I F I G
Research
Report

Institut für Informatik

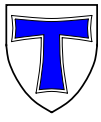
Cellular Automata With Sparse Communication

Martin Kutrib Andreas Malcher

IFIG Research Report 0903
May 2009

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 0903, May 2009

Cellular Automata With Sparse Communication

Martin Kutrib[1] and Andreas Malcher[2]

Institut für Informatik, Universität Giessen
Arndtstraße 2, D-35392 Giessen, Germany

Abstract. We investigate cellular automata whose internal inter-cell communication is bounded. The communication is quantitatively measured by the number of uses of the links between cells. Bounds on the sum of all communications of a computation as well as bounds on the maximal number of communications that may appear between each two cells are considered. It is shown that even the weakest non-trivial device in question, that is, one-way cellular automata where each two neighboring cells may communicate constantly often only, accept rather complicated languages. We investigate the computational capacity of the devices in question and prove an infinite strict hierarchy depending on the bound on the total number of communications during a computation. Despite their sparse communication even for the weakest devices, by reduction of Hilbert's tenth problem undecidability of several problems is derived. Finally, the question whether a given real-time one-way cellular automaton belongs to the weakest class is shown to be undecidable. This result can be adapted to answer an open question posed in [Vollmar, R.: Zur Zustandsänderungskomplexität von Zellularautomaten. In: Beiträge zur Theorie der Polyautomaten—zweite Folge, Braunschweig (1982) 139–151 (in German)].

F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Cellular automata*; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*Parallelism and concurrency*; F.1.3 [**Computation by Abstract Devices**]: Complexity Measures and Classes—*Complexity hierarchies*; F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages—*Decision problems*

Additional Key Words and Phrases: —

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

1 Introduction

Parallel computational models are appealing and widely used in order to describe, understand, and manage parallel processes occurring in real life. One principal task in order to employ a parallel computational model in an optimal way is to understand how cooperation of several processors is organized optimally. To this end, it is essential to know which communication and which amount of communication must or should take place between several processors. From the viewpoint of energy and the costs of communication links, it would be desirable to communicate a minimal number of times with a minimum amount of information transmitted. On the other hand, it would be interesting to know how much communication is necessary in a certain parallel model to accomplish a certain task.

In this paper, we study the parallel computational model of cellular automata which are linear arrays of identical copies of deterministic finite automata, where the single nodes, which are called cells, are homogeneously connected to their both immediate neighbors. They work synchronously at discrete time steps. In the general case, in every time step the state of each cell is communicated to its neighbors. That is, on one hand the state is sent regardless of whether it is really required, and on the other hand, the number of bits sent is determined by the number of states. The latter question has been dealt with in [8, 9, 16–18, 22] where the bandwidth of the inter-cell links is bounded by some constant being independent of the number of states. The former question concerns the amount of communication necessary for a computation. In [19, 20] two-way cellular automata are considered where the number of proper state changes is bounded. There are strong relations to inter-cell communication. Roughly speaking, a cell can remember the states received from its neighbors. As long as these do not change, no communication is necessary. Here we investigate cellular automata where the communication is quantitatively measured by the number of uses of the links between cells. Bounds on the sum of all communications of a computation as well as bounds on the maximal number of communications that may appear between each two cells are considered.

In the next section we present some basic notions and definitions, and introduce the classes of communication bounded cellular automata. Examples of constructions for important types of languages are presented. Then, in Section 3 some computational capacity aspects are investigated, where an infinite strict hierarchy depending on the bound on the total number of communications during an computation is shown. Since the proof methods used in connection with the number of state changes in [19, 20] apply also for the devices in question we adapt and summarize some of the known results.

Section 4 is devoted to decidability problems. We consider the weakest nontrivial device in question, that is, one-way cellular automata where each two neighboring cells may communicate constantly often only, and show by reduction of Hilbert's tenth problem undecidability of several problems. It turns out that also the question whether or not a given real-time one-way cellular automaton belongs to the weakest class of cellular automata with sparse communication

is undecidable. This result can be adapted to answer an open question posed in [21].

2 Definitions and Preliminaries

We denote the positive integers and zero $\{0, 1, 2, ...\}$ by $\mathbb{N}$. 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 $|w|_x$, and for a set of words X, we define $|w|_X = \sum_{x \in X} |w|_x$. We use $\subseteq$ for inclusions and $\subset$ for strict inclusions. For a function $f : \mathbb{N} \to \mathbb{N}$ we denote its i-fold composition by $f^{[i]}$, $i \in \mathbb{N}$, where $f^{[0]}$ denotes the identity.

A cellular automaton is a linear array of identical deterministic finite state machines, sometimes called cells. Except for the leftmost cell and rightmost cell each one is connected to its both nearest neighbors. We identify the cells by positive integers. The state transition depends on the current state of each cell and on the information which is currently sent by its neighbors. The information sent by a cell depends on its current state and is determined by so-called communication functions. The two outermost cells receive a boundary symbol on their free input lines once during the first time step from the outside world. Subsequently, these input lines are never used again. A formal definition is:

Definition 1. *A* cellular automaton (*CA*) *is a system* $\langle S, F, A, B, \#, b_l, b_r, \delta \rangle$, *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, B *is the set of* communication symbols, $\# \notin B$ *is the* boundary symbol, $b_l, b_r : S \to B \cup \{\perp\}$ *are* communication functions *which determine the information* to be sent *to the left and right neighbors, where* $\perp$ *means* nothing to send, *and* $\delta : (B \cup \{\#, \perp\}) \times S \times (B \cup \{\#, \perp\}) \to S$ *is the* local transition function.

A *configuration* of a cellular automaton $\langle S, F, A, B, \#, b_l, b_r, \delta \rangle$ at time $t \geq 0$ is a description of its global state, which is actually a mapping $c_t : [1, \ldots, n] \to S$, for $n \geq 1$. The operation starts at time 0 in a so-called *initial configuration*. For a given input $w = a_1 \cdots a_n \in A^+$ we set $c_{0,w}(i) = a_i$, for $1 \leq i \leq n$. During its course of computation a CA steps through a sequence of configurations, whereby successor configurations are computed according to the global transition function Δ: Let c_t, $t \geq 0$, be a configuration. Then its successor configuration $c_{t+1} = \Delta(c_t)$ is as follows. For $2 \leq i \leq n-1$, $c_{t+1}(i) = \delta(b_r(c_t(i-1)), c_t(i), b_l(c_t(i+1)))$, and for the leftmost and rightmost cell we set $c_1(1) = \delta(\#, c_0(1), b_l(c_0(2)))$, $c_{t+1}(1) = \delta(\perp, c_t(1), b_l(c_t(2)))$, for $t \geq 1$, and $c_1(n) = \delta(b_r(c_0(n-1)), c_0(n), \#)$, $c_{t+1}(n) = \delta(b_r(c_t(n-1)), c_t(n), \perp)$, for $t \geq 1$. Thus, the global transition function Δ is induced by δ.

An input w is accepted by a CA $\mathcal{M}$ if at some time i during its course of computation the leftmost cell enters an accepting state. The *language accepted*

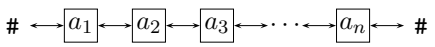


Fig. 1. A two-way cellular automaton.

