ELSEVIER Theoretical Computer Science 290 (2003) 2075–2084

Theoretical Computer Science

www.elsevier.com/locate/tcs

Note

Characterization of real time iterative array by alternating device

Véronique Terrier

Département d'Informatique, GREYC, Campus II, Université de Caen, 14032 Caen Cedex, France

Received 5 October 2001; received in revised form 7 May 2002; accepted 13 May 2002
Communicated by B. Durand

Abstract

In this paper, we show that real time k-dimensional iterative arrays are equivalent through reverse to real time one-way alternating k-counter automata.

Keywords: Iterative array; Cellular automata; Alternating counter automata

1. Introduction

Cellular arrays are recognized to be a very relevant and natural model of massively parallel computation. Alternating computation, extension of nondeterminism, has also a parallel behavior. This common feature of modeling parallelism has been stressed by Ito et al. [5] for a kind of two-dimensional cellular arrays and alternating finite automata working on two-dimensional languages: they have shown that deterministic two-dimensional on-line tessallation acceptors are equivalent to two-way (right and down moves) two-dimensional alternating finite automata through 180°-rotation. As real time one-dimensional cellular automata can be viewed as a deterministic two-dimensional on-line tessallation acceptor where the input is restricted to square tape whose the symbols but the top row are all blank, they conclude that real time one-dimensional cellular automata can be simulated by two-dimensional alternating finite automata. Further, if we consider rebound automaton introduced in [7] which is a two-dimensional alternating finite automaton where the input are also square tapes whose

E-mail address: veroniqu@info.unicaen.fr (V. Terrier).

0304-3975/02/$ - see front matter
PII: S0304-3975(02)00489-9

the symbols but the top row are all blank, it follows that alternating rebound automata starting from the bottom right corner with only up and left moves are equivalent to real time cellular automata. Zhang et al. [8] have shown that alternating rebound automata are equivalent to real time two-way alternating one counter automata. So real time one-dimensional cellular automata (*real time* 1-*CA*) can be simulated by real time two-way alternating one counter automata (*real time* $2ACA(1)$). Here, we will precise the relationship between alternating counter automata and cellular arrays in considering iterative arrays which are cellular arrays with sequential input mode. We will show that real time one-way alternating k-counter automata (*real time* $1ACA(k)$) are equivalent through reverse to real time k-dimensional iterative arrays (*real time* k-*IA*). As consequence for one-dimensional array, we have (*real time* $1ACA(k))^{\text{reverse}} =$ *real time* 1-*IA* $\subsetneq$ *real time* 1-*CA* $\subseteq$ *real time* $2ACA(1)$. Concerning iterative arrays, Cole [2] has shown that the power of real-time IA strictly increases with the dimension of the space and that real time 1-IA is not closed under concatenation, Kleene closure and reversal. Moreover Rosenberg [6] has set that for any k, real time k-IA are not closed under concatenation, Kleene closure, reversal, sequential machine mapping nor the operations of taking derivatives and quotients. The same results have been obtained by Inoue et al. [4] for alternating counter automata: real time $1\text{ACA}(k)$ is less powerful than real time $1\text{ACA}(k+1)$ and real time $1\text{ACA}(k)$ is not closed under concatenation, Kleene closure, reversal and length-preserving homomorphism. As real time k-IA is obviously closed under complementation, the question whether real time $1\text{ACA}(k)$ is closed under complementation becomes trivial.

The definitions of iterative arrays and one-way alternating counter automata are specified in Section 2. In Section 3, we describe how to simulate through reverse a real time k-IA by a real time 1ACA (k) and in Section 4 the converse.

2. Definitions

2.1. Iterative array

A k-dimensional iterative array is a k-dimensional array of finite automata (cells) indexed by Z^k. All cells are identical except the communicating cell indexed by $(0,\ldots,0)$ which receives sequentially the input and gives the result. The cells evolve synchronously at discrete time steps according their local neighborhood and also according the input for the communicating cell. Formally a k-IA is defined by $(\Sigma, S, F, V, \delta_0, \delta, \lambda)$ where Σ is the input alphabet, S the set of states, $F \subset S$ the set of accepting states; $V = \{(x_1^1,\ldots,x_k^1),\ldots,(x_1^v,\ldots,x_k^v)\} \subset Z^k$ is the neighborhood; δ from $S^{|V|}$ into S is the transition function of the noncommunicating cells and δ_0 from $(\Sigma \cup \{B\}) \times S^{|V|}$ into S the transition function of the communicating cell (B is a blank symbol); $\lambda \in S$ is the quiescent state which verifies $\delta(\lambda,\ldots,\lambda) = \lambda$.

