Natural Computing (2022) 21:3–15
https://doi.org/10.1007/s11047-020-09810-6

Iterative arrays with finite inter-cell communication

Martin Kutrib[1] · Andreas Malcher[1]

Accepted: 21 September 2020 / Published online: 24 October 2020

Abstract
Iterative arrays whose internal inter-cell communication is quantitatively restricted are investigated. The quantity of communication is measured by counting the number of uses of the links between cells. In particular, iterative arrays are studied where the maximum number of communications per cell occurring in accepting computations is drastically bounded by a constant number. Additionally, the iterative arrays have to work in realtime. We study the computational capacity of such devices. For example, a result is that a strict and dense hierarchy with respect to the constant number of communications exists. Due to their very restricted communication, the question arises whether the usually studied decidability problems such as, for example, emptiness, finiteness, inclusion, or equivalence become decidable for such devices. However, it can be shown that all such decidability questions remain undecidable even if only four communications per cell are allowed. Finally, the undecidability results are shown to hold as well for one-way and two-way cellular automata having at most four communications per cell.

Keywords Cellular automata · Iterative arrays · Communication bounds · Computational capacity · Decidability questions

1 Introduction

Devices of homogeneous, interconnected, parallel acting automata have widely been investigated from a computational capacity point of view. Multi-dimensional devices with nearest neighbor connections whose cells are finite automata are commonly called *cellular automata* (CA). The cells work synchronously at discrete time steps. If the input mode is sequential to a distinguished communication cell, such devices are called *iterative arrays* (IA).

In connection with formal language recognition one-dimensional iterative arrays have been introduced in Cole (1969), where it was shown that the language family accepted by realtime IAs forms a Boolean algebra not closed under concatenation and reversal. In Chang et al. (1987) it is shown that for every context-free grammar a two-dimensional lineartime iterative array parser exists. A realtime IA acceptor for prime numbers has been constructed in Fischer (1965). A characterization of various types of IAs in terms of restricted Turing machines and several results, especially speed-up theorems, are given in Ibarra and Palis (1985), (1988). Several more results concerning formal languages can be found, for example, in the survey (Kutrib 2009).

It is obvious that inter-cell communication is an essential resource for iterative arrays that can be measured qualitatively as well as quantitatively. In the first case, the number of different messages to be communicated by the cells is bounded by some fixed constant. IAs with this restricted inter-cell communication have been investigated in Umeo and Kamikawa (2002, 2003) with respect to the algorithmic design of sequence generation. In particular, it is shown that several infinite, non-regular sequences such as exponential or polynomial, Fibonacci, and prime sequences can be generated in realtime. In connection with language recognition and decidability questions multi-dimensional IAs and one-dimensional (one-way) CAs with restricted communication have intensively been studied in Kutrib and Malcher (2009a, 2011), Worsch (2000).

For a quantitative measure of communication in iterative arrays the number of uses of the links between cells is counted. Additionally, it is distinguished between bounds

✉ Martin Kutrib
kutrib@informatik.uni-giessen.de

Andreas Malcher
andreas.malcher@informatik.uni-giessen.de

[1] Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany

on the *sum* of all communications of an accepting computation and bounds on the *maximum number* of communications *per cell* occurring in accepting computations. There are quite a few results in the literature with respect to these measures. Results for (one-way) cellular automata may be found in Kutrib and Malcher (2010a, 2010b). In Kutrib and Malcher (2009b, 2010b) also cellular automata are investigated that are restricted with respect to the qualitative *and* the quantitative measure. The main results are in both cases hierarchy results and the undecidability of almost all commonly studied decidability questions such as emptiness, finiteness, equivalence, inclusion, regularity, and context-freeness. It should be noted that already a finite amount of communication per cell is sufficient to obtain undecidability results for cellular automata. First results on iterative arrays with restricted communication are presented in Malcher (2018) and comprise again hierarchy results as well as undecidability results for the above questions. Concerning the measure on the maximum communication per cell it has been shown that the undecidability results hold as long as at least a logarithmic number of communications per cells is allowed. Moreover, it is stated as an open question whether the undecidable questions become decidable when the allowed communication is even more restricted, namely, bounded by a constant number.

In this paper, we can answer the latter question negatively even if only four communications per cell are allowed. In addition, we establish a strict and dense hierarchy with respect to the constant number of communications. The paper is organized as follows. In the next section, we present some basic notions and definitions, introduce the classes of max communication bounded iterative arrays, and give an illustrative example. Then, in Sect. 3 we show that for every $k \geq 2$, IAs with at most $k+1$ communications per cell are more powerful than devices with at most k communications per cell. For $k \in \{0,1,2\}$ it turns out that devices with at most k communications per cell can accept regular languages only. Section 4 is devoted to showing the undecidability of the usually studied decidability questions for IAs working in realtime and having at most four communications per cell. This is done by a reduction of the halting problem for counter machines (Minsky 1961). The ideas and constructions of Sect. 4 are applied in Sect. 5 where the known undecidability results for realtime one-way and two-way cellular automata with max bounded communication can essentially be improved to hold for machines with at most four communications per cell as well.

