ELSEVIER

Theoretical Computer Science 180 (1997) 229–241

Theoretical Computer Science

Alternation on cellular automata [1]

Martín Matamala *

Departamento de Ingeniería Matemática, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile, Casilla 170-correo 3 Santiago, Chile

Received November 1995; revised December 1995
Communicated by E. Goles

Abstract

In this paper we consider several notions of alternation in cellular automata: non-uniform, uniform and weak alternation. We study relations among these notions and with alternating Turing machines. It is proved that the languages accepted in polynomial time by alternating Turing machines are those accepted by alternating cellular automata in polynomial time for all the proposed alternating cellular automata. In particular, this is true for the weak model where the difference between existential and universal states is omitted for all the cells except the first one. It is proved that real time alternation in cellular automata is strictly more powerful than real time alternation in Turing machines, with only one read–write tape. Moreover, it is shown that in linear time uniform and weak models agree.

Keywords: Cellular automata; Alternation; Nondeterminism

1. Introduction

A *one-dimensional cellular automata* (CA) is a bi-infinite line of identical finite machines each being in a state which is represented by a symbol taken in a finite alphabet Q. The transition function for all finite machines is given by a function $f: Q^3 \to Q$. The new state for a finite machine is obtained by looking its own state, x, and the states y and z of their right and left neighbors, respectively, and then by computing $f(z,x,y)$. A step on the cellular automata is achieved by computing, simultaneously, a new state for each finite machine.

A *configuration* for a CA is a function $C : \mathbb{Z} \to Q$ which assigns to each cell an element q in Q. Let C_t be a configuration for a CA at time t. Then a configuration at time $t+1$, C_{t+1}, is given by

$$C_{t+1}(i) = f(C_t(i-1), C_t(i), C_t(i+1)), \quad i \in \mathbb{Z}.$$

[1] This work was supported under grant CEE Marie Curie ERBCISTGT 920031.

* E-mail: mmatamal@dim.uchile.cl

PII S0304-3975(96)00214-9

Cellular automata have been studied in several contexts [2, 4, 5, 7, 10]. One of them is to study cellular automata as language acceptors [5, 9, 10]. In this sense, we consider $(U, \#, Q_a)$, where U is the input alphabet (a subset of Q), $\#$ is the quiescent state having the quiescent property $f(\#,\#,\#) = \#$ and $Q_a \subseteq Q$ is the set of accepting states. Initially, an input $u = u_0 \dots u_{n-1}$, $u_i \in U, i = 0, \dots, n-1$, is put in cells $0, \dots, n-1$ defining configuration C_0^u given by:

$$C_0^u(i) = \begin{cases} \# & i < 0 \text{ or } i \geqslant n \\ u_i & i \in \{0, \dots, n-1\} \end{cases} \quad : \quad \cdots \# u_0 u_1 \dots u_{n-1} \# \cdots$$

The input u is accepted if at some time the first cell enters an accepting state.

As for Turing machines, several generalizations can be introduced in cellular automaton models. Nondeterminism in finite automata or Turing machines gives to these devices the possibility of multiple transitions. So, the next state (next head state, next tape state, next move) for finite automata (Turing machine) can be chosen from a set of possible transitions.

Nondeterminism was defined for cellular automata by Smith in [10], where several language complexity results were obtained. In [8], it was studied another way to introduce nondeterminism on cellular automata. There, the authors relate their notion to space and time complexity classes in Turing machines.

In model given in [10], all the cells make an independent choice in the set of all possible next states given by a transition function $f: Q^3 \to 2^Q$. We will call that, nonuniform model.

An *nonuniform nondeterministic cellular automata* (NCA) is like a CA except that the transition function for the finite machines is a function $f: Q^3 \to 2^Q$. So, a step in NCA is achieved by choosing, independently for each finite machine, a state belongs to $f(x, y, z)$, where y is the state of the finite machine and, x and z are the states of its left and right neighbors, respectively.

In the uniform version of the above definition, at each computation step some determination for f is chosen, i.e. a function $f': Q^3 \to Q$ such that $f'(x, y, z) \in f(x, y, z)$ for every $(x, y, z) \in Q^3$ and this determination is used to update the cells.

We will see in Section 4 (Theorem 5) that the nonuniform models are at least as much powerful than uniform ones. This result will be obtained using a synchronization procedure.

We denote a (U)NCA, defined as above, by $A = (Q, f)$.

Another possible nondeterministic version consists in permitting nondeterministic transitions only in the first cell. So, we give the following definition.

A *weak nondeterministic cellular automata* (WNCA) is like a CA but the transition function for cell at *origin* (the cell associated to position 0) is a function $f_0: Q \to 2^Q$. We denote a WNCA by $A = (Q, f, f_0)$, where $f: Q^3 \to Q$ is the transition function for any cell outside the origin.

