I F I G
RESEARCH
REPORT

INSTITUT FÜR INFORMATIK

On Interacting Automata with Limited Nondeterminism

Thomas Buchholz   Andreas Klein
Martin Kutrib

IFIG Research Report 9806
May 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 9806, May 1998

On Interacting Automata with Limited Nondeterminism

Thomas Buchholz[1] Andreas Klein[2]
Martin Kutrib[3]

Institute of Informatics, University of Giessen
Arndtstr. 2, D-35392 Giessen, Germany

Abstract. One-way and two-way cellular language acceptors with restricted nondeterminism are investigated. The number of nondeterministic state transitions is regarded as limited resource which depends on the length of the input.
We center our attention to real-time, linear-time and unrestricted-time computations. A speed-up result that allows any linear-time computation to be sped-up to real-time is proved. The relationships to deterministic arrays are considered. For an important subclass a characterization in terms of deterministic language families and ϵ-free homomorphisms is given. Finally we prove strong closure properties of languages acceptable with a constant number of nondeterministic transitions.

CR Subject Classification (1998): F.1, F.4.3, B.6.1, E.4

[1]E-mail: buchholz@informatik.uni-giessen.de
[2]E-mail: andreas.klein@informatik.uni-giessen.de
[3]E-mail: kutrib@informatik.uni-giessen.de

1 Introduction

Systems of interconnected parallel acting automata have extensively been investigated from a language theoretic point of view.

The specification of such a system includes the type and specification of the single automata (these are in almost all cases finite or pushdown automata), the interconnection scheme (which sometimes implies a dimension of the system), a local and/or global transition function and the input and output modes. Various types have been studied for a long time (e.g., [1, 2, 4, 5, 7, 8, 14, 20, 21, 23, 24, 25, 27]).

One kind of system is of particular interest: the cellular automata. In this well-investigated model homogeneously connected deterministic or nondeterministic finite automata work synchronously at discrete time steps.

Here we are investigating linear arrays with very simple interconnection schemes. Each node is connected to its both immediate neighbors or to its right immediate neighbor only. Correspondingly they are called two-way or one-way cellular automata (CA or OCA) resp. nondeterministic two-way or nondeterministic one-way cellular automata (NCA or NOCA). Although deterministic and nondeterministic finite automata have the same computing capability, nondeterminism can strengthen the power of the deterministic parallel devices.

Nondeterministic arrays have been investigated e.g. in [24], where it was proved that NCAs can exactly accept the context-sensitive languages, in [8], where the equivalence of NCAs and NOCAs without time restrictions has been shown, and in [16], where it was shown in terms of homogeneous trellis automata that the real-time NOCA languages contain the ε-free context-free languages as well as a NP-complete language and form an AFL closed under intersection.

Here we are interested in a refinement of the amount of nondeterminism in order to identify the power and limitations of known cellular automata. The limitation of the number of allowed nondeterministic transitions is done according to a mapping $g : \mathbb{N} \to \mathbb{N}_0$ depending on the length of the input. By regarding the nondeterminism as a restrictable resource of the arrays the spectrum is drawn from the pure deterministic ($g(n) = 0$) to the full nondeterministic ($g(n) = t(n)$) devices.

In Section 2 we introduce such gG-CAs and gG-OCAs (g guess (O)CAs) and define the notions in terms of formal language processing. Even for constant mappings g the resulting arrays are powerful devices whose language recognition capabilities are neither affected by reducing the time complexity from linear- to real-time nor by restricting the information flow to one-way. Section 3 is devoted to the possibility to reduce the number of nondeterministic transitions. In Section 4 a speed-up result is shown that is stronger than all known results. Comparisons between various devices especially to deterministic ones are made in section 5. In Section 6 a characterization of an important subclass in terms of deterministic language families and ε-free homomorphisms are given. Thus, the nondeterminism can be replaced by the homomorphism and vice versa. Finally, in section 7 the strong closure properties of the real-time G-(O)CA languages are shown. They form an AFL closed under intersection and reversal.

2 Basic notions

We denote the rational numbers by $\mathbb{Q}$, the integers by $\mathbb{Z}$, the positive integers $\{1, 2, \ldots\}$ by $\mathbb{N}$, the set $\mathbb{N} \cup \{0\}$ by $\mathbb{N}_0$ and the powerset of a set S by 2^S. The empty word is denoted by ε and the reversal of a word w by w^R. For the length of w we write $|w|$. The set of mappings from M to N is denoted by N^M.

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

Definition 1 *A* nondeterministic (two-way) cellular automaton *(*NCA*) is a system* $(S, \delta_{nd}, \#, A, F)$, *where*

a) S *is the finite, nonempty set of* states,
b) $\# \notin S$ *is the* boundary state,
c) $A \subseteq S$ *is the nonempty set of* input symbols,
d) $F \subseteq S$ *is the nonempty set of* accepting states,
e) $\delta_{nd} : \left(S \cup \{\#\}\right)^3 \rightarrow (2^S \setminus \emptyset)$ *is the (nondeterministic)* local transition function.

Let $\mathcal{M} = (S, \delta_{nd}, \#, A, F)$ be an NCA. A *configuration* of $\mathcal{M}$ at some time $t \geq 0$ is a description of its global state, which is actually a mapping $c_t : [1, \ldots, n] \rightarrow S$ for $n \in \mathbb{N}$. The configuration at time 0 is defined by the initial sequence of states. For a given input word $w = w_1 \cdots w_n \in A^+$ we set $c_{0,w}(i) := w_i$, $1 \leq i \leq n$. During its course of computation an NCA steps nondeterministically through a sequence of configurations, whereby successor configurations are chosen according to the global transition Δ_{nd}:

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

$$c' \in \Delta_{nd}(c) \iff s'_1 \in \delta_{nd}(\#, s_1, s_2), s'_2 \in \delta_{nd}(s_1, s_2, s_3), \ldots, s'_n \in \delta_{nd}(s_{n-1}, s_n, \#)$$

Thus, Δ_{nd} is induced by δ_{nd}. For $i \in \mathbb{N}_0$ the i-fold composition of Δ_{nd} is defined as follows:

$$\Delta^0_{nd}(c) := \{c\}$$

$$\Delta^{i+1}_{nd}(c) := \bigcup_{c' \in \Delta^i_{nd}(c)} \Delta_{nd}(c')$$

$\pi_i(s_1 \cdots s_n) := s_i$ 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. The concatenation of a specific register of all cells forms a *track*.

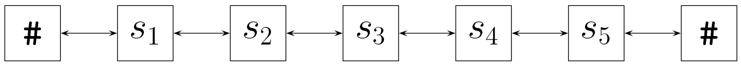

Figure 1: A (two-way) cellular automaton.

If the flow of information is restricted to one-way, the resulting device is a *nondeterministic one-way cellular automaton* (NOCA). I.e. the next state of each cell depends on the state of the cell itself and the state of its immediate neighbor to the right.

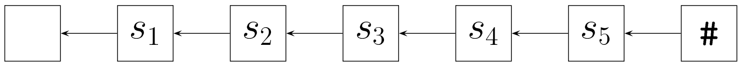

Figure 2: A one-way cellular automaton.

An NCA (NOCA) is *deterministic* if $\delta_{nd}(s_1, s_2, s_3)$ ($\delta_{nd}(s_1, s_2)$) is a singleton for all states $s_1, s_2, s_3 \in S \cup \{\#\}$. Deterministic cellular arrays are denoted by CA resp. OCA.

Definition 2 *Let $\mathcal{M} = (S, \delta_{nd}, \#, A, F)$ be an* NCA *or an* NOCA.

a) *A word $w \in A^+$ is accepted by $\mathcal{M}$ if there exists a time step $t_w \in \mathbb{N}$ such that there exists a configuration $c_{t_w} \in \Delta_{nd}^{t_w}(c_{0,w})$ where $c_{t_w}(1) \in F$.*

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 $t_w \leq t(|w|)$ time steps, then L is said to be of time complexity t.*

The family of all languages which can be accepted by an NCA (NOCA) with time complexity t is denoted by $\mathscr{L}_t(\text{NCA})$ ($\mathscr{L}_t(\text{NOCA})$). If t equals the *identity function* $id(n) := n$, acceptance is said to be in *real-time* and we write $\mathscr{L}_{rt}(\text{NCA})$ ($\mathscr{L}_{rt}(\text{NOCA})$). In the sequel we will use a corresponding notion for other types of acceptors. The *linear-time* languages $\mathscr{L}_{lt}(\text{NCA})$ ($\mathscr{L}_{lt}(\text{NOCA})$) are defined according to

$$\mathscr{L}_{lt}(\text{NCA}) := \bigcup_{k \in \mathbb{Q},\, k \geq 1} \mathscr{L}_{k \cdot id}(\text{NCA})$$

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 this reason a *deterministic local transition* $\delta_d : \left(S \cup \{\#\}\right)^3 \to S$ is provided and the global transition induced by δ_d is denoted by Δ_d.

Let $g : \mathbb{N} \to \mathbb{N}_0$ be a mapping that gives the number of allowed nondeterministic transitions dependent on the length of the input. The resulting system

$(S, \delta_{nd}, \delta_d, s_0, \texttt{\#}, A, F)$ is a gG-CA (gG-OCA) if starting with the initial configuration $c_{0,w}$ (for some $w \in A^+$) the possible configurations at some time t are given by the global transition as follows:

$$\begin{array}{ll} \{c_{0,w}\} & \text{if } t = 0 \\ \Delta_{nd}^{t}(c_{0,w}) & \text{if } t \leq g(|w|) \\ \displaystyle\bigcup_{c' \in \Delta_{nd}^{g(|w|)}(c_{0,w})} \Delta_d^{t-g(|w|)}(c') & \text{otherwise} \end{array}$$

Observe, that all nondeterministic transitions have to be applied before the deterministic ones and that for some $s_1, s_2, s_3 \in S \cup \{\texttt{\#}\}$ the nondeterministic part may contain the deterministic one: $\delta_d(s_1, s_2, s_3) \in \delta_{nd}(s_1, s_2, s_3)$.

