Shrinking One-Way Cellular Automata

Martin Kutrib, Andreas Malcher, Matthias Wendlandt

Shrinking One-Way Cellular Automata

Martin Kutrib, Andreas Malcher, and Matthias Wendlandt

Institut für Informatik, Universität Giessen
Arndtstr. 2, 35392 Giessen, Germany
{kutrib,malcher,matthias.wendlandt}@informatik.uni-giessen.de

Abstract. We investigate cellular automata as acceptors for formal languages. In particular, we consider real-time one-way cellular automata (OCA) with the additional property that during a computation any cell of the OCA has the ability to dissolve itself and we call this model shrinking one-way cellular automata (SOCA). It turns out that real-time SOCA are more powerful than real-time OCA, since they can accept certain linear-time OCA languages. Additionally, a construction is provided that enables real-time SOCA to accept the reversal of real-time iterative array languages. Finally, restricted real-time SOCA are investigated which are allowed to dissolve only a number of cells that is bounded by some function. In this case, an infinite strict hierarchy of language classes is obtained.

1 Introduction

Devices of homogeneous, interconnected, parallel acting automata have widely been investigated from a computational capacity point of view. In particular, many results are known about *cellular automata* (see, for example, the surveys [10, 11]) which are linear arrays of identical copies of deterministic finite automata, where the single nodes, which are sometimes called cells, are homogeneously connected to their both immediate neighbors. Additionally, they work synchronously at discrete time steps. The computational power of cellular automata (CA) can be measured by their ability to accept formal languages. Initially, each word is written symbolwise into the cells. Then, the transition function is synchronously applied to each cell at discrete time steps, and the input is accepted if there is a time step at which the leftmost cell enters an accepting state. The in a way simplest model of CA is that of real-time one-way cellular automata (OCA) [4]. In this model, every cell is connected with its right neighbor only which restricts the flow of information from right to left. Additionally, the available time to accept an input is restricted to the length of the input. The class of languages accepted by real-time OCA is properly included both in the class of languages accepted by real-time CA and by linear-time OCA. Furthermore, it is incomparable with the class of context-free languages. More results on real-time OCA such as, for example, closure properties and decidability questions and references to the literature may be found in [10, 11].

The first notions of cellular automata date back to Ulam and von Neumann [14] whose work was biologically motivated by finding a way to design

a self-replicating machine. While the birth of new cells and the death of existing cells are natural processes in biology, these features are not reflected, at least to our knowledge and in connection with formal language recognition, in the existing literature. In this paper, we introduce *shrinking* one-way cellular automata (SOCA). Here, the transition function of each cell is extended so that each cell has the ability to dissolve itself. Dissolving of a cell means that the cell itself and all information stored in the cell is deleted. Moreover, the cell to the left of a dissolved cell is directly connected with the right neighbor of the dissolved cell. In this way, the number of cells available is shrinking. Nevertheless, real-time is still defined by the initial length of the input and the input is accepted whenever the leftmost cell at some computation step enters an accepting state. A related concept is that of *fault tolerant* CA (see, for example, [6, 12, 15, 18]). These are CA where cells may become defective, that is, they cannot process the information from its neighbors any longer, but only can forward the unchanged information to its neighbors. Hence, defective cells can no longer take part in the computation, but they are still existing.

The paper is organized as follows. In Section 2 we provide a formal definition of SOCA and an illustrating example. Moreover, real-time SOCA are compared to conventional real-time OCA. To this end, a technique is developed how to embed languages. It turns out that such embeddings of languages are still acceptable by conventional OCA as long as the unembedded language is accepted in time $n+r(n)$, where r is some sublinear function. If the unembedded language is accepted in time $2n$, but not in real time, then it is shown that the embedded language is accepted by a real-time SOCA, but not by any real-time OCA. In Section 3 we investigate an interesting relation of real-time SOCA to real-time iterative arrays (IA) which are similar to real-time CA, but process their input sequentially. It is known due to Cole [3] that the language classes induced by real-time IA and real-time OCA are incomparable. Here, we obtain that real-time IA are very close to real-time SOCA. It is shown that for every language L accepted by a real-time IA, a real-time SOCA can effectively be constructed which accepts all words of even or odd length from L, or the language $\$L^R$, that is, the reversal of L headed by a new symbol $\$$. In Section 4 we study the relation between the dissolving of cells and additional time. It turns out that real-time SOCA where the number of dissolved cells in accepting computations is bounded by some function r depending on the length of the input, can be simulated by conventional OCA which work in time $n + r(n)$. This enables us to utilize results known for the time hierarchy between real-time and linear-time OCA and to obtain an infinite hierarchy with regard to the number of dissolved cells.

2 Preliminaries and Definitions

We denote the set of non-negative integers by $\mathbb{N}$. Let A denote a finite set of letters. Then we write A^* for the set of all finite words (strings) built with letters from A. The empty word is denoted by λ, the reversal of a word w by w^R, and

for the length of w we write $|w|$. For the number of occurrences of a subword x in w we use the notation $|w|_x$. A subset of A^* is called a language over A. We use $\subseteq$ for set inclusion and $\subset$ for strict set inclusion. For a set S and a symbol a we abbreviatory write S_a for $S \cup \{a\}$. We use complexity functions $f : \mathbb{N} \to \mathbb{R}^+$ tacitly assuming that the range are integers by setting $f'(n) = \lceil f(n) \rceil$ and denote f' also by f. The i-fold composition of a function f is denoted by $f^{[i]}$, $i \geq 1$. Function f is said to be *increasing* if $m < n$ implies $f(m) \leq f(n)$. The *inverse* of an increasing and unbounded function $f : \mathbb{N} \to \mathbb{N}$ is defined as $f^{-1}(n) = \min\{\, m \in \mathbb{N} \mid f(m) \geq n \,\}$. The notation $f(O(n)) \leq O(f(n))$ means $\forall\, c \geq 1 : \exists\, n_0, c' \geq 1 : \forall\, n \geq n_0 : f(c \cdot n) \leq c' \cdot f(n)$. In order to avoid technical overloading in writing, two languages L and L' are considered to be equal, if they differ at most by the empty word, that is, $L \setminus \{\lambda\} = L' \setminus \{\lambda\}$. Throughout the article two devices are said to be *equivalent* if and only if they accept the same language.