The notion of configuration for NCA, UNCA and WNCA is as for CA. If C_t is a configuration at time t of a NCA (Q, f), then a configuration at time $t+1$, C_{t+1}, is

such that

$$C_{t+1}(i) \in f(C_t(i-1), C_t(i), C_t(i+1)).$$

We impose that $c_{t+1}(i) = c_{t+1}(j)$ whenever $c_t|_{\{i-1,i,i+1\}} = c_t|_{\{j-1,j,j+1\}}$, in the case of UNCA. This is equivalent to choose some determination f' for f and update cells with this determination.

For a WNCA, (Q, f, f_0) the next configuration is given by

$$C_{t+1}(i) = f(C_t(i-1), C_t(i), C_t(i+1)), \quad i \neq 0$$

and

$$C_{t+1}(0) \in f_0(C_t(-1), C_t(0), C_t(1)).$$

We will prove that in polynomial time these models have the same power. This will be a consequence of simulation results.

The results will be given in a more general framework by introducing the alternation in cellular automata. Alternation was introduced for finite automata and later for Turing machines as models for parallel computations [3]. It consists to classify the states (head state) of a finite automaton (Turing machine) as existential or universal ones. An existential state, as in the nondeterministic case, is interpreted as an option in the computation. The universal ones are associated to transitions where the device makes two or more simultaneous actions. Since alternation is strongly related to nondeterminism we will have three kinds of alternation in cellular automata: nonuniform, uniform and weak alternation. Other kinds of alternations could be defined but we will study only relations between these three models. This work will focus on suitable time or space restriction on models, so as to show where these models and Turing machines differ or agree.

The paper is organized as follows. Section 2 is devoted to definition of alternation on cellular automata and on Turing machine. Moreover, we define time complexity classes. In Section 3 we give simulations of alternating cellular automata models by alternating Turing machines (Theorem 1) and vice versa (Theorem 2). In the first case each step in the alternating cellular automata is simulated by $O(n)$ steps in the alternating Turing machine, where n is the size of the current configuration in the alternating cellular automata. In the second case, n consecutive actualizations in alternating Turing machine are performed by $2n$ actualizations with a weak alternating cellular automata. Theorem 1 is a natural extension of theorems given in [10]. Theorem 2 uses a different technique. It consists in moving the Turing machine tape representation around the Turing machine head representation which will be fixed at the origin cell of the cellular automata. It is easy to see that this simulation can be extended to uniform cellular automata and nonuniform ones. So, polynomial time restrictions are equivalent in all models.

In Section 4 we study special time restrictions: real time and linear time. In the former, we prove that there exists a language which is recognized by a cellular automata in real time and which is not recognized in real time by any alternating Turing machine, with only one read–write tape.

Relations between weak and uniform models will be given, permitting to conclude that in linear time both models agree. To complete these relations we prove that the nonuniform model is the most powerful of all the proposed models.

2. Definitions

2.1. Alternation on cellular automata and Turing machines

An *acceptor* (*uniform*) (*weak*) *alternating cellular automata* (U)(W)ACA is a 4-tuple $\mathscr{A} = (U, \#, Q_a, A)$ where $A = (Q, f, (f_0))$ is a (U)(W)NCA whose states are partitioned into existential and universal. $U \subset Q$ is the input alphabet, # is the quiescent state and $Q_a \subseteq Q$ is the set of accepting states. The (U)(W)NCA satisfies the quiescent property given by

$$f(\#,\#,\#) = \{\#\} \qquad f_0(\#,\#,\#) = \{\#\}$$

and # cannot be created.

In order to define when an input is accepted by an (U)(W)ACA we introduce the notion of *computation tree* [1]. A *computation tree*, $T(\mathscr{A}, u)$, for a (U)(W)ACA $\mathscr{A}$ on input u, is a finite tree whose nodes are labeled by configurations and whose root is labeled by C_0^u.

A *computation tree* for an ACA is built as follows:

Let α be a configuration with $r+s$ nonquiescent states. Cells $i_1, \ldots, i_s$ are in existential states and cells $j_1, \ldots, j_r$ are in universal ones.

The node labeled by α has K children labeled by configurations $\beta^{1,1,\ldots,1}, \ldots, \beta^{n_1,n_2,\ldots,n_r}$ where

$$f(\alpha_{j_k-1}, \alpha_{j_k}, \alpha_{j_k+1}) = \{q_1^k, \ldots, q_{n_k}^k\} \subseteq Q \quad \text{and} \quad K = \prod_{k=1}^{r} n_k.$$

We choose $\alpha'_{i_k} \in f(\alpha_{i_k-1}, \alpha_{i_k}, \alpha_{i_k+1})$ for $k = 1, \ldots, s$ and then we build configurations $\beta^{1,1,\ldots,1}, \ldots, \beta^{n_1,n_2,\ldots,n_r}$ as follows:

For every $(l_1, \ldots, l_r)$, $\beta^{l_1,\ldots,l_r}(i_k) = \alpha'_{i_k}$, $k = 1, \ldots, s$ and $\beta^{l_1,\ldots,l_r}(j_k) = q_{l_k}^k$, $k = 1, \ldots, r$.