Up to now we have g not required to be computable. Of course, for almost all applications we will have to do so but some of our general results can be developed without such requirement. Since in the sequel the (certainly computable) constant mappings g are playing an important role the following example might motivate their use.

Let $L \subseteq \{\texttt{a}\}^+$ be an arbitrary not necessarily recursively enumerable language that does not contain the word $\texttt{a}$. By

$$g(n) := \begin{cases} 1 & \text{if } \texttt{a}^n \notin L \\ n & \text{if } \texttt{a}^n \in L \end{cases}$$

a gG-OCA can accept L in real-time simply by verifying whether or not more than one nondeterministic transitions have been performed.

3 Guess reduction

The next two results show that $k+1$ guesses per cell are not better than k guesses. On the other hand, they yield the possibility to reduce the number of nondeterministic transitions by a constant as long as one remains.

Theorem 3 *Let $g : \mathbb{N} \to \mathbb{N}_0$, $g(n) \geq 1$, be a mapping and $k \in \mathbb{N}_0$ be a constant number. Then*

$$\mathscr{L}_t((g+k)\text{G-CA}) \subseteq \mathscr{L}_t(g\text{G-CA})$$

holds for all time complexities $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq g(n) + k$.

Proof. It suffices to prove the assertion for $k = 1$. For a given $(g+1)$G-CA $\mathcal{M} = (S, \delta_{nd}, \delta_d, \texttt{\#}, A, F)$ we define a gG-CA $\mathcal{M}' = (S', \delta'_{nd}, \delta'_d, \texttt{\#}, A, F)$ as follows.

Let $S_\# := S \cup \{\#\}$.

$S' := S \cup \big(S \times S^{S_\# \times S_\# \times S_\#}\big)$
$\forall\, s_1, s_2, s_3 \in S_\#,\ f_1, f_2, f_3 \in S^{S_\# \times S_\# \times S_\#}$:

$$\begin{aligned}
\delta'_d(s_1, s_2, s_3) &:= \delta_d(s_1, s_2, s_3) && (1)\\
\delta'_d\big((s_1, f_1), (s_2, f_2), (s_3, f_3)\big) &:= f_1(s_1, s_2, s_3) && (2)\\
\delta'_{nd}(s_1, s_2, s_3) &:= \big\{(s, f) \mid s \in \delta_{nd}(s_1, s_2, s_3) \\
&\qquad \wedge f \in S^{S_\# \times S_\# \times S_\#} \text{ such that} \\
&\qquad \forall\, \bar{s}_1, \bar{s}_2, \bar{s}_3 \in S_\# : \\
&\qquad\quad f(\bar{s}_1, \bar{s}_2, \bar{s}_3) \in \delta_{nd}(\bar{s}_1, \bar{s}_2, \bar{s}_3)\big\} && (3)\\
\delta'_{nd}\big((s_1, f_1), (s_2, f_2), (s_3, f_3)\big) &:= \big\{(s, f_2) \mid s \in \delta_{nd}(s_1, s_2, s_3)\big\} && (4)
\end{aligned}$$

We are going to show $L(\mathcal{M}) = L(\mathcal{M}')$ and that the time complexities are identical.

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 triples of states of $\mathcal{M}$ (rule(3)). The second result is stored in an additional register. It is a finite table which contains one row for every $(s_1, s_2, s_3) \in S_\#^3$. Thus, $\mathcal{M}$ can compute the first component of the states of $\mathcal{M}'$, too.

The deterministic transition of $\mathcal{M}'$ applied to states from $S_\#$ corresponds to the deterministic transition of $\mathcal{M}$ (rule (1)). Again, it follows that $\mathcal{M}$ can compute the same states. The deterministic transition of $\mathcal{M}'$ applied to states from $S_\# \times S^{S_\# \times S_\# \times S_\#}$ is an application of the second component of the state of the cell itself to the actual first components of the cell itself and of its both neighbors (rule (2)). This corresponds to a nondeterministic transition of $\mathcal{M}$. Since deterministic transitions of $\mathcal{M}'$ always result in states from S an application of rule (2) can only happen at time $g(n) + 1$. But at that time $\mathcal{M}$ performs its last nondeterministic transition and, hence, can simulate $\mathcal{M}'$. It follows $L(\mathcal{M}') \subseteq L(\mathcal{M})$.

On the other hand, the first $g(n)$ nondeterministic transitions of $\mathcal{M}$ are simulated by $\mathcal{M}'$ in the first components of the states (rules (3),(4)). At time $g(n) + 1$ the first deterministic transition of $\mathcal{M}'$ is the application of the nondeterministically chosen second component to the actual first components (rule (2)) such that a nondeterministic transition (of $\mathcal{M}$) is deterministically simulated (by $\mathcal{M}'$). From time $g(n) + 2$ to $t(n)$ automaton $\mathcal{M}'$ simulates $\mathcal{M}$ directly (rule (1)). It follows $L(\mathcal{M}) \subseteq L(\mathcal{M}')$. □

The next theorem does not follow for structural reasons since in general the single cells are not able to recognize the time step $g(n)$. So we have to ensure that despite of the additional nondeterministic transitions no additional inputs are accepted.

Theorem 4 *Let $g : \mathbb{N} \to \mathbb{N}_0$, $g(n) \geq 1$, be a mapping and $k \in \mathbb{N}_0$ be a constant number. Then*

$$\mathscr{L}_t(g\text{G-CA}) \subseteq \mathscr{L}_t((g+k)\text{G-CA})$$

holds for all time complexities $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq g(n) + k$.

Proof. It suffices to prove the assertion for $k = 1$.

For a given gG-CA $\mathcal{M} = (S, \delta_{nd}, \delta_d, \#, A, F)$ we define a $(g+1)$G-CA $\mathcal{M}' = (S', \delta'_{nd}, \delta'_d, \#, A, F')$ as follows.

$S' := S \cup S^2$

$F' := F \cup \{(s_1, s_2) \in S^2 \mid s_2 \in F\}$

$\forall\, s_1, s_2, s_3, s_4, s_5, s_6 \in S_{\#}$:

$$\begin{aligned}
\delta'_{nd}(s_1, s_2, s_3) &:= \big\{(s'_1, s'_2) \mid s'_1 \in \delta_{nd}(s_1, s_2, s_3) \\
&\qquad \wedge\, s'_2 = \delta_d(s_1, s_2, s_3)\big\} && (1)\\
\delta'_{nd}\big((s_1, s_2), (s_3, s_4), (s_5, s_6)\big) &:= \big\{(s'_1, s'_2) \mid s'_1 \in \delta_{nd}(s_1, s_3, s_5) \\
&\qquad \wedge\, s'_2 = \delta_d(s_1, s_3, s_5)\big\} && (2)\\
\delta'_d\big((s_1, s_2), (s_3, s_4), (s_5, s_6)\big) &:= \delta_d(s_2, s_4, s_6) && (3)\\
\delta'_d(s_1, s_2, s_3) &:= \delta_d(s_1, s_2, s_3) && (4)
\end{aligned}$$

We have to show $L(\mathcal{M}) = L(\mathcal{M}')$ and that $\mathcal{M}$ and $\mathcal{M}'$ are of the same time complexity.

If δ'_{nd} is applied to pairs from S^2 its result is independent on the second components (rule (2)). The first component of the result corresponds to an application of δ_{nd} whereas the second component of the result corresponds to an application of δ_d. Thus, $\mathcal{M}$ can compute the first components of the first $g(n)$ configurations of $\mathcal{M}'$, respectively. The last nondeterministic transition of $\mathcal{M}'$ at time $g(n)+1$ is computed by $\mathcal{M}$ with respect to the second components only, because the $(g(n)+1)$-th transition of $\mathcal{M}$ is a deterministic one. But since the next transition of $\mathcal{M}'$ at time $g(n)+2$ uses the second components only (rule (3)), $\mathcal{M}$ computes the corresponding configuration correctly. From time $g(n)+2$ to $t(n)$ the transitions δ_d and δ'_d are identical. By the definition of F' the case $g(n)+1 = t(n)$ is handled. It follows $L(\mathcal{M}') \subseteq L(\mathcal{M})$.

For the converse, it follows directly from the definition of $\mathcal{M}'$ that $\mathcal{M}'$ simulates $\mathcal{M}$ whereby additionally to the nondeterministic transitions in the first registers a deterministic transition in the second registers is simulated (rules (1),(2)). Since the first deterministic transition of $\mathcal{M}'$ uses the second components only (rule (3)), the last simulation of δ_{nd} is not regarded. It follows $L(\mathcal{M}) \subseteq L(\mathcal{M}')$. □

The following corollary summarizes the last two theorems.

Corollary 5 *Let $g : \mathbb{N} \to \mathbb{N}_0$, $g(n) \geq 1$, be a mapping and $k \in \mathbb{N}_0$ be a constant number. Then*

$$\mathscr{L}_t(g\text{G-CA}) = \mathscr{L}_t((g+k)\text{G-CA})$$

holds for all time complexities $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq g(n) + k$.

By the techniques shown above the following corollary concerning one-way arrays is easily derived.

Corollary 6 *Let $g : \mathbb{N} \to \mathbb{N}_0$, $g(n) \geq 1$, be a mapping and $k \in \mathbb{N}_0$ be a constant number. Then*

$$\mathscr{L}_t(g\text{G-OCA}) = \mathscr{L}_t((g+k)\text{G-OCA})$$

holds for all time complexities $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq g(n) + k$.

