I F I G
RESEARCH
REPORT

INSTITUT FÜR INFORMATIK

ONE GUESS ONE-WAY CELLULAR ARRAYS

Thomas Buchholz   Andreas Klein
Martin Kutrib

IFIG RESEARCH REPORT 9801
FEBRUARY 1998

Institut für Informatik
JLU Gießen
Arndtstraße 2
D-35392 Giessen, Germany
Tel: +49-641-99-32141
Fax: +49-641-99-32149
mail@informatik.uni-giessen.de
www.informatik.uni-giessen.de

JUSTUS-LIEBIG-
UNIVERSITÄT
GIESSEN

IFIG Research Report

IFIG Research Report 9801, February 1998

One guess one-way cellular arrays

Thomas Buchholz[1] Andreas Klein

Martin Kutrib[2]

Institute of Informatics, University of Giessen

Arndtstr. 2, D-35392 Giessen, Germany

Abstract. One-way cellular automata with restricted nondeterminism are investigated. The number of allowed nondeterministic state transitions is limited to a constant. It is shown that a limit to exactly one step does not decrease the language accepting capabilities. We prove a speed-up result that allows any linear-time computation to be sped-up to real-time. Some relationships to deterministic arrays are considered. Finally we prove several closure properties of the real-time languages.

[1]E-mail: buchholz@informatik.uni-giessen.de

[2]E-mail: kutrib@informatik.uni-giessen.de

1 Introduction

Linear arrays of finite automata can be regarded as models for massively parallel computers. Mainly they differ in how the automata are interconnected and in how the input is supplied. Various types have been studied for a long time [1, 2, 3, 4, 5, 8, 11, 14, 15, 16, 17, 19, 21]. Here we are investigating arrays with a very simple interconnection pattern. Each node is connected to its right immediate neighbor only. They are usually called one-way cellular automata (OCA).

Although deterministic and nondeterministic finite automata have the same computing capability, nondeterminism can strengthen the power of OCAs under some time resource bounds.

Nondeterministic OCAs have been investigated e.g. in [8], where $\mathscr{L}(\text{NOCA}) = \mathscr{L}(\text{NCA})$ was proved, and in [12], where it was shown in terms of homogeneous trellis automata that $\mathscr{L}_{rt}(\text{NOCA})$ contains the ε-free context-free languages as well as a NP-complete language, and is an AFL closed under intersection.

Here we consider arrays with restricted nondeterminism. We limit the number of allowed nondeterministic transitions. Moreover, all nondeterministic transitions have to appear before the deterministic ones. The main object of the present paper is to investigate arrays that are limited to exactly one nondeterministic transition step.

The paper is organized as follows: In section 2 we define the basic notions in terms of formal language processing. Section 3 is devoted to speed-up results and the possibility to reduce the number of nondeterministic transitions. Especially, it is shown that for one guess OCAs real-time is as powerful as linear-time. In section 4 comparisons are made to deterministic one-way and two-way devices. In section 5 various closure properties of the real-time one-guess OCA languages are shown.

2 Basic notions

We denote the integers by $\mathbb{Z}$, the positive natural numbers $\{1, 2, \dots\}$ by $\mathbb{N}$, the set $\mathbb{N} \cup \{0\}$ by $\mathbb{N}_0$ and the powerset of a set S by $\wp(S)$.

A nondeterministic one-way cellular automaton is a linear array of nondeterministic finite automata, sometimes called cells, each of them is connected to its nearest neighbor to the right. For our convenience we identify the cells by natural numbers. The state transition depends on the actual state of each cell and the actual state of its neighbor. The transition function is applied to all cells synchronously at discrete time steps. More formally:

Definition 1 *A nondeterministic one-way cellular automaton (*NOCA*) is a system $(S, \delta, \#)$, where*

a) S *is the finite, nonempty set of* states,

b) $\# \in S$ *is the* boundary state,

c) $\delta : S^2 \to \wp(S)$ *is the* local transition function *satisfying* $\forall\, s_1, s_2 \in S : (\delta(s_1, s_2) \neq \emptyset)$ *and* $(\delta(s_1, s_2) = \{\#\} \iff s_1 = \#)$.

Let $\mathcal{M}$ be an NOCA with n cells. A configuration of $\mathcal{M}$ at some time $i \geq 0$ is a description of its global state, which is actually a mapping $c_i : [1, \ldots, n] \to S$. During its course of computation an NOCA steps nondeterministically through a sequence of configurations. The configuration c_0 at time 0 is defined by the initial sequence of states in an NOCA, while subsequent configurations are chosen according to the global transition Δ:

Let $n \in \mathbb{N}$ be an arbitrary natural number and c resp. c' be defined by $s_1, \ldots, s_n \in S$ resp. $s'_1, \ldots, s'_n \in S$.

$$c' \in \Delta(c) \iff s'_1 \in \delta(s_1, s_2), s'_2 \in \delta(s_2, s_3), \ldots, s'_n \in \delta(s_n, \#)$$

The i-fold composition of Δ is defined as follows:

$$\begin{aligned} \Delta^0(c) &:= c \\ \Delta^{i+1}(c) &:= \bigcup_{c' \in \Delta^i(c)} \Delta(c') \end{aligned}$$

$\pi_i(s_1 \cdots s_n) := s_i$ selects the ith component of $s_1 \cdots s_n$. If the state set is a Cartesian product of some smaller sets $S = S_0 \times S_1 \times \cdots \times S_r$, we will use the notion “register” for the single parts of a state. Accordingly we define $c_i^k(j) := \pi_k(\pi_j(c_i))$.

If the flow of information is extended to two-way, the resulting device is a *nondeterministic two-way cellular automaton* (NCA). I.e. the next state of each cell depends on the state of the cell itself and the states of its both immediate neighbors (to the left and to the right).

An NOCA (NCA) is deterministic if $\delta(s_1, s_2)$ $(\delta(s_1, s_2, s_3))$ is a singleton for all states $s_1, s_2, s_3 \in S$. Deterministic cellular arrays are denoted by OCA resp. CA.