So, configurations $\beta^{1,1,\ldots,1}, \ldots, \beta^{n_1,n_2,\ldots,n_r}$ agree in sites i_k and may differ in sites j_k.

For WACA and UACA models the *configurations* are divided into universal and existential. A configuration is existential (resp. universal) when the state at origin is existential (resp. universal).

A *computation tree* for WACA or UACA is a tree such that the children of any internal node labeled by an universal (existential) configuration include all (one) of the next configurations.

A tree $T(\mathscr{A}, u)$ is *accepting* if all its leaves are *accepting configurations*, i.e., configurations where the state at origin belongs to Q_a. We say that an ACA, UACA or WACA $\mathscr{A}$ *accepts* u if there exists an accepting tree for $\mathscr{A}$ on u. The language

accepted by a (W)(U)ACA $\mathscr{A}$ is given by

$$L(\mathscr{A}) = \{u/\mathscr{A} \text{ accepts } u\}.$$

An alternating Turing machine (ATM) is a generalization of a nondeterministic Turing machine (NTM), which is defined analogously to ACA. The states in an ATM are divided into existential and universal ones. The definition of a configuration is slightly different than that for CA's: it describes the current state of the machine, the contents of the tape (we suppose only one tape is used) and the position of the read–write head. A new configuration is obtained by performing a step in the Turing machine, i.e., by reading the tape symbol and, in a nondeterministic way, computing the new head state, the new tape symbol and the next head move. A configuration is existential (universal) if the head state is an existential (universal) one. A computation tree for an ATM M on input u, $T(M,u)$, is a finite tree whose nodes are labeled by configurations of M on u, such that the root is the initial configuration and the children of any internal node labeled by a universal (existential) configuration include all (one) of the next configurations. The notion of accepting tree, acceptance and language accepted by M are analogous to those of alternating cellular automata.

2.2. Languages and complexity classes

We say that $\mathscr{A}$, an ATM or (W)(U)ACA, is $t(n)$-time bounded if for any $u \in L(\mathscr{A})$ there exists an accepting tree whose height is less than $t(|u|)$, where $|u|$ is the length of the input word u.

We define, for a (U)(W)ACA and for an ATM the following complexity classes:

$(W)(U)ACA(t(n)) = \{L(\mathscr{A})/\mathscr{A}$ is a (W)(U)ACA $t(n)$-time bounded$\}$,

$ATM(t(n)) = \{L(\mathscr{A})/\mathscr{A}$ is an ATM $t(n)$-time bounded$\}$.

3. Simulating ACA in ATM

In this section, we prove that the evolution of an ACA C can be simulated by an ATM M. M simulates one step of C for a configuration of size n in $3n + 5$ steps. In the first $n + 1$ steps, M moves from left to right copying the state in cell i into cell $i + 1$. In the following $2(n + 2)$ steps, M moves from right to left making the actualization of all the cells.

Theorem 1. *Let C be an ACA. Then there exists an ATM M such that $L(C) = L(M)$. If C has time complexity $t(n)$ then M has time complexity $O(t^2(n))$.*

Proof. The simulation of one step of C can be done through $3n + 5$ steps of M. Set $Q_M = Q_C \times \{R,L,H\} \cup \{w\}$ where w is the accepting state, and the alphabet

$$A_M = Q_C \cup (Q_C \times \{*\}) \cup Q_C^2 \cup (Q_C^2 \times \{*\}).$$

Let α be the current configuration of C with nonquiescent states lying through j to $j+n$, then the associated configuration of M is such that:

- cell 0 contains $\binom{\alpha_0}{*}$.
- cell $i \neq 0$ contains α_i.
- M's head scans cell j and M is in state $(\#, R)$.

One step of C on α is simulated as follows by M:

First, M moves rightwards until it reads # (on cell $j+n$). If M is in state (p,R) and reads q (resp. $(q,*)$), then it enters state (q,R), writes (q,p) (resp. $(q,p,*)$) and moves rightwards.

Second, M is in state (p,R) and reads #, then it enters state $(\#,L)$, writes $(\#,p)$ and makes no move.

Third, M moves leftwards until it finds # (on cell $j-1$). If M is in state (r,L) and reads (q,p) (resp. $(q,p,*)$), then M enters state (q,H), writes (p,r) (resp. $(p,r,*)$) and makes no move. At this moment, M reads (p,r) and is in state (q,H). So, if $s \in f(p,q,r)$, then M enters state (q,L), may write s (resp. $(s,*)$) and moves leftwards.

Fourth, M is in state (p,R), reads # then, M enters state $(\#,H)$, write $(\#,p)$ and makes no move. Now, if $s \in f(\#,\#,p)$ then M enters state $(\#,R)$, may write s and makes no move.

At the beginning of each step, M's tape looks as follows:

#	α_j	α_{j+1}	$\cdots$	$(\alpha_0,*)$	$\cdots$	α_{j+n-1}	#

After the first and second steps M's tape is given by

#	α_j	α_{j+1}	$\cdots$	$(\alpha_0,*)$	$\cdots$	α_{j+n-1}	#	#
	#	α_j	$\cdots$	α_{-1}	$\cdots$	α_{j+n-2}	α_{j+n-1}	#

M's head is at cell $j+n$ and M is in state $(\#,L)$.

