

One Guess One-Way Cellular Arrays

Thomas Buchholz, Andreas Klein, and Martin Kutrib

Institute of Informatics, University of Giessen
Arndtstr. 2, D-35392 Giessen, Germany
{buchholz,kutrib}@informatik.uni-giessen.de

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 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. Here we are investigating arrays with a parallel input mode and 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 [7] where $\mathcal{L}(\text{NOCA}) = \mathcal{L}(\text{NCA})$ was proved, and in [10] where it was shown in terms of homogeneous trellis automata that $\mathcal{L}_{rt}(\text{NOCA})$ 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.

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 2^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

1. S is the finite, nonempty set of states,
2. $\# \in S$ is the boundary state,
3. $\delta : S^2 \rightarrow 2^S$ is the local transition function satisfying

$$\forall s_1, s_2 \in S : (\delta(s_1, s_2) \neq \emptyset) \text{ and } (\delta(s_1, s_2) = \{\#\} \iff s_1 = \#).$$

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 $c_i : [1, \dots, n] \rightarrow S$. 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 $s_1, \dots, s_n \in S$ resp. $s'_1, \dots, s'_n \in S$.

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

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

$$\Delta^0(c) := c, \quad \Delta^{i+1}(c) := \bigcup_{c' \in \Delta^i(c)} \Delta(c')$$

$\pi_i(s_1 \dots s_n) := s_i$ selects the i th component of s_1, \dots, s_n . If the state set is a Cartesian product of some smaller sets $S = S_0 \times S_1 \times \dots \times S_r$, we will use the notion ‘register’ for the single parts of a state.

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 $A \subseteq S$.

1. 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 $\pi_1(c_{t_0}) \in F$.
2. $L(\mathcal{M}) = \{w \in A^+ \mid w \text{ is accepted by } \mathcal{M}\}$ is the language accepted by \mathcal{M} .
3. Let $t : \mathbb{N} \rightarrow \mathbb{N}$, $t(n) \geq n$, 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 $\mathcal{L}_{t(n)}(\text{NOCA})$. If t equals the *identity function* $id(n) := n$ acceptance is said to be in real-time and we write $\mathcal{L}_{rt}(\text{NOCA})$.

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 $\delta_d(s_1, s_2) \in \delta_{nd}(s_1, s_2)$. 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 $g : \mathbb{N} \rightarrow \mathbb{N}$ 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

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

is denoted by $g\text{G-OCA}$ (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 $t : \mathbb{N} \rightarrow \mathbb{N}$, $t(n) \geq n$, be a mapping and $k \in \mathbb{N}_0$ be a constant number. Then $\mathcal{L}_{t(n)+k}(1\text{G-OCA}) = \mathcal{L}_{t(n)}(1\text{G-OCA})$ 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 $c_{t(n)+k}(1)$. \square

Deterministic OCAs can be sped-up from $(n+t(n))$ -time to $(n+\frac{t(n)}{k})$ -time [1, 11]. 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 $\mathcal{L}_{rt}(\text{OCA}) \subset \mathcal{L}_{(1+\varepsilon)\cdot id}(\text{OCA}) = \mathcal{L}_{rt}(\text{CA})$ is a proper one [17, 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 $t : \mathbb{N} \rightarrow \mathbb{N}$, $t(n) \geq n$, be a mapping and $k \in \mathbb{N}$ be a constant number. Then $\mathcal{L}_{k\cdot t(n)}(1\text{G-OCA}) = \mathcal{L}_{t(n)}(1\text{G-OCA})$ holds.*

Proof. The inclusion $\mathcal{L}_{t(n)}(1\text{G-OCA}) \subseteq \mathcal{L}_{k\cdot t(n)}(1\text{G-OCA})$ follows from the definition. In order to prove $\mathcal{L}_{k\cdot t(n)}(1\text{G-OCA}) \subseteq \mathcal{L}_{t(n)}(1\text{G-OCA})$ let L be a language belonging to $\mathcal{L}_{k\cdot t(n)}(1\text{G-OCA})$ 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 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.

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 $x_1x_2 \dots x_{n/k}$ 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.

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 we may assume that the first $n/2$ cells and the last $n/2$ cells are distinguishable which can be provided by guessing and a simple verification task.

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. Two initially empty registers of each of the first $n/2$ cells work as a queue in a first-in first-out manner through which the arriving symbol stream is successively piped (cf. Fig. 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.

				ab	cd	ef	gh
			ab	cd	ef	gh	
			cd	ab	ef	gh	
		ab	ef	cd	gh		
		cd	ab	ef	gh		
	ab	ef	cd	gh			
	cd	ab	ef	gh			
ab	ef	cd	gh				

Fig. 1. Example to the proof of Theorem 4 with $k = 2$

												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2
												1	1	1	1
												0	0	0	0
												2	2	2	2

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. \square

4 Comparisons with Deterministic Cellular Arrays

In order to compare the real-time computing power of 1G-OCA's to the well-investigated deterministic devices we prove the following theorem which states that the computing power of real-time OCA's is strictly increased by adding one nondeterministic step to that device.