We denote by $\langle (u_1,\ldots,u_k), t\rangle$ the state of the cell $(u_1,\ldots,u_k)$ at time t. At time 0, all cells are in the quiescent state λ: $\langle (u_1,\ldots,u_k), 0\rangle = \lambda$. IA has a sequential input mode, that means at time $i = 1,\ldots,n$ the ith digit a_i of the input $a_1 \cdots a_n$ is fed to the cell $(0,\ldots,0)$ and at time $i > n$ the blank symbol B is fed to the cell $(0,\ldots,0)$. At time t

on the cell $(u_1,\dots,u_k)$ we have

$$\langle (u_1,\dots,u_k),t\rangle = \begin{cases} \delta_0(a_t,\langle (x_1^1,\dots,x_k^1),t-1\rangle,\dots,\langle (x_1^v,\dots,x_k^v),t-1\rangle) & \text{if } u_1=\cdots=u_k=0,\\ \delta(\langle (u_1+x_1^1,\dots,u_k+x_k^1),t-1\rangle,\dots, & \\ \quad \langle (u_1+x_1^v,\dots,u_k+x_k^v),t-1\rangle) & \text{else.} \end{cases}$$

Let $f:N\to N$ be a strictly increasing function. An IA recognizes a language L in time f if and only if it accepts the words $w\in L$ of length n in $f(n)$ steps i.e. if the communicating cell enters an accepting state at time $f(n)$. The real time corresponds to $f(n)=n$ and the linear time to $f(n)=cn$ for some constant $c>1$. Cole [2] has shown that the computing capability of the array is preserved even if the neighborhood is restricted to the Von Neumann one: $V_{\text{Von Neumann}}=\{(x_1,\dots,x_k)\in Z^k:\ \sum |x_i|\leqslant 1\}$ and that the same is true for the Moore neighborhood: $V_{\text{Moore}}=\{(x_1,\dots,x_k)\in Z^k:\ |x_i|\leqslant 1\}$. Therefore in the following, we will only consider the Moore neighborhood. The working area of a real time 1-IA on the input $a_1\cdots a_7$ is depicted in Fig. 1.

2.2. One-way alternating counter automata

As defined in [4] a one-way alternating k-counter automaton is a one-way finite automaton with k integer counters which has both existential and universal branching modes. Formally, an 1ACA(k) M is a sextuplet $(\Sigma,S,U,F,s_{\text{init}},\delta)$ where Σ is the input alphabet, S the set of states, $U\subset S$ the set of universal states, $F\subset S$ the set of accepting states, s_{init} the initial state and $\delta:S\times(\Sigma\cup\{\$\})\times\{0,1\}^k\to\mathscr{P}(S\times\{-1,0,1\}^k\times\{right, nomove\})$ the transition function ($\$ is an end-marker). $S\backslash U$ describes the set of existential states.

The input $x\in\Sigma^*$ is delimited with the end-marker $ placed on the right. M starts the computation in initial state s_{init}, its head pointing on the first input symbol and with its k counters empty. In each step, according its current state s, the symbol a scanned and whether its counters are zero or not, M enters a state s', modifies the counters c_i of $\varepsilon_i=-1,0$ or 1 and moves eventually its head to the right for any $(s',(\varepsilon_1,\dots,\varepsilon_k),move)\in\delta(s,a,(\mathbb{I}(c_1),\dots,\mathbb{I}(c_k)))$. $\mathbb{I}$ refers to the indicator function: $\mathbb{I}(c)=1$ if $c\neq 0$ and 0 if $c=0$. We assume that M can enter an accepting state only when reaching the right end-marker $. A *configuration* of M is a description of M at some step of the computation, it is an element $(a_i\cdots a_n,s,(c_1,\dots,c_k))$ of $\Sigma^*\times S\times Z^k$ giving the portion of the input $a_1\cdots a_n$ unread, the current state s and the current values $c_1,\dots,c_k$ of the k counters. For two configurations I and J we write $I\vdash J$ if J is a successor of I in one step of computation. A configuration is *universal* or *existential* according its state s is universal or existential. A configuration is *accepting* if its state is accepting and the whole input is read. The *initial* configuration of M on input $w=a_1\cdots a_n\in\Sigma^*$ is $(a_1\cdots a_n,s_{\text{init}},(0,\dots,0))$.