We show M's tape during the third step. At the begin of this step it looks as follows:

#	α_j	α_{j+1}	$\cdots$	$(\alpha_0,*)$	$\cdots$	α_k	α_{k+1}	$\cdots$	s_{j+n-1}	s_{j+n}	#
	#	α_j	$\cdots$	α_{-1}	$\cdots$	α_{k-1}	α_k	$\cdots$			

.

M's head is at cell $k+1$ and M is in state (α_{k+2},L).

The first sub-step leaves M's tape as follows

#	α_j	α_{j+1}	$\cdots$	$(\alpha_0,*)$	$\cdots$	α_k	α_k	$\cdots$	s_{j+n-1}	s_{j+n}	#
	#	α_j	$\cdots$	α_{-1}	$\cdots$	α_{k-1}	α_{k+2}	$\cdots$			

.

M is in state (α_{k+1},H) and M's head is at cell $k+1$.

The second sub-step, leaves M's tape as follows:

#	α_j	α_{j+1}	$\cdots$	$(\alpha_0,*)$	$\cdots$	α_k	s_{k+1}	$\cdots$	s_{j+n-1}	s_{j+n}	#
	#	α_j	$\cdots$	α_{-1}	$\cdots$	α_{k-1}		$\cdots$			

M is in state (α_{k+1},L) and M's head is at cell k.

The simulation finishes when M finds the symbol #. At this time M's tape looks as follows:

#	s_{j-1}	s_j	s_{j+1}	$\cdots$	$(s_0,*)$	$\cdots$	s_k	s_{k+1}	$\cdots$	s_{j+n-1}	s_{j+n}	#

M is in state $(\#,R)$ and M's head is at cell $j-1$.

The simulation of one transition of C takes $(n+1)+2(n+2)=3n+5$ steps, then we get the quadratic bound:

States of the form (q,H) will be existential if q is existential (resp. are universal if q is universal). Other states lead to deterministic moves.

Now, since accepting configurations in C were defined as having an accepting state in the origin, the accepting state w is entered in M's head when a state $\binom{q}{*}$ appears in the origin, with q an accepting state. □

Now, we want to give the inverse result, i.e., we want to simulate the behavior of an ATM into an ACA. Next theorem shows how to simulate the evolution of an ATM into a WACA. We could make it by associating a pair $\binom{q}{\sigma}$ to the scanned symbol and leave unchanged the other cells position. Since M could make nondeterministic transitions in any position, the WACA should go at the first position to make the simulation of M's transition and then came back to the position where M's head should move. This solution is 'sequential' like and we should obtain a simulation of M's evolution in quadratic time. We can improve this result with a more parallel solution.

Theorem 2. *Let M be an ATM. Then there exists a WACA C accepting $L(M)$. If M has $t(n)$-time complexity then C has $2t(n)$-time complexity.*

Proof. The idea is to move M's tape around the origin. M's head will always be at origin represented by vector states $\begin{pmatrix} a \\ q \\ \sigma \end{pmatrix}$ where $a \in \{0,1\}$ is a control character, q is the state of M and σ is the symbol scanned by M. When M moves right (resp. left) two families of signal, $\binom{rR}{\cdot}$ and $\binom{rL}{\cdot}$ (resp. $\binom{lL}{\cdot}$ and $\binom{lR}{\cdot}$) are produced moving M's head representation one position to the left (resp. right) of the origin. Signals $\binom{rR}{\cdot}$ and $\binom{lR}{\cdot}$ move rightwards while signals $\binom{rL}{\cdot}$ and $\binom{lL}{\cdot}$ move leftwards.

Before giving formal proof we show in an example how the simulation evolves. Suppose the evolution on M is

$$\#\underbrace{\sigma_0}_{q_0}\sigma_1\cdots\sigma_{n-1}\# \quad \to \#\sigma_0'\underbrace{\sigma_1}_{q_0'}\sigma_2\cdots\sigma_{n-1}\# \quad \to \#\sigma_0'\sigma_1'\underbrace{\sigma_2}_{q_0''}\sigma_3\cdots\sigma_{n-1}\# \to$$

$$\to \#\sigma_0'\underbrace{\sigma_1'}_{q_0''}\sigma_2'\cdots\sigma_{n-1}\# \to \#\underbrace{\sigma_0'}_{q_0'''}\sigma_1''\cdots\sigma_{n-1}\#$$