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

An important subclass of cellular automata are so-called *one-way cellular automata* (OCA), where the flow of information is restricted to one way from right to left. For a formal definition it suffices to require that b_r maps all states to $\perp$, and that the leftmost cell does not receive the boundary symbol during the first time step.

In the following we study the impact of communication in cellular automata. The communication is measured by the number of uses of the links between cells. It is understood that whenever a communication symbol not equal to $\perp$ is sent, a communication takes place. Here we do not distinguish whether either or both neighboring cells use the link. More precisely, the number of communications between cell i and cell $i+1$ up to time step t is defined by

$$\mathrm{com}(i,t) = |\{\, j \mid 0 \leq j < t \text{ and } (b_r(c_j(i)) \neq \perp \text{ or } b_l(c_j(i+1)) \neq \perp) \,\}| \,.$$

For computations we now distinguish the maximal number of communications between two cells and the total number of communications. Let $c_0, c_1, \ldots, c_{t(|w|)}$ be the sequence of configurations computed on input w by some cellular automaton with time complexity $t(n)$, that is, the *computation on* w. Then we define $\mathrm{mcom}(w) = \max\{\, \mathrm{com}(i, t(|w|)) \mid 1 \leq i \leq |w| - 1 \,\}$ and $\mathrm{scom}(w) = \sum_{i=1}^{|w|-1} \mathrm{com}(i, t(|w|))$. Let $f : \mathbb{N} \to \mathbb{N}$ be a mapping. If all $w \in L(\mathcal{M})$ are accepted with computations where $\mathrm{mcom}(w) \leq f(|w|)$, then $\mathcal{M}$ is said to be *max communication bounded by* f. Similarly, if all $w \in L(\mathcal{M})$ are accepted with computations where $\mathrm{scom}(w) \leq f(|w|)$, then $\mathcal{M}$ is said to be *sum communication bounded by* f. In general, it is not expected to have tight bounds on the exact number of communications but tight bounds on their numbers in the order of magnitude. For the sake of readability we denote the class of CAs that are max communication bounded by some function $g \in O(f)$ by MC(f)-CA, where it is understood that f gives the order of magnitude. Corresponding notations are used for OCAs and sum communication bounded CAs and OCAs. (SC(f)-CA and SC(f)-OCA).

The family of all languages which are accepted by some device X with time complexity t is denoted by $\mathscr{L}_t(X)$. In the sequel we are particularly interested in fast computations and call the time complexity $t(n) = n$ *real time* and write $\mathscr{L}_{rt}(X)$. To illustrate the definitions we start with an example.

Lemma 2. *The language* $\{\, a^n b^n \mid n \geq 1 \,\}$ *belongs to* $\mathscr{L}_{rt}(MC(1)\text{-}OCA)$.

Proof. The acceptance of the language is governed by two signals. The rightmost cell is sending a signal B with maximum speed to the left whereas the unique cell which has an a in its input and has a right neighbor with a b in its input is sending a signal A with speed $1/2$ to the left. When both signals meet in a cell, an accepting state is assumed. Obviously, $\{\, a^n b^n \mid n \geq 1 \,\}$ is accepted and each cell performs only a finite number of communications. □

By an obvious generalization of the above construction with suitable signals having a certain speed we obtain that the languages $\{\, a^n b^n c^n \mid n \geq 1 \,\}$,

$\{ a^n b^m c^n d^m \mid n, m \geq 1 \}$, and $\{ a_1^n a_2^n \cdots a_k^n \mid n \geq 1 \}$, for $k \geq 1$ and different symbols $a_1, a_2, \ldots, a_k$, are accepted by real-time MC(1)-OCAs as well. The languages are non context free.

For the language $\{ a^n b^{n_1} c^m b^{n_2} \mid n, m \geq 1 \wedge n_1, n_2 \geq 0 \wedge n_1 + n_2 = n \}$ the above technique of suitable signals having an appropriate speed cannot be applied, since the block of cs may be arbitrary large. Here, the first idea is to use two different signals B and $\circ$. All b-cells send a signal B to the left which is matched against the a-cells. All c-cells send a signal $\circ$ to the left which does not affect the matching of a-cells and B-signals. This approach implies that some cells may forward an arbitrary number of signals B or $\circ$ and leads to a real-time OCA which is not an MC(1)-OCA. But, we can overcome this problem by applying the following technique. Whenever some cell has sent a signal X to the left, it sends $\perp$ in the next time steps as long as no other signal $Y \neq X$ has to be sent to the left. The cells which obtain some signal X for the first time store this in their state. The information $\perp$ arriving in the next time steps can then be interpreted as "nothing has changed," that is, each $\perp$ is interpreted as a signal X and is suitably processed. It can be observed that in this way each cell performs only a finite number of communications as long as only a finite number of blocks of identical signals has to be sent to the left.

Lemma 3. *The language* $\{ a^n b^{n_1} c^m b^{n_2} \mid n, m \geq 1 \wedge n_1, n_2 \geq 0 \wedge n_1 + n_2 = n \}$ *belongs to* $\mathscr{L}_{rt}(MC(1)\text{-}OCA)$.

Proof. All b-cells send a signal B which is forwarded by all b-cells and c-cells and is matched against the a-cells. That is, when a signal B arrives in an a-cell it is stopped and the cell is marked as a matched cell. When a signal B arrives at a marked a-cell, the signal is forwarded to the left as long as it arrives at an unmarked a-cell where it is stopped and marks the cell as matched. The c-cells send a signal $\circ$ which is forwarded by b-cells to the left. All a-cells are forwarding $\circ$-signals to the left as long as they are stopped by an a-cell which has not yet sent a signal B to the left. These $\circ$-signals do not affect the matching of a-cells with B-signals, but they carry the information that the first block of bs has been processed. Initially, in the rightmost cell a signal $\triangleleft$ is started which checks the correct formatting and forces a marked a-cell which has not been used to forward a B-signal to enter an accepting state. Additionally, the signal $\triangleleft$ is stopped. In this way, a real-time OCA acceptor has been constructed. By applying the technique described above we obtain a real-time MC(1)-OCA, since we have one block of signals B followed by one block of signals $\circ$, which is followed by another block of signals B. Thus, the assertion follows. An example schematically accepting $a^4 b^2 c^2 b^2$ is depicted in Figure 2. □

A straightforward generalization yields the next lemma.

Lemma 4. *Let* $k \geq 0$ *be a constant. Then language* $L_k = \{ a^n w \mid n \geq 1 \wedge w \in (b^* c^*)^k b^* \wedge |w|_b = n \}$ *belongs to* $\mathscr{L}_{rt}(MC(1)\text{-}OCA)$.

Let us finally remark that a helpful tool for later constructions is the closure under the Boolean operations which can be proven in the same way as for unrestricted real-time OCAs.