In general, we cannot reduce the number of nondeterministic transitions to 0 even if g is a constant mapping. In Theorem 14 it will be shown that $\mathscr{L}_{rt}(\text{OCA}) = \mathscr{L}_{rt}(0\text{G-OCA})$ is a proper subfamily of $\mathscr{L}_{rt}(1\text{G-OCA})$. Therefore, the condition $g(n) \geq 1$ is necessary.

4 Speed-up

It is known that several types of cellular automata can be sped-up by a constant amount of time as long as the remaining time complexity does not fall below real-time. A proof in terms of trellis automata can be found in [5]. In [17, 18] the speed-up results are shown for deterministic and nondeterministic cellular and iterative automata. The proofs are based on sequential machine characterizations of the parallel devices. Both techniques can be adapted to G-(O)CAs in a straightforward manner. Exemplarily, a corresponding result has been shown for 1G-OCAs in [3].

Lemma 7 *Let $g : \mathbb{N} \to \mathbb{N}_0$ be a mapping and $k \in \mathbb{N}_0$ be a constant number. Then*

$$\mathscr{L}_{t+k}(g\text{G-CA}) = \mathscr{L}_t(g\text{G-CA}) \quad \textit{and} \quad \mathscr{L}_{t+k}(g\text{G-OCA}) = \mathscr{L}_t(g\text{G-OCA})$$

hold for all time complexities $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$.

It is sometimes convenient to have such results for constructions: For example, after the k-th nondeterministic step a deterministic $t(n)$-time (O)CA can be simulated and subsequently the resulting $(t(n) + k)$-time kG-(O)CA can be sped-up to a $t(n)$-time kG-(O)CA again. Observe that in case of real-time it is not possible to speed-up the deterministic (O)CA by k time steps before its simulation.

Deterministic CAs and OCAs can be sped-up from $(n+t(n))$-time to $(n+\frac{t(n)}{k})$-time [1, 17, 18]. Thus, linear-time is close by real-time. The question whether every linear-time CA can be sped-up to real-time is an open problem. Related to that question is a closure property of the family $\mathscr{L}_{rt}(\text{CA})$: The real-time CA languages are closed under reversal iff the linear-time and real-time CA languages are identical [15].

Both problems are solved for OCAs. The family $\mathscr{L}_{rt}(\text{OCA})$ is closed under reversal [5] and the real-time OCA languages are a proper subfamily of the linear-time OCA languages ($\mathscr{L}_{rt}(\text{OCA}) \subset \mathscr{L}_{(1+\varepsilon)\cdot id}(\text{OCA}) = \mathscr{L}_{lt}(\text{OCA})$) [5, 26].

Now we are going to show a stronger result for G-(O)CAs from which follows that real-time is as powerful as linear-time. Since we need just one nondeterministic transition and one-way information flow only, the strong speed-up is shown for 1G-OCAs (i.e. the weakest device in question) at first. Subsequently, the result follows for G-(O)CAs as corollary.

In order to prove the theorem we introduce a technique that is later on referenced as *packing-and-checking*. The basic idea is to guess the input in a packed form on the left of the array. The verification of the guess can be done by a deterministic OCA in real-time what is shown by the next two lemmas.

Let B be an arbitrary alphabet that does neither contain the blank symbol $\mathtt{e}$ nor the border symbol $\#$. Following the idea each cell of the OCA has k registers for the packed part of the input and one register for its original input. The next two mappings extract the packed resp. the original input from a cell.

$$\begin{aligned} h_{k,1} : \left(B \cup \{\mathtt{e}\}\right)^k \times B &\to \left(B \cup \{\mathtt{e}\}\right)^k \\ w_1, \dots, w_k, w_{k+1} &\mapsto w_1 \cdots w_k \end{aligned}$$

and

$$\begin{aligned} h_{k,2} : \left(B \cup \{\mathtt{e}\}\right)^k \times B &\to B \\ w_1, \dots, w_k, w_{k+1} &\mapsto w_{k+1} \end{aligned}$$

The following lemma allows to verify whether (after a guess) the concatenation of the first k registers of all cells yields to a word beginning with n symbols from B and ending with $(k-1)\cdot n$ blank symbols $\mathtt{e}$. I.e., whether the packed input has the correct length and is contained in the leftmost $\lceil \frac{n}{k} \rceil$ cells.

Lemma 8 *Let $k > 1$ be a constant number. Then*

$$L_{k,1} := \left\{ w = w_1 \cdots w_n \mid w_1, \dots, w_n \in \left(B \cup \{\mathtt{e}\}\right)^k \times B \wedge h_{k,1}(w) \in B^n \mathtt{e}^{(k-1)\cdot n} \right\}$$

is a real-time OCA *language.*

Proof. A corresponding OCA has to perform two checking tasks (cf. Figure 3). The first is to verify that $h_{k,1}(w)$ is of the form $B^*\mathtt{e}^+$. Therefore, the cell which contains the last symbol of the packed input generates a signal $^\bullet$ in the corresponding register. The signal passes through the registers and cells in descending order and must not meet a symbol $\mathtt{e}$. Otherwise an error signal is

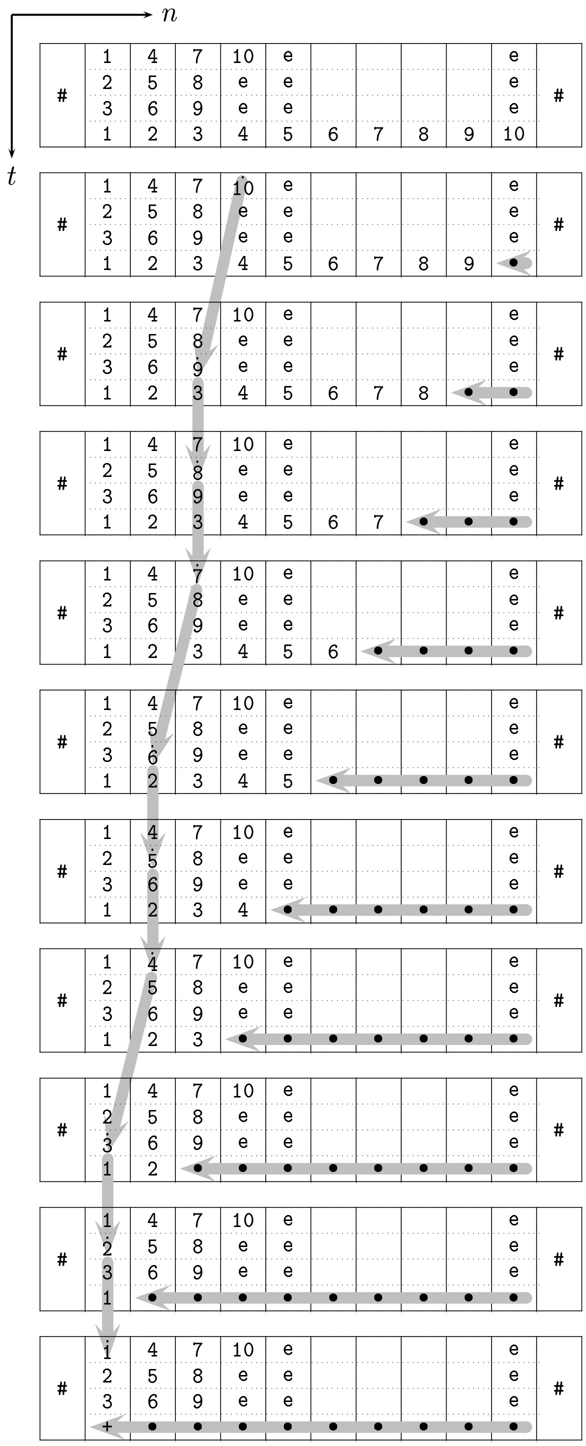

Figure 3: Example to the proof of Lemma 8 ($k = 3$).

generated that prohibits the leftmost cell to accept. Additionally, error signals are generated if a register of a cell contains the blank symbol followed by a nonblank symbol.

The second task is to verify that the length of the packed input meets the length of the array. It is simply performed by initiating a left moving signal in the rightmost cell at the first time step. The lengths are identical iff it arrives at the left border exactly when the signal $^\bullet$ of the first task arrives in the first register of the leftmost cell. $\square$

To verify the guess it remains to show that the packed input is identical to the original input on the $(k+1)$-th track:

Lemma 9 *Let $k > 1$ be a constant number. Then*

$$L_{k,2} := \{w = w_1 \cdots w_n \mid w_1, \ldots, w_n \in (B \cup \{\mathsf{e}\})^k \times B \wedge h_{k,1}(w) = h_{k,2}(w)\mathsf{e}^{(k-1)\cdot n}\}$$

is a real-time OCA *language.*

Proof. Since $\mathscr{L}_{rt}(\text{OCA})$ is closed under intersection [19] we may assume that the input belongs to the language $L_{k,1}$ of Lemma 8.

The $\lceil \frac{n}{k} \rceil$ cells containing the packed input are able to identify themselves by the contents of their first k registers. For what follows every cell has another k registers which work like a first-in-first-out (FIFO) queue (cf. Figure 4). The next input symbol to the queue is the content of the $(k+1)$-th register, respectively.