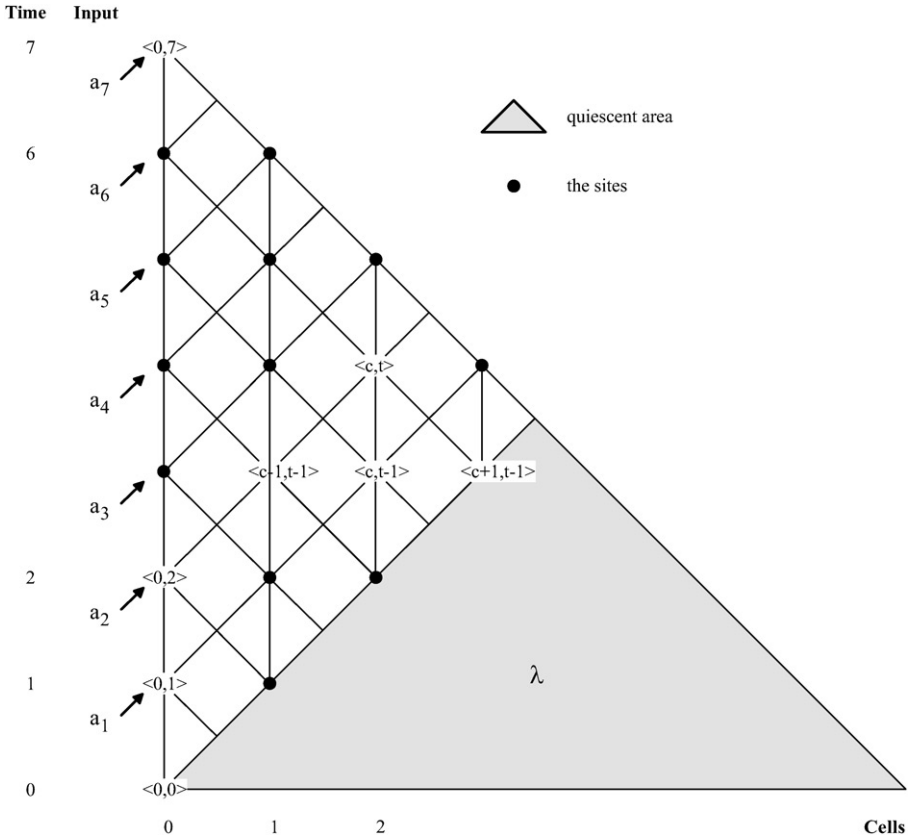

Fig. 1. A one-dimensional iterative array.

A *computation tree* of M is a labeled tree which satisfies the following conditions:

1. each node is labeled by a configuration;
2. if an internal node is labeled by an existential configuration I, it has exactly one child J such that $I \vdash J$;
3. if an internal node is labeled by an universal configuration I with successors $\{J: I \vdash J\} = \{J_1, \ldots, J_r\}$, it has exactly r children $J_1, \ldots, J_r$.

An *accepting computation tree* of M on input w is a computation tree whose root is labeled by the initial configuration of M on w and whose leaves are labeled by accepting configurations.

An input w is accepted by M in t steps if there is an accepting computation tree of M on w whose height is t. M recognizes a language L in time f if it accepts the words $w \in L$ of length n in $f(n)$ steps. The real time corresponds to $f(n) = n$ and the linear time to $f(n) = cn$ for any constant $c > 1$. Note that a machine M which works in real time moves its head to the right at each step, its transition function is from $S \times (\Sigma \cup \{\$\}) \times \{0,1\}$ into $\mathscr{P}(S \times \{-1,0,1\}^k)$ and its computation trees have $n+1$ levels.

Observe that, without loss of generality, for any constant integer r, we can consider that M discriminates whether its counters are $-r-1,\ldots,0,\ldots,r+1$ or different, and modify the counters of $-r,\ldots,0,\ldots,r$. Indeed to come down to the usual definition, it suffices to divide the counters by r and to record the remainders and the signs in the state. Actually it is as for IA whose neighborhood can be restricted to the Von Neumann one.

3. Simulation of a real time k-dimensional iterative array by a real time one-way alternating k-counter automaton

Let $A=(\Sigma,S,F,V_{\text{Moore}},\delta_0,\delta,\lambda)$ be a k-dimensional iterative array and L the words accepted in real time by A. First, we describe a 1ACA(k) M which recognizes L^{R} the reverse of L in time $2n-1$. Second, we simulate M by a 1ACA(k) M' working in real time.

Let $(x_1^1,\ldots,x_k^1),\ldots,(x_1^v,\ldots,x_k^v)$ the $v=3^k$ relative positions of the Moore neighborhood. Let $w=a_1\cdots a_n$ be a word of L. Let $\langle(c_1,\ldots,c_k),t\rangle$ denote the state of the cell $(c_1,\ldots,c_k)$ at time t of the k-IA A on the input w. We will define an 1ACA(k) M which admits an accepting computation tree on the input w^{R} such that at level $2(n-t)$ the nodes with counter values $c_1,\ldots,c_k$ record the state $\langle(c_1,\ldots,c_k),t\rangle$ of the k-IA. The alternating machine M will go back one transition of the IA with an existential branching where it guesses the v antecedents following by an universal branching where it updates the counters. By instance in dimension $k=1$ the transition $\delta(\langle c-1,t-1\rangle,\langle c,t-1\rangle,\langle c+1,t-1\rangle)=\langle c,t\rangle$ will be depicted by