Theorem 6. $\mathcal{L}_{rt}(\text{OCA}) \subset \mathcal{L}_{rt}(\text{1G-OCA})$

Proof. Obviously, we have an inclusion between the families since the nondeterministic part of the state transition can be designed to be deterministic. In [3] $L = \{w^{|w|} \mid w \in A^+\} \in \mathcal{L}_{rt}(\text{1G-OCA})$ has been shown for an arbitrary alphabet A .

In [16] it was shown that it does not belong to $\mathcal{L}_{rt}(\text{OCA})$: L intersected with the regular language $\{\mathbf{a}\}^+$ is the unary language $\{\mathbf{a}^{n^2} \mid n \in \mathbb{N}\}$ which is not regular and thus not a real-time OCA language. Thus the inclusion is a proper one. \square

It is known that $\mathcal{L}_{rt}(\text{OCA})$ is closed under inverse homomorphism [14], injective length multiplying homomorphism [5, 6] and inverse deterministic gsm mappings [10] but is not closed under ε -free homomorphism [14]. There is another relation between $\mathcal{L}_{rt}(\text{OCA})$ and $\mathcal{L}_{rt}(\text{1G-OCA})$. If we build the closure under ε -free homomorphisms of $\mathcal{L}_{rt}(\text{OCA})$ we obtain exactly the family $\mathcal{L}_{rt}(\text{1G-OCA})$. To prove the assertion we need the result from [3] that $\mathcal{L}_{rt}(\text{OCA})$ is closed under another weak kind of homomorphism.

Definition 7. Let $h : A^* \rightarrow B^*$ be an ε -free homomorphism. h is structure preserving if for every two $\mathbf{a}, \mathbf{a}' \in A$ with $h(\mathbf{a}) = \mathbf{b}_1 \cdots \mathbf{b}_m$ and $h(\mathbf{a}') = \mathbf{b}'_1 \cdots \mathbf{b}'_n$ the sets $\{\mathbf{b}_1, \dots, \mathbf{b}_m\}$ and $\{\mathbf{b}'_1, \dots, \mathbf{b}'_n\}$ are disjoint if $\mathbf{a} \neq \mathbf{a}'$.

Lemma 8. $\mathcal{L}_{rt}(\text{OCA})$ is closed under structure preserving homomorphism.

Theorem 9. 1. Let L be a language belonging to $\mathcal{L}_{rt}(\text{OCA})$ and h be an ε -free homomorphism. Then $h(L)$ belongs to $\mathcal{L}_{rt}(\text{1G-OCA})$.

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

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

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

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

We introduce an alphabet $\bar{B} := \{\bar{\mathbf{b}}_{1,1}, \dots, \bar{\mathbf{b}}_{1,n_1}, \bar{\mathbf{b}}_{2,1}, \dots, \bar{\mathbf{b}}_{m,n_m}\}$ of different symbols and a structure preserving homomorphism $h' : A^* \rightarrow \bar{B}^*$:

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

Since $\mathcal{L}_{rt}(\text{OCA})$ is closed under structure preserving homomorphism $h'(L)$ is a real-time OCA language. Define an ε -free length preserving homomorphism $h'' : \bar{B}^* \rightarrow B^*$: $h''(\bar{\mathbf{b}}_{1,1}) = \mathbf{b}_{1,1}, \dots, h''(\bar{\mathbf{b}}_{m,n_m}) = \mathbf{b}_{m,n_m}$.

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 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 $h(L(\mathcal{M}'')) = L$. \square

Adding two-way communication to deterministic cellular arrays yields more powerful real-time devices. It is well known that $\mathcal{L}_{rt}(\text{CA})$ is a proper superset of $\mathcal{L}_{rt}(\text{OCA})$ [14]. The following two theorems relate both augmentations.

Theorem 10. $\mathcal{L}_{rt}(\text{CA}) \subseteq \mathcal{L}_{rt}(\text{1G-OCA})$

Proof. In [4, 17] it has been shown that $L \in \mathcal{L}_{rt}(\text{CA})$ if and only if $L^R \in \mathcal{L}_{2id}(\text{OCA})$, where L^R denotes the reversal of L . From Theorem 4 we know $\mathcal{L}_{2id}(\text{1G-OCA}) = \mathcal{L}_{rt}(\text{1G-OCA})$. The inclusion $\mathcal{L}_{2id}(\text{OCA}) \subseteq \mathcal{L}_{2id}(\text{1G-OCA})$ holds due to structural reasons. In [3] the closure of $\mathcal{L}_{rt}(\text{1G-OCA})$ under reversal was shown what proves the assertion. \square

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

Proof. The idea is to construct a brute-force CA which tries all possible choices of the $\mathcal{L}_{rt}(\text{1G-OCA})$. 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 $\mathcal{L}_{rt}(\text{1G-OCA})$ 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.