The rightmost $n - \lceil \frac{n}{k} \rceil$ cells shift the content of the $(k+1)$-th track successively leftward. Thus, they are implementing the input stream to the FIFO queue. At the end of the input stream – marked by a $\bullet$ – each of the leftmost $\lceil \frac{n}{k} \rceil$ cells compares its FIFO content to its packed input. If the comparisons of all cells are successful the input is accepted by a signal $\mathbf{+}$ in the $(k+1)$-th registers. Since the $\mathbf{+}$ is generated one time step after the arrival of the end-of-input-stream marker $\bullet$ the OCA works in $n+1$ time, but can be sped-up by one time step to real-time. $\square$

Now we are prepared to prove the strong speed-up result for 1G-OCAs:

Theorem 10 *Let $k \in \mathbb{N}$ be a constant number. Then*

$$\mathscr{L}_{k \cdot t}(\text{1G-OCA}) = \mathscr{L}_t(\text{1G-OCA})$$

holds for all time complexities $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$.

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

Let L be a language belonging to $\mathscr{L}_{k \cdot t}(\text{1G-OCA})$ and let $\mathcal{M}$ be a 1G-OCA that accepts L with time complexity $k \cdot t$. We construct a 2G-OCA $\mathcal{M}'$ that simulates $\mathcal{M}$ in time $t(n) + 1$. The underlying technique is packing-and-checking.

The idea is as follows: On an input of length n each cell i with $1 \leq i \leq \lceil \frac{n}{k} \rceil$ of $\mathcal{M}'$ guesses the initial states of the cells $k(i-1)+1, k(i-1)+2, \ldots, ki$ in its first k registers and remembers its original input in its $(k+1)$-th register (here we may assume that all cells $i > n$ initially contain an e). Based on

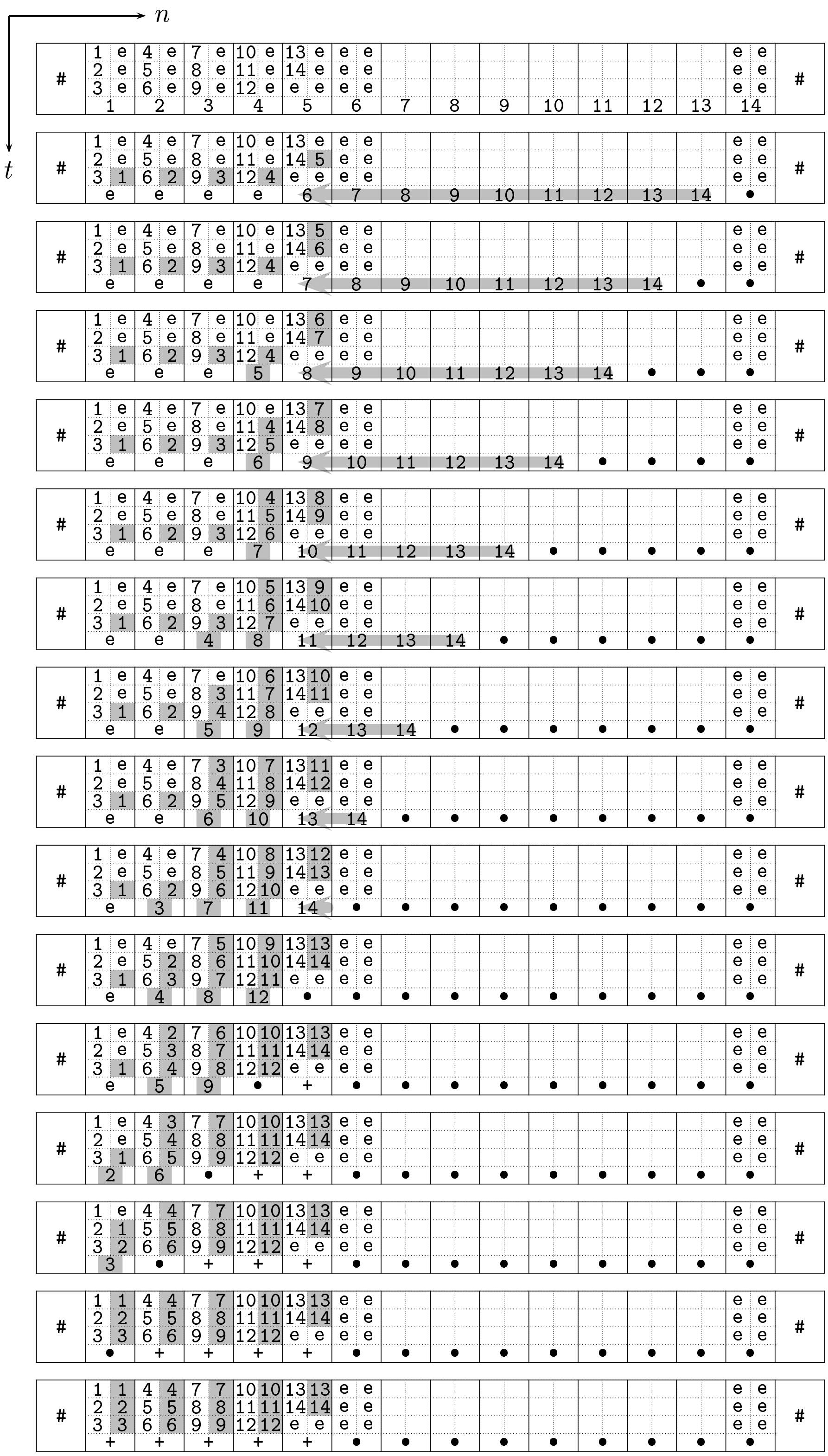

Figure 4: Example to the proof of Lemma 9 ($k = 3$).

this compressed representation $\mathcal{M}'$ can simulate k time steps of $\mathcal{M}$ per time step which yields the required speed-up. For this reason $\mathcal{M}'$ needs a second nondeterministic transition and another $t(n)$ time steps.

In parallel $\mathcal{M}'$ has to check whether the guesses of the initial states were correct, which can be done by the simulation of the acceptor of Lemma 9. For the verification n time steps are needed. By Corollary 6 and Lemma 7 there exists a 1G-OCA accepting $L(\mathcal{M})$ with time complexity $t(n)$. $\square$

Although the simulation on the compressed representation may be faster than real-time a speed-up below real-time is, of course, not possible due to the time needed for packing-and-checking. The necessary conditions for the strong speed-up are the possibility to simulate the original device with k-fold speed on a compressed representation of the input, the possibility to speed-up the resulting device by a constant amount of time, and the possibility to reduce the number of nondeterministic transitions by a constant. Due to Corollaries 5 and 6 and Lemma 7 these conditions are met by gG-OCAs and gG-CAs as well.

Corollary 11 *Let $g : \mathbb{N} \to \mathbb{N}_0$, $g(n) \geq 1$, be a mapping and $k \in \mathbb{N}$ be a constant number. Then*

$$\mathscr{L}_{k \cdot t}(g\text{G-OCA}) = \mathscr{L}_t(g\text{G-OCA}) \quad \textit{and} \quad \mathscr{L}_{k \cdot t}(g\text{G-CA}) = \mathscr{L}_t(g\text{G-CA})$$

hold for all time complexities $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$.

In [8] $\mathscr{L}(\text{NOCA}) = \mathscr{L}(\text{NCA})$ has been shown. The proof is based on a set of signals along which verifications of previous guesses are done. The last verification needs another id time steps such that the simulation of an NCA by an NOCA is at the cost of additional id time steps. Thus, $\mathscr{L}_{t+id}(\text{NOCA}) = \mathscr{L}_t(\text{NCA})$ is proved. As an application of our strong speed-up result we can strengthen the simulation avoiding the increase of the time complexity.

Theorem 12

$$\mathscr{L}_t(\text{NOCA}) = \mathscr{L}_t(\text{NCA})$$

holds for all time complexities $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$.

Proof.

From [8] $\mathscr{L}_{t+id}(\text{NOCA}) = \mathscr{L}_t(\text{NCA})$ is known. In order to apply Corollary 11 let $g := t + id$ and $k = 2$. By $\mathscr{L}_{t+id}(\text{NOCA}) = \mathscr{L}_{t+id}((t+id)\text{G-OCA})$ it follows $\mathscr{L}_{t+id}(\text{NOCA}) = \mathscr{L}_{\frac{t+id}{2}}(\text{NOCA})$. Since $id \leq t$ we obtain $\mathscr{L}_{\frac{t+id}{2}}(\text{NOCA}) \subseteq \mathscr{L}_{\frac{2 \cdot t}{2}}(\text{NOCA}) = \mathscr{L}_t(\text{NOCA})$.

$\square$

Throughout the present paper real-time and linear-time acceptors are of particular interest. Therefore, we emphasize their specific relations.

Corollary 13 *Let $g : \mathbb{N} \to \mathbb{N}_0$, $g(n) \geq 1$, be a mapping. Then*

$$\mathscr{L}_{rt}(g\text{G-OCA}) = \mathscr{L}_{lt}(g\text{G-OCA}) \quad \textit{and} \quad \mathscr{L}_{rt}(g\text{G-CA}) = \mathscr{L}_{lt}(g\text{G-CA}).$$

5 Ranging in the hierarchy of deterministic (O)CA languages

The present section is devoted to compare the computing power of the kG-(O)CAs to the well investigated deterministic devices. We proceed by ranging in them into the following hierarchy of deterministic (O)CA language families.

It is known that the real-time OCA languages are properly included in the linear-time OCA languages, which in turn are identical to the reversal of the real-time CA languages ($\mathscr{L}_{rt}(\text{OCA}) \subset \mathscr{L}_{lt}(\text{OCA}) = \mathscr{L}_{rt}^{R}(\text{CA})$) [5, 17, 26]. In [4, 14] the inclusion $\mathscr{L}_{lt}(\text{CA}) \subseteq \mathscr{L}(\text{OCA})$ has been shown. Together with the inclusions for structural reasons we obtain $\mathscr{L}_{rt}(\text{OCA}) \subset \mathscr{L}_{lt}(\text{OCA}) = \mathscr{L}_{rt}^{R}(\text{CA}) \subseteq \mathscr{L}_{lt}(\text{CA}) \subseteq \mathscr{L}(\text{OCA}) \subseteq \mathscr{L}(\text{CA})$.