Definition 2 *Let A be an Alphabet and $\mathcal{M} = (S, \delta, \#)$ be an* NOCA *with $A \subseteq S$.*

a) A word $w \in A^+$ is accepted by $\mathcal{M}$ with final states $F \subseteq S$ in t time steps if there exists a $t_0 \leq t$ such that there exists a configuration $c_{t_0} \in \Delta^{t_0}(c_0)$ where $\pi_1(c_{t_0}) \in F$.

b) $L(\mathcal{M}) = \{w \in A^+ \mid w$ is accepted by $\mathcal{M}\}$ is the formal language accepted by $\mathcal{M}$.

c) Let $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$, be a mapping. If all $w \in L(\mathcal{M})$ are accepted within $t(|w|)$ time steps, then L is said to be of time complexity t.

The family of all languages which can be accepted by an NOCA with time complexity t is denoted by $\mathscr{L}_{t(n)}(\text{NOCA})$. If t equals the *identity function* $id(n) := n$ acceptance is said to be in real-time and we write $\mathscr{L}_{rt}(\text{NOCA})$.

There is a natural way to restrict the nondeterminism of the arrays. One can limit the number of allowed nondeterministic state transitions of the cells.

For the following let us suppose the local transition consists of a deterministic and nondeterministic part δ_d and δ_{nd}, where $\delta_d(s_1, s_2) \subseteq \delta_{nd}(s_1, s_2)$. At a whole $\delta = \delta_{nd}$ remains nondeterministic, but with this distinction the restriction is easily defined. Δ_{nd} denotes the global transition based on δ_{nd} and Δ_d the deterministic one based on δ_d.

Let $g : \mathbb{N} \to \mathbb{N}$ be a mapping for which $g(n) \leq t(n)$ holds. g gives the number of allowed nondeterministic transitions. An NOCA of length n for which the i-fold global transition Δ^i is defined as

$$\Delta^i := \begin{cases} \Delta^i_{nd} & \text{if } i \leq g(n) \\ \Delta_d^{i-g(n)}\left(\Delta_{nd}^{g(n)}\right) & \text{otherwise} \end{cases}$$

is denoted by gG-OCA (g guess OCA). Observe that all nondeterministic transitions have to be applied before the deterministic ones. In the sequel we are mainly interested in NOCAs allowed to guess constant times (i.e. $g(n) = k$, $k \in \mathbb{N}_0$).

3 Speed-up and guess reduction

It is known [4] that deterministic OCAs can be sped-up by a constant amount of time as long as the remaining time complexity does not fall below real-time. For constructions it is sometimes convenient to have a corresponding result for 1G-OCAs: For example, after the first nondeterministic step a deterministic $t(n)$-time OCA can be simulated and subsequently the resulting $(t(n) + 1)$-time 1G-OCA can be sped-up to a $t(n)$-time 1G-OCA again. Observe that in case of real-time it is not possible to speed-up the deterministic OCA by 1 time step before its simulation.

Lemma 3 *Let $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$, be a mapping and $k \in \mathbb{N}_0$ be a constant number. Then $\mathscr{L}_{t(n)+k}(\text{1G-OCA}) = \mathscr{L}_{t(n)}(\text{1G-OCA})$ holds.*

Proof. Let $\mathcal{M}$ be an 1G-OCA with time complexity $t(n) + k$ which passes through the configurations c_0 to $c_{t(n)+k}$. An 1G-OCA $\mathcal{M}'$ which simulates $\mathcal{M}$ in time $t(n)$ works as follows. Each cell consists of $k + 1$ registers each may contain a state from S, thus $S' := S^{k+1}$. k of the registers are initially empty.

The rightmost cell "knows" its input in advance (i.e. the border state). Therefore it can compute the states $c_1(n), \dots, c_{k+1}(n)$ in the first transition and store them in its registers. Subsequently it simulates one step of cell n of $\mathcal{M}$ in every time step. At the second time step cell $n-1$ observes the filled registers of its neighbor and can compute the states $c_2(n-1), \dots, c_{k+2}(n-1)$ in one time step. Again, subsequently it simulates one transition per time step. The behavior of cells $n-1$ to 1 is identical. Thus at time n the first cell computes the states $c_n(1), \dots, c_{n+k}(1)$, and at time $t(n)$ the state $c_{t(n)+k}(1)$. □

Deterministic OCAs can be sped-up from $(n + t(n))$-time to $(n + \frac{t(n)}{k})$-time [1, 13]. Thus linear-time (i.e. k times real-time, $k \geq 1$) is close by real-time. By the way, it is not the same, since the inclusion $\mathscr{L}_{rt}(\text{OCA}) \subset \mathscr{L}_{(1+\varepsilon)\cdot id}(\text{OCA}) = \mathscr{L}_{rt}(\text{CA})$ is a proper one [20, 4]. For 1G-OCAs we have the following stronger result, from which follows that real-time is as powerful as linear-time.

Theorem 4 *Let $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$, be a mapping and $k \in \mathbb{N}$ be a constant number. Then $\mathscr{L}_{k\cdot t(n)}(\text{1G-OCA}) = \mathscr{L}_{t(n)}(\text{1G-OCA})$ holds.*

Proof. From the definition we obtain the inclusion $\mathscr{L}_{t(n)}(\text{1G-OCA}) \subseteq \mathscr{L}_{k\cdot t(n)}(\text{1G-OCA})$.
It remains to show that $\mathscr{L}_{k\cdot t(n)}(\text{1G-OCA}) \subseteq \mathscr{L}_{t(n)}(\text{1G-OCA})$.