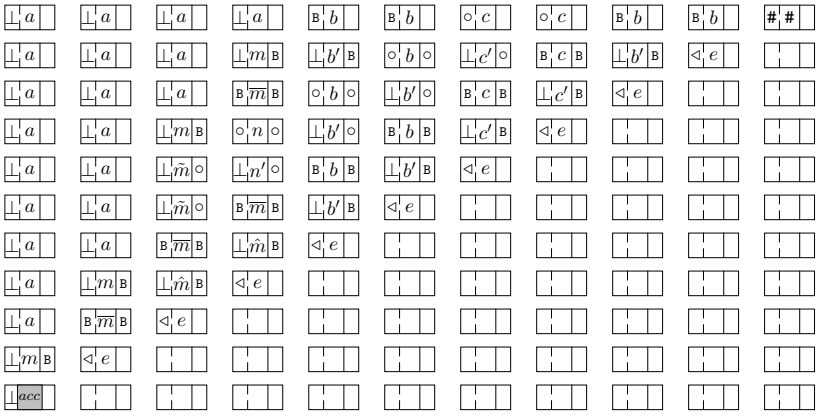


Fig. 2. Schematic computation of an MC(1)-OCA accepting $a^4b^2c^2b^2$ in real time. The left part of each cell denotes the communication channel to the left neighboring cell. The middle and right part of each cell depict the current state of each cell whereas in the right part a B ($\circ$) is stored when a $\perp$ arriving in the next time step has to be interpreted as a B-signal ($\circ$-signal).

Lemma 5. *The family $\mathscr{L}_{rt}(MC(1)\text{-}OCA)$ is closed under union, intersection, and complementation.*

3 Computational Capacity

In order to identify the computational power of communication bounded real-time devices we begin by describing the relationship to previous works. In [19, 20] two-way cellular automata are considered where the number of proper state changes is bounded. Similar as in the present paper the sum of all state changes or the maximal number of the state changes of single cells are bounded. By applying the technique of saving communication steps by storing the last signal received in the state and to interpret an arriving $\perp$ suitably, it is not hard to see, that such a device can be simulated by the corresponding communication bounded device. Whether or not state change bounded devices are strictly weaker than communication bounded ones is an open problem. However, the restrictions introduced in [19, 20] have been investigated with respect to communication in cellular automata, and the proof methods used apply also for the devices in question. So, we adapt some of the results shown in connection with state changes in the next theorem.

Theorem 6 ([19, 20]). *1. $\mathscr{L}_{rt}(MC(1)\text{-}CA) \subset \mathscr{L}_{rt}(SC(n)\text{-}CA)$.*
2. $REG \subset \mathscr{L}_{rt}(MC(1)\text{-}CA) \subset \mathscr{L}_{rt}(MC(\sqrt{n})\text{-}CA) \subset \mathscr{L}_{rt}(MC(n)\text{-}CA)$.
3. $\mathscr{L}_{rt}(MC(1)\text{-}CA) \subset NL$.

Next we turn to show an infinite proper hierarchy of real-time SC(f)-CA families. We start with the top of the hierarchy.

Theorem 7. *Let $f : \mathbb{N} \to \mathbb{N}$ be a function. If $f \in o(n^2/\log(n))$, then language $L = \{\, wcw^R \mid w \in \{a, b\}^+ \,\}$ is not accepted by any real-time $SC(f)$-CA.*

Proof. In contrast to the assertion, assume that L is accepted by some real-time SC(f)-CA $\mathcal{M}$. We consider accepting computations on wcw^R.

We claim that for any constant $k > 0$, there must exist a length $n_k \geq 2$ such that for all $w \in \{a,b\}^{2n_k}$ there is a cell $j(w)$, where $n_k + 1 \leq j(w) \leq 2n_k$, such that the number of communications occurring between cells $j(w)$ and $j(w)+1$ is at most $k4n_k/\log(4n_k)$.

If the claim would be wrong, then there would be a constant $k > 0$, such that for all lengths n_k there is a word $w \in \{a,b\}^{2n_k}$ such that for all $n_k+1 \leq j(w) \leq 2n_k$, the number of communications occurring between cells $j(w)$ and $j(w)+1$ is at least $k4n_k/\log(4n_k)$. Therefore, the total number of communications during an accepting computation on wcw^R is at least $k4n_k(2n_k - n_k - 1)/\log(4n_k)$ which is of order $\Omega(n_k^2/\log(n_k)$. Since for all lengths n_k there is such a word w, a contradiction to the assumption $f \in o(n^2/\log(n))$ follows, and the claim is shown.

Now we turn to derive an upper bound on the number of possibilities for some r communications between two cells in real-time computations. To this end, we have to consider the information to be communicated as well as the time steps at which the communications take place. There are $\binom{n}{r}$ possibilities to choose time steps, and $(|B|+1)^2 - 1$ possibilities to use a link, where both cells must not send $\perp$ simultaneously in order to have a communication at all. So, there are at most

$$\begin{aligned}\binom{n}{r}((|B|+1)^2-1)^r &\leq \frac{n^r}{(r/2)^{r/2}} 2^{\log(|B|+1)2r} \leq \frac{n^r 2^{r/2}}{r^{r/2}} 2^{\log(|B|+1)2r} \\ &\leq 2^{\log(n)r + r/2 + \log(|B|+1)2r - \log(r)r/2} \\ &\leq 2^{k_0 \log(n) r}, \text{ for some constant } k_0 \geq 1,\end{aligned}$$

possibilities. Next we choose $k < 1/(16k_0)$ and apply the claim shown above. So, there is an $n_k \geq 2$ such that for all $w \in \{a,b\}^{2n_k}$ there is a cell $j(w)$, where $n_k+1 \leq j(w) \leq 2n_k$, such that the number of communications occurring between cells $j(w)$ and $j(w)+1$ is at most $r = k4n_k/\log(4n_k)$. For these communications there are at most

$$2^{k_0 \log(4n_k+1) r} \leq 2^{k_0 2 \log(4n_k) k \frac{4n_k}{\log(4n_k)}} \leq 2^{k_0 2 \log(4n_k) \frac{1}{16k_0} \frac{4n_k}{\log(4n_k)}} \leq 2^{\frac{n_k}{2}}$$

possibilities. Since there are 2^{n_k} words of length n_k, there must exist two words

$$\begin{aligned} w_1 &= u_1 u_2 \cdots u_{n_k} u_{n_k+1} \cdots u_{j(w_1)} u_{j(w_1)+1} \cdots u_{2n_k} \text{ and} \\ w_2 &= v_1 v_2 \cdots v_{n_k} v_{n_k+1} \cdots v_{j(w_2)} v_{j(w_2)+1} \cdots v_{2n_k} \end{aligned}$$

with accepting computations on $w_1 c w_1^R$ and $w_2 c w_2^R$ that differ in their first n_k symbols, and that imply exactly the same communications between cells $j(w_1)$ and $j(w_1)+1$ on one hand and between cells $j(w_2)$ and $j(w_2)+1$ on the other hand. Therefore, also the input $u_1 u_2 \cdots u_{n_k} u_{n_k+1} \cdots u_{j(w_1)} v_{j(w_2)+1} \cdots v_{2n_k} c w_2^R$ is accepted, which is a contradiction since it does not belong to L. □

In order to define witness languages that separate the levels of the hierarchy, for all $i \geq 1$, the functions $\varphi_i : \mathbb{N} \to \mathbb{N}$ are defined by $\varphi_1(n) = 2^n$, and $\varphi_i(n) = 2^{\varphi_{i-1}(n)}$, for $i \geq 2$, and we set $L_i = \{\, w\$^{\varphi_i(|w|)-2|w|} w^R \mid w \in \{a,b\}^+ \,\}$.