$$\begin{pmatrix} a_t\cdots a_1 \\ \langle c,t\rangle \\ c \end{pmatrix}$$
$$\downarrow$$
$$\begin{pmatrix} a_{t-1}\cdots a_1 \\ \langle c-1,t-1\rangle,\langle c,t-1\rangle,\langle c+1,t-1\rangle \\ c \end{pmatrix}$$
$$\swarrow \quad \downarrow \quad \searrow$$
$$\begin{pmatrix} a_{t-1}\cdots a_1 \\ \langle c-1,t-1\rangle \\ c-1 \end{pmatrix}\begin{pmatrix} a_{t-1}\cdots a_1 \\ \langle c,t-1\rangle \\ c \end{pmatrix}\begin{pmatrix} a_{t-1}\cdots a_1 \\ \langle c+1,t-1\rangle \\ c+1 \end{pmatrix}$$

Formally the 1ACA(k) M is defined by the sextuplet $(\Sigma,S_M,U_M,F_M,s_{\text{init}},\delta_M)$ where the set of states $S_M=\{s_{\text{init}}\}\cup S\cup S^v$ ($v=3^k$ the size of the neighborhood), the set of universal states $U_M=S^v$, the set of accepting states $F_M=\{(\lambda,\ldots,\lambda)\}$ (λ the quiescent state of A) and the transition function is defined by

- $\delta_M(s_{\text{init}},a,(0,\ldots,0))=\{((s^1,\ldots,s^v),(0,\ldots,0),\text{right}):\ \delta_0(a,s^1,\ldots,s^v)\in F\}$
- For $s\in S$, $\delta_M(s,a,(i_1,\ldots,i_k))=\{((s^1,\ldots,s^v),(0,\ldots,0),\text{right}):\ \delta(s^1,\ldots,s^v)=s$ if $(i_1,\ldots,i_k)\neq(0,\ldots,0)$, else $\delta_0(a,s^1,\ldots,s^v)=s\}$
- For $(s^1,\ldots,s^v)\in S^v$, $\delta_M((s^1,\ldots,s^v),a,(i_1,\ldots,i_k))=\{(s^p,(x_1^p,\ldots,x_k^p),\text{no move}):\ p=1,\ldots,v\}$

Fact 1. *A word $w=a_1\cdots a_n$ is accepted in real time by the k-IA A if and only if its reverse $w^{\mathrm{R}}=a_n\cdots a_1$ is accepted by the $1ACA(k)$ M.*

Proof. First assume that w is accepted in real time by A. Consider the computation tree of M on the input $w^{\mathrm{R}}=a_n\cdots a_1$. At level 0, the root labeled by the existential configuration $(a_n\cdots a_1, s_{\mathrm{init}}, (0,\ldots,0))$ has one successor $(a_{n-1}\cdots a_1, (\langle (x_1^1,\ldots,x_k^1), n-1\rangle,\ldots,\langle (x_1^v,\ldots,x_k^v), n-1\rangle), (0,\ldots,0))$; it is possible as $\delta_0(a_n, \langle (x_1^1,\ldots,x_k^1), n-1\rangle,\ldots, \langle (x_1^v,\ldots,x_k^v), n-1\rangle) = \langle (0,\ldots,0), n\rangle \in F$. At level $2(n-t)$ each node is labeled by an existential configuration $(a_t\cdots a_1, \langle (c_1,\ldots,c_k), t\rangle, (c_1,\ldots,c_k))$ and has one successor labeled by an universal configuration $(a_{t-1}\cdots a_1, (\langle (c_1+x_1^1,\ldots,c_k+x_k^1), t-1\rangle,\ldots,\langle (c_1+x_1^v,\ldots,c_k+x_k^v), t-1\rangle), (c_1,\ldots,c_k))$. At level $2(n-t)+1$ each node is labeled by an universal configuration $(a_{t-1}\cdots a_1, (\langle (c_1+x_1^1,\ldots,c_k+x_k^1), t-1\rangle,\ldots,\langle (c_1+x_1^v,\ldots, c_k+x_k^v), t-1\rangle), (c_1,\ldots,c_k))$ and has v successors labeled by existential configurations $(a_{t-1}\cdots a_1, \langle (c_1+x_1^p,\ldots,c_k+x_k^p), t-1\rangle, (c_1+x_1^p,\ldots,c_k+x_k^p))$. In particular at level $2n-1$, as $\langle (c_1,\ldots,c_k), 0\rangle = \lambda$, each node is labeled by the configuration $(\varepsilon, (\lambda,\ldots,\lambda), (c_1,\ldots,c_k))$ which is an accepting configuration. Thus, this computation tree is an accepting one and $w^{\mathrm{R}}=a_n\cdots a_1$ is accepted by M in $2n-1$ steps.