These steps are simulated in C by

$$
\begin{array}{ccccccccccc}
\# & \# & \# & \sigma_0 & \sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 & \sigma_5 & \sigma_6 & \\
\# & \# & \# & \begin{pmatrix} 0 \\ \sigma_0 \\ q_0 \end{pmatrix} & \sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 & \sigma_5 & \sigma_6 & \\
\# & \# & \begin{pmatrix} rL \\ \# \end{pmatrix} & \begin{pmatrix} 1 \\ \sigma_0 \\ q_0 \end{pmatrix} & \begin{pmatrix} rR \\ \sigma_1 \end{pmatrix} & \sigma_2 & \sigma_3 & \sigma_4 & \sigma_5 & \sigma_6 & \\
\# & \# & \sigma_0' & \begin{pmatrix} 0 \\ \sigma_1 \\ q_1 \end{pmatrix} & \sigma_2 & \begin{pmatrix} rR \\ \sigma_3 \end{pmatrix} & \sigma_3 & \sigma_4 & \sigma_5 & \sigma_6 & \\
\# & \# & \begin{pmatrix} rL \\ \sigma_0' \end{pmatrix} & \begin{pmatrix} 1 \\ \sigma_1 \\ q_1 \end{pmatrix} & \begin{pmatrix} R \\ \sigma_2 \end{pmatrix} & \sigma_3 & \begin{pmatrix} rR \\ \sigma_4 \end{pmatrix} & \sigma_4 & \sigma_5 & \sigma_6 & \\
\# & \begin{pmatrix} rL \\ \sigma_0' \\ \# \end{pmatrix} & \sigma_1' & \begin{pmatrix} 0 \\ \sigma_2 \\ q_2 \end{pmatrix} & \sigma_3 & \begin{pmatrix} rR \\ \sigma_4 \end{pmatrix} & \sigma_4 & \begin{pmatrix} rR \\ \sigma_5 \end{pmatrix} & \sigma_5 & \sigma_6 & \\
\begin{pmatrix} rL \\ \# \\ \# \end{pmatrix} & \sigma_0' & \begin{pmatrix} lL \\ \sigma_1' \end{pmatrix} & \begin{pmatrix} 1 \\ \sigma_2 \\ q_2 \end{pmatrix} & \begin{pmatrix} lR \\ \sigma_3 \end{pmatrix} & \sigma_4 & \begin{pmatrix} rR \\ \sigma_5 \end{pmatrix} & \sigma_5 & \begin{pmatrix} rR \\ \sigma_6 \end{pmatrix} & \sigma_6 & \\
\# & \begin{pmatrix} lL \\ \# \end{pmatrix} & \sigma_0' & \begin{pmatrix} 0 \\ \sigma_1' \\ q_3 \end{pmatrix} & \sigma_2' & \begin{pmatrix} lR \\ \sigma_3 \\ \sigma_4 \end{pmatrix} & \sigma_5 & \begin{pmatrix} rR \\ \sigma_6 \end{pmatrix} & \sigma_6 & \begin{pmatrix} rR \\ \# \end{pmatrix} & \\
\# & \# & \begin{pmatrix} lL \\ \sigma_0' \end{pmatrix} & \begin{pmatrix} 1 \\ \sigma_1' \\ q_2 \end{pmatrix} & \begin{pmatrix} lR \\ \sigma_2' \end{pmatrix} & \sigma_3 & \begin{pmatrix} lR \\ \sigma_4 \\ \sigma_5 \end{pmatrix} & \sigma_6 & \begin{pmatrix} rR \\ \# \end{pmatrix} & \# & \\
\# & \begin{pmatrix} lL \\ \# \end{pmatrix} & \# & \begin{pmatrix} 0 \\ \sigma_0' \\ q_4 \end{pmatrix} & \sigma_1'' & \begin{pmatrix} lR \\ \sigma_2' \\ \sigma_3 \end{pmatrix} & \sigma_4 & \begin{pmatrix} lR \\ \sigma_5 \\ \sigma_6 \end{pmatrix} & \# & \# &
\end{array}
$$

and signal evolutions are given by

$$
a\begin{pmatrix} rL \\ b \\ c \end{pmatrix} d \to \begin{cases} \begin{pmatrix} rL \\ c \\ a \end{pmatrix} b\ ? & b \neq \# \\ \# \ \#? & b = \# \end{cases} \qquad ab\begin{pmatrix} rL \\ c \end{pmatrix} d \to \begin{cases} a\begin{pmatrix} lL \\ a \end{pmatrix} c\ ? & a \neq \# \\ \#\#\#\ ? & a = \# \end{cases}
$$

$$
a\begin{pmatrix} lR \\ b \\ c \end{pmatrix} d \to \begin{cases} ?\ b\begin{pmatrix} lR \\ c \\ d \end{pmatrix} b\ ? & b \neq \# \\ ?\ \#\ \# & b = \# \end{cases} \qquad a\begin{pmatrix} rR \\ b \end{pmatrix} c\ d \to \begin{cases} ?\ b\begin{pmatrix} rR \\ d \end{pmatrix} d & b \neq \# \\ ?\#\#\# & b = \# \end{cases}
$$