Lemma 8. *Let $i \geq 1$ be an integer and $f : \mathbb{N} \to \mathbb{N}$ be a function. If $f \in o((n \log^{[i]}(n))/\log^{[i+1]}(n))$, then language L_i is not accepted by any real-time SC(f)-CA.*

Proof. First we note that a cell that carries initially a \$ cannot usefully communicate until it can receive information for the first time from one of the cells not carrying initially a \$. Up to that time its state is the same as the states of its neighbors. For example, some cell j with $n/2 \leq j \leq n - |w|$ cannot usefully communicate before time step $n - |w| - j$. On the other hand, its last (useful) communication that may influence the overall computation result must appear at time $n - j + 1$. So, there are at most $|w| + 1$ time steps for useful communications. Now we can continue as in the proof of Theorem 7 to show that for all cells $n/2 \leq j \leq n - |w|$ the number of communications occurring in accepting computations between cells j and $j + 1$ is of order $\Omega(|w|/\log(|w|))$.

So, in total we have at least $\Omega((|w|/\log(|w|))(n/2 - |w|))$ communications. Since $n = \varphi_i(|w|)$, we substitute $|w|$ by $\varphi_i^{-1}(n) = \log^{[i]}(n)$ and obtain

$$\Omega\left(\frac{\log^{[i]}(n)}{\log^{[i+1]}(n)}\left(\frac{n}{2} - \log^{[i]}(n)\right)\right) = \Omega\left(\frac{n \log^{[i]}(n)}{\log^{[i+1]}(n)}\right)$$

which concludes the proof. □

Lemma 9. *Let $i \geq 1$ be an integer. Then language L_i is accepted by some real-time SC($n \log^{[i]}(n)$)-CA.*

Proof. We sketch the construction of a real-time SC($n \log^{[i]}(n)$)-CA $\mathcal{M}$ that accepts L_i. Basically, $\mathcal{M}$ has to perform two tasks. One is to match the input prefix w with the suffix w^R, the other one is to verify the number of \$s.

The first task is realized as follows. The prefix w is shifted successively to the right and the suffix w^R is shifted successively to the left. When the words arrive in the middle of the array, the symbols are matched one by one. Since initially, a cell carrying not a \$ cannot know whether it belongs to the prefix or suffix, the symbols are shifted to the left as well as to the right on different tracks. The shifting to the 'wrong' directions do not affect the computation. Altogether, the first task requires communication for every shifted symbol. To shift the prefix w and the suffix w^R to the center, to match them, and to send the result back to the leftmost cell not more than $O(|w|n) = O(n \log^{[i]}(n))$ communications are necessary.

The second task combines known methods to simulate a stack and to construct the function φ_i in time in order to verify the number of \$s. One subtask is to simulate a pushdown store without any loss of time, where the top of the stack is simulated by the leftmost cell. Details of the construction can be found in [1, 4, 6]. Initially, the unique cell carrying not a \$ with a right neighbor carrying a \$ sends a signal to the left. The leftmost cell pushes tokens onto the stack until it receives the signal. So, it can push $|w|$ tokens. Also at initial time a time constructor for the function φ_i is started, that is, a subtask that distinguishes exactly the time steps $\varphi_i(j)$, for $j \geq 1$, at the leftmost cell. At any time step $\varphi_i(j)$ a symbol is popped from the stack. If in addition a signal that

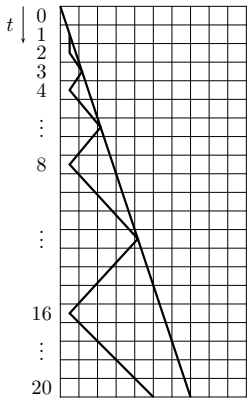


Fig. 3. Space-time diagram showing the time constructor for the function 2^n.

is initially started in the rightmost cell arrives in the leftmost cell exactly at a time step that is distinguished by the time constructor and at which the last symbol is popped from the stack, then the total length of the input is verified to be $\varphi_i(|w|)$.

So, it remains to be shown how to set up the time constructor and to analyze the number of communications. For more details on time-constructible function see [1, 2, 7, 13]. The time constructor is inductively constructed as follows. For $\varphi_1(n) = 2^n$ (cf. [3]), at initial time the leftmost cell emits a signal which moves with speed 1/3 to the right. In addition, another signal is emitted which moves with maximal speed and bounces between the slow signal and the leftmost cell. It is easy to see that the signal passes through the leftmost cell exactly at the time steps 2^j, $j \geq 1$ (see Figure 3).

Now assume that there is a time constructor for φ_i. Then another one for the function 2^n is implemented on a different track. Both are working together as follows. Basically, the time constructor for 2^n is idle until the cells receive from the left the fast signal from the time constructor for φ_i, that is, after the signal has distinguished the leftmost cell. This signal causes the cells to perform one transition of the time constructor for 2^n cell by cell from left to right. We omit minor details of the construction. Altogether, when the delayed time constructor for 2^n distinguishes the leftmost cell for the jth time, it has received 2^j signals from time constructor for φ_i, that is, at time $\varphi_i(2^j) = \varphi_{i+1}(j)$.

For the number of communications of the second task we have the communications within the leftmost $|w|$ cells for the simulation of the stack. These are at most $O(|w|n) = O(n \log^{[i]}(n))$. In addition, we have i time constructors working together. Each consists basically of two signals. So, not more than $i2n \in O(n)$ communications are necessary. □

Now we are prepared to derive the infinite hierarchy.

Theorem 10. *Let $i \geq 0$ be an integer. Then $\mathscr{L}_{rt}(SC(n \log^{[i+1]}(n))\text{-}CA)$ is properly included in $\mathscr{L}_{rt}(SC(n \log^{[i]}(n))\text{-}CA)$.*

Proof. The inclusion is trivial. For $i = 0$, consider the linear context-free language $L = \{ wcw^R \mid w \in \{a, b\}^+ \}$. In [15] it is shown that any linear context-

free language is accepted by some real-time CA. So, obviously it is accepted by a real-time SC(n^2)-CA. Since $n\log(n) \in o(n^2/\log(n))$, language L does not belong to $\mathscr{L}_{rt}$(SC($n\log(n)$)-CA) by Theorem 7.

For $i \geq 1$, a witness for the properness of the inclusion is language L_i. By Lemma 9 it belongs to $\mathscr{L}_{rt}$(SC($n\log^{[i]}(n)$)-CA).