Conversely assume that $w^{\mathrm{R}}=a_n\cdots a_1$ is accepted by M. Consider an accepting computation tree of M on w^{R}. As M is specified, this tree must have the following structure: it is a strict alternance of existential branching with one input symbol consumed and universal branching with v children. So its height is $2n-1$. And at level $2n-1$, the v^{n-1} leaves are labeled by accepting configurations of the form $(\varepsilon, (\lambda,\ldots,\lambda), (c_1,\ldots,c_k))$. As all states s of the leaves $(\varepsilon, s, (c_1,\ldots,c_k))$ are identical, we will get by induction that each state s of any existential configurations $(a_t\cdots a_1, s, (c_1,\ldots,c_k))$ is uniquely determined by the input part $a_t\cdots a_1$ and the counters $c_1,\ldots,c_k$. More precisely, we will get that for t $(0<t<n)$, the states $\langle (c_1,\ldots,c_k), t\rangle$ of the IA correspond to the states s of all existential configurations $(a_t\cdots a_1, s, (c_1,\ldots,c_k))$ which label the nodes at level $2(n-t)>0$. Indeed at level $2n-2$, the nodes are labeled by existential configurations $(a_1, s, (c_1,\ldots,c_k))$ with $s=\lambda$ if $(c_1,\ldots,c_k)\neq(0,\ldots,0)$ else $s=\delta_0(a_1,\lambda,\ldots,\lambda)$; as initially all cells of the IA are in quiescent state λ, the labels at level $2n-2$ are clearly of the form $(a_1, \langle (c_1,\ldots,c_k), 1\rangle, (c_1,\ldots,c_k))$. Further, at level $2(n-t)>0$ the nodes are labeled by existential configurations $(a_t\cdots a_1, s, (c_1,\ldots,c_k))$ and have v grandchildren at level $2(n-t+1)$ labeled by existential configurations $(a_{t-1}\cdots a_1, s^p, (c_1+x_1^p,\ldots,c_k+x_k^p))$ such that $s=\delta_0(a, s^1,\ldots,s^v)$ if $(c_1,\ldots,c_k)=(0,\ldots,0)$, else $s=\delta(s^1,\ldots,s^v)$; so by induction the state $\langle (c_1,\ldots,c_k), t\rangle$ of the IA A corresponds to the state s of all labels $(a_t\cdots a_1, s, (c_1,\ldots,c_k))$ at level $2(n-t)>0$. Finally, the root labelled by an existential configuration $(a_n\cdots a_1, s_{\mathrm{init}}, (0,\ldots,0))$ has v grandchildren labeled by existential configurations $(a_{n-1}\cdots a_1, s^p, (x_1^p,\ldots,x_k^p))$ such that $\delta_0(a_n, s^1,\ldots,s^v)\in F$; hence the states $s^p=\langle (x_1^p,\ldots,x_k^p), n-1\rangle$ of the grandchildren of the root verify $\delta_0(a_n, s^1,\ldots,s^v)\in F$; so the state $\langle (0,\ldots,0), n\rangle$ of the IA is an accepting one and $a_1\cdots a_n\in L$. □

Finally, we will show the following fact.

Fact 2. *The $1ACA(k)$ M working in time $2n-1$ can be simulated by a real time $1ACA(k)$ M'.*

Proof. As done in [1], the idea is to simulate a sequence of $\exists\forall\exists\forall$ branchings of M by a sequence of $\exists\forall$ branchings of M'. In a sequence of $\exists\forall\exists\forall$ branchings M starting in an existential state s, guesses $s^1,\ldots,s^v$ the successors of s according the current symbol a_t; by an universal branching M updates for all s^p the counters c_i of x_i^p; M guesses $s^{p,1},\ldots,s^{p,v}$ the successors of each s^p according the current symbol a_{t+1}; by an universal branching M updates for all $s^{p,q}$ the counters c_i of x_i^q. Now M' from the state s proceeds in this way. In an existential branching, M' guesses an input symbol b (hopefully the next one); it guesses $s^{p,1},\ldots,s^{p,v}$ the v^2 successors according the current symbol a_t and b. In an universal branching M' checks whether the guessed symbol b corresponds to the current symbol a_{t+1}; it updates for all $s^{p,q}$ the counters c_i of $x_i^p + x_i^q$. Actually we consider that M' can discriminate whether its counters are -1, 0, 1 or different, and can modify the counters of -1, 0, 1 as well of -2 and 2.