Most of the following results are obtained for 1G-(O)CAs but hold for kG-(O)CAs, too. On the other hand, due to the homomorphic characterization (Section 6) and the enormous increase of computing power by adding just one nondeterministic transition the 1G-(O)CAs are an important subclass of the gG-(O)CAs.

At first we show that the computing power of real-time OCAs is, in fact, strictly increased by adding one nondeterministic step to that device.

Theorem 14 $\mathscr{L}_{rt}(\text{OCA}) \subset \mathscr{L}_{rt}(\text{1G-OCA})$

Proof. Obviously, we have an inclusion between the families since the nondeterministic part of the state transition can be designed to be deterministic.

Let L be the language $\{\mathtt{a}^p \mid p \text{ is a prime number}\}$. In [9] it is shown that L belongs to $\mathscr{L}_{rt}(\text{CA})$. Since L is a unary language from the hierarchy $L \in \mathscr{L}_{lt}(\text{OCA})$ follows. For structural reasons we obtain $L \in \mathscr{L}_{lt}(\text{1G-OCA})$. By Theorem 10 a corresponding linear-time 1G-OCA can be sped-up to real-time, hence, L belongs to $\mathscr{L}_{rt}(\text{1G-OCA})$.

On the other hand, L does not belong to $\mathscr{L}_{rt}(\text{OCA})$ since it is not a regular language [23]. Thus, the inclusion is a proper one. $\square$

The previous result can be strengthened furthermore. Admittedly, we loose the strictness of the inclusion. The question of the strictness is strongly related to the famous open problem whether or not the real-time CA languages are a proper subfamily of the CA languages.

Theorem 15 $\mathscr{L}_{lt}(\text{CA}) \subseteq \mathscr{L}_{rt}(\text{1G-OCA})$

Proof. Let $L \in \mathscr{L}_{lt}(\text{CA})$. Since $\mathscr{L}_{lt}(\text{CA})$ is closed under reversal [24] there exists a linear-time CA that accepts L^R. This CA, in turn, can be sped-up by a multiplicative and additive constant [18]. We obtain a CA $\mathcal{M} = (S, \delta_d, \#, A, F)$ that accepts L^R with time complexity $2n - 1$.

Now in a first step a deterministic OCA $\mathcal{M}' = (S', \delta'_d, \texttt{\#}, A', F)$ is constructed such that $\mathcal{M}'$ accepts the language $\{\mathsf{e}^{|w|}w^R \mid w \in L(\mathcal{M})\}$ with time complexity $2n-2$. Here we assume $\mathsf{e} \notin S$ and w.l.o.g. $n > 1$.

$S' := \big(S \cup \{\mathsf{e}\}\big) \cup \big(S \cup \{\mathsf{e}\}\big)^2$

$A' := A \cup \{\mathsf{e}\}$

$\forall\, s_1, s_2 \in S \cup \{\mathsf{e}\}:$

$$\begin{aligned} \delta'_d(s_1, \texttt{\#}) &:= (s_1, \mathsf{e}) \\ \delta'_d(s_1, s_2) &:= (s_1, s_2) \end{aligned}$$

$\forall\, (s_1, s_2), (s_3, s_4) \in \big(S \cup \{\mathsf{e}\}\big)^2:$

$$\delta'_d\big((s_1, s_2), (s_3, s_4)\big) := \begin{cases} \delta_d(s_4, s_2, s_1) & \text{if } (s_1 \neq \mathsf{e} \wedge s_2 \neq \mathsf{e} \wedge s_4 \neq \mathsf{e}) \\ \delta_d(\texttt{\#}, s_2, s_1) & \text{if } (s_1 \neq \mathsf{e} \wedge s_2 \neq \mathsf{e} \wedge s_4 = \mathsf{e}) \\ \delta_d(s_4, s_2, \texttt{\#}) & \text{if } (s_1 = \mathsf{e} \wedge s_2 \neq \mathsf{e} \wedge s_4 \neq \mathsf{e}) \\ \mathsf{e} & \text{otherwise} \end{cases}$$

The basic idea is that during an intermediate step the cells of $\mathcal{M}'$ are collecting the information needed to simulate one step of the CA (cf. Figure 5). Due to the one-way information flow a cell i thereby can collect information from the cells $i+1$ and $i+2$ and, thus, simulate one step of the CA cell $i+1$. Therefore, the relevant part of the configuration shifts in space to the left.

Since $\mathscr{L}_{rt}(\text{OCA})$ is closed under reversal Lemma 9 holds for $L^R_{k,2}$, too, and $L^R_{k,2}$ is a real-time OCA language.

The cells of a 1G-OCA $\mathcal{M}''$ that accepts the language $\{w^R \mid w \in L(\mathcal{M})\}$ are constructed such that they can store two input symbols. Under input w^R the 1G-OCA $\mathcal{M}''$ guesses in its first step the configuration $\mathsf{e}^{|w|}w^R$ whereby two adjacent symbols are stored in one cell, respectively. The verification of the guess corresponds to the acceptance of the language $L^R_{k,2}$ for $k = 2$.

In parallel to the verification $\mathcal{M}''$ simulates the OCA $\mathcal{M}'$ with double speed on the compressed input. Therefore, $\mathcal{M}''$ accepts with time complexity $1 + \frac{2n-2}{2} = n$. Since $L(\mathcal{M}'') = \{w^R \mid w \in L(\mathcal{M})\} = \big(L(\mathcal{M})\big)^R = \big(L^R\big)^R = L$ the theorem follows. □

The next result fits the family $\mathscr{L}_{rt}(\text{1G-OCA})$ (and due to Theorem 23 the family $\mathscr{L}_{rt}(\text{1G-CA})$ as well) into the known hierarchy of deterministic language families. Moreover, it shows that the increase of computing power gained in adding one nondeterministic transition to real-time OCAs cannot be achieved in general (i.e. for arrays without time limits). Conversely, we can avoid the nondeterministic transitions without reducing the computing power.

Theorem 16 $\mathscr{L}(\text{1G-OCA}) = \mathscr{L}(\text{OCA})$

Proof. In [14] the equivalence of OCAs and a restricted online single-tape Turing machine has been shown. Such a *sweeping machine* (SM) works as follows: The semi-infinite working tape is bounded to the right. Initially all cells

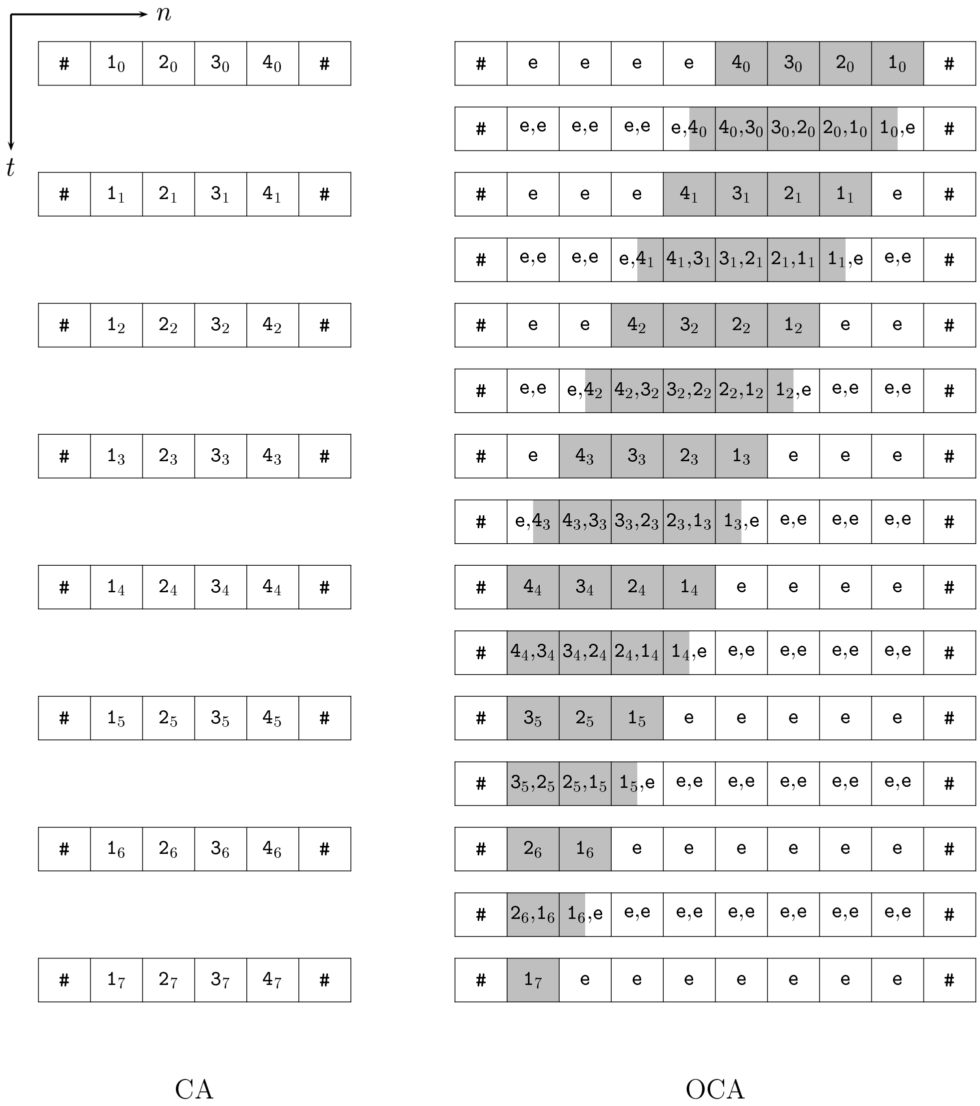

Figure 5: Example to the proof of Theorem 15.

except the rightmost one are empty. The rightmost cell contains an endmarker #. The SM starts scanning the # and performs successively right to left sweeps over the nonempty part of the working tape. At the beginning of its i-th sweep, $1 \leq i \leq n$, it reads the i-th input symbol and moves its read-write head from # to the left. Subsequently, it continues moving leftward rewriting the nonempty cells passed through (but does not erase them). The sweep ends on the first empty cell which may be rewritten. Now the read-write head resets to the rightmost cell and the SM starts the next sweep. After reading the whole input an end-of-input symbol is assumed to be the permanent input. When reading the end-of-input symbol for the first time the SM rewrites the empty cell at the end of its sweep by # and subsequent sweeps are bounded by the left and right # endmarkers, thus, not expanding the working tape.