Since $n\log^{[i+1]}(n) = \frac{n(\log^{[i+1]}(n))^2}{\log^{[i+1]}(n)}$ and

$$\lim_{n\to\infty} \frac{\frac{n(\log^{[i+1]}(n))^2}{\log^{[i+1]}(n)}}{\frac{n\log^{[i]}(n)}{\log^{[i+1]}(n)}} = \lim_{n\to\infty} \frac{n(\log^{[i+1]}(n))^2}{n\log^{[i]}(n)} = \lim_{n\to\infty} \frac{(\log^{[i+1]}(n))^2}{2^{\log^{[i+1]}(n)}} = 0$$

we have $n\log^{[i+1]}(n) \in o((n\log^{[i]}(n))/\log^{[i+1]}(n))$. Therefore, by Lemma 8 language L_i does not belong to $\mathscr{L}_{rt}$(SC($n\log^{[i+1]}(n)$)-CA). □

4 Decidability Questions

This section is devoted to decidability problems. In fact, the results show undecidability of various questions, even for the weakest non-trivial device under consideration, that is, for real-time MC(1)-OCAs. Needless to say, the undecidability carries over to all the other (stronger) devices considered.

Two of the common techniques to show undecidability results are reductions of Post's Correspondence Problem or reductions of the emptiness and finiteness problem on Turing machines using the set of valid computations. Both techniques have been used successfully to obtain results for variants of cellular automata [10–12, 14]. Taking a closer look at these known techniques, it is not clear yet whether they can be applied to MC(1)-OCAs. Here, we first show that emptiness is undecidable for real-time MC(1)-OCAs by reduction of Hilbert's tenth problem which is known to be undecidable. The problem is to decide whether a given polynomial $p(x_1, \ldots, x_n)$ with integer coefficients has an integral root. That is, to decide whether there are integers $\alpha_1, \ldots, \alpha_n$ such that $p(\alpha_1, \ldots, \alpha_n) = 0$. In [5] Hilbert's tenth problem has been used to show that emptiness is undecidable for certain multi-counter machines. As is remarked in [5], it is sufficient to restrict the variables $x_1, \ldots, x_n$ to take non-negative integers only. If $p(x_1, \ldots, x_n)$ contains a constant summand, then we may assume that it has a negative sign. Otherwise, $p(x_1, \ldots, x_n)$ is multiplied with -1. Such a polynomial then has the following form: $p(x_1, \ldots, x_n) = t_1(x_1, \ldots, x_n) + \ldots + t_r(x_1, \ldots, x_n)$, where each $t_j(x_1, \ldots, x_n)$ $(1 \leq j \leq r)$ is a term of the form $t_j(x_1, \ldots, x_n) = s_j x_1^{i_{j,1}} \ldots x_n^{i_{j,n}}$ with $s_j \in \{+1, -1\}$ and $i_{j,1}, \ldots, i_{j,n} \geq 0$. Additionally, we may assume that the summands are ordered according to their sign, i.e., there exists $1 \leq q \leq r$ such that $s_1 = \ldots = s_q = 1$ and $s_{q+1} = \ldots = s_r = -1$. Moreover, constant terms are occurring only at the end of the sum. I.e., $t_r = \ldots = t_{r-c+1} = -1$, if p contains $c > 0$ constant terms. Finally, let $i_j = \sum_{t=1}^{n} i_{j,t}$.

Now, we consider a polynomial $p(x_1, \ldots, x_n)$ with integer coefficients that has the above form. We first look at the positive terms t_j of $p(x_1, \ldots, x_n)$ with $1 \leq j \leq q$ and define languages $L(t_j)$ as follows.

$$L(t_j) = \{b_1^{\alpha_1} \dots b_{i_{j,1}}^{\alpha_1} b_{i_{j,1}+1}^{\alpha_2} \dots b_{i_{j,1}+i_{j,2}}^{\alpha_2} \dots b_{i_{j,1}+\dots+i_{j,n-1}+1}^{\alpha_n} \dots b_{i_j}^{\alpha_n} \cdot f_j(\underbrace{\alpha_1, \dots, \alpha_1}_{i_{j,1}}, \dots, \underbrace{\alpha_n, \dots, \alpha_n}_{i_{j,n}})\text{¢} \mid \alpha_1, \dots, \alpha_n \geq 0\}$$

where $f_j : \mathbb{N}^{i_j} \to \{\$_1, \$_2, \dots, \$_{i_j}\}^*$ is inductively defined by the following rules with $1 \leq i \leq i_j - 1$.

$$f_j(\alpha_1, \dots, \alpha_{i_j}) = \left(f_j^{(i_j-1)}(\alpha_1, \dots, \alpha_{i_j-1})\$_{i_j}\right)^{\alpha_{i_j}}$$
$$f_j^{(i)}(\alpha_1, \dots, \alpha_i) = \left(f_j^{(i-1)}(\alpha_1, \dots, \alpha_{i-1})\$_i\right)^{\alpha_i - 1} f_j^{(i-1)}(\alpha_1, \dots, \alpha_{i-1})$$
$$f_j^{(0)} = \lambda$$

For the negative, non-constant terms t_j with $q+1 \leq j \leq r$ the definition of $L(t_j)$ is identical except for the fact that each symbol $\$_k$ is replaced by some symbol $€_k$. For each negative, constant term t_j, we define $L(t_j) = \{€_1\}$.

Lemma 11. *For $1 \leq j \leq r$, $|f_j(\alpha_1, \dots, \alpha_{i_j})| = \alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_{i_j}$.*

Proof. It can be shown by induction that for $1 \leq i \leq i_j - 1$, $|f_j^{(i)}(\alpha_1, \dots, \alpha_i)| = \alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_i - 1$. Then, $|f_j(\alpha_1, \dots, \alpha_{i_j})| = (|f_j^{(i_j-1)}(\alpha_1, \dots, \alpha_{i_j-1})| + 1) \cdot \alpha_{i_j} = (\alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_{i_j-1} - 1 + 1) \cdot \alpha_{i_j} = \alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_{i_j}$. □

Thus, if $w \in L(t_j)$ and w contains ℓ symbols \$ or €, respectively, then there exist non-negative integers $\alpha_1, \dots, \alpha_n$ such that $t_j(\alpha_1, \dots, \alpha_n) = s_j \cdot \ell$. In other words, the number of symbols \$ or € occurring in $L(t_j)$ denote all evaluations of t_j on non-negative integers. Furthermore, symbols \$ or € denote evaluations with positive or negative sign, respectively.