Formally the 1ACA(k) M' is defined by the sextuplet $(\Sigma, S_{M'}, U_{M'}, F_{M'}, s_{\text{init}}, \delta_{M'})$ where the set of states $S_{M'} = \{s_{\text{init}}\} \cup S \cup ((\Sigma \cup \{\$\}) \times S^{v^2})$, the set of universal states $U_{M'} = (\Sigma \cup \{\$\}) \times S^{v^2}$, the set of accepting states $F_{M'} = \{(\$, \lambda, \ldots, \lambda), \lambda\}$ and the transition function is defined by

- For $s \in \{s_{\text{init}}\} \cup S$, $\delta_{M'}(s, a, (i_1, \ldots, i_k)) = \{((b, s^{1,1}, \ldots, s^{v,v}), (0, \ldots, 0)):$ it exists $(s^1, \ldots, s^v)$ such that $((s^1, \ldots, s^v), (0, \ldots, 0), \text{right}) \in \delta_M(s, a, (i_1, \ldots, i_k))$ and for all $p = 1, \ldots, v$ $((s^{p,1}, \ldots, s^{p,v}), (0, \ldots, 0), \text{right}) \in \delta_M(s^p, b, (\mathbb{I}(i_1 + x_1^p), \ldots, \mathbb{I}(i_k + x_k^p)))\}$
- For $(b, s^{1,1}, \ldots, s^{v,v}) \in U_{M'}$,

$$\delta_M((b, s^{1,1}, \ldots, s^{v,v}), a, (i_1, \ldots, i_k)) = \begin{cases} \emptyset & \text{if } a \neq b, \\ \{(s^{p,q}, (x_1^p + x_1^q, \ldots, x_k^p + x_k^q)): \ p, q = 1, \ldots, v\} & \text{else.} \end{cases}$$ □

According to Facts 1 and 2 we get:

Proposition 1. *The languages recognized in real time by k-IA are recognized in real time by* 1*ACA*(k).

4. Simulation through reverse of a real time 1ACA(k) by a real time k-IA

Let $M = (\Sigma, S, U, F, s_{\text{init}}, \delta)$ be a given 1ACA(k) and L the words accepted in real time by M. From M we will construct a k-IA which recognizes L^{R} in real time. In this purpose, we will first cast the accepting computation trees of M in a trellis; then we will embed this trellis in the working area of a real time k-IA.

The nodes of the trellis will be labeled by $[(c_1, \ldots, c_k), i]$ defined in this following way. Let $a_n \cdots a_1$ be a given input. For any integers $c_1, \ldots, c_k$ and i with $-n \leqslant c_1, \ldots, c_k \leqslant n$ and $0 \leqslant i \leqslant n$, $[(c_1, \ldots, c_k), i]$ represents the set of states $s \in S$ such that the configuration $(a_{n-i} \cdots a_1, s, (c_1, \ldots, c_k))$ is the root of an accepting computation tree. Note that by definition $a_n \cdots a_1 \in L \Leftrightarrow s_{\text{init}} \in [(0, \ldots, 0), 0]$.

Recall that an universal configuration leads to acceptance if and only if all its successors lead to acceptance and an existential configuration leads to acceptance if and only if it has a successor which leads to acceptance. Hence, we have that an universal state s belongs to $[(c_1,\dots,c_k),i]$ if and only if for all $(s',(x_1,\dots,x_k))\in\delta(s,a_{n-i},(\mathbb{I}(c_1),\dots,\mathbb{I}(c_k)))$, s' belongs to $[(c_1+x_1,\dots,c_k+x_k),i+1]$ and an existential state s belongs to $[(c_1,\dots,c_k),i]$ if and only if it exists $(s',(x_1,\dots,x_k))\in\delta(s,a_{n-i},(\mathbb{I}(c_1),\dots,\mathbb{I}(c_k)))$ such that s' belongs to $[(c_1+x_1,\dots,c_k+x_k),i+1]$. Moreover as M works in real time, at level n the leaves $(\varepsilon,s,(c_1,\dots,c_k))$ are accepting if and only if $s\in F$. So $[(c_1,\dots,c_k),n]=F$.

Formally we introduce a function g from $\Sigma\times\mathscr{P}(S)^{|V_{\text{Moore}}|}\times\{0,1\}^k$ into $\mathscr{P}(S)$ defined by $g(a,(S_1,\dots,S_v),(i_1,\dots,i_k))=\{s\in U:$ for all $(s',(x_1^r,\dots,x_k^r))\in\delta(s,a,(i_1,\dots,i_k))$, $s'\in S_r\}\cup\{s\in S\setminus U:$ it exists $(s',(x_1^r,\dots,x_k^r))\in\delta(s,a,(i_1,\dots,i_k))$, such that $s'\in S_r\}$. According the above observation we get the following fact.