and

$$ab\begin{pmatrix}0\\q\\c\end{pmatrix}d\ e\to a\begin{pmatrix}lL\\b\end{pmatrix}\begin{pmatrix}1\\q\\c\end{pmatrix}\begin{pmatrix}lR\\d\end{pmatrix}\ e\to ?a\begin{pmatrix}0\\q'\\b\end{pmatrix}c'\ \begin{pmatrix}lR\\d\\e\end{pmatrix}$$

when M's transition is $(q,c)\to(q',c',l)$ and

$$ab\begin{pmatrix}0\\q\\c\end{pmatrix}d\ e\to a\begin{pmatrix}rL\\b\end{pmatrix}\begin{pmatrix}1\\q\\c\end{pmatrix}\begin{pmatrix}rR\\d\end{pmatrix}\ e\to\begin{pmatrix}rL\\b\\a\end{pmatrix}\ c'\begin{pmatrix}0\\q'\\d\end{pmatrix}\ e\ ?$$

when M's transition is $(q,c)\to(q',c',r)$. Here, symbol ? at position i signifies that the information to update site i is incompleted.

Let (q_t,γ^t,i_t) be the configuration at time t for M. It can be proved, by induction on t that:

$\forall t\geqslant 0$

- $C^{2t+1}(0)=\begin{pmatrix}0\\q_t\\\gamma^t_{i_t}\end{pmatrix}$
- $\forall j\geqslant 1 C^{2t+|j|}(j)=\gamma^t_{i_t+j}$
- M moves right at time t then $\forall j\geqslant 2$

$$C^{2t+2+j}|_{\{-j-1,-j\}}=\begin{cases}\begin{pmatrix}rL\\\gamma^t_{i_t-j}\\\gamma^t_{i_t-j-1}\end{pmatrix}\gamma^t_{i_t-j+1} & \gamma^t_{i_t-j+1}\neq\#\\ \#\ \# & \text{otherwise}\end{cases}$$

$$C^{2t+2+j}|_{\{j,j+1\}}=\begin{cases}\gamma^t_{i_t+j+1}\begin{pmatrix}rR\\\gamma^t_{i_t+j+2}\end{pmatrix} & \gamma^t_{i_t+j+1}\neq\#\\ \#\ \# & \text{otherwise}\end{cases}$$

- M moves left at time t then $\forall j\geqslant 2$

$$C^{2t+2+j}|_{\{-j-1,-j\}}=\begin{cases}\begin{pmatrix}lL\\\gamma^t_{i_t-j-2}\end{pmatrix}\gamma^t_{i_t-j-1} & \gamma^t_{i_t-j-1}\neq\#\\ \#\ \# & \text{otherwise}\end{cases}$$

$$C^{2t+2+j}|_{\{j,j+1\}}=\begin{cases}\gamma^t_{i_t+j-1}\begin{pmatrix}lR\\\gamma^t_{i_t+j}\\\gamma^t_{i_t+j+1}\end{pmatrix} & \gamma^t_{i_t+j-1}\neq\#\\ \#\ \# & \text{otherwise.}\end{cases}$$

Therefore, each transition in M is simulated by two steps in C.

States $\begin{pmatrix} 1 \\ \sigma \\ q \end{pmatrix}$ are existential (resp. universal) when q is existential (resp. universal). All other states are deterministic. Moreover,

$$Q'_a = \left\{ \begin{pmatrix} 0 \\ \sigma \\ q \end{pmatrix} / q \text{ accepting state for } M \right\}.$$

So, any accepting computation tree for M on u has associated an accepting computation tree of C on u with twice its height. □

Remark 1. For UACA and for ACA we can make an analogous construction obtaining real time simulations. It is due to the fact that UACA and ACA can make nondeterministic transition in any cell.

Previous results prove that polynomial time languages for (W)(U)ACA and ATM agree. This fact came from polynomial simulation of these models.

Now, we analyze another time restrictions: linear and real time.

4. Real and linear time on ATM and ACA

4.1. $rATM \subset rACA$

In this subsection, we prove that there exists a language recognized in real time by a deterministic CA and which is not recognized by an ATM in real time.

This language is the palindrome set defined by

$$\mathscr{P} = \{w \in U^* / w = w^t\}, \quad \text{where } w = w_1 w_2 \dots w_n \text{ and } w^t = w_n \dots w_2 w_1.$$

Proposition 1. *$\mathscr{P} \in rCA$ and $P \notin rATM$.*

Proof. The former affirmation is proved in [5] for cellular automaton where the input is put step by step in the first cell and later in [10] it was proved also for the cellular automata defined like those defined here.

In order to prove the second affirmation we observe that if there were an ATM accepting $\mathscr{P}$ it should read all its inputs to give any answer. So, since we want to make it in real time the machine at time i reads the ith position and moves right. Since the machine cannot read symbols already read then it has to works like a finite automata. In [3, 6] it is proved that the power of a finite automata, nondeterministic finite automata and alternating finite automata are the same when they work as language recognizers. Furthermore, it is known that $\mathscr{P}$ is not a regular language which concludes the proof. □

Corollary. *$rATM \subset rACA$.*

Proof. Since $CA(t(n)) \subseteq ACA(t(n))$ from Proposition 1 we get that $\mathscr{P} \in rACA$. From Remark 1 we know that $rATM \subseteq rACA$ and so the result holds. □

4.2. Linear time in ACA

In this subsection, we prove that UACA and WACA agree in linear time. For that we prove two theorems which execute the simulation of UACA by WACA and reciprocally of WACA by UACA.