It is easily seen and shown in [4] that under input $w_1 \cdots w_n$, $w_i \in A$, such a sweeping machine can partition its working tape during the first $n+1$ sweeps as depicted in Figure 6 (here we may assume that each cell of the working tape has two registers, thus, speaking loosely of a tape of length $2n$). The cells of the left partition are divided into three tracks. On the first one the input is stored. The second and third track is empty. The cells of the right partition are containing zeroes.

<table>
<tr><td rowspan="3">#</td><td>w_1</td><td>w_2</td><td>$\cdots$</td><td>w_{n-1}</td><td>w_n</td><td rowspan="3">0</td><td rowspan="3">0</td><td rowspan="3">$\cdots$</td><td rowspan="3">0</td><td rowspan="3">0</td><td rowspan="3">#</td></tr>
<tr><td></td><td></td><td>$\cdots$</td><td></td><td></td></tr>
<tr><td></td><td></td><td>$\cdots$</td><td></td><td></td></tr>
</table>

Figure 6: Partitions of the SM in the proof of Theorem 16.

Let $\mathcal{M} = (S, \delta_{nd}, \delta_d, \texttt{\#}, A, F)$ be an arbitrary 1G-OCA. A SM that accepts $L(\mathcal{M})$ works as follows:

On the right partition a $|S|$-ary counter is realized where the least significant digit is at the right. Obviously, the SM can increment the counter during one sweep. The generation of a carry-over at the most significant digit synchronizes the computation on the left partition (it occurs after $|S|^n$ sweeps, respectively).

Suppose now on one of the tracks of the left partition the configuration c_i at some time $i \geq 1$ of $\mathcal{M}$ is stored. Then the SM can compute the successor configuration c_{i+1} during one sweep. It suffices to remember the content $c_i(j)$ of a cell j, $1 \leq j \leq n$, while moving one cell to the left. On the new cell the SM finds the state $c_i(j-1)$ and now is able to simulate the local transformation $\delta_d\big(c_i(j-1), c_i(j)\big)$ in order to obtain $c_{i+1}(j-1)$.

During the whole computation the SM keeps the original input on the first track of the left partition unchanged.

On the second track again a $|S|$-ary counter is realized that produces successively all configurations of $\mathcal{M}$ after its guess (at time 1) (here we identify the state set S by the ordered set $\{0, \ldots, |S|-1\}$). Since the original input is available on the first track the SM can verify whether a counter value corresponds to a valid configuration at time 1 (i.e., whether the state given by the digit of a cell $j-1$ belongs to $\delta_{nd}\big(c_0(j-1), c_0(j)\big) = \delta_{nd}(w_{j-1}, w_j)$).

The counter on the second track is incremented if and only if the synchronizing counter on the right partition overflows, thus, each $|S|^n$ sweeps. In parallel during such sweeps the second track is copied onto the third one.

During the following $|S|^n - 1$ sweeps the content of the third track is updated according to the local transformation δ_d of $\mathcal{M}$. Hence, the configurations $c_2, \ldots, c_{|S|^n}$ are generated successively. Since $\mathcal{M}$ has at most $|S|^n$ different configurations of length n, the simulation started with the configuration c_1 is completely performed.

On the other hand, all possible configurations c_1 are generated at the second track such that the SM simulates all possible computations of $\mathcal{M}$ on input

$w_1 \cdots w_n$. The SM accepts the input if one of the simulations is accepting, it rejects if the counter on the left partition overflows before one accepting computation has been simulated.

It follows $\mathscr{L}(\text{1G-OCA}) \subseteq \mathscr{L}(\text{OCA})$ and for structural reasons $\mathscr{L}(\text{OCA}) \subseteq \mathscr{L}(\text{1G-OCA})$ what proves the theorem. □

The following Corollary gives an upper bound for the family $\mathscr{L}_{rt}(\text{1G-OCA})$.

Corollary 17 $\mathscr{L}_{rt}(\text{1G-OCA}) \subseteq \mathscr{L}(\text{OCA})$

Now we have $\mathscr{L}_{lt}(\text{CA}) \subseteq \mathscr{L}_{rt}(\text{1G-OCA}) \subseteq \mathscr{L}(\text{OCA})$. As will be shown by Theorem 23 the real-time computing power of 1G-OCAs is not increased when two-way information flow is provided.

A summary of the relations between several language families is depicted in Figure 7.

6 Homomorphic characterization

In [22] Myhill has proved that the regular languages are exactly the closure of the finite languages under union, concatenation and iteration. Such results open the possibility to characterize certain language families by, in some sense, simpler ones and some kind of operations. Besides they shed some light on the structure of the family itself they may be used as powerful reduction tool in order to simplify some proofs or constructions.

In the present section we are going to characterize the real-time 1G-(O)CA languages by the closure of $\mathscr{L}_{rt}((\text{O})\text{CA})$s under ε-free homomorphism. Thus replacing the nondeterminism by ε-free homomorphisms and vice versa. An application of the homomorphic characterization yields the result $\mathscr{L}_{rt}(\text{1G-CA}) = \mathscr{L}_{rt}(\text{1G-OCA})$ from which follows that for such devices one-way information flow is sufficient.

In order to prove the characterization for real-time 1G-OCAs we need the closure of $\mathscr{L}_{rt}(\text{OCA})$ under a weak kind of homomorphism.

Definition 18 *Let $h : A^* \to B^*$ be an ε-free homomorphism. h is* structure preserving *iff for every two $\mathtt{a}, \mathtt{a}' \in A$ with $\mathtt{a} \neq \mathtt{a}'$ and $h(\mathtt{a}) = \mathtt{b}_1 \cdots \mathtt{b}_m$ 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.*

Lemma 19 $\mathscr{L}_{rt}(\text{OCA})$ *is closed under structure preserving homomorphism.*

Proof. Let $A = \{\mathtt{a}_1, \ldots, \mathtt{a}_m\}$ 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:

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

where $\mathtt{b}_{i,j} \in B$. Observe that the $\mathtt{b}_{i,j}$ are not necessarily different.

A deterministic generalized sequential machine (gsm) [13] is defined as follows: The input alphabet is B, the output alphabet is A and the set of states is $S = \{s_{i,j} \mid 1 \le i \le m \text{ and } 2 \le j \le n_i\} \cup \{s_0, s_e\}$ 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 $1 \le i \le m$ and $s_{k,l} \in S \setminus \{s_0, s_e\}$:

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

The gsm reads an input word w' from B^* and emits an output word w from A^*. If additionally the 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 [16] 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 $w = \mathtt{a}_1 \cdots \mathtt{a}_p \in L$ we have $h(w) = \mathtt{b}_{1,1} \cdots \mathtt{b}_{1,n_1} \cdots \mathtt{b}_{p,1} \cdots \mathtt{b}_{p,n_p}$. Since h is structure preserving the $\mathtt{b}_{i,1}$ are all different, thus, if started in state s_0 the computation path of gsm under input $h(\mathtt{a}_i)$, $1 \le i \le m$, is $(s_0, \mathtt{b}_{i,1}) \xrightarrow{\mathtt{a}_i} s_0$ if $n_i = 1$. The output is written on top of the arrow. If $n_i > 1$ we obtain $(s_0, \mathtt{b}_{i,1}) \xrightarrow{\varepsilon} (s_{i,2}, \mathtt{b}_{i,2}) \xrightarrow{\varepsilon} (s_{i,3}, \mathtt{b}_{i,3}) \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} (s_{i,n_i}, \mathtt{b}_{i,n_i}) \xrightarrow{\mathtt{a}_i} s_0$. In both cases gsm maps $h(\mathtt{a}_i) \mapsto \mathtt{a}_i$. 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 $gsm^{-1}(L)$. There must exist a word $w = \mathtt{a}_1 \cdots \mathtt{a}_p \in L$ such that $gsm(w') = w$. We consider an arbitrary symbol in w, say $\mathtt{a}_i$. The possible transition steps of gsm that emit $\mathtt{a}_i$ are $(s_{i,n_i}, \mathtt{b}_{i,n_i}) \xrightarrow{\mathtt{a}_i} s_0$ and $(s_0, \mathtt{b}_{i,1}) \xrightarrow{\mathtt{a}_i} s_0$. 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 $(s_0, \mathtt{b}_{i,1}) \xrightarrow{\mathtt{a}_i} s_0$ 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 $(s_{i,n_i}, \mathtt{b}_{i,n_i}) \xrightarrow{\mathtt{a}_i} s_0$ 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 $w' = \mathtt{b}_{1,1} \cdots \mathtt{b}_{1,n_1} \cdots \mathtt{b}_{p,1} \cdots \mathtt{b}_{p,n_p} = h(\mathtt{a}_1) \cdots h(\mathtt{a}_p) = h(w) \in h(L)$. $\square$

Now we can use the previous lemma to prove the characterization.