Fact 3. *The data* $[(c_1,\dots,c_k),i]$ *with* $-n\leqslant c_1,\dots,c_k\leqslant n$ *and* $0\leqslant i\leqslant n$ *verify these equations of recurrence*:

$$[(c_1,\dots,c_k),n]=F$$

$$[(c_1,\dots,c_k),i]=g(a_{n-i},([(c_1+x_1^1,\dots,c_k+x_k^1),i+1],\dots,[(c_1+x_1^v,\dots,c_k+x_k^v),i+1]),(\mathbb{I}(c_1),\dots,\mathbb{I}(c_k))).$$

So to construct an IA which simulates M, we have to embed the computation of the $[(c_1,\dots,c_k),i]$'s related to the input $a_n\cdots a_1$ in the working area of the IA on the input $a_1\cdots a_n$. Let show the following fact.

Fact 4. *Each* $[(c_1,\dots,c_k),i]$ *can be computed on the site* $\langle(\lfloor c_1/(k+1)\rfloor,\dots,\lfloor c_k/(k+1)\rfloor),n-i+\lceil(c_1+\cdots+c_k)/(k+1)\rceil\rangle$.

Proof. See Fig. 2 for the case $k=1$. For $i=n$, the sites $\langle(\lfloor c_1/(k+1)\rfloor,\dots,\lfloor c_k/(k+1)\rfloor),\lceil(c_1+\cdots+c_k)/(k+1)\rceil\rangle$ can be characterized by an iterative array, then $F=[(c_1,\dots,c_k),i]$ can be recorded on these sites. For $i<n$, using $k+1$ steps of the equations of recurrence we can compute the data $[(c_1,\dots,c_k),i]$ from the input symbols $a_{n-i},\dots,a_{n-i-k}$ and the antecedents $[(c_1+\varepsilon_1^1+\cdots+\varepsilon_1^{k+1},\dots,c_k+\varepsilon_k^1+\cdots+\varepsilon_k^{k+1}),i+k+1]$, where $\varepsilon_i^j=-1,0$ or 1. By induction, these antecedents are computed on the surface

$$S=\left\{\left\langle\left(\left\lfloor\frac{c_1+\varepsilon_1^1+\cdots+\varepsilon_1^{k+1}}{k+1}\right\rfloor,\dots,\left\lfloor\frac{c_k+\varepsilon_k^1+\cdots+\varepsilon_k^{k+1}}{k+1}\right\rfloor\right),n-i-k-1+\left\lceil\frac{c_1+\cdots+c_k+\varepsilon_1^1+\cdots+\varepsilon_1^{k+1}+\cdots+\varepsilon_k^1+\cdots+\varepsilon_k^{k+1}}{k+1}\right\rceil\right\rangle:\ \varepsilon_i^j=-1,0\text{ or }1\right\}.$$

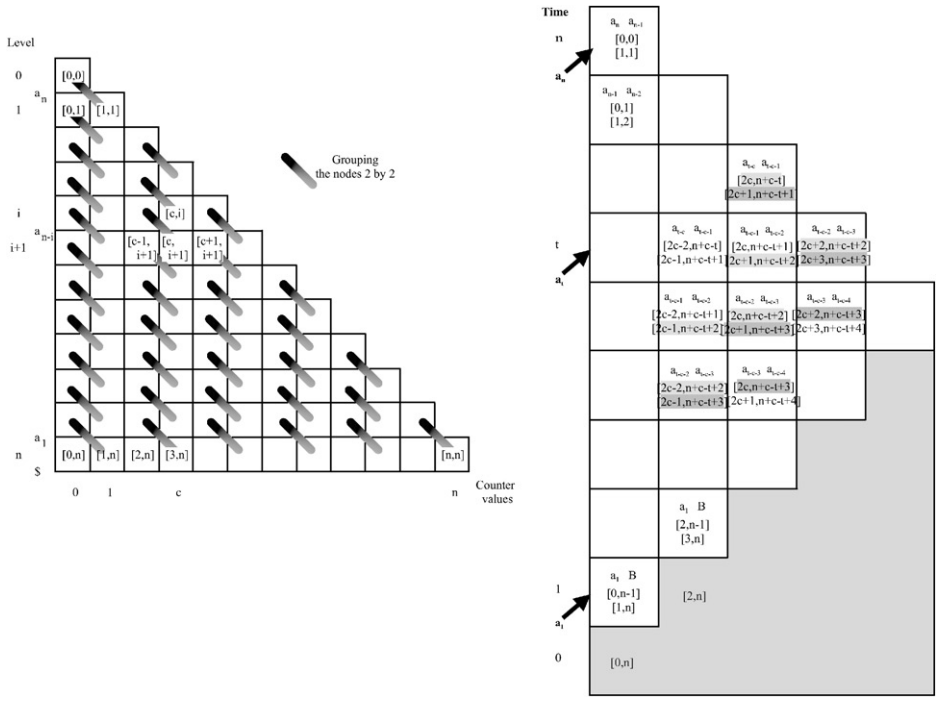

Fig. 2. From 1ACA(k) to k-IA.