Let L be a language belonging to $\mathscr{L}_{k\cdot t(n)}(\text{1G-OCA})$ and let $\mathcal{M}$ be an 1G-OCA that accepts L with time complexity $k \cdot t(n)$. We construct an 1G-OCA $\mathcal{M}'$ that simulates $\mathcal{M}$ in time $t(n)$. The underlying technique will later be referenced as *pack-and-check technique.*

The idea is as follows: on an input of length n each cell i with $1 \leq i \leq n/k$, of $\mathcal{M}'$, guesses the initial states of the cells $k(i-1)+1, k(i-1)+2, \dots, ki$ and additionally what each cell of $\mathcal{M}$ might have guessed with respect to these initial states. (For simplicity here we assume that n is a multiple of k. The other cases are omitted since their handling is only a technical challenge.) Based on this compressed representation $\mathcal{M}'$ can simulate k time steps of $\mathcal{M}$ per time step which yields the required speed-up.

In parallel $\mathcal{M}'$ has to check whether the guesses of the initial states were correct. Therefore each cell $(n/k)j+i$ with $1 \leq i \leq n/k$ and $0 \leq j \leq k-1$ guesses the initial states of the cells $(i-1)n/k+1, (i-1)n/k+2, \dots in/k$, too. So we are concerned with an interim configuration of the form $x_1 x_2 \dots x_{n/k}$ where $|x_i| = k$ and each x_i might contain the compressed initial input. Now $\mathcal{M}'$ subsequently verifies – the first checking task – that the initial states guessed in the cells corresponding to x_j and x_{j+1}, are the same, $1 \leq j < n/k$. Additionally it checks – the second one – whether the guessed initial states $x_{n/k}$ are really the packed initial states of all cells. So in total it can be ensured that the simulation of $\mathcal{M}$ is based on the correct data.

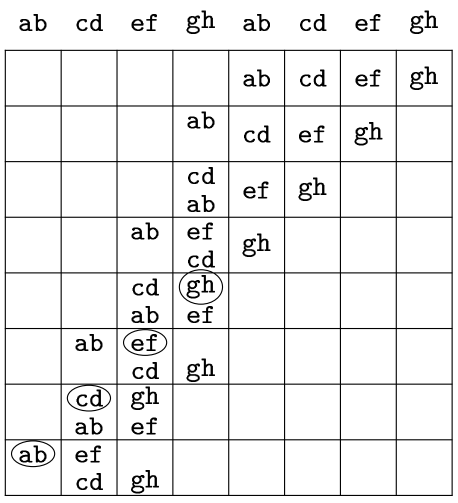

Figure 1: Example to the proof of theorem 4 with $k = 2$

To complete the proof we have to show how the two checking tasks can be realized. For the first task w.l.o.g. we may assume that $k = 2$. Further it is convenient to assume that the first $n/2$ cells and the last $n/2$ cells are distinguishable which can be provided by allowing each cell to guess whether it belongs to the first or last part and subsequently checking in n time steps that no cell which guessed 'last' occurs left from a cell which guessed 'first' and vice versa. Further it can easily be checked that the numbers of cells in the left and right part are identical.

The first checking task is then performed as follows. The last $n/2$ cells shift there guessed initial states in some register with maximum speed to the left. Each of the first $n/2$ cells are equipped with some 2 registers that are initially empty and work as a queue in a first-in first-out manner through which the arriving symbol stream is successively piped (cf. figure 1). Additionally in the rightmost cell a signal is generated in the first time step which moves leftward with maximum speed. If it enters one of the first $n/2$ cells it checks whether the cell's guessed initial states which were stored in some register are equal to the initial states that are currently in the position to leave the queue next. If they are not equal the signal prohibits the cell and the cells left from this cell to become final.

To perform the second checking task each cell is equipped with a counter modulo k which is initialized to $k - 1$ and decremented by one at each time step. Further at each time step a cell takes over the guessed packed input symbols of its right neighbour if its counter differs from $k - 1$ such that they are shifted through the cells. Otherwise a cell holds the packed input symbols which it actually contains (cf. figure 2). Again in the rightmost cell a signal is generated in the first time step which moves leftward with maximum speed. If it enters a cell it checks whether the symbol in the packed representation at position $r + 1$ equals to the

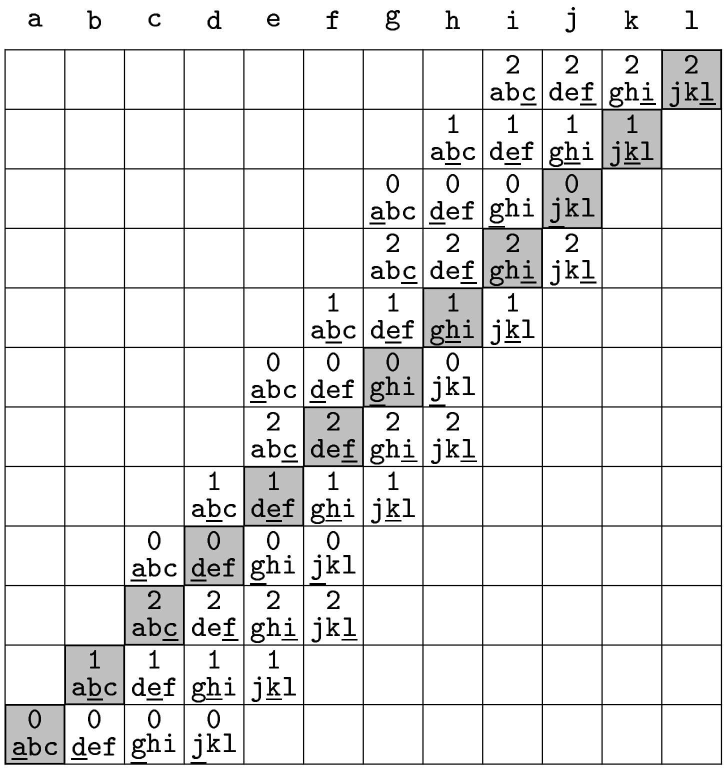

Figure 2: Example to the proof of theorem 4 with $k = 3$

(real) initial state of that cell where r denotes the value of its counter. Similarly if there exists some cell where the equality check fails this signal prohibits this cell and the cells left from this cell to become final. $\square$

The next result shows that $k + 1$ guesses per cell are not better than k guesses.