Theorem 20

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

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

Proof.

a) Let $L \in \mathscr{L}_{rt}(\text{OCA})$ be a language over the alphabet $A = \{\mathtt{a}_1, \ldots, \mathtt{a}_m\}$ and an ε-free 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 $\bar{B} := \{\bar{\mathtt{b}}_{1,1}, \ldots, \bar{\mathtt{b}}_{1,n_1}, \bar{\mathtt{b}}_{2,1}, \ldots, \bar{\mathtt{b}}_{m,n_m}\}$ 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 $\mathscr{L}_{rt}(\text{OCA})$ is closed under structure preserving homomorphism $h'(L)$ is a real-time OCA language. Define an ε-free length preserving homomorphism $h'' : \bar{B}^* \to B^*$: $h''(\bar{\mathtt{b}}_{1,1}) = \mathtt{b}_{1,1}, \ldots, h''(\bar{\mathtt{b}}_{m,n_m}) = \mathtt{b}_{m,n_m}$.

Obviously, we have $h(L) = h''(h'(L))$.

A 1G-OCA $\mathcal{M}'$ accepting $h(L)$ 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 $\mathcal{M}$ accepting $h'(L)$. As shown in Lemma 7 we can speed-up $\mathcal{M}'$ by one time step.

Trivially, for each $w \in h(L)$ there exists a $w' \in L$ such that $h(w') = w$. Define $w'' := h'(w')$ then w'' is a preimage of w under h'' and, thus, $\mathcal{M}'$ accepts w.

On the other hand, if $\mathcal{M}'$ accepts an input w then there must exist a w' such that $h''(w') = w$ and $w' \in h'(L)$ hold. Therefore, there exists a $w'' \in L$ such that $h'(w'') = w'$. It follows $h''(h'(w'')) = w$ and, hence, $w \in h(L)$.

b) Let $\mathcal{M} = (S, \delta_{nd}, \delta_d, \texttt{\#}, A, F)$ be a real-time 1G-OCA with global transformations Δ_{nd} and Δ_d, configurations c_t and $L = L(\mathcal{M})$. Define a language $L' \subseteq (A \times S)^*$ as follows:

$$\begin{aligned} L' := \Big\{ (a_1, s_1) \cdots (a_n, s_n) \mid\ & w = a_1 \cdots a_n \in L \\ & \wedge \exists\, c_1 \in \Delta_{nd}(c_{0,w}) : c_1(1) = s_1, \ldots, c_1(n) = s_n \\ & \wedge \big(\Delta_d^{n-1}(c_1)\big)(1) \in F \Big\} \end{aligned}$$

L' is a real-time OCA language: The cells of a corresponding OCA $\mathcal{M}'$ are verifying the condition $s_i \in \delta_{nd}(a_i, a_{i+1})$, $1 \leq i \leq n-1$, and $s_n \in \delta_{nd}(a_n, \texttt{\#})$

during their first time step. If the verification fails an error signal is sent to the left preventing the OCA from accepting. Otherwise cell i enters state s_i. For these cases the configurations of $\mathcal{M}$ (under input $a_1 \cdots a_n$) and $\mathcal{M}'$ (under input $(a_1, s_1) \cdots (a_n, s_n)$) at time 1 are identical.

During the next $n-1$ time steps $\mathcal{M}'$ simulates the deterministic behavior of $\mathcal{M}$ from time 2 to n. Therefore, $\mathcal{M}'$ accepts if $\mathcal{M}$ accepts the input $a_1 \cdots a_n$ by running through the configuration $s_1 \cdots s_n$ at time step 1. Thus, if $a_1 \cdots a_n \in L$. It follows $L' \in \mathscr{L}_{rt}(\text{OCA})$.

An ε-free homomorphism $h : (A \times S)^* \to A^*$, $h\big((a,s)\big) := a$, maps a pair from $A \times S$ to its first component. It follows for all $w = (a_1, s_1) \cdots (a_n, s_n) \in L'$: $h(w) = a_1 \cdots a_n \in L$ and, hence, $h(L') = L$. □

Now we apply the characterization and its proof.

Lemma 21 *Let L be a language belonging to $\mathscr{L}_{rt}(\text{1G-CA})$. Then there exist an ε-free homomorphism h and a language $L' \in \mathscr{L}_{rt}(\text{CA})$ such that $h(L') = L$ holds.*

Proof. The proof is only a slight modification of part b) of the proof of Theorem 20. It can easily be adapted. □

Lemma 22 *Let L be a language belonging to $\mathscr{L}_{rt}(\text{CA})$ and h be an ε-free homomorphism. Then $h(L)$ belongs to $\mathscr{L}_{rt}(\text{1G-OCA})$.*

Proof. Theorem 20 and Theorem 27 show that $\mathscr{L}_{rt}(\text{1G-OCA})$ is closed under ε-free homomorphism. The Lemma follows immediately from the fact $\mathscr{L}_{lt}(\text{CA}) \subseteq \mathscr{L}_{rt}(\text{1G-OCA})$ shown by Theorem 15. □

Theorem 23 $\mathscr{L}_{rt}(\text{1G-CA}) = \mathscr{L}_{rt}(\text{1G-OCA})$

Proof. $\mathscr{L}_{rt}(\text{1G-OCA}) \subseteq \mathscr{L}_{rt}(\text{1G-CA})$ follows for structural reasons.

For the converse suppose contrarily there exists a $L \in \mathscr{L}_{rt}(\text{1G-CA})$ that does not belong to $\mathscr{L}_{rt}(\text{1G-OCA})$. From Lemma 21 we obtain that there must be a language $L' \in \mathscr{L}_{rt}(\text{CA})$ and an ε-free homomorphism h such that $h(L') = L$. By Lemma 22 it follows $h(L') \in \mathscr{L}_{rt}(\text{1G-OCA})$, a contradiction. □

Figure 7 summarizes the relations between the language families.

7 Closure properties

In the following we are studying the closure properties of nondeterministic cellular automata. Figure 7 separates the language families into five groups where four of them are corresponding to well-known classes.

The first group is formed by the families $\mathscr{L}(\text{NCA}) = \mathscr{L}(\text{NOCA})$ that are identical to the context-sensitive languages (NSPACE(n)). From the group

$$
\begin{array}{cccccccc}
\mathscr{L}(\text{NCA}) & = & \multicolumn{2}{c}{\mathscr{L}(\text{NOCA})} & & = & & \mathscr{L}_1 \\
\cup\!| & & \rotatebox{45}{$\subsetneq$} & \rotatebox{135}{$\subsetneq$} & & & & \cup\!| \\
\mathscr{L}_{lt}(\text{NCA}) & = & \mathscr{L}_{lt}(\text{NOCA}) & \mathscr{L}(k\text{G-CA}) & = & \mathscr{L}(\text{CA}) & = & \mathscr{L}_{1,det} \\
\parallel & & \parallel & \cup\!| & & \cup\!| & & \cup \\
\mathscr{L}_{rt}(\text{NCA}) & = & \mathscr{L}_{rt}(\text{NOCA}) & \mathscr{L}(k\text{G-OCA}) & = & \mathscr{L}(\text{OCA}) & \supset & \mathscr{L}_2 \\
\cup\!| & & \rotatebox{135}{$\subsetneq$} & \rotatebox{45}{$\subsetneq$} & & \cup\!| & & \\
\mathscr{L}_{lt}(k\text{G-CA}) & = & \multicolumn{2}{c}{\mathscr{L}_{lt}(k\text{G-OCA})} & \supseteq & \mathscr{L}_{lt}(\text{CA}) & & \cup \\
\parallel & & \multicolumn{2}{c}{\parallel} & & \cup\!| & & \\
\mathscr{L}_{rt}(k\text{G-CA}) & = & \multicolumn{2}{c}{\mathscr{L}_{rt}(k\text{G-OCA})} & & \mathscr{L}_{rt}(\text{CA}) & \supset & \mathscr{L}_{2,det} \\
\parallel & & \multicolumn{2}{c}{\parallel} & & \parallel & & \\
\mathscr{L}_{lt}(1\text{G-CA}) & = & \multicolumn{2}{c}{\mathscr{L}_{lt}(1\text{G-OCA})} & & \mathscr{L}_{lt}^{R}(\text{OCA}) & & \cup \\
\parallel & & \multicolumn{2}{c}{\parallel} & & \cup & & \\
\mathscr{L}_{rt}(1\text{G-CA}) & = & \multicolumn{2}{c}{\mathscr{L}_{rt}(1\text{G-OCA})} & \supset & \mathscr{L}_{rt}(\text{OCA}) & \supset & \mathscr{L}_3
\end{array}
$$

Figure 7: Relations between language families.

$\mathscr{L}_{lt}(\text{NCA}) = \mathscr{L}_{lt}(\text{NOCA}) = \mathscr{L}_{rt}(\text{NCA}) = \mathscr{L}_{rt}(\text{NOCA})$ the family $\mathscr{L}_{rt}(\text{NOCA})$ has been well investigated in terms of homogeneous trellis automata [16]. The closure properties of the group $\mathscr{L}(k\text{G-CA}) = \mathscr{L}(\text{CA})$ are corresponding to the closure properties of the deterministic context-sensitive languages (DSPACE(n)), whereas the group $\mathscr{L}(k\text{G-OCA}) = \mathscr{L}(\text{OCA})$ has been studied in terms of iterative arrays [4, 14].

