarrow{R} i$ (i.e. the right son of x is associated with the state i) stands for
$\exists y\,(Right - son(x,y) \wedge Y_i(y))$.
Given a trellis automaton $\mathcal{T}$, a formula $\phi \in MSO[<, 2x]$ accepting $\mathcal{L}_\omega(\mathcal{T})$ is the following:
$\exists Y_1, \ldots, Y_m$ (

$$\bigwedge_{i \neq j} \neg \exists y\,(Y_i(y) \wedge Y_j(y)) \wedge \tag{15}$$

$$\forall x \left(Odd(x) \Rightarrow \left(\bigwedge_{a,b \in \Sigma} (Q_a(x-1) \wedge Q_b(x)) \Rightarrow \bigvee_{\{i : \langle j,k,i \rangle \in f, \langle a,j \rangle, \langle b,k \rangle \in g\}} Y_i(x) \right) \right) \wedge \tag{16}$$

$$\forall z \left(\bigwedge_{i,j \in Q} \left((z \xrightarrow{L} i \wedge z \xrightarrow{R} j) \Rightarrow \bigvee_{\{k : \langle i,j,k \rangle \in f\}} Y_k(z) \right) \right) \wedge \tag{17}$$

$$\bigvee_{i \in F} \forall x \exists y\,(x < y \wedge Column(1,y) \wedge Y_i(y)) \tag{18}$$

). □

The fact that $MSO[<, 2x]$ is undecidable, is well known (see [4, 14]). Moreover, if trellis ω-languages were closed under complementation, then $MSO[<, 2x]$ would be a characterisation of trellis ω-languages. This would follow from the fact that the ω-language which encodes the function *double* (as well as its complement) can be accepted by a trellis automaton, and from the closure of trellis ω-languages under union, intersection and projection (see Proposition 12).

Proposition 15 *There exists a deterministic trellis automaton accepting the ω-language encoding the double function.*

Proof. We define a deterministic trellis automaton $\mathcal{T} = \langle \Sigma, Q, Q_0, f, g \rangle$ which accepts ω-words having the form $0^x 1.0^{x-1}.2.0^\omega$ with $x \geq 1$. This automaton is as follows:

- $\Sigma = \{0, 1, 2\}$;
- $Q = \{0, 1, m_l, m_r, ok\}$; $Q_0 = \{ok\}$;
- $g = \{\langle 0,0 \rangle, \langle 1, m_r \rangle, \langle 2,1 \rangle\}$;
- $f = \{\ \langle 0,0,0 \rangle, \langle 0, m_r, m_l \rangle, \langle m_r, 0, 0 \rangle, \langle 0, m_l, m_0 \rangle, \langle m_l, 0, m_r \rangle,$
  $\langle 0,1,1 \rangle, \langle 1,0,0 \rangle, \langle m_r, 1, 1 \rangle, \langle m_l, 1, ok \rangle, \langle ok, 0, ok \rangle\}$.

□

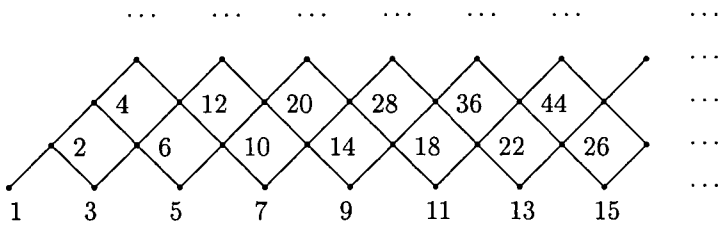

Figure 18: The upward unbounded trellis structure on $\mathbb{N}$.

6 Conclusions

In this paper we have introduced the classes of ω-languages accepted by Y-tree and trellis systolic automata, and we have studied some of their expressiveness, closure and decidability properties. We have, then, defined two monadic second order logics, $EMSO[<, adj]$ and $MSO[<, 2x]$, which allow to logically define the two mentioned classes of languages. The problem whether $EMSO[<, adj]$ is a characterisation of the class of ω-languages accepted by Y-tree automata remains open, and the first step of our future work will be devoted to solve it. The question whether $MSO[<, 2x]$ is a characterisation of the class of ω-languages accepted by trellis languages is, conversely, a problem which hardly will have an answer. Moreover, this paper provides the theoretical framework for investigating expressiveness issues of monadic second theories of time granularity interpreted on different types of ω-layered metric temporal structures (see [11] for a discussion on the open related questions). Our future work will consider also such an investigation.

References

[1] K. Culik II, J. Gruska, A. Salomaa, Systolic trellis automata. Part I, *International Journal of Computer Mathematics*, **15** (1984) 195–212.

[2] K. Culik II, J. Gruska, A. Salomaa, Systolic trellis automata, International *Journal of Computer Mathematics*, **16** (1984) 3–22.

[3] K. Culik II, J. Gruska, A. Salomaa, Systolic trellis automata: Stability, decidability and complexity, *Information and Control*, **71** (1986) 218–230.

[4] C.C. Elgot, M.O. Rabin, Decidability and undecidability of extensions of secon (first) order theory of (generalized) successor, *The Journal of Symbolic Logic*, **31** (1966) 169–181.

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

[6] E. Fachini, R. Franes, M. Napoli, M. Parente, BC-tree systolic automata: characterization and properties, *Journal of Computer and Artificial Intelligence*, **1** (1989) 53–82.

[7] E. Fachini, A. Monti, Chomsky hierarchy and systolic Y-tree automata, *Funtamenta Informaticae*, **29** (1996) 325–339.

[8] J. Gruska, Synthesis, structure and power of systolic computations, *Theoretical Computer Science*, **71** (1990) 47–78.

[9] H. Ibarra, S.M. Kim, Characterizations and computational complexity of systolictrellis automata, *Theoretical Computer Science*, **29** (1984) 123–153.

[10] A. Montanari, A. Peron, A. Policriti, Theories of ω-layered metric temporal structures: Expressiveness and decidability, *Logic Journal of the IGPL*, **7** (1999) 79–102.

[11] A. Montanari, A. Peron, A. Policriti, The way to go: Multi-level temporal logics, Proceedings of IWTS'99: 1st International Workshop on Specification and Verification of Timed Systems, N. Yonezaki (Ed.) Kyoto Research Institute of Mathematical Science, march 1999.

[12] A. Monti, A. Peron, A logical characterization of systolic languages, in: *Proc. STACS'98*, LNCS Vol. 1373 (Springer, Berlin,1998), 466–476.

[13] A. Monti, A. Peron, Systolic tree ω-languages: The operational and the logical view, *Theoretical Computer Science*, **233** (2000) 1–18.

[14] R.M. Robinson, Restricted set-theoretical definitions in aritmetic, Proc. Amer. Mat. Soc. Vol. 9, 1958, 238–242.

[15] M.Y. Vardi, Nontraditional applications of automata theory, LNCS Vol. 789 (Springer, Berlin,1994), 575–597.

[16] W. Thomas, Automata on infinite objects, in: J. van Leeuwen, ed., *Handbook of Theoretical Computer Science, Vol. B*, (Elsevier, Amsterdam, 1990) 133–191.

[17] W. Thomas, Languages, Automata and Logic, in: G. Rozemberg and A. Salomaa, eds., *Handbook of formal languages, Vol. 3*, Springer, 1997, 389–455.

Received March, 2001