2.1 One-Way (Shrinking) Cellular Automata

A one-way cellular automaton is a linear array of identical deterministic finite state machines, called cells. Except for the rightmost cell each one is connected to its nearest neighbor to the right. The state transition depends on the current state of a cell itself and the current state of its neighbor, where the rightmost cell receives information associated with a boundary symbol on its free input line. The state changes take place simultaneously at discrete time steps. A one-way cellular automaton is said to be shrinking if the cells may dissolve themselves. If a cell dissolves itself the next cell to its left is connected to the next cell to its right. The input mode for cellular automata is called parallel. One can suppose that all cells fetch their input symbol during a pre-initial step.

Definition 1. *A* shrinking one-way cellular automaton (*SOCA*) *is a system* $\langle S, F, A, \#, \delta \rangle$*, where* S *is the finite, nonempty set of* cell states*,* $F \subseteq S$ *is the set of* accepting states*,* $A \subseteq S$ *is the nonempty set of* input symbols*,* $\# \notin S$ *is the permanent* boundary symbol*, and* $\delta : S \times S_\# \to S \cup \{\mathsf{dissolve}\}$ *is the* local transition function*.*

A configuration c_t *of* M *at time* $t \geq 0$ *is a string of the form* $S^*\#$*, that reflects the cell states from left to right. The computation starts at time 0 in a so-called* initial configuration*, which is defined by the input* $w = a_1 a_2 \cdots a_n \in A^+$*. We set* $c_0 = a_1 a_2 \cdots a_n \#$*. Successor configurations are computed according to the global transition function* Δ*. Let* $c_t = x_1 x_2 \cdots x_n \#$*,* $t \geq 0$*, be a configuration, then its successor* $c_{t+1} = y_1 y_2 \cdots y_n \#$ *is defined as follows:*

$$c_{t+1} = \Delta(c_t) \iff \begin{cases} y_i = h(\delta(x_i, x_{i+1})), i \in \{1, 2, \ldots, n-1\} \\ y_n = h(\delta(x_n, \#)) \end{cases}$$

where $h : S \cup \{\mathsf{dissolve}\} \to S \cup \{\lambda\}$ *is the homomorphism that maps states to states and erases* dissolve*, that is,* $h(p) = p$ *for* $p \in S$*, and* $h(\mathsf{dissolve}) = \lambda$*. Thus,* Δ *is induced by* δ*.*

Fig. 1. A (shrinking) one-way cellular automaton.

An SOCA is *non-shrinking* if $\delta(s_1, s_2) \neq \mathsf{dissolve}$ for all $s_1, s_2 \in S_{\#}$. Non-shrinking SOCA are referred to as one-way cellular automata. They are denoted by OCA.

An input w is accepted by an SOCA M if at some time step during the course of its computation the leftmost cell enters an accepting state, that is, the leftmost symbol of the configuration is an accepting state. The *language accepted by* M is denoted by $L(M)$. Let $t : \mathbb{N} \to \mathbb{N}$ be a mapping. If all $w \in L(M)$ are accepted with at most $t(|w|)$ time steps, then M is said to be of time complexity t.

In general, the family of languages accepted by some device X with time complexity t is denoted by $\mathscr{L}_t(X)$. The index is omitted for arbitrary time. Actually, arbitrary time is exponential time due to the space bound. If t is the identity function n, acceptance is said to be in *real time* and we write $\mathscr{L}_{rt}(X)$. The *linear-time* languages $\mathscr{L}_{lt}(X)$ are defined according to $\mathscr{L}_{lt}(X) = \bigcup_{k\in\mathbb{Q},\, k\geq 1} \mathscr{L}_{k\cdot n}(X)$.

Example 2. The language $L = \{\, \$w \mid w \in \{a,b\}^* \text{ and } |w|_a = |w|_b \,\}$ is accepted by a real-time SOCA.

The basic idea to accept language L is that every cell x with $x \in \{a,b\}$ having the border cell as right neighbor sends a signal to the left that eventually dissolves a cell y with $y \in \{a,b\}$ and $y \neq x$. Subsequently, this cell dissolves itself in the following step. Meanwhile the signal is forwarded to the left by cells carrying the same input symbol x. Thus, the transition function δ of an SOCA accepting L may be sketched as follows with $x \in \{a,b\}$. We set $\delta(x, \#) = \delta(x', \#) = \overline{x}$, $\delta(\overline{x}, \#) = \mathsf{dissolve}$, and $\delta(x, \overline{x}) = \delta(x, x') = x'$. If an x-signal reaches a cell carrying an input symbol $y \in \{a,b\}$ with $y \neq x$, the cell dissolves itself as well. We set $\delta(y, \overline{x}) = \delta(y, x') = \mathsf{dissolve}$. If the input belongs to L, then all cells carrying a's and b's have been dissolved up to the next to last computation step. Thus, the only $ cell has the border symbol as right neighbor and can enter an accepting state *acc*. If the only $ cell sees a state x' or $\overline{x}$ to its right, it enters a rejecting state *rej*. If the input contains no symbol $ at all, then the automaton can never enter an accepting state. If the input contains more than one $, there is a cell having a $-cell as right neighbor. This cell sends a signal to the left that prevents all cells passed through from dissolving and accepting.