Theorem 3. *Let C be a WACA. Then, there exists a UACA C' such that $L(C) = L(C')$. If C is $t(n)$-time bounded then C' is $t(n)$-time bounded.*

Proof. Let f, f_0 and Q be the transition functions and the state set for C. We define the transition function for C' g as f in states which belong to Q and as f_0 in states which belong to $\{*\} \times Q$.

Let $T(C,u)$ be a computation tree for C on u. We build a computation tree, $T(C',u)$ for C' on u, which is accepting if and only if $T(C,u)$ is accepting. Let α a configuration in $T(C,u)$. We associate in $T(C',u)$ a configuration $\Phi(\alpha)$ given by

$$\Phi(\alpha) : \cdots \alpha_{-r} \cdots \begin{pmatrix} * \\ \alpha_0 \end{pmatrix} \alpha_1 \cdots \alpha_l \cdots$$

It is easy to see that the children for $\Phi(\alpha)$ in $T(C',u)$ are $\Phi(\beta^i)$ where β^i are the children of α in $T(C,u)$ □

Theorem 4. *Let C be an UACA. Then there exists a WACA C' such that $L(C) = L(C')$. If C is $t(n)$-time bounded then C' is $O(2t(n))$ time bounded.*

Proof. Let $T(C,u)$ be an accepting tree for u. Let p be its height. We build an accepting tree $T(C',u)$ for u of height $2p$. Let α^h, $h \geqslant 1$ a configuration labeling a node in $T(C,u)$, at level h, let $e_1, e_2, \ldots, e_h$ be the choices leading from the root to α^h and $\alpha^0, \alpha^1, \ldots, \alpha^h$ the configurations labeling the path in $T(C,u)$ defined by the choices $e_1, \ldots, e_h$. We associates to α^h a configuration in $T(C',u)$, $\Phi = \Phi(\alpha^h, \alpha^{h-1}, \ldots, \alpha^0, e_h, \ldots, e_0)$ given by

$$\Phi : \cdots \begin{pmatrix} L \\ e_{h-1} \\ \alpha_{-3}^{h-1} \\ \alpha_{-3}^{h-2} \end{pmatrix} \alpha_{-2}^{h-1} \begin{pmatrix} L \\ e_h \\ \alpha_{-1}^{h} \\ \alpha_{-1}^{h-1} \end{pmatrix} \begin{pmatrix} * \\ \alpha_0^h \end{pmatrix} \begin{pmatrix} R \\ e_h \\ \alpha_1^h \\ \alpha_1^{h-1} \end{pmatrix} \alpha_2^{h-1} \begin{pmatrix} R \\ e_{h-1} \\ \alpha_3^{h-1} \\ \alpha_3^{h-2} \end{pmatrix} \cdots$$

The transition in C' transforms Φ into Φ' given by

$$\Phi' : \cdots \alpha_{-3}^{h-1} \begin{pmatrix} L \\ e_h \\ \alpha_{-2}^{h} \\ \alpha_{-2}^{h-1} \end{pmatrix} \alpha_{-1}^{h} \begin{pmatrix} * \\ e_{h+1} \\ \alpha_0^h \end{pmatrix} \alpha_1^h \begin{pmatrix} R \\ e_h \\ \alpha_2^h \\ \alpha_2^{h-1} \end{pmatrix} \alpha_3^{h-1}$$

States $\binom{*}{u}$ are existential for C' if and only if u is existential for C (universal otherwise). So, when α_0^h is an existential state, α^h will have only one child in $T(C,u)$, α^{h+1} which is obtained by updating α^h via some determination of f. In this case $\Phi(\alpha^h)$ will have $\Phi'(\alpha^h)$ as the only child and e_{h+1} gives the determination chosen by C. Then updating once more C' we obtain a configuration $\Phi''(\alpha^h)$, which is equal to $\Phi(\alpha^{h+1})$.

When α_0^h is an universal state, α^h will have as children all the configurations obtained from α^h by applying some determination of f. So, $\Phi(\alpha^h)$ will have as children all the configurations $\Phi_j(\alpha^h)$ given by

$$\Phi_j(\alpha^h) : \cdots \begin{pmatrix} \# \\ \alpha_{-3}^{h-1} \end{pmatrix} \begin{pmatrix} L \\ e_h \\ \alpha_{-2}^h \\ \alpha_{-2}^{h-1} \end{pmatrix} \begin{pmatrix} \# \\ \alpha_{-1}^h \end{pmatrix} \begin{pmatrix} * \\ j \\ \alpha_0^h \end{pmatrix} \begin{pmatrix} \# \\ \alpha_1^h \end{pmatrix} \begin{pmatrix} R \\ e_h \\ \alpha_2^h \\ \alpha_2^{h-1} \end{pmatrix} \begin{pmatrix} \# \\ \alpha_3^{h-1} \end{pmatrix}$$

where $j \in \{1,2,\ldots,K\}$ and $K = \prod_{i \in Q^3} |f(i)|$. □

Corollary. *lUACA = lWACA.*

Proof. Since $O(2n) = O(n)$ the proof comes directly from the last theorems. □

Theorem 5. *UACA*$(t(n)) \subseteq$ *ACA* $(t(n))$.