Theorem 5 *Let $g : \mathbb{N} \to \mathbb{N}$, $g(n) \le t(n)$, be a mapping and $k \in \mathbb{N}_0$ be a constant number. Then $\mathscr{L}_{t(n)}((g + k)\text{G-OCA}) = \mathscr{L}_{t(n)}(g\text{G-OCA})$ holds for all mappings $t : \mathbb{N} \to \mathbb{N}$, $t(n) \ge n$.*

Proof. It suffices to show the theorem for $k = 1$. Since an application to $g_k = g + k$ yields $g_{k-1} = g + k - 1$, a subsequent application to g_{k-1} yields $g_{k-2} = g + k - 2$ and so on to $g_{k-k} = g$.

Let $\mathcal{M}$ be a $(g+1)$G-OCA. We construct a gG-OCA $\mathcal{M}'$ which simulates $\mathcal{M}$ without any loss of time.

In its first (nondeterministic) step $\mathcal{M}'$ simulates the first step of $\mathcal{M}$ and, additionally, another nondeterministic step of $\mathcal{M}$ for all possible pairs of states of $\mathcal{M}$. The second result is stored in an additional register. It is a table $S^2 \times S$, which contains one row for every $(s_1, s_2) \in S^2$. After $g(n)$ time steps the first deterministic step of $\mathcal{M}'$ is as follows.

Every cell takes the actual state of itself and its neighbor and selects the corresponding row in the table. The next state is the third component of that row. Since the third components were nondeterministically chosen a nondeterministic transition is simulated deterministically. From time $g(n) + 2$ to $t(n)$ $\mathcal{M}'$ simulates $\mathcal{M}$ directly. □

4 Comparisons with deterministic cellular arrays

In order to compare the real-time computing power of 1G-OCAs to the well-investigated deterministic devices we prove the following lemma.

Lemma 6 *Let A be an arbitrary alphabet.*
$$L = \{w^{|w|} \mid w \in A^+\} \in \mathscr{L}_{rt}(\text{1G-OCA}).$$

Proof. For each symbol a in A we introduce a new symbol $\bar{\mathsf{a}}$ and denote the resulting alphabet by $\bar{A}$, i.e. $\bar{A} = \{\bar{\mathsf{a}} \mid \mathsf{a} \in A\}$, by h we denote the homomorphism that maps $\mathsf{a} \mapsto \bar{\mathsf{a}}$ for all $\mathsf{a} \in A$ and $\bar{\mathsf{a}} \mapsto \mathsf{a}$ for all $\bar{\mathsf{a}} \in \bar{A}$.

On input w an 1G-OCA $\mathcal{M}$ might work as follows to accept L in time $n+1$. Due to lemma 3 we can speed-up $\mathcal{M}$ to real-time subsequently. In the first time step each cell is allowed to nondeterministically stay in its initial state or to switch into its barred analogue. At time step 2 there might be a (guessed) configuration $x = x_1x_2 \ldots x_k$ where the odd indexed words belong to A^+ and the even indexed ones to $\bar{A}^+$ or vice versa. Now obviously $w \in L$ if and only if $h(x_i) = x_{i+1}$ for all $i = 1, \ldots, k-1$ and $|x_1| = k$. $\mathcal{M}$ is constructed to accept if and only if x is of that form.

Therefore $\mathcal{M}$ performs two tasks in parallel. The first is to check whether $h(x_i) = x_{i+1}$ for all $i = 1, \ldots, k-1$. Therefore it uses the queue technique which was described as first task in the proof of theorem 4 (see also figure 1). The second task is to verify that $|x_1| = k$. This can be realized as follows (cf. figure 3): in the second time step the rightmost cell and each cell in a state from A with a neighbor in a state from $\bar{A}$ or vice versa generates signals that moves leftward with maximal speed. They can be active or passive. If such an active signal enters an (unmarked) cell it marks that cell as being visited exactly once and becomes passive remembering whether or not the state of the cell was from A. As long as it subsequently passes through cells with states belonging to the same alphabet it remains passive. Otherwise it becomes active again and remains active as long as it enters marked cells which are additionally marked as being visited more than once. One can easily verify that the leftmost cell is marked as being visited once exactly if the number of generated signals is equal to the number of symbols in x_1, i.e. if and only if $|x_1| = k$. □

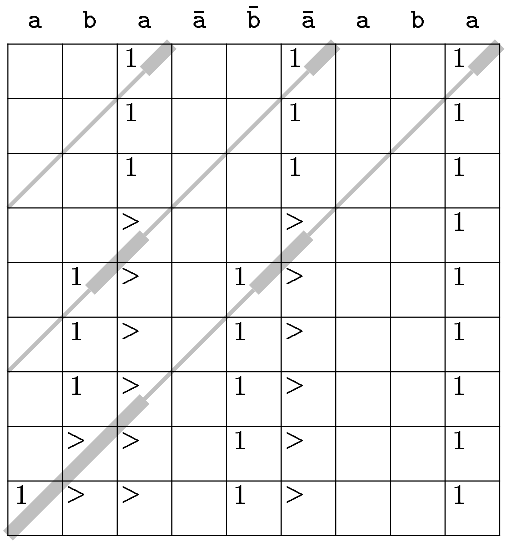

Figure 3: Checking whether $|x_1| = k$.

The following theorem states that the computing power of real-time OCAs is strictly increased by adding one nondeterministic step to that device.