To show that the SOCA works in real time, notice that in every second computation step the cell next to the border cell is dissolved. Furthermore, such a cell initiates a signal to the left that will dissolve another cell somewhere to the left. So, at some time step $2i$ there are i cells dissolved next to the border cell and i signals for dissolving a cell are sent. In particular, for an input word $\$w$ with $w \in L$, we have $|w|$ is even and at time $2\frac{|w|}{2}$ exactly $\frac{|w|}{2}$ cells have been dissolved next to the border cell and exactly $\frac{|w|}{2}$ cells have been dissolved somewhere to the left. So, only the $-cell at the left border remains which accepts in the next,

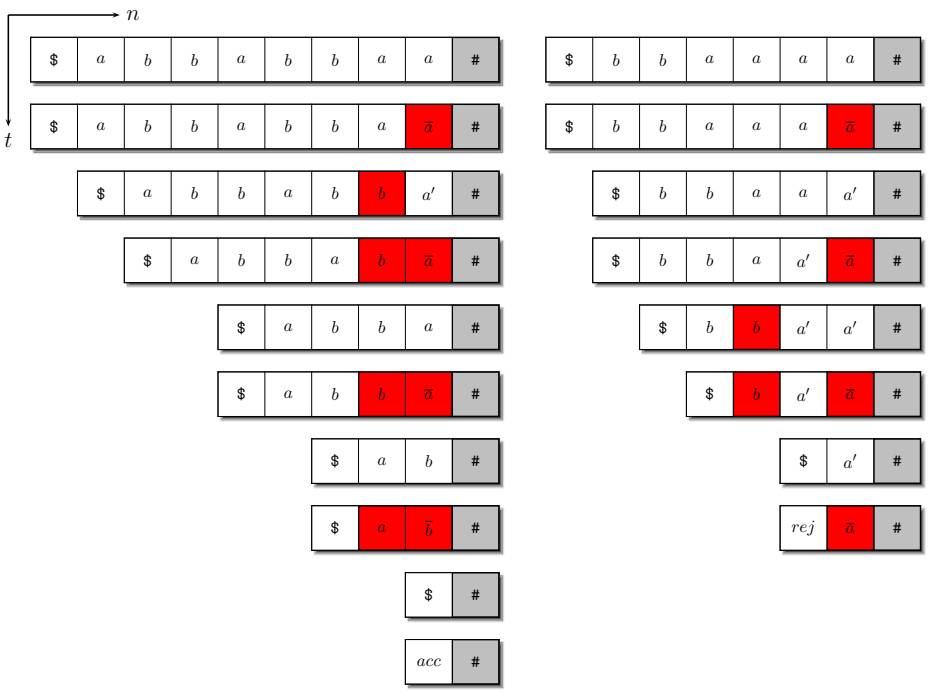

Fig. 2. The computation on the left accepts the input $\$abbabbaa$ and the computation on the right rejects the input $\$bbaaaa$. Cells to be dissolved in the next time step are depicted with background.

that is, $(|w|+1)$st step, hence, in real time. Two example computations can be found in Figure 2. □

It is worth mentioning that it is still an open problem whether or not the deterministic context-free language L of Example 2 can be accepted by a real-time OCA.

2.2 Does Shrinking Really Help?

It is well known that the family of real-time OCA languages is a proper subfamily of the linear-time OCA languages [11]. This section is devoted to comparing the power of real-time shrinking one-way cellular automata to the power of classical real-time OCA. It turns out that the former are more powerful.

Let $L \subseteq A^*$ be a language and $\$ \notin A$ be a letter. The embedding of a $\$$ after every symbol from A is given by the homomorphism $\text{emb} : A^* \to A^*_{\$}$, where $\text{emb}(a) = a\$$ for $a \in A$. Next we show that this embedding does not help OCA. We consider time complexities from real time to linear time of the form $n + r(n)$, where $r : \mathbb{N} \to \mathbb{N}$ is a linear or sublinear function. In order to avoid functions with strange behavior, we require that r meets the weak properties increasing

and $r(O(n)) \leq O(r(n))$. Recall that the latter means $\forall\, c \geq 1 : \exists\, n_0, c' \geq 1 : \forall\, n \geq n_0 : r(c \cdot n) \leq c' \cdot r(n)$. It can easily be verified that many of the commonly considered linear and sublinear time complexity functions have this property.

Lemma 3. *Let $r : \mathbb{N} \to \mathbb{N}$ be an increasing function so that $r(O(n)) \leq O(r(n))$. A language L belongs to the family $\mathscr{L}_{n+r(n)}(OCA)$ if and only if* emb(L) *belongs to $\mathscr{L}_{n+r(n)}(OCA)$.*

Proof. Let $M = \langle S, F, A_{\$}, \#, \delta \rangle$ be an OCA accepting emb$(L) \subseteq A_{\$}^*$ in $n + r(n)$ time. We construct an OCA $M' = \langle S', F', A, \#, \delta' \rangle$ accepting $L \subseteq A^*$ as follows. The simple basic idea is to let each cell of M' simulate two adjacent cells of M, that is, a cell receiving an input from A together with its neighboring cell receiving a \$. In each step of M' two steps of M are simulated, which is possible since the simulated input of M appears compressed in M'. A formal construction is:

$$
\begin{aligned}
S' = A \cup S^2, \qquad & F' = \{\, (q_1, q_2) \in S' \mid q_1 \in F \,\}, \\
\delta'(p_1, \#) &= (\delta(\delta(p_1, \$), \delta(\$, \#)), \delta(\delta(\$, \#), \#)) \\
&\quad \text{for all } p_1 \in A, \\
\delta'(p_1, q_1) &= (\delta(\delta(p_1, \$), \delta(\$, q_1)), \delta(\delta(\$, q_1), \delta(q_1, \$))) \\
&\quad \text{for all } p_1, q_1 \in A, \\
\delta'((p_1, p_2), (q_1, q_2)) &= (\delta(\delta(p_1, p_2), \delta(p_2, q_1)), \delta(\delta(p_2, q_1), \delta(q_1, q_2))) \\
&\quad \text{for all } (p_1, p_2), (q_1, q_2) \in S'^2,
\end{aligned}
$$

in all other cases δ' does not change the current state of a cell. Let w be some word in L. So, M may use $2|w| + r(2|w|)$ time steps to accept emb(w). In order to complete the simulation on input w, the OCA M' takes at most $|w| + \frac{1}{2} r(2|w|)$ steps. Since $r(O(n)) \leq O(r(n))$, there is a constant $c' \geq 1$ so that $c' \cdot r(|w|) \geq r(2|w|)$. Therefore, M' takes at most $|w| + \frac{c'}{2} \cdot r(|w|)$ steps. Strong speed-up theorems for several devices are obtained in [7, 8]. In particular, it is possible to speed up the time beyond real time linearly. Here we choose a speed-up constant $\frac{2}{c'}$ and apply this technique to M'. The resulting OCA obeys the time complexity $n + r(n)$ and accepts L.

For the converse simulation, now let M be an $n + r(n)$-time OCA accepting $L \subseteq A^*$. First, M is sped up to $n + \frac{1}{2} r(n)$ time. The further construction idea for an OCA M' that accepts emb(L) in $n + r(n)$ time is as follows.

At initial time a signal is sent from the right border to the left. It checks the correct format of the input. If the format is incorrect the left border cell is prevented from accepting. So we safely may assume the format to be correct. All cells remember whether their input symbol is a \$ or a symbol from A (the leftmost cell must be an A-cell). Now, in each other time step an A-cell does nothing, while a \$-cell copies the current state of its right neighbor as part of its own state. In the following step, a \$-cells does nothing, while an A-cell simulates one transition of M.

Let w be some word in L. So, M may use $|w| + \frac{1}{2} r(|w|)$ time steps to accept w. In order to complete the simulation on input emb(w), the OCA M' takes at most

$2|w| + r(|w|)$ steps. Since $|\mathrm{emb}(w)| = 2|w|$ and r is increasing, this is less than $2|w| + r(2|w|)$ and M' obeys the time complexity $n + r(n)$. □

On the other hand, the embedding together with the possibility of cells to dissolve themselves strengthens the power of real-time OCA significantly. As mentioned before, the proper inclusion $\mathscr{L}_{rt}(\mathrm{OCA}) \subset \mathscr{L}_{lt}(\mathrm{OCA})$ is known.

Lemma 4. *Let L be a language from $\mathscr{L}_{lt}(OCA) \setminus \mathscr{L}_{rt}(OCA)$. Then* $\mathrm{emb}(L)$ *belongs to $\mathscr{L}_{rt}(SOCA)$.*

Proof. As in the proof of Lemma 3, we utilize the speed-up techniques obtained in [7, 8]. So, we safely may assume that for any linear-time OCA language there exists a $(2n-2)$-time OCA M that accepts it.

Now let $L \in \mathscr{L}_{lt}(\mathrm{OCA}) \setminus \mathscr{L}_{rt}(\mathrm{OCA})$. A real-time SOCA M' accepting $\mathrm{emb}(L)$ works as follows. During the first transition every cell checks whether its right neighbor carries a correct input symbol. That is, a cell with input symbol belonging to the alphabet of L must have a `$` neighbor and vice versa (except for the rightmost cell). If the input format is incorrect a failure signal is sent to the left that prevents all cells passed through from dissolving and accepting. During the second transition all cells with `$` input dissolve themselves, while the others remain in their states. Finally, the OCA M is simulated on the remaining cells. On input of length n every other cell is dissolved during the first two time steps. Then the simulation of M takes $2\frac{n}{2} - 2$ time steps. Altogether the SOCA M' works in n time steps, that is, in real time. □

Lemma 3 and Lemma 4 have shown the next theorem.

Theorem 5. *Let L be a language from $\mathscr{L}_{lt}(OCA) \setminus \mathscr{L}_{rt}(OCA)$. Then* $\mathrm{emb}(L)$ *belongs to $\mathscr{L}_{rt}(SOCA)$ but does not belong to $\mathscr{L}_{rt}(OCA)$. In particular, the family $\mathscr{L}_{rt}(OCA)$ is properly included in $\mathscr{L}_{rt}(SOCA)$.*

3 Real-Time Shrinking OCA and Iterative Arrays

Iterative arrays (IA) are another simple model for massively parallel computations, which sometimes are called cellular automata with sequential input. In connection with formal language processing iterative arrays have been introduced in [3]. What makes these devices interesting in the current context is the incomparability of the families of languages accepted by OCA and IA in real time. Moreover, the latter devices accept non-semilinear unary languages while the former devices accept only regular unary languages. Conversely, for example, all linear context-free languages belong to $\mathscr{L}_{rt}(\mathrm{OCA})$ but there is a deterministic linear context-free language not accepted by any real-time iterative array. In the following, we investigate to what extent real-time shrinking OCA can simulate real-time IA.

For the formal constructions we need to introduce IA in more detail. Basically, they are semi-infinite linear arrays of finite automata, again called cells. But now the information flow is two-way, that is, each cell except the leftmost one

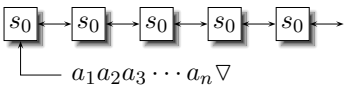

Fig. 3. An iterative array.

is connected to its both nearest neighbors (one to the left and one to the right). The input is supplied sequentially to the distinguished communication cell at the origin which is connected to its immediate neighbor to the right only. For this reason, we have two different local transition functions. The state transition of all cells but the communication cell depends on the current state of the cell itself and the current states of its both neighbors. The state transition of the communication cell additionally depends on the current input symbol (or if the whole input has been consumed on a special end-of-input symbol). The finite automata work synchronously at discrete time steps. Initially they are in the so-called quiescent state.