Example 12. Let $t_j(x_1, x_2, x_3, x_4) = x_1^2 x_2 x_4$. Then, $i_{j,1} = 2$, $i_{j,2} = 1$, $i_{j,3} = 0$, $i_{j,4} = 1$, and $i_j = 4$. Then,

$$L(t_j) = \{b_1^{\alpha_1} b_2^{\alpha_1} b_3^{\alpha_2} b_4^{\alpha_4} f_j(\alpha_1, \alpha_1, \alpha_2, \alpha_4)\text{¢} \mid \alpha_1, \alpha_2, \alpha_4 \geq 0\}$$

with

$$f_j^{(1)}(\alpha_1) = \$_1^{\alpha_1 - 1}$$
$$f_j^{(2)}(\alpha_1, \alpha_1) = \left(\$_1^{\alpha_1-1}\$_2\right)^{\alpha_1-1} \$_1^{\alpha_1-1}$$
$$f_j^{(3)}(\alpha_1, \alpha_1, \alpha_2) = \left(\left(\$_1^{\alpha_1-1}\$_2\right)^{\alpha_1-1} \$_1^{\alpha_1-1}\$_3\right)^{\alpha_2-1} \left(\$_1^{\alpha_1-1}\$_2\right)^{\alpha_1-1} \$_1^{\alpha_1-1}$$
$$f_j(\alpha_1, \alpha_1, \alpha_2, \alpha_4) = \left(\left(\left(\$_1^{\alpha_1-1}\$_2\right)^{\alpha_1-1} \$_1^{\alpha_1-1}\$_3\right)^{\alpha_2-1} \left(\$_1^{\alpha_1-1}\$_2\right)^{\alpha_1-1} \$_1^{\alpha_1-1}\$_4\right)^{\alpha_4}$$

For example, to evaluate $t_j(2, 3, x_3, 2) = 2^2 \cdot 3 \cdot 2 = 24$ we consider the word

$$b_1^2 b_2^2 b_3^3 b_4^2 \$_1\$_2\$_1\$_3\$_1\$_2\$_1\$_3\$_1\$_2\$_1\$_4\$_1\$_2\$_1\$_3\$_1\$_2\$_1\$_3\$_1\$_2\$_1\$_4\text{¢} \in L(t_j)$$

and to evaluate $t_j(2,1,x_3,2) = 2^2 \cdot 1 \cdot 2 = 8$ we consider the word

$$b_1^2 b_2^2 b_3 b_4^2 \$_1 \$_2 \$_1 \$_4 \$_1 \$_2 \$_1 \$_4 ¢ \in L(t_j).$$

Lemma 13. *For $1 \leq j \leq r$, let t_j be a non-constant term. Then language $L(t_j)^R$ belongs to $\mathscr{L}_{rt}(MC(1)\text{-}OCA)$.*

Proof. For the sake of simplicity we describe a real-time MC(1)-OCA which has information flow from left to right and accepts in the rightmost cell. Obviously, considering the reversal of the input, an equivalent MC(1)-OCA with information flow from right to left and acceptance in the leftmost cell can be constructed. We next sketch the construction of a real-time MC(1)-OCA accepting $L(t_j)$ where t_j is a positive term. The construction for negative terms is identical except for changing symbols $\$_k$ with symbols $€_k$. An MC(1)-OCA accepting $L(t_j)$ has to compute two main tasks.

First, the same number of symbols $b_{i_{j,1}+\ldots+i_{j,k-1}+1}, \ldots, b_{i_{j,1}+\ldots+i_{j,k}}$ for $1 \leq k \leq n$ has to be checked. This can be realized with an obvious generalization of the construction given in the proof of Lemma 2. Thus, this task can be done by some MC(1)-OCA.

The second task is to compute f_j with variables $\alpha_1, \ldots, \alpha_{i_j}$ which are given in the first part of the input. Here, the task to be done can be described as follows. If $k = i_j$, then the number of symbols b_k has to be checked against the number of symbols $\$_k$. If $1 \leq k < i_j$, we consider blocks of maximum length of the input which start with the symbol $\$_k$, end with the symbol $\$_{k+1}$, and contain exactly one symbol $\$_{k+1}$. Now, the number of symbols b_k has to be checked against the number of symbols $\$_k$ and $\$_{k+1}$ within each block.

Let us first assume that $\alpha_k \geq 2$ for $1 \leq k \leq i_j$. The main idea of the construction is that each b_k-cell sends some signal b_k with maximum speed to the right. If $k = i_j$, each $\$_k$ is matched against some signal b_k and the signal is stopped. If $1 \leq k < i_j$, then each $\$_k$ is matched against some signal b_k and the signal is changed to some signal b'_k which is forwarded with maximum speed to the right, but does not check any matchings between b_k and $\$_k$. Whenever a signal b'_k arrives at some $\$_{k+1}$-cell, i.e., at the end of a block, the signal is reset to the signal b_k and the next block is checked. If the signal b_k arrives at some $\$_{k+1}$-cell, the cell is marked as matched and the signal b_k is forwarded to the right to check the next block. It can be observed that each $\$_k$-cell is marked as matched by all signals b_m with $1 \leq m \leq k$, if the input belongs to $L(t_j)$. Otherwise, some $\$_k$-cell is not suitably marked as matched. Thus, in the leftmost cell a signal is started which checks the correct format of the input as well as the correct markings in the $\$_k$-cells and enables the ¢-cell to enter an accepting state.

If $\alpha_k = 1$ for some $1 \leq k \leq i_j$, then the above-mentioned blocks consist of exactly one symbol $\$_{k+1}$, if $\alpha_{k+1} \geq 2$. In general, we observe that $\alpha_k = 1$ implies that the corresponding block consists of exactly one symbol $\$_t$, where $t = \min\{s \mid (k+1 \leq s \leq i_j - 1 \wedge \alpha_s > 1) \vee s = i_j\}$. For example, if all $\alpha_k = 1$, then each block consists of $\$_{i_j}$ only. It is straightforward to adapt the above construction to work also for these special cases.

If $\alpha_k = 0$ for some $1 \leq k \leq i_j$, i.e., there is no symbol b_k in the input. Then, t_j is equal to zero and the input must not contain any input symbols $\$_m$ with $1 \leq m \leq i_j$. But this case concerns the correct format and can be checked with the signal starting in the leftmost cell.

Let us finally prove that each cell does not perform more than a constant number of communication steps. We use the same technique as in the proof of Lemma 3 to forward blocks of identical signals. We first consider the b_k-cells with $1 \leq k \leq i_j$. Through each such cell pass at most i_j many different b_k-signals and the final checking signal. Additionally, due to the structure of the input the b_k-signals are arriving in blocks. Thus, the number of communication steps is bounded by some constant. For the $\$_k$-cells with $1 \leq k \leq i_j$, we can observe that at most $2i_j$ many different signals b_k or b'_k and the final checking signal pass through each such cell. Additionally, the signals b_k and b'_k are arriving in blocks and, moreover, a cell which has changed from forwarding signals b'_k to signals b_k for some fixed k will never forward again signals b'_k, since a changing from b'_k to b_k results from a correct matching of symbol and signal and marked cells cannot emit signals b'_k (see Figure 4). Thus, within each block of b_k- and b'_k-signals passing through one cell there are at most two changes from b_k to b'_k to b_k. Altogether, the number of communication steps is bounded by some constant. Obviously, the ¢-cell never communicates and therefore we obtain that also the second task can be done by an MC(1)-OCA. □

We next consider the following regular languages R_k depending on the sign of t_k. We set $R_k = b_1^* \dots b_{i_{k,1}}^* \dots b_{i_{k,1}+\dots+i_{k,n-1}+1}^* \dots b_{i_k}^* \{\$_1, \dots, \$_{i_k}\}^* ¢$ if $s_k = 1$, $R_k = b_1^* \dots b_{i_{k,1}}^* \dots b_{i_{k,1}+\dots+i_{k,n-1}+1}^* \dots b_{i_k}^* \{€_1, \dots, €_{i_k}\}^* ¢$ if $s_k = -1$ and t_k is non-constant, and $R_k = €_1^* ¢$ otherwise. Then, we define

$$\tilde{L}(t_j) = \{a_1^{\alpha_1} \dots a_n^{\alpha_n} w_1 \dots w_{j-1} b_1^{\alpha_1} \dots b_{i_{j,1}}^{\alpha_1} \dots b_{i_{j,1}+\dots+i_{j,n-1}+1}^{\alpha_n} \dots b_{i_j}^{\alpha_n} \cdot f_j(\underbrace{\alpha_1, \dots, \alpha_1}_{i_{j,1}}, \dots, \underbrace{\alpha_n, \dots, \alpha_n}_{i_{j,n}}) ¢ w_{j+1} \dots w_r \mid \alpha_1, \dots, \alpha_n \geq 0 \text{ and } w_i \in R_i\}$$

and consider $\tilde{L}(p) = \bigcap_{i=1}^{r} \tilde{L}(t_j)^R$.