Theorem 7 $\mathscr{L}_{rt}(\text{OCA}) \subset \mathscr{L}_{rt}(\text{1G-OCA})$

Proof. Obviously, we have an inclusion between the families since the nondeterministic part of the state transition can be designed to be deterministic. The language from lemma 6 belongs to $\mathscr{L}_{rt}(\text{1G-OCA})$. In [18] it was shown that it does not belong to $\mathscr{L}_{rt}(\text{OCA})$: L intersected with the regular language $\{\mathtt{a}\}^+$ is the unary language $\{\mathtt{a}^{n^2} \mid n \in \mathbb{N}\}$ which is not regular and thus not a real-time OCA language. Thus the inclusion is a proper one. □

It is known that $\mathscr{L}_{rt}(\text{OCA})$ is closed under inverse homomorphism [16], injective length multiplying homomorphism [6, 7] and inverse deterministic gsm mappings [12] but is not closed under ε-free homomorphism [16]. There is another relation between $\mathscr{L}_{rt}(\text{OCA})$ and $\mathscr{L}_{rt}(\text{1G-OCA})$. If we build the closure under ε-free homomorphisms of $\mathscr{L}_{rt}(\text{OCA})$ we obtain exactly the family $\mathscr{L}_{rt}(\text{1G-OCA})$. To prove the assertion we need the result that $\mathscr{L}_{rt}(\text{OCA})$ is closed under another weak kind of homomorphism.

Definition 8 *Let $h : A^* \to B^*$ be an ε-free homomorphism. h is structure preserving if for every two $\mathtt{a}, \mathtt{a}' \in A$ with $h(\mathtt{a}) = \mathtt{b}_1 \cdots \mathtt{b}_m$ and $h(\mathtt{a}') = \mathtt{b}'_1 \cdots \mathtt{b}'_n$ the sets $\{\mathtt{b}_1, \ldots, \mathtt{b}_m\}$ and $\{\mathtt{b}'_1, \ldots, \mathtt{b}'_n\}$ are disjoint if $\mathtt{a} \neq \mathtt{a}'$.*

Lemma 9 $\mathscr{L}_{rt}(\text{OCA})$ *is closed under structure preserving homomorphism.*

Proof. Let $A = \{\mathsf{a}_1, \dots, \mathsf{a}_m\}$ be an alphabet, $L \subseteq A^*$ be a language belonging to $\mathscr{L}_{rt}(\text{OCA})$ and $h : A^* \to B^*$ be a structure preserving homomorphism:

$$h(\mathsf{a}_1) = \mathsf{b}_{1,1} \cdots \mathsf{b}_{1,n_1}, \dots, h(\mathsf{a}_m) = \mathsf{b}_{m,1} \cdots \mathsf{b}_{m,n_m},$$

where $\mathsf{b}_{i,j} \in B$.

A deterministic generalized sequential machine (gsm) [10] is defined as follows: The input alphabet is B, the output alphabet is A and the set of states is $S = \{s_{i,j} \mid 1 \leq i \leq m \text{ and } 2 \leq j \leq n_i\} \cup \{s_0, s_e\}$ where s_0 is starting and final state. The gsm does its computation according to the local transformation $\delta : S \times B \to S \times A^*$.
For all $1 \leq i \leq m$ and $s_{k,l} \in S \setminus \{s_0, s_e\}$:

$$\begin{aligned}
\delta(s_0, b_{i,j}) &= \begin{cases} (s_{i,2}, \varepsilon) & \text{if } j = 1 \text{ and } n_i > 1 \\ (s_0, \mathsf{a}_i) & \text{if } j = 1 \text{ and } n_i = 1 \\ (s_e, \varepsilon) & \text{otherwise} \end{cases} \\
\delta(s_{k,l}, b_{i,j}) &= \begin{cases} (s_{k,l+1}, \varepsilon) & \text{if } k = i \text{ and } l = j \text{ and } n_i > l \\ (s_0, \mathsf{a}_i) & \text{if } k = i \text{ and } l = j \text{ and } n_i = l \\ (s_e, \varepsilon) & \text{otherwise} \end{cases} \\
\delta(s_e, b_{i,j}) &= (s_e, \varepsilon)
\end{aligned}$$

The gsm reads an input word w' from B^* and emits an output word w from A^*. If additionally gsm stops in a final state we write formally $gsm(w') = w$. For a given language $L_{in} \subseteq B^*$ it defines the language $gsm(L_{in}) = \{w \in A^* \mid \exists\, w' \in L_{in} : gsm(w') = w\}$.

Since $\mathscr{L}_{rt}(\text{OCA})$ is closed under inverse deterministic gsm mappings with final states the language $gsm^{-1}(L) = \{w' \in B^* \mid \exists\, w \in L : gsm(w') = w\}$ belongs to $\mathscr{L}_{rt}(\text{OCA})$, too. Now it suffices to show $h(L) = gsm^{-1}(L)$.

For an arbitrary $w = \mathsf{a}_1 \cdots \mathsf{a}_p \in L$ we have $h(w) = \mathsf{b}_{1,1} \cdots \mathsf{b}_{1,n_1} \cdots \mathsf{b}_{p,1} \cdots \mathsf{b}_{p,n_p}$. Since h is structure preserving the $\mathsf{b}_{i,1}$ are all different, thus, if started in state s_0 the computation path of gsm under input $h(\mathsf{a}_i)$, $1 \leq i \leq m$, is $(s_0, \mathsf{b}_{i,1}) \xrightarrow{\mathsf{a}_i} s_0$ if $n_i = 1$. The output is written on top of the arrow. If $n_i > 1$ we obtain $(s_0, \mathsf{b}_{i,1}) \xrightarrow{\varepsilon} (s_{i,2}, \mathsf{b}_{i,2}) \xrightarrow{\varepsilon} (s_{i,3}, \mathsf{b}_{i,3}) \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} (s_{i,n_i}, \mathsf{b}_{i,n_i}) \xrightarrow{\mathsf{a}_i} s_0$. In both cases gsm maps $h(\mathsf{a}_i) \mapsto \mathsf{a}_i$. Since starting and final states are identical we have $gsm(h(w)) = w$ and therefore $h(w) \in gsm^{-1}(L)$.