Definition 6. *An* iterative array (*IA*) *is a system* $\langle S, F, A, \triangledown, s_0, \delta, \delta_0 \rangle$*, where* S *is the finite, nonempty set of* cell states*,* $F \subseteq S$ *is the set of* accepting states*,* A *is the finite, nonempty set of* input symbols*,* $\triangledown \notin A$ *is the* end-of-input symbol*,* $s_0 \in S$ *is the* quiescent state*,* $\delta : S^3 \to S$ *is the* local transition function for non-communication cells *satisfying* $\delta(s_0, s_0, s_0) = s_0$*, and* $\delta_0 : A_\triangledown \times S^2 \to S$ *is the* local transition function for the communication cell.

The notions of configurations are a straightforward adaption from OCA. An input is accepted if at some time step during the course of its computation the communication cell enters an accepting state. Acceptance is said to be in real time if all w in the accepted language are accepted within at most $|w| + 1$ time steps.

The next result reveals an interesting relation between real-time IA and SOCA. The proof shows a basic construction that will later be modified for further relations.

Theorem 7. *Let* L *belong to* $\mathscr{L}_{rt}(IA)$*. Then* $\{ w \mid w^R \in L$ *and* $|w|$ *is even* $\}$ *is accepted by a real-time SOCA.*

Proof. Given a language L accepted by some IA M in real time, the claimed real-time SOCA M' is constructed in three steps.

The first step is to embed the computation of M into a one-way device M'. To overcome the problem to simulate two-way information flow by one-way information flow, the cells of M' collect the information necessary to simulate one transition of M in an intermediate step. So, the first step of M is simulated in the second step of M' and so on. Moreover, the configuration flows with speed $1/2$ to the left. The behavior is depicted in Figure 4, where for this step it is assumed that the input is available where it is consumed.

The second step is to speed up the computation of M'. Assume that in every computation step two input symbols are fed to M'. Since all cells of an IA are

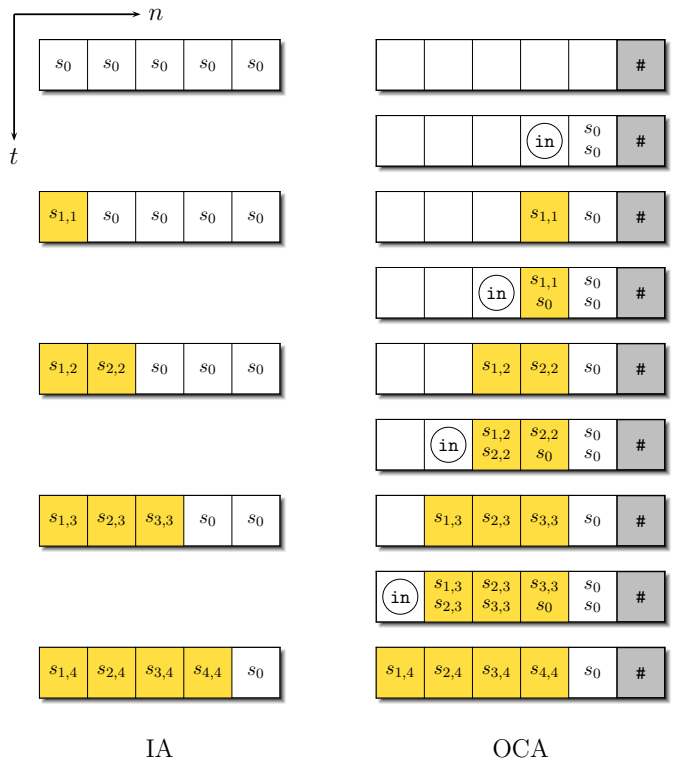

Fig. 4. Simulation of two-way information flow by one-way information flow. The input to the OCA is provided at the cells marked by the circled `in`.

initially in the same, that is, quiescent state, the computation can be set up such that each cell simulates two cells of M. Hence, this together with the compressed input allows some OCA M'' to simulate two steps of M' at once at every time step.

The third step is to construct the desired real-time SOCA M'''. Based on M'' we first explain how cells dissolve themselves. This happens at every other time step beginning at time step two. More precisely, all cells maintain an internal counter modulo two. So, they can detect even time steps. At these time steps, precisely the cell which carries two input symbols and whose right neighbor simulates a state of M dissolves itself. Since the simulated configuration flows to the left with speed $1/2$ and at every other time step one cell dissolves itself, the cell simulating the communication cell of M arrives at the leftmost cell at real time as desired (see Figure 5). It remains to be shown how the input is provided as requested. To this end, it is sufficient that during the first transition each cell copies the state of its right neighbor, that is its input, as part of its own state.