Lemma 14. *The language $\tilde{L}(p)$ belongs to $\mathscr{L}_{rt}$(MC(1)-OCA).*

Proof. Since $\mathscr{L}_{rt}$(MC(1)-OCA) is closed under intersection, we have to show that each $\tilde{L}(t_j)^R$ belongs to $\mathscr{L}_{rt}$(MC(1)-OCA). If t_j is a constant term, then $\tilde{L}(t_j)^R$ is a regular language and therefore is in $\mathscr{L}_{rt}$(MC(1)-OCA). Now, let t_j be a non-constant term. As in the proof of Lemma 13 we describe a real-time MC(1)-OCA accepting $\tilde{L}(t_j)$ which has information flow from left to right and accepts in the rightmost cell. Then, $\tilde{L}(t_j)^R$ belongs to $\mathscr{L}_{rt}$(MC(1)-OCA). Due to Lemma 13 we know that an MC(1)-OCA accepting $L(t_j)$ can be constructed. We generalize this construction by concatenating the regular languages $a_1^* \dots a_n^* R_1 \dots R_{j-1}$ and $R_{j+1} \dots R_r$ to $L(t_j)$ from right and left, respectively. It can be observed that this can be done by an MC(1)-OCA. It remains to be shown that for $1 \leq k \leq n$ the number of symbols a_k is equal to each the number of symbols $b_{i_{j,1}+\dots+i_{j,k-1}+1}, \dots, b_{i_{j,1}+\dots+i_{j,k}}$. This can be achieved by an obvious

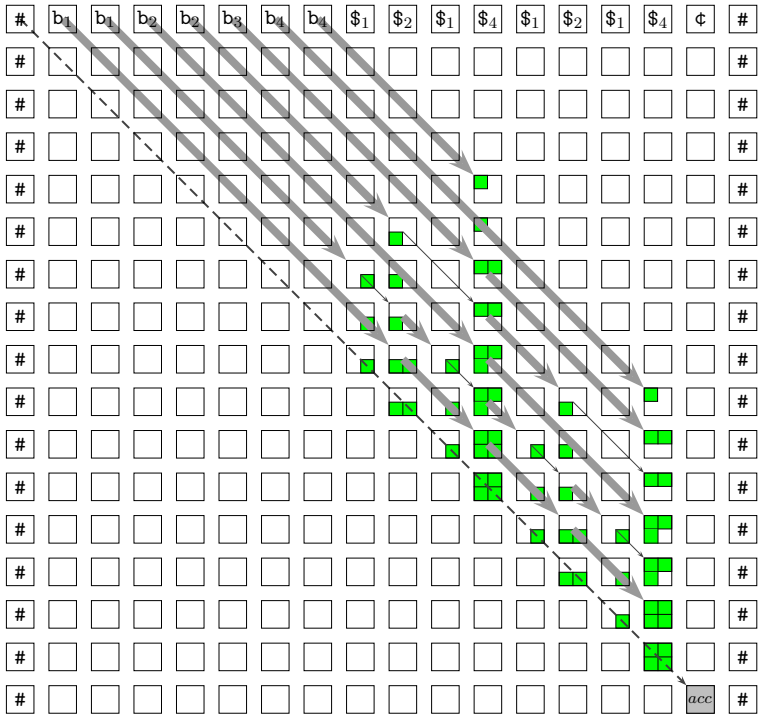


Fig. 4. Schematic computation of an MC(1)-OCA which accepts the input word $b_1^2 b_2^2 b_3 b_4^2 \$_1 \$_2 \$_1 \$_4 \$_1 \$_2 \$_1 \$_4 ¢ \in L(t_j)$ with $t_j(x_1, x_2, x_3, x_4) = x_1^2 x_2 x_4$ in real time. The signals b_k are depicted as large arrows whereas the signals b'_k are depicted as small arrows. The final signal which checks the correct format and the correct matchings is depicted as a dashed arrow. A correct matching of signal b_4 is depicted as an upper left box in the corresponding cell. The remaining matchings of signals b_3, b_2, and b_1 are depicted as upper right box, lower left box, and lower right box, respectively. Then, the final signal can check the correct matchings in each $\$_k$-cell with the help of the original input and the marked boxes.

generalization of the construction given in the proof of Lemma 3. All a_k-cells send signals a_k to the right. Whenever the $(j-1)$st ¢-cell has been passed, the matching of $b_{i_{j,1}+\ldots+i_{j,k-1}+m}$-cells $(1 \leq m \leq i_{j,k})$ with the signal a_k starts. Due to Lemma 3, this task can be done by some MC(1)-OCA. This implies $\tilde{L}(t_j)^R \in \mathscr{L}_{rt}(\text{MC(1)-OCA})$ and shows the lemma. □

Finally, let X be the set of all occurring symbols $\$_k$ and Y be the set of all occurring symbols €$_k$. Then, we define $L(p) = \{w \in \tilde{L}(p) \mid |w|_X = |w|_Y\}$.

Lemma 15. *The language $L(p)$ belongs to $\mathscr{L}_{rt}(MC(1)\text{-}OCA)$.*

Proof. A real-time MC(1)-OCA accepting $L(p)$ has to check whether the input belongs to $\tilde{L}(p)$ as well as to check the equal number of $-and €-symbols. The first task can be done by an MC(1)-OCA due to Lemma 14. The second task is a variation of the language presented in Lemma 4. All €-symbols are sending a signal € to the left which is matched against one $-cell. It can be observed that the €-signals pass on their way to the left only a finite number of blocks which contain no $- or €-symbols. According to the discussion before Lemma 3, we obtain that the second task can be realized by an MC(1)-OCA as well. This shows the lemma. □

Theorem 16. *Given an arbitrary real-time MC(1)-OCA $\mathcal{M}$, it is undecidable whether $L(\mathcal{M})$ is empty.*

Proof. Due to Lemma 15 we can construct a real-time MC(1)-OCA $\mathcal{M}$ accepting $L(p)$. By the construction of $L(p)$, it is not difficult to observe that $\mathcal{M}$ accepts the empty set if and only if $p(x_1, \ldots, x_n)$ has no solution in the non-negative integers. Since Hilbert's tenth problem is undecidable, we obtain that the emptiness problem for real-time MC(1)-OCAs is undecidable. □

Corollary 17. *The problems finiteness, infiniteness, universality, equivalence, inclusion, regularity, and context-freedom are undecidable for arbitrary real-time MC(1)-OCAs.*

Proof. Undecidability of emptiness implies immediately the undecidability of inclusion and equivalence.