Now let w' be a word in $gsm^{-1}(L)$. It must exist a word $w = \mathsf{a}_1 \cdots \mathsf{a}_p \in L$ such that $gsm(w') = w$. We consider an arbitrary symbol in w, say a_i. The possible transition steps of gsm that emit a_i are $(s_{i,n_i}, \mathsf{b}_{i,n_i}) \xrightarrow{\mathsf{a}_i} s_0$ and $(s_0, \mathsf{b}_{i,1}) \xrightarrow{\mathsf{a}_i} s_0$. Note that these steps result in the state s_0 respectively. Since n_i is uniquely defined by h we have for $n_i = 1$ the transition

$(s_0, \mathtt{b}_{i,1}) \overset{\mathtt{a}_i}{\to} s_0$ and since h is structure preserving (i.e. $\mathtt{b}_{i,1}$ is uniquely determined) $gsm^{-1}(\mathtt{a}_i) = \mathtt{b}_{i,1}$. For $n_i > 1$ the transition $(s_{i,n_i}, \mathtt{b}_{i,n_i}) \overset{\mathtt{a}_i}{\to} s_0$ must take place to emit the symbol $\mathtt{a}_i$. The only way to enter state s_{i,n_i} is from state s_{i,n_i-1} with input $\mathtt{b}_{i,n_i-1}$ whereby the empty word is emitted. We can trace back the computation until we reach state s_0 again. The only way to enter s_0 is by transition steps that emit nonempty symbols. Therefore $gsm^{-1}(\mathtt{a}_i) = \mathtt{b}_{i,1} \cdots \mathtt{b}_{i,n_i}$. Since starting and final states are identical we obtain the unique word $gsm^{-1}(w)$ which must be w': $w' = gsm^{-1}(w) = \mathtt{b}_{1,1} \cdots \mathtt{b}_{1,n_1} \cdots \mathtt{b}_{p,1} \cdots \mathtt{b}_{p,n_p}$. It follows $w' = \mathtt{b}_{1,1} \cdots \mathtt{b}_{1,n_1} \cdots \mathtt{b}_{p,1} \cdots \mathtt{b}_{p,n_p} = h(\mathtt{a}_1) \cdots h(\mathtt{a}_p) = h(w) \in h(L)$. □

Theorem 10

a) *Let L be a language belonging to $\mathscr{L}_{rt}(\mathrm{OCA})$ and h be an ε-free homomorphism. Then $h(L)$ belongs to $\mathscr{L}_{rt}(\mathrm{1G\text{-}OCA})$.*

b) *Let L be a language belonging to $\mathscr{L}_{rt}(\mathrm{1G\text{-}OCA})$. Then there exist an ε-free homomorphism and a language $L' \in \mathscr{L}_{rt}(\mathrm{OCA})$ such that $h(L') = L$ holds.*

Proof.

a) Let L be a language over the alphabet $A = \{\mathtt{a}_1, \ldots, \mathtt{a}_m\}$ and a homomorphism $h : A^* \to B^*$ be defined according to

$$h(\mathtt{a}_1) = \mathtt{b}_{1,1} \cdots \mathtt{b}_{1,n_1}, \ldots, h(\mathtt{a}_m) = \mathtt{b}_{m,1} \cdots \mathtt{b}_{m,n_m},$$

where $\mathtt{b}_{i_j} \in B$.

We introduce an alphabet $\bar{B} := \{\bar{\mathtt{b}}_{1,1}, \ldots, \bar{\mathtt{b}}_{1,m_1}, \bar{\mathtt{b}}_{2,1}, \ldots, \bar{\mathtt{b}}_{m,n_m}\}$ of different symbols and a structure preserving homomorphism $h' : A^* \to \bar{B}^*$:

$$h'(\mathtt{a}_1) = \bar{\mathtt{b}}_{1,1} \cdots \bar{\mathtt{b}}_{1,n_1}, \ldots, h'(\mathtt{a}_m) = \bar{\mathtt{b}}_{m,1} \cdots \bar{\mathtt{b}}_{m,n_m}.$$

Since $\mathscr{L}_{rt}(\mathrm{OCA})$ is closed under structure preserving homomorphism $h'(L)$ is a real-time OCA language. Define an ε-free length preserving homomorphism $h'' : \bar{B}^* \to B^*$: $h''(\bar{\mathtt{b}}_{1,1}) = \mathtt{b}_{1,1}, \ldots, h''(\bar{\mathtt{b}}_{m,n_m}) = \mathtt{b}_{m,n_m}$. Obviously, we have $h(L) = h''(h'(L))$.

An 1G-OCA $\mathcal{M}'$ that accepts $h(L)$ in $n+1$ time steps works as follows. Since h'' is length preserving, in the first time step every cell can guess the inverse image of its initial state under h''. During the next n time steps $\mathcal{M}'$ simulates a real-time OCA $\mathcal{M}$ that accepts $h'(L)$. As shown in lemma 3 we can speed-up $\mathcal{M}$ by one time-step.

b) Let $\mathcal{M}$ be an 1G-OCA accepting L in real-time. $\mathcal{M}$ can be simulated by another 1G-OCA $\mathcal{M}'$ which works in $(n+1)$-time as follows. In the first step $\mathcal{M}'$ guesses the state of each cell of $\mathcal{M}$ at time 2. In the second step $\mathcal{M}'$ verifies its guess. During the steps 3 to $n+1$ $\mathcal{M}'$ works exactly as $\mathcal{M}$ during the steps 2 to n. Now let $\mathcal{M}''$ be an OCA which simulates the computation of $\mathcal{M}'$ during the time steps 2 to $n+1$ and

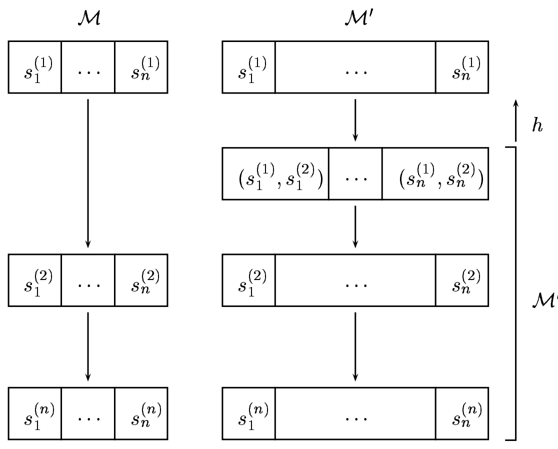
$\mathcal{M}$
$\mathcal{M}'$
$s_1^{(1)}$ … $s_n^{(1)}$
$(s_1^{(1)}, s_1^{(2)})$ … $(s_n^{(1)}, s_n^{(2)})$
$s_1^{(2)}$ … $s_n^{(2)}$
$s_1^{(n)}$ … $s_n^{(n)}$
h
$\mathcal{M}''$