The SOCA M''' simulates the given IA M on even length inputs. However, the last step of M depends on the end-of-input symbol. So far, M''' does not

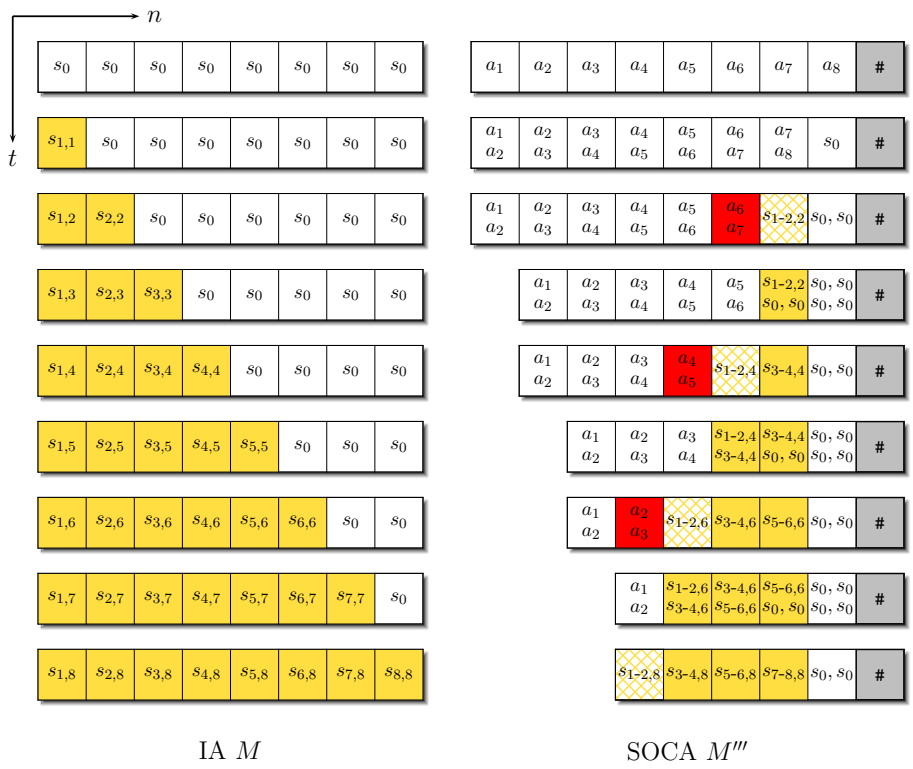

Fig. 5. Simulation of a real-time IA by a real-time SOCA on even length input. The internal modulo two counters are not depicted. The red cells dissolve themselves. The state of cell i at time k is denoted by $s_{i,k}$. We write $s_{i\text{-}j,k}$ for $s_{i,k}, s_{i+1,k}, \ldots, s_{j,k}$. The crosshatched cells are those who simulate a further transition under the assumption that they receive the last input symbol.

simulate this last step of M. Therefore, the construction has slightly to be extended. Whenever, a cell of M''' storing input symbols changes to a state that simulates states of M, it simulates one more transition of M under the assumption that the cell has received the last input symbols and, thus, is the leftmost one. The result is stored in an extra register. Finally, a state is accepting if this extra register contains an accepting state of M. The transition that fills the extra register of the real leftmost cell corresponds to the missing last transition of M. □

The proof of Theorem 7 can be modified to accept inputs of odd length instead of even length in a straightforward way. Even more complicated cycles of move and dissolve operations can be realized.

Theorem 8. *Let L belong to $\mathscr{L}_{rt}(IA)$. Then $\{ w \mid w^R \in L$ and $|w|$ is odd$\}$ is accepted by a real-time SOCA.*

Example 9. Basically, a function $f : \mathbb{N} \to \mathbb{N}$ is said to be IA-constructible if there is an IA which indicates by states of the communication cell the time steps $f(n)$, for $n \geq 1$. For details and further results on IA-constructibility we refer to [1, 5, 13]. However, the family of such functions is rich. For example, the functions $f(n) = 2^n$ and $f(n) = p_n$, where p_n is the nth prime number, are IA-constructible. Therefore, the non-semilinear unary languages $\{ a^{2^n} \mid n \geq 0 \}$ and $\{ a^{p_n} \mid n \geq 2 \}$ belong to the family $\mathscr{L}_{rt}(\mathrm{IA})$. By Theorems 7 and 8 they are accepted by an SOCA in real time as well. On the other hand, real-time OCA accept only regular, that is, semilinear unary languages [17]. □

A further relation between real-time IA and SOCA has been foreshadowed by Example 2. The words of the language of the example have the property that the leftmost cell can identify itself. Clearly this is realized by concatenating a fixed new symbol to the left. In general, we have the following result.

Theorem 10. *Let $L \subseteq A^*$ be a language from $\mathscr{L}_{rt}(IA)$ and $\$ \notin A$ be a letter. Then $\{ \$w \mid w^R \in L \}$ is accepted by a real-time SOCA.*

Example 11. It is shown in [16] that the real-time deterministic context-free language $L = \{ c^m a^{k_0} b a^{k_1} b \cdots b a^{k_m} b \cdots b a^{k_\ell} b d^n \mid m, n, k_i \geq 0, \ell \geq 1, k_m = n \}$ does not belong to $\mathscr{L}_{rt}(\mathrm{OCA})$. Since the family $\mathscr{L}_{rt}(\mathrm{OCA})$ is closed under reversal and left quotient with a fixed new symbol, the language $\$L^R$ does not belong to $\mathscr{L}_{rt}(\mathrm{OCA})$, neither. On the other hand, all real-time deterministic context-free languages belong to $\mathscr{L}_{rt}(\mathrm{IA})$ [11]. Now Theorem 10 shows that $\$L^R$ is accepted by some SOCA in real time. □

4 Dissolving versus Time

In this section, we investigate to what extent the possibility of an SOCA to dissolve cells can be captured by providing additional time. We will obtain that this can always be achieved. This result enables us together with a suitable embedding of symbols to translate the time hierarchy known to exist in between real-time and linear-time conventional OCA ([9]) into a hierarchy for SOCA with regard to the number of cells dissolved. We start by defining classes of SOCA where the maximum number of dissolved cells in accepting computations is put in relation to the length of the input processed.