Consider the language $L(\mathcal{M})\{a\}^*$ for some new alphabet symbol a. Obviously, $L(\mathcal{M})\{a\}^* \in \mathscr{L}_{rt}(\text{MC(1)-OCA})$ and $L(\mathcal{M})\{a\}^*$ is finite if and only if $L(\mathcal{M}) = \emptyset$. Since emptiness is undecidable, finiteness and infiniteness are undecidable as well.

Lemma 5 shows the closure under complementation. Therefore, universality is undecidable.

To show the undecidability of regularity we consider the language $L' = L(\mathcal{M})\{\, a^n b^n \mid n \geq 1 \,\}$ for some new alphabet symbols a, b. A straightforward adaption of the proof of Lemma 2 shows that L' belongs to $\mathscr{L}_{rt}(\text{MC(1)-OCA})$. An obvious application of the pumping lemma for regular languages shows that the regularity of L' implies the finiteness of $L(\mathcal{M})$. Thus, regularity is undecidable. Similarly, the undecidability of context-freedom is shown by using the language $L_{\mathcal{M}}\{\, a^n b^n c^n \mid n \geq 1 \,\}$ for some new alphabet symbols a, b, c. □

Theorem 18. *It is undecidable for an arbitrary real-time OCA $\mathcal{M}$ whether $\mathcal{M}$ is a real-time MC(1)-OCA.*

Proof. Let $\mathcal{M}'$ be a real-time MC(1)-OCA and consider the language $L_{\mathcal{M}'} = \{\, a^{|w|} w \mid w \in L(\mathcal{M}') \,\}$ where a is some new alphabet symbol. A real-time OCA $\mathcal{M}$ accepting $L_{\mathcal{M}'}$ can be described as follows. The correct number of a- and non-a-symbols can be checked in the same way as it is done for the language $\{\, a^n b^n \mid n \geq 1 \,\}$ (see Lemma 2). The cells initially carrying non-a symbols are simulating the given real-time MC(1)-OCA $\mathcal{M}'$. Whenever the leftmost non-a-cell enters an accepting state of $\mathcal{M}'$, which can be detected by its left neighboring cell, some signal A is sent with maximum speed to the left. This signal forces all a-cells to communicate in every time step. In the rightmost cell, some signal is started which checks the correct input and enters an accepting state in the leftmost cell whenever the format is correct, the number of a- and non-a-symbols is correct, and the A-signal has reached the leftmost cell. It can be observed that the number of communication steps in each cell in the first block of a-cells depends on the length of w, if $w \in L_{\mathcal{M}'}$. Thus, $\mathcal{M}$ is an MC(1)-OCA if and only if $L_{\mathcal{M}'}$ is finite. Since finiteness is undecidable for MC(1)-OCAs due to Corollary 17, we obtain that the question whether $\mathcal{M}$ is a real-time MC(1)-OCA is undecidable as well. □

In conclusion we remark that the results can also be adapted to cellular automata where the number of proper state changes is bounded [19, 20], which answers an open question posed in [21]. Analyzing the proofs of Lemma 13, Lemma 14, and Lemma 15 with respect to the number of state changes shows that the language $L(p)$ is accepted by real-time cellular automata where the number of state changes of each cell is bounded by a constant. So, from the above we derive immediately that the problems emptiness, finiteness, infiniteness, universality, inclusion, equivalence, regularity, and context-freedom are undecidable for these automata, as well as that it is undecidable for an arbitrary real-time OCA whether its number of state changes of each cell is bounded by a constant.

References

1. Buchholz, T., Kutrib, M.: Some relations between massively parallel arrays. Parallel Comput. **23**(11) (1997) 1643–1662
2. Buchholz, T., Kutrib, M.: On time computability of functions in one-way cellular automata. Acta Inform. **35** (1998) 329–352
3. Choffrut, C., Čulik II, K.: On real-time cellular automata and trellis automata. Acta Inform. **21** (1984) 393–407
4. Čulik II, K., Yu, S.: Iterative tree automata. Theoret. Comput. Sci. **32** (1984) 227–247
5. Ibarra, O.H.: Reversal-bounded multicounter machines and their decision problems. J. ACM **25** (1978) 116–133
6. Kutrib, M.: Cellular automata – a computational point of view. In: New Developments in Formal Languages and Applications. Springer (2008) 183–227
7. Kutrib, M.: Cellular automata and language theory. In: Encyclopedia of Complexity and System Science. Springer (to appear)
8. Kutrib, M., Malcher, A.: Fast cellular automata with restricted inter-cell communication: Computational capacity. In: Theoretical Computer Science (IFIPTCS 2006). Volume 209 of IFIP, Springer (2006) 151–164
9. Kutrib, M., Malcher, A.: Fast iterative arrays with restricted inter-cell communication: Constructions and decidability. In: Mathematical Foundations of Computer Science (MFCS 2006). Volume 4162 of LNCS, Springer (2006) 634–645
10. Kutrib, M., Malcher, A.: Fast reversible language recognition using cellular automata. Inform. Comput. **206** (2008) 1142–1151
11. Malcher, A.: Descriptional complexity of cellular automata and decidability questions. J. Autom., Lang. Comb. **7** (2002) 549–560
12. Malcher, A.: On the descriptional complexity of iterative arrays. IEICE Trans. Inf. Syst. **E87-D** (2004) 721–725
13. Mazoyer, J., Terrier, V.: Signals in one-dimensional cellular automata. Theoret. Comput. Sci. **217** (1999) 53–80
14. Seidel, S.R.: Language recognition and the synchronization of cellular automata. Technical Report 79-02, Department of Computer Science, University of Iowa (1979)
15. Smith III, A.R.: Cellular automata and formal languages. In: Switching and Automata Theory (SWAT 1970), IEEE (1970) 216–224
16. Umeo, H.: Linear-time recognition of connectivity of binary images on 1-bit inter-cell communication cellular automaton. Parallel Comput. **27** (2001) 587–599
17. Umeo, H., Kamikawa, N.: A design of real-time non-regular sequence generation algorithms and their implementations on cellular automata with 1-bit inter-cell communications. Fund. Inform. **52** (2002) 257–275
18. Umeo, H., Kamikawa, N.: Real-time generation of primes by a 1-bit-communication cellular automaton. Fund. Inform. **58** (2003) 421–435
19. Vollmar, R.: On cellular automata with a finite number of state changes. Computing **3** (1981) 181–191

20. Vollmar, R.: Some remarks about the 'efficiency' of polyautomata. Internat. J. Theoret. Phys. **21** (1982) 1007–1015
21. Vollmar, R.: Zur Zustandsänderungskomplexität von Zellularautomaten. In: Beiträge zur Theorie der Polyautomaten – zweite Folge –, Braunschweig (1982) 139–151 (in German)
22. Worsch, T.: Linear time language recognition on cellular automata with restricted communication. In: LATIN 2000: Theoretical Informatics. Volume 1776 of LNCS, Springer (2000) 417–426