Figure 4: An 1G-OCA can be simulated by an OCA and a homomorphism.

h be the ε-free homomorphism that maps a pair of states of $\mathcal{M}$ to the first component $h(s_1, s_2) = s_1$. Thus $h(L(\mathcal{M}'')) = L$. □

Adding two-way communication to deterministic cellular arrays yields more powerful real-time devices. It is well known that $\mathscr{L}_{rt}(\text{CA})$ is a proper superset of $\mathscr{L}_{rt}(\text{OCA})$ [16]. The following two theorems relate both augmentations.

Theorem 11 $\mathscr{L}_{rt}(\text{CA}) \subseteq \mathscr{L}_{rt}(\text{1G-OCA})$

Proof. In [4, 20] it has been shown that $L \in \mathscr{L}_{rt}(\text{CA})$ if and only if $L^R \in \mathscr{L}_{2id}(\text{OCA})$, where L^R denotes the reversal of L. From theorem 4 we know $\mathscr{L}_{2id}(\text{1G-OCA}) = \mathscr{L}_{rt}(\text{1G-OCA})$. For structural reasons $\mathscr{L}_{2id}(\text{OCA}) \subseteq \mathscr{L}_{2id}(\text{1G-OCA})$ must hold. In theorem 16 the closure of $\mathscr{L}_{rt}(\text{1G-OCA})$ under reversal is shown what proves the assertion. □

Theorem 12 $\mathscr{L}_{rt}(\text{1G-OCA}) \subseteq \mathscr{L}(\text{CA})$

Proof. The idea is to construct a brute-force CA which tries all possible choices of the $\mathscr{L}_{rt}(\text{1G-OCA})$. In order to realize such a behavior we need two mechanisms. One has to select successively all possible choices. The other mechanism is the simulation of the $\mathscr{L}_{rt}(\text{1G-OCA})$ on the actual choice.

To control the mechanisms we use synchronization by a modified fssp. If the synchronization is started at both border cells simultaneously it can be done in exactly n time steps, where n is the length of the array. This process can be repeated such that the array fires every n time steps.

In the $\mathscr{L}_{rt}(\text{1G-OCA})$ computation every cell can nondeterministically choose a state in the first time step. Let S denote the state set. To generate all possible choices of the whole array it suffices to set up a $|S|$-ary counter. At most every possible number on the counter corresponds to one choice. Actually, it may happen that some of the numbers are invalid choices since the nondeterministically step of a cell may depend on its input symbol. But such cases are detectable by the cells themself which can set up an error flag that prevents the array to accept. To increment the counter a signal is send from the border cell containing the least significant digit to the opposite. It needs n time steps.

Subsequently n time steps of the $\mathscr{L}_{rt}$(1G-OCA) are simulated in a straightforward manner. The input is accepted if one of the choices leads to an accepting simulation. Otherwise it is rejected when the border cell containing the most significant digit generates a carry-over.

□

5 Closure properties

The family $\mathscr{L}_{rt}$(1G-OCA) has strong closure properties.

Lemma 13 $\mathscr{L}_{rt}$(1G-OCA) *is closed under union, intersection and set difference.*

Proof. Using the same two channel technique of [17] and [8] the assertion is easily seen. Each cell consists of two registers in which acceptors for both languages are simulated in parallel. □

Theorem 14 $\mathscr{L}_{rt}$(1G-OCA) *is an AFL (i.e. is closed under intersection with regular sets, inverse homomorphism, ε-free homomorphism, union, concatenation and positive closure).*

Proof. Closure under intersection with regular sets and union have been shown in lemma 13.

Assume there is a language $L' \in \mathscr{L}_{rt}$(1G-OCA) and an ε-free homomorphism h' such that $L'' := h'(L') \notin \mathscr{L}_{rt}$(1G-OCA). From theorem 10 follows that there exists a language $L \in \mathscr{L}_{rt}$(OCA) and an ε-free homomorphism h such that $h(L) = L'$. Therefore we have $L'' = h'(h(L))$. Since $h' \circ h$ is an ε-free homomorphism too, theorem 10 is contradicted. The closure under ε-free homomorphism follows.

Let $L \in \mathscr{L}_{rt}$(1G-OCA) be a language over A and $h : B^* \to A^*$ be a homomorphism. From theorem 10 we obtain a real-time OCA language L' over A' and a length preserving homomorphism $h' : A'^* \to A$ with $h'(L') = L$. Let h_1 be the homomorphism with $h_1((x, x')) = x'$ for every $x \in B$, $x' \in A'$ with $h(x) = h'(x')$. Further let p_1 be an ε-free homomorphism with $p_1((x, x')) = x$ for $x \in B$, $x' \in A'$. Then $p_1(h_1^{-1}(L')) = h^{-1}(h'(L')) = h^{-1}(L)$. Since $\mathscr{L}_{rt}$(OCA) is closed under inverse homomorphism [16], $h_1^{-1}(L')$ belongs to $\mathscr{L}_{rt}$(OCA). Now theorem 10 implies $h^{-1}(L) \in \mathscr{L}_{rt}$(1G-OCA) which proves the closure under inverse homomorphism.