Let M be an SOCA and $f : \mathbb{N} \to \mathbb{N}$ be an increasing function. If all $w \in L(M)$ are accepted with computations where the number of dissolved cells is bounded by $f(|w|)$, then M is said to be *dissolving bounded* by f. The class of SOCA that are dissolving bounded by f is denoted by f-SOCA.

The next result says that every dissolving bounded SOCA can be simulated by a conventional OCA with additional time steps depending only on the number of dissolved cells.

Theorem 12. *Let M be an f-SOCA working in real time. Then an equivalent conventional OCA M' working in time $n + f(n)$ can effectively be constructed.*

Next, we utilize results on a time hierarchy in between real-time and linear-time known for OCA to obtain a hierarchy for SOCA according to the number of dissolved cells. The hierarchy result uses the notion of OCA-constructible functions. A function $f : \mathbb{N} \to \mathbb{N}$ with $f(n) \geq n$, is *OCA-constructible* if there exists a length-preserving homomorphism h and a language $L \in \mathscr{L}_{rt}(\text{OCA})$ such that $h(L) = \{\, a^{f(n)-n}b^n \mid n \geq 1 \,\}$. More details on OCA-constructibility may be found in [2]. Further, the language $L_d \subset \{0, 1, (,), |\}^+$ is used whose words are of the form $x\,(\,x_1 \,|\, y_1\,) \cdots (\,x_n \,|\, y_n\,)\,y$, where $x, x_i, y, y_i \in \{0,1\}^*$ for $1 \leq i \leq n$, and $(\,x \,|\, y\,) = (\,x_i \,|\, y_i\,)$ for at least one $i \in \{1, \ldots, n\}$.

Now, the witness languages for the time hierarchy defined in [9] are as follows. Let $f : \mathbb{N} \to \mathbb{N}$ be an increasing OCA-constructible function and language $\hat{L}_f \subseteq B^+$ be a witness for the constructibility, that is, $\hat{L}_f \in \mathscr{L}_{rt}(\text{OCA})$ such that $h(\hat{L}_f) = \{\, a^{f(n)-n}b^n \mid n \geq 1 \,\}$ for a length-preserving homomorphism h. From $\hat{L}_f$ and L_d the language $L_f \subseteq ((A \cup \{\sqcup\}) \times B)^+$ is constructed as follows:

1. The second component of each word w in L_f is a word of $\hat{L}_f$ that implies that w is of length $f(m)$ for some $m \geq 1$.
2. The first component of w contains exactly $f(m) - m$ blank symbols and m non-blank symbols.
3. The non-blank symbols in the first component of w form a word in L_d.

Theorem 13. *Let $r_1, r_2 : \mathbb{N} \to \mathbb{N}$ be two increasing functions. If r_1^{-1} is OCA-constructible, $r_2(O(n)) \leq O(r_2(n))$, and $r_2 \cdot \log(r_2) \in o(r_1)$, then*

$$\mathscr{L}_{rt}(r_2\text{-}SOCA) \subset \mathscr{L}_{rt}(r_1\text{-}SOCA).$$

Proof. In [9] the proper inclusion $\mathscr{L}_{n+r_2(n)}(\text{OCA}) \subset \mathscr{L}_{n+r_1(n)}(\text{OCA})$ has been shown with $L_{r_1^{-1}}$ as witness language. Let A' be the alphabet of $L_{r_1^{-1}}$ and M_1 be an $(n + r_1(n))$-time OCA accepting $L_{r_1^{-1}}$. By applying speed-up techniques we obtain an equivalent OCA M_1' working in time $n + r_1(n) - 2$.

Here, we will consider language $L_{\$,r_1^{-1}}$ instead, where a word w belongs to $L_{\$,r_1^{-1}}$ if and only if it is of the form $A'^*(A'\$)^*$, $|w|_\$ = r_1(|w|)$, and $h(w) \in L_{r_1^{-1}}$, for the homomorphism $h(a) = a$, for $a \in A$, and $h(\$) = \lambda$. That is, we take the words from $L_{r_1^{-1}}$ and insert exactly one $\$$ after each of the rightmost $r_1(|w|)$ symbols.

A real-time r_1-SOCA M accepting $L_{\$,r_1^{-1}}$ is constructed as follows. During the first transition, every cell with input from A' applies the homomorphism that witnesses the time constructibility of r_1 to the content of its second register and checks whether its right neighbor carries a correct input symbol. That is, if the homomorphic image is a b the cell must have a $\$$ neighbor, if it is an a the cell must not have a $\$$ neighbor. In addition, every cell with input $\$$ must have a neighbor whose homomorphic image is b or the border symbol. If the check fails, a failure signal is sent to the left that prevents all cells passed through from dissolving and accepting. During the second transition all cells with $\$$ input dissolve themselves, while the others remain in their states. Finally, the $(n+r_1(n)-2)$-time OCA M_1' accepting $L_{r_1^{-1}}$ is simulated on the remaining cells.

So, the SOCA M accepts an input w if and only if it is of the form $A'^*(A'\$)^*$, the input without $\$$ symbols belongs to $L_{r_1^{-1}}$, that is $h(w) \in L_{r_1^{-1}}$, and the number of $\$$, that is the number of b, is m for input length $|w| = r_1^{-1}(m)$. The latter means $|w|_{\$} = r_1(|w|)$. We conclude that M is an r_1-SOCA that accepts $L_{\$,r_1^{-1}}$.

In order to derive the time complexity of M, we recall that $r_1(n)$ cells are dissolved during the second time step on inputs of length n. Then, the simulation of M_1' on inputs of length $n - r_1(n)$ takes $n - r_1(n) + r_1(n - r_1(n)) - 2 \leq n - 2$ time steps, since r_1 is increasing. Altogether the r_1-SOCA M works in n time steps, that is, in real time.