Observe that

$$\left\lfloor \frac{c_r}{k+1} \right\rfloor - 1 \leqslant \left\lfloor \frac{c_r + \varepsilon_r^1 + \cdots + \varepsilon_r^{k+1}}{k+1} \right\rfloor \leqslant \left\lfloor \frac{c_r}{k+1} \right\rfloor + 1 \quad \text{for all } r = 1, \ldots, k$$

and

$$n - i - k - 1 + \left\lceil \frac{c_1 + \cdots + c_k + \varepsilon_1^1 + \cdots + \varepsilon_1^{k+1} + \cdots + \varepsilon_k^1 + \cdots + \varepsilon_k^{k+1}}{k+1} \right\rceil$$
$$\leqslant n - i + \left\lceil \frac{c_1 + \cdots + c_k}{k+1} \right\rceil - 1.$$

So the antecedents could be sent on the set of sites

$$P = \left\{ \left\langle \left(\left\lfloor \frac{c_1}{k+1} \right\rfloor + x_1^r, \ldots, \left\lfloor \frac{c_k}{k+1} \right\rfloor + x_k^r \right), n - i + \left\lceil \frac{c_1 + \cdots + c_k}{k+1} \right\rceil - 1 \right\rangle : (x_1^r, \ldots, x_k^r) \in V_{\text{Moore}} \right\}.$$

Remark that the moves of the data from S to P depend of the remainders of the c_i's modulo $k+1$ which can be computed by finite automata. Moreover the input symbols $a_{n-i-k}, \ldots, a_{n-i}$ can be spread from the initial cell 0 at times $t = n - i - k, \ldots, n - i$

towards the site $\langle(\lfloor c_1/(k+1)\rfloor,\ldots,\lfloor c_k/(k+1)\rfloor),n-i+\lceil(c_1+\cdots+c_k)/(k+1)\rceil\rangle$. So we have all the required information to compute $[(c_1,\ldots,c_k),i]$ on the site $\langle(\lfloor c_1/(k+1)\rfloor,\ldots,\lfloor c_k/(k+1)\rfloor),n-i+\lceil(c_1+\cdots+c_k)/(k+1)\rceil\rangle$. □

In particular, $[(0,\ldots,0),0]$ is computed on the site $((0,\ldots,0),n)$. Hence, the IA can test in real time on the input $a_1\cdots a_n$ whether $s_{\text{init}}\in[(0,\ldots,0),0]$, i.e. whether $a_n\cdots a_1\in L^{\mathrm{R}}$. From which one may conclude.

Proposition 2. *The languages recognized in real time by* $1ACA(k)$ *are recognized in real time by k-IA.*

5. Conclusion

In the same way, linear time k-IA can be simulated by linear time 1ACA(k).

Furthermore, compared to sequential input mode, parallel input mode admits an implicit synchronization of the cells at initial time. So synchronized alternating device studied in [3] is suitable for simulating cellular array with parallel input mode. We could show that real time one-way cellular automata can be simulated by linear time one-way synchronized alternating finite automata and real time cellular automata can be simulated through reverse by real time one-way synchronized alternating counter automata.

What about the converse, efficient simulations of linear time 1ACA and synchronized 1ACA by cellular arrays?

References

[1] T. Buchholz, A. Klein, M. Kutrib, Real-time language recognition by alternating cellular automata, IFIP Internat. Conf. on Theoretical Computer Science, Lecture Notes in Computer Science, Vol. 1872, Springer, Berlin, 2000, pp. 213–225.

[2] S.N. Cole, Real-time computation by n-dimensional iterative arrays of finite-state machine, IEEE Trans. Comput. C18 (1969) 349–365.

[3] J. Hromkovic, K. Inoue, A note on real time one-way synchronized alternating one-counter automata, Theoret. Comput. Sci. 108 (1993) 393–400.

[4] K. Inoue, A. Ito, I. Takanami, A note on real time one-way alternating multicounter machines, Theoret. Comput. Sci. 88 (1991) 287–296.

[5] A. Ito, K. Inoue, I. Takanami, Deterministic two-dimensional on-line tessallation acceptors are equivalent to two-way two-dimensional alternating finite automata through 180°-rotation, Theoret. Comput. Sci. 66 (1989) 273–287.

[6] A. Rosenberg, Real-time definable languages, J. ACM 14 (4) (1967) 645–662.

[7] K. Sugata, H. Umeo, K. Morita, On the computing abilities of rebound automata, IECE Trans. (1977) 367–374.

[8] L. Zhang, J. Xu, K. Inoue, A. Ito, Y. Wang, Alternating rebound turing machines, IECE Trans. Fund. E82-A (5) (1999) 745–755.