Now we are exploring the strong closure properties of the remaining group $\mathscr{L}_{lt}(k\text{G-CA}) = \mathscr{L}_{lt}(k\text{G-OCA}) = \mathscr{L}_{rt}(k\text{G-CA}) = \mathscr{L}_{rt}(k\text{G-OCA}) = \mathscr{L}_{lt}(1\text{G-CA}) = \mathscr{L}_{lt}(1\text{G-OCA}) = \mathscr{L}_{rt}(1\text{G-CA}) = \mathscr{L}_{rt}(1\text{G-OCA})$ on the hand of the simplest structured 1G-OCAs.

Lemma 24 $\mathscr{L}_{rt}(1\text{G-OCA})$ *is closed under union and intersection.*

Proof. Using the same two channel technique of [8] and [24] the assertion is easily seen. Each cell consists of two registers in which acceptors for both languages are simulated in parallel. □

Theorem 25 $\mathscr{L}_{rt}(1\text{G-OCA})$ *is closed under concatenation and iteration.*

Proof. Let $L_1, L_2 \in \mathscr{L}_{rt}(1\text{G-OCA})$ 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 3 we can construct a 1G-OCA that accepts L_1L_2 in time $n+1$ which according to lemma 7 can be sped-up to work in real-time. The closure under concatenation follows.

The closure under iteration follows analogously. During the first time step a corresponding acceptor guesses the cells which contain the first symbols of each subword. Subsequently acceptors for the basic language are simulated on each subword in parallel. A leftmoving signal that is started at the right border collects the results of the simulations. □

It is known that $\mathscr{L}_{rt}(\mathrm{OCA})$ is closed under reversal [5] which is a long-standing open problem for $\mathscr{L}_{rt}(\mathrm{CA})$.

Theorem 26 $\mathscr{L}_{rt}(\text{1G-OCA})$ *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 $\mathscr{L}_{rt}(\mathrm{OCA})$.

Let $\mathcal{M}$ be a 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. Since the input as well as its (guessed) reversal is stored on different tracks and 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. □

Now some closure properties concerning homomorphisms are shown.

Theorem 27 $\mathscr{L}_{rt}(\text{1G-OCA})$ *is closed under ε-free homomorphism.*

Proof. Suppose contrarily there is a language $L' \in \mathscr{L}_{rt}(\text{1G-OCA})$ and an ε-free homomorphism h' such that $L'' := h'(L') \notin \mathscr{L}_{rt}(\text{1G-OCA})$. From Theorem 20 follows that there exist a language $L \in \mathscr{L}_{rt}(\mathrm{OCA})$ 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 20 is contradicted. The closure under ε-free homomorphism follows. □

Corollary 28 $\mathscr{L}_{rt}(\text{1G-OCA})$ *is closed under injective length-multiplying homomorphism.*

We can relate an unsolved closure property to the power of deterministic two-way CAs. In [15] it has been shown that the family $\mathscr{L}_{rt}(\text{CA})$ is closed under reversal iff the linear-time and real-time CA languages are identical.

Lemma 29 *If the family* $\mathscr{L}_{rt}(\text{CA})$ *is closed under* ε*-free homomorphism then* $\mathscr{L}_{rt}(\text{CA}) = \mathscr{L}_{lt}(\text{CA})$ *holds.*

Proof. Assume $\mathscr{L}_{rt}(\text{CA})$ is closed under ε-free homomorphism. Due to the inclusion $\mathscr{L}_{rt}(\text{OCA}) \subset \mathscr{L}_{rt}(\text{CA})$ and the Theorems 20 and 27 one obtains $\mathscr{L}_{rt}(\text{1G-OCA}) \subseteq \mathscr{L}_{rt}(\text{CA})$. The lemma follows from $\mathscr{L}_{rt}(\text{CA}) \subseteq \mathscr{L}_{lt}(\text{CA}) \subseteq \mathscr{L}_{rt}(\text{1G-OCA})$. □

Theorem 30 $\mathscr{L}_{rt}(\text{1G-OCA})$ *is closed under inverse homomorphism.*

Proof. Let $L \in \mathscr{L}_{rt}(\text{1G-OCA})$ be a language over some alphabet A and $h : B^* \to A^*$ be a homomorphism. From Theorem 20 we obtain a real-time OCA language L' over some alphabet A' and a length preserving homomorphism $h' : A'^* \to A$ with $h'(L') = L$.

Let $h_2 : \{(x, x') \in B \times A' \mid h(x) = h'(x')\}^* \to A'^*$ be the homomorphism with $h_2\big((x, x')\big) = x'$. Further let $h_1 : (B \times A')^* \to B^*$ be an ε-free homomorphism with $h_1\big((x, x')\big) := x$. Then $h_1\big(h_2^{-1}(L')\big) = h^{-1}\big(h'(L')\big) = h^{-1}(L)$ holds.

Since $\mathscr{L}_{rt}(\text{OCA})$ is closed under inverse homomorphism [23], $h_2^{-1}(L')$ belongs to $\mathscr{L}_{rt}(\text{OCA})$. Now Theorem 20 implies $h^{-1}(L) \in \mathscr{L}_{rt}(\text{1G-OCA})$ what proves the closure under inverse homomorphism. □

Theorem 31 $\mathscr{L}_{rt}(\text{1G-OCA})$ *is not closed under arbitrary homomorphism.*

Proof. The Dyck languages and the regular languages are real-time OCA languages [8] and therefore real-time 1G-OCA languages. Chomsky [6] has shown that every context-free language is the homomorphic image of the intersection of a regular language and a Dyck language.

In contrast to the assertion we assume $\mathscr{L}_{rt}(\text{1G-OCA})$ is closed under homomorphism. Since it contains the regular as well as the Dyck languages it contains the context-free languages.

Ginsburg, Greibach and Harrison [12] have shown that every recursively enumerable language is the homomorphic image of the intersection of two context-free languages. Due to our assumption all recursively enumerable languages have to be contained in $\mathscr{L}_{rt}(\text{1G-OCA})$ from which a contradiction follows. □

Since homomorphisms are specific substitutions it follows:

Corollary 32 $\mathscr{L}_{rt}(\text{1G-OCA})$ *is not closed under arbitrary substitutions.*

The theory of *abstract families of languages* (AFL) has been founded in [10]. Since the closure properties of AFLs are not independent of each other we now derive two more properties from known results about AFLs.

Corollary 33 $\mathscr{L}_{rt}(\text{1G-OCA})$ *is an* AFL *(i.e. is closed under intersection with regular sets, inverse homomorphism, ε-free homomorphism, union, concatenation and iteration).*

Lemma 34 $\mathscr{L}_{rt}(\text{1G-OCA})$ *is closed under ε-free gsm mappings and inverse gsm mappings.*

Proof. In [10] it has been shown that every AFL is closed under ε-free gsm and inverse gsm mappings. □

Lemma 35 $\mathscr{L}_{rt}(\text{1G-OCA})$ *is closed under ε-free substitution.*

Proof. In [11] 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 24 and Corollary 33. □

We have shown the inclusion $\mathscr{L}_{rt}(\text{1G-OCA}) \subseteq \mathscr{L}(\text{CA})$ and that $\mathscr{L}_{rt}(\text{1G-OCA})$ is closed under reversal. Up to now it is not known whether the family is closed under complement or set difference. 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 Informatique Théorique et Applications. 18 (1984), 307–325.

[2] Buchholz, Th. and Kutrib, M. *On time computability of functions in one-way cellular automata.* Acta Informatica 35 (1998), 329–352.

[3] Buchholz, Th., Klein, A., and Kutrib, M. *One guess one-way cellular arrays.* Mathematical Foundations in Computer Science 1998, LNCS 1450, 1998, pp. 807–815.

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

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

[6] Chomsky, N. *Context-free grammars and pushdown storage.* Technical Report QPR 65, Massachusetts Institute of Technology, 1962.

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

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

[9] Fischer, P. C. *Generation of primes by a one-dimensional real-time iterative array*. Journal of the ACM 12 (1965), 388–394.

[10] Ginsburg, S. and Greibach, S. *Abstract families of languages*. In *Studies in Abstract Families of Languages 87*, Memoirs of the American Mathematical Society. American Mathematical Society, Providence, Rhode Island, 1969, pp. 1–32.

[11] Ginsburg, S. and Hopcroft, J. E. *Two-way balloon automata and AFL*. Journal of the ACM 17 (1970), 3–13.

[12] Ginsburg, S., Greibach, S., and Harrison, M. *One-way stack automata*. Journal of the ACM 14 (1967), 389–418.

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

[14] Ibarra, O. H. and Jiang, T. *On one-way cellular arrays*. SIAM Journal on Computing 16 (1987), 1135–1154.

[15] Ibarra, O. H. and Jiang, T. *Relating the power of cellular arrays to their closure properties*. Theoretical Computer Science 57 (1988), 225–238.

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

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

[18] Ibarra, O. H., Kim, S. M., and Moran, S. *Sequential machine characterizations of trellis and cellular automata and applications*. SIAM Journal on Computing 14 (1985), 426–447.

[19] Kasami, T. and Fuji, M. *Some results on capabilities of one-dimensional iterative logical networks*. Electronics and Communications in Japan 51-C (1968), 167–176.

[20] Kutrib, M. *Pushdown cellular automata*. Theoretical Computer Science 215 (1999), 239–261.

[21] 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.

[22] Myhill, J. *Finite automata and the representation of events.* Technical Report TR 57-624, WADC, 1957.

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

[24] Smith III, A. R. *Real-time language recognition by one-dimensional cellular automata.* Journal of Computer and System Sciences 6 (1972), 233–253.

[25] Terrier, V. *On real time one-way cellular array.* Theoretical Computer Science 141 (1995), 331–335.

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

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