To show a proper inclusion between language classes $\mathscr{L}_{rt}(r_2\text{-SOCA})$ and $\mathscr{L}_{rt}(r_1\text{-SOCA})$, we first notice that the first language class is included in the latter since $r_2 \cdot \log(r_2) \in o(r_1)$ holds. Now, we assume by way of contradiction that the language classes $\mathscr{L}_{rt}(r_2\text{-SOCA})$ and $\mathscr{L}_{rt}(r_1\text{-SOCA})$ are identical. Then we obtain that $L_{\$,r_1^{-1}}$ is accepted by some real-time r_2-SOCA as well. An application of Theorem 12 gives that then $L_{\$,r_1^{-1}}$ also belongs to $\mathscr{L}_{n+r_2(n)}(\text{OCA})$. So, it remains to be shown that the latter is a contradiction.

The contradiction is derived along the lines of the proof of Lemma 3. Given an $(n + r_2(n))$-time OCA N accepting $L_{\$,r_1^{-1}}$, an OCA N' accepting $L_{r_1^{-1}}$ is constructed, such that each cell receiving an input symbol whose homomorphic image is a b simulates two adjacent cells, that is, the cell itself together with its neighboring cell receiving a $\$$. In each such step two original steps are simulated. The rightmost a-cell starts to simulate two steps at time step 1, the next a-cell at time step 2 and so on. In this way, the leftmost cell starts to simulate two steps from time step $|w| - r_1(|w|)$ on.

Let w be some word in $L_{r_1^{-1}}$. So, N may use $|w| + r_1(|w|) + r_2(|w| + r_1(|w|))$ time steps to accept the associated padded word from $L_{\$,r_1^{-1}}$. In order to complete the simulation on input w, the OCA N' takes at most

$$|w| - r_1(|w|) + \frac{1}{2}(2r_1(|w|) + r_2(|w| + r_1(|w|))) \\ = |w| + \frac{1}{2}r_2(|w| + r_1(|w|)) \leq |w| + \frac{1}{2}r_2(2|w|)$$

time steps. Since $r_2(O(n)) \leq O(r_2(n))$, there is a constant $c' \geq 1$ so that $c' \cdot r_2(|w|) \geq r_2(2|w|)$. Therefore, N' takes at most $|w| + \frac{c'}{2} \cdot r_2(|w|)$ steps. Now, the time beyond real time is sped-up linearly by a constant $\frac{2}{c'}$. The resulting OCA obeys the time complexity $n + r_2(n)$ and accepts $L_{r_1^{-1}}$, which is a contradiction. □

Example 14. Let $0 \leq p < q \leq 1$ be two rational numbers. Then there is the following proper inclusion

$$\mathscr{L}_{rt}(n^p\text{-SOCA}) \subset \mathscr{L}_{rt}(n^q\text{-SOCA}).$$

Another example is given by the i-fold iterated logarithms $\log^{[i]}$. Let $i < j$ be two positive integers. Then there is the following proper inclusion

$$\mathscr{L}_{rt}(\log^{[j]}\text{-SOCA}) \subset \mathscr{L}_{rt}(\log^{[i]}\text{-SOCA}).$$

□

References

1. Buchholz, T., Kutrib, M.: Some relations between massively parallel arrays. Parallel Comput. 23, 1643–1662 (1997)
2. Buchholz, T., Kutrib, M.: On time computability of functions in one-way cellular automata. Acta Inform. 35, 329–352 (1998)
3. Cole, S.N.: Real-time computation by n-dimensional iterative arrays of finite-state machines. IEEE Trans. Comput. C-18, 349–365 (1969)
4. Dyer, C.R.: One-way bounded cellular automata. Inform. Control 44, 261–281 (1980)
5. Fischer, P.C.: Generation of primes by a one-dimensional real-time iterative array. J. ACM 12, 388–394 (1965)
6. Harao, M., Noguchi, S.: Fault tolerant cellular automata. J. Comput. System Sci. 11, 171–185 (1975)
7. Ibarra, O.H., Kim, S.M., Moran, S.: Sequential machine characterizations of trellis and cellular automata and applications. SIAM J. Comput. 14, 426–447 (1985)
8. Ibarra, O.H., Palis, M.A.: Some results concerning linear iterative (systolic) arrays. J. Parallel Distributed Comput. 2, 182–218 (1985)
9. Klein, A., Kutrib, M.: Fast one-way cellular automata. Theoret. Comput. Sci. 295, 233–250 (2003)
10. Kutrib, M.: Cellular automata – a computational point of view. In: New Developments in Formal Languages and Applications, chap. 6, pp. 183–227. Springer (2008)
11. Kutrib, M.: Cellular automata and language theory. In: Encyclopedia of Complexity and System Science, pp. 800–823. Springer (2009)
12. Kutrib, M., Löwe, J.T.: Space- and time-bounded nondeterminism for cellular automata. Fund. Inform. 58, 273–293 (2003)
13. Mazoyer, J., Terrier, V.: Signals in one-dimensional cellular automata. Theoret. Comput. Sci. 217, 53–80 (1999)
14. von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press (1966), edited and completed by Arthur W. Burks
15. Nishio, H., Kobuchi, Y.: Fault tolerant cellular spaces. J. Comput. System Sci. 11, 150–170 (1975)
16. Okhotin, A.: Comparing linear conjunctive languages to subfamilies of the context-free languages. In: Theory and Practice of Computer Science (SOFSEM 2011). LNCS, vol. 6543, pp. 431–443. Springer (2011)
17. Seidel, S.R.: Language recognition and the synchronization of cellular automata. Tech. Rep. 79-02, Department of Computer Science, University of Iowa (1979)
18. Umeo, H.: A simple design of time-efficient firing squad synchronization algorithms with fault-tolerance. IEICE Trans. Inf. Syst. E87-D, 733–739 (2004)