Let S denote the state set of a real-time 1G-OCA. To generate all of its possible time step 1 configurations it suffices to set up a $|S|$ -ary counter. At most every possible number on the counter corresponds to one configuration. 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 $\mathcal{L}_{rt}(\text{1G-OCA})$ 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. \square

5 Closure Properties

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

Lemma 12. $\mathcal{L}_{rt}(1G-OCA)$ is closed under union, intersection and set difference.

Proof. Using the same two channel technique of [15] and [7] the assertion is easily seen. Each cell consists of two registers in which acceptors for both languages are simulated in parallel. \square

Theorem 13. $\mathcal{L}_{rt}(1G-OCA)$ 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 12.

Assume there is a language $L' \in \mathcal{L}_{rt}(1G-OCA)$ and an ε -free homomorphism h' such that $L'' := h'(L') \notin \mathcal{L}_{rt}(1G-OCA)$. From Theorem 9 follows that there exists a language $L \in \mathcal{L}_{rt}(OCA)$ 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 9 is contradicted. The closure under ε -free homomorphism follows.

Let $L \in \mathcal{L}_{rt}(1G-OCA)$ be a language over A and $h : B^* \rightarrow A^*$ be a homomorphism. From Theorem 9 we obtain a real-time OCA language L' over A' and a length preserving homomorphism $h' : A'^* \rightarrow A$ 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 $\mathcal{L}_{rt}(OCA)$ is closed under inverse homomorphism [14], $h_1^{-1}(L')$ belongs to $\mathcal{L}_{rt}(OCA)$. Now Theorem 9 implies $h^{-1}(L) \in \mathcal{L}_{rt}(1G-OCA)$ which proves the closure under inverse homomorphism.

Now let $L_1, L_2 \in \mathcal{L}_{rt}(1G-OCA)$ 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. \square

Corollary 14. $\mathcal{L}_{rt}(1G-OCA)$ is closed under ε -free substitution.

Proof. In [8] 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 12 and Theorem 13. \square

In [3] it has been shown that $\mathcal{L}_{rt}(1G-OCA)$ is closed under reversal and that theorem 11 proves the inclusion $\mathcal{L}_{rt}(1G-OCA) \subseteq \mathcal{L}(CA)$. 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., Čulik II, K.: On real time and linear time cellular automata. *RAIRO Inform. Théor.* **18** (1984) 307–325
- [2] Buchholz, Th., Kutrib, M.: On time computability of functions in one-way cellular automata. *Acta Inf.* **35** (1998) 329–352
- [3] Buchholz, Th., Klein, A., Kutrib, M.: One guess one-way cellular arrays. Research Report 9801, Institute of Informatics, University of Giessen, Gießen, 1998.
- [4] Choffrut, C., Čulik II, K.: On real-time cellular automata and trellis automata. *Acta Inf.* **21** (1984) 393–407
- [5] Čulik II, K., Gruska, J., Salomaa, A.: Systolic trellis automata I. *Internat. J. Comput. Math.* **15** (1984) 195–212
- [6] Čulik II, K., Gruska, J., Salomaa, A.: Systolic trellis automata II. *Internat. J. Comput. Math.* **16** (1984) 3–22
- [7] Dyer, C. R.: One-way bounded cellular automata. *Inform. Control* **44** (1980) 261–281
- [8] Ginsburg, S., Hopcroft, J. E.: Two-way balloon automata and AFL. *J. Assoc. Comput. Mach.* **17** (1970) 3–13
- [9] Ibarra, O. H., Jiang, T.: On one-way cellular arrays. *SIAM J. Comput.* **16** (1987) 1135–1154
- [10] Ibarra, O. H., Kim, S. M.: Characterizations and computational complexity of systolic trellis automata. *Theoret. Comput. Sci.* **29** (1984) 123–153
- [11] Ibarra, O. H., Palis, M. A.: Some results concerning linear iterative (systolic) arrays. *J. Parallel and Distributed Comput.* **2** (1985) 182–218
- [12] Kutrib, M.: Pushdown cellular automata. *Theoret. Comput. Sci.* (1998)
- [13] Kutrib, M., Richstein, J.: Real-time one-way pushdown cellular automata languages. In: *Developments in Language Theory II. At the Crossroads of Mathematics, Computer Science and Biology*, World Scientific, Singapore, (1996) 420–429
- [14] Seidel, S. R.: Language recognition and the synchronization of cellular automata. Technical Report 79-02, Department of Computer Science, University of Iowa, 1979
- [15] Smith III, A. R.: Real-time language recognition by one-dimensional cellular automata. *J. Comput. System Sci.* **6** (1972) 233–253
- [16] Terrier, V.: Language recognizable in real time by cellular automata. *Complex Systems* **8** (1994) 325–336
- [17] Umeo, H., Morita, K., Sugata, K.: Deterministic one-way simulation of two-way real-time cellular automata and its related problems. *Inform. Process. Lett.* **14** (1982) 158–161
- [18] Vollmar, R.: *Algorithmen in Zellularautomaten*. Teubner, Stuttgart, 1979