Now let $L_1, L_2 \in \mathscr{L}_{rt}(\text{1G-OCA})$ and $\mathcal{M}_1$, $\mathcal{M}_2$ be acceptors for L_1 and L_2. We construct a 2G-OCA $\mathcal{M}'$ that accepts the concatenation L_1L_2 in $n+1$ time steps. To accept an input w_1w_2, $w_1 \in L_1$, $w_2 \in L_2$, $\mathcal{M}'$ guesses in its first time step the cell in which the first symbol of w_2 occurs. In the remaining time steps 2 to $n+1$ $\mathcal{M}'$ simulates $\mathcal{M}_1$ in the left part on w_1 and $\mathcal{M}_2$ in the right part of the array on w_2. Due to theorem 5 we can construct an 1G-OCA that accepts L_1L_2 in time $n+1$ which according to lemma 3 can be sped-up to work in real-time.

The closure under positive closure follows analogously. □

Corollary 15 $\mathscr{L}_{rt}(\text{1G-OCA})$ *is closed under* ε*-free substitution.*

Proof. In [9] it has been shown that an AFL that is closed under intersection is also closed under ε-free substitution. Thus the assertion follows from lemma 13 and theorem 14. □

Theorem 16 $\mathscr{L}_{rt}(\text{1G-OCA})$ *is closed under reversal.*

Proof. Let A be an arbitrary alphabet. In [8] it has been shown that the language

$$L_R = \{w \in A^+ \mid w = w^R\}$$

belongs to $\mathscr{L}_{rt}(\text{OCA})$.

Let $\mathcal{M}$ be an $\mathscr{L}_{rt}(\text{1G-OCA})$ that accepts a language $L \subseteq A^*$ in real-time.

An $\mathscr{L}_{rt}(\text{1G-OCA})$ $\mathcal{M}'$ that accepts L^R in $n+1$ time steps works as follows. On input $w = w_1 \cdots w_n$ every cell $1 \leq i \leq n$ of $\mathcal{M}'$ guesses the symbol w_{n-i+1} and stores it in an additional register. If the guesses are correct then $\mathcal{M}'$ has the symbols $w_n w_{n-1} \cdots w_2 w_1 = w^R$ on its additional track. Furthermore the cell in the center of the array is nondeterministically marked (if n is even the two cells in the center). Altogether, after the first time step $\mathcal{M}'$ performs three tasks in parallel.

One is to simulate $\mathcal{M}$ on w^R because $w \in L^R$ iff $w^R \in L = L(\mathcal{M})$.

The second task is to verify that the cell(s) in the center is (are) marked. It is realized by a signal which moves with speed $\frac{1}{2}$ from the marked cell(s) to the left. Since accepting is in real-time it can only be done at the time step the signal arrives at the left border. In this case the center was marked.

The last task is to check that the guessed word w^R was correct. Note that every cell can remember its original input. Therefore, the input as well as its (guessed) reversal could be stored on different tracks at time step 1. Since the center is marked $\mathcal{M}'$ can simulate two real-time OCAs for the language L_R where the input is the left half of one track and the right half of the other track respectively. □

It has been shown that $\mathscr{L}_{rt}$(1G-OCA) is closed under reversal and that there is the inclusion $\mathscr{L}_{rt}$(1G-OCA) $\subseteq \mathscr{L}$(CA). Up to now it is not known whether the family is closed under complement. A negative answer would imply that there exists a CA language which is not a real-time CA language.

References

[1] Bucher, W. and Čulik II, K. *On real time and linear time cellular automata.* RAIRO Inform. Théor. 18 (1984), 307–325.

[2] Buchholz, Th. and Kutrib, M. *On time computability of functions in one-way cellular automata.* Acta Inf. (1998).

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

[4] Choffrut, C. and Čulik II, K. *On real-time cellular automata and trellis automata.* Acta Inf. 21 (1984), 393–407.

[5] Cole, S. N. *Real-time computation by n-dimensional iterative arrays of finite-state machines.* IEEE Trans. Comput. C-18 (1969), 349–365.

[6] Čulik II, K., Gruska, J., and Salomaa, A. *Systolic trellis automata I.* Internat. J. Comput. Math. 15 (1984), 195–212.

[7] Čulik II, K., Gruska, J., and Salomaa, A. *Systolic trellis automata II.* Internat. J. Comput. Math. 16 (1984), 3–22.

[8] Dyer, C. R. *One-way bounded cellular automata.* Inform. Control 44 (1980), 261–281.

[9] Ginsburg, S. and Hopcroft, J. E. *Two-way balloon automata and AFL.* J. Assoc. Comput. Mach. 17 (1970), 3–13.

[10] Hopcroft, J. E. and Ullman, J. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachusetts, 1979.

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

[12] Ibarra, O. H. and Kim, S. M. *Characterizations and computational complexity of systolic trellis automata.* Theoret. Comput. Sci. 29 (1984), 123–153.

[13] Ibarra, O. H. and Palis, M. A. *Some results concerning linear iterative (systolic) arrays.* J. Parallel and Distributed Comput. 2 (1985), 182–218.

[14] Kutrib, M. *Pushdown cellular automata.* Theoret. Comput. Sci. (1998).

[15] Kutrib, M. and Richstein, J. *Real-time one-way pushdown cellular automata languages.* Developments in Language Theory II. At the Crossroads of Mathematics, Computer Science and Biology, 1996, pp. 420–429.

[16] Seidel, S. R. *Language recognition and the synchronization of cellular automata.* Technical Report 79-02, Department of Computer Science, University of Iowa, 1979.

[17] Smith III, A. R. *Real-time language recognition by one-dimensional cellular automata.* J. Comput. System Sci. 6 (1972), 233–253.

[18] Terrier, V. *Language recognizable in real time by cellular automata.* Complex Systems 8 (1994), 325–336.

[19] Terrier, V. *On real time one-way cellular array.* Theoret. Comput. Sci. 141 (1995), 331–335.

[20] Umeo, H., Morita, K., and Sugata, K. *Deterministic one-way simulation of two-way real-time cellular automata and its related problems.* Inform. Process. Lett. 14 (1982), 158–161.

[21] Vollmar, R. *Algorithmen in Zellularautomaten.* Teubner, Stuttgart, 1979.