Proof. Let C be an UACA given by $(Q, f, U, Q_a, \#)$. Let $C' = (Q', g, U, Q'_a, \#)$ be the ACA simulating C. We take $Q' = \{0,1,\ldots,K\} \times Q \cup Q$ where K is the number of possible determinations for f. We define the maps $\varphi^i, i = 1,2$. φ^1 is defined from Q' into Q. It is the identity function over Q and is the second projection over the set $\{0,1,\ldots,K\} \times Q$. φ^2 is the first projection over the set $\{0,1,\ldots,K\} \times Q$ and is defined as 1 over Q. Function g is defined as follows:

$$g(u,v,w) = \begin{cases} \{\begin{pmatrix} j \\ c_j \end{pmatrix} : \; c_j = f^j(\phi^1(u), \phi^1(v), \phi^1(w)), j = 1,\ldots,K\} \\ \qquad\qquad \text{if } \phi^2(u) = \phi^2(v) = \phi^2(w) \neq 0, \\ \begin{pmatrix} 0 \\ \varphi^1(v) \end{pmatrix} \qquad\qquad \text{otherwise.} \end{cases}$$

So, at time $t \geqslant 1$ the nonquiescent part of a configuration for C' will be $\begin{pmatrix} e_j^t \\ C_j^t \end{pmatrix}, \ldots, \begin{pmatrix} e_i^t \\ C_i^t \end{pmatrix}$.

Let C^t be an accepting configuration for C on u at time t. Since acceptance is defined in the cell at origin, C' must be sure that its accepting configurations are only those obtained by uniform elections on cell that may modify the values of cell at origin at time t. At time $s \leqslant t$, these cells are those belonging to $\{-(t-s),\ldots,(t-s)\}$. It is easy to see, by induction on t that if e_i^t is the value of the first coordinate (for these cells being in a non-quiescent state) of the ith cell at time t then $e_i^t \neq 0$ if and only if

$\forall 1 \leqslant t' \leqslant t \ \forall j \in \{i-(t-t'), i+t-t'\} \ \ e_j^{t'} = e_i^{t'}$. So, accepting trees for C on u have associated at least one accepting tree for C' on u.

5. Conclusion

It was proved that alternating cellular automata and alternating Turing machine have the same power when there are no restrictions in the sources. This is true even when polynomial restriction is made on the time. This is no longer true when we considered real time since we can then separate the class defined by alternating cellular automata and alternating Turing machine.

We observe that all the results obtained here apply to the nondeterministic case when we consider only existential states.

From Theorems 1, 2 and 3 we have seen that general alternation in cellular automata is as powerful as weak alternation when we consider the polynomial time class. This could be interesting for simulation of alternation, in particular random simulation, because our result says that if simulation time is reasonably larger (polynomial), then one can make random choices only in the first position and still obtain the same language complexity.

References

[1] J.L. Balcázar, J. Díaz and J.Gabarró. *Structural Complexity II*, EATCS monograph Series, Vol. 22 (Springer, Berlin, 1990).

[2] W. Bucher and K. Culik II, On real time and linear time cellular automata, *RAIRO Inform. Theor.* **18** (4) (1984) 307–325.

[3] A.K. Chandra, D.C. Kozen and L.J. Stockmeyer, Alternation, *J. ACM* **28** (1981) 114–133.

[4] J.H. Changa, O.H. Ibarra and A. Vergis, On the power of one-way communication. *J. ACM* **35** (3) (1988) 697–726.

[5] S.N. Cole, Real-time computation by n-dimensional iterative arrays of finite-state machines, *IEEE Trans. comput.* **c18** (14) (1969) 349–365.

[6] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata theory, Languages and Computation* (Addison-Wesley, Reading, MA, 1979).

[7] O. Ibarra and T. Jiang, On one-way cellular arrays, *SIAM J. Comput.* **16** (6) (1987) 1135–1154.

[8] K. Krithivasan and M. Mahajan. Nondeterministic, probabilistic and alternating computations on cellular array models, preprint, Department of Computer Science and Engineering, Indian Institute of Technology, Madras 600 036, India.

[9] J. Mazoyer and N. Reimen, A linear speed-up theorem for cellular automata, *Theoret. Comput. Sci.* **101** (1992) 59–98.

[10] A.R. Smith III, Real-time language recognition by one-dimensional cellular automata, *J. Comput. System Sci.* **6** (1972) 223–253.