We would like to note that a preliminary version of this paper has been presented in Kutrib and Malcher (2019). Here, we have provided the full proofs of the results in Sect. 3. Moreover, the undecidability results in Kutrib and Malcher (2019) have been obtained by a reduction of Hilbert's tenth problem. Here, a different reduction is chosen which leads to more precise results in Sect. 4. Moreover, the results presented in Sect. 5 are new.

2 Definitions and preliminaries

We denote the non-negative integers by $\mathbb{N}$. Let Σ denote a finite set of letters. Then we write Σ^* for the *set of all finite words* (strings) consisting of letters from Σ. The *empty word* is denoted by λ, and we set $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. A subset of Σ^* is called a *language* over Σ. A language L over some alphabet $\{a_1, a_2, \ldots, a_k\}$ is said to be *letter-bounded*, if $L \subseteq a_1^* a_2^* \cdots a_k^*$. For the *reversal of a word* w we write w^R and for its *length* we write $|w|$. In general, we use $\subseteq$ for *inclusions* and $\subset$ for *strict inclusions*.

A one-dimensional iterative array is a linear, semi-infinite array of identical deterministic finite state machines, sometimes called cells. Except for the leftmost cell each one is connected to its both nearest neighbors (see Fig. 1). For convenience we identify the cells by their coordinates, that is, by non-negative integers. The distinguished leftmost cell at the origin is connected to its right neighbor and, additionally, equipped with a one-way read-only input tape. At the outset of a computation the input is written on the input tape with an infinite number of end-of-input symbols to the right, and all cells are in the so-called quiescent state. The finite state machines work synchronously at discrete time steps. The state transition of all cells but the communication cell depends on the current state of the cell itself 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 state transition of the communication cell additionally depends on the input symbol to be read next. The head of the one-way input tape is moved to the right in each step. With an eye towards recognition problems the machines have no extra output tape but the states are partitioned into accepting and rejecting states.

Formally, an *iterative array* (IA) denotes a system $\langle S, F, A, B, \triangledown, s_0, b_l, b_r, \delta, \delta_0 \rangle$, where S is the finite, nonempty set of *cell states*, $F \subseteq S$ is the set of *accepting states*, A is the finite set of *input symbols*, B is the finite set of *communication symbols*, $\triangledown \notin A$ is the *end-of-input symbol*, $s_0 \in S$ is the *quiescent state*, $b_l, b_r : S \to B \cup \{\perp\}$

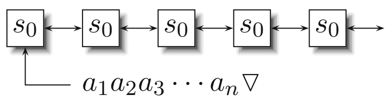

Fig. 1 Initial configuration of an iterative array

are *communication functions* which determine the information to be sent to the left and right neighbors, where $\perp$ means nothing to send and $b_l(s_0) = b_r(s_0) = \perp$, $\delta : (B \cup \{\perp\}) \times S \times (B \cup \{\perp\}) \to S$ is the *local transition function for all but the communication cell* satisfying $\delta(\perp, s_0, \perp) = s_0$, and $\delta_0 : (A \cup \{\triangledown\}) \times S \times (B \cup \{\perp\}) \to S$ is the *local transition function for the communication cell*.

Let M be an IA. A configuration of M at some time $t \geq 0$ is a description of its global state which is a pair (w_t, c_t), where $w_t \in A^*$ is the remaining input sequence and $c_t : \mathbb{N} \to S$ is a mapping that maps the single cells to their current states. The configuration (w_0, c_0) at time 0 is defined by the input word w_0 and the mapping c_0 that assigns the quiescent state to all cells, while subsequent configurations are chosen according to the global transition function Δ that is induced by δ and δ_0 as follows: Let (w_t, c_t), $t \geq 0$, be a configuration. Then its successor configuration $(w_{t+1}, c_{t+1}) = \Delta(w_t, c_t)$ is as follows.

$$c_{t+1}(i) = \delta(b_r(c_t(i-1)), c_t(i), b_l(c_t(i+1)))$$

for all $i \geq 1$, and $c_{t+1}(0) = \delta_0(a, c_t(0), b_l(c_t(1)))$, where $a = \triangledown$ and $w_{t+1} = \lambda$ if $w_t = \lambda$, as well as $a = a_1$ and $w_{t+1} = a_2 a_3 \cdots a_n$ if $w_t = a_1 a_2 \cdots a_n$.

We remark that we obtain the classical definition of IA, if we set $B = S$ and $b_l(s) = b_r(s) = s$ for all $s \in S$.

An input w is accepted by an IA M if at some time i during the course of its computation the communication cell enters an accepting state. The *language accepted by* M is denoted by $L(M)$. Let $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n + 1$ be a mapping. If all $w \in L(M)$ are accepted with at most $t(|w|)$ time steps, then M and $L(M)$ are said to be of time complexity t.

The family of all languages which are accepted by some type of device X with time complexity t is denoted by $\mathscr{L}_t(X)$. If t is the function $n + 1$, acceptance is said to be in *realtime* and we write $\mathscr{L}_{rt}(X)$. Since for nontrivial computations an IA has to read at least one end-of-input symbol, realtime has to be defined as $(n+1)$-time.

In the following we study the impact of communication in iterative arrays. 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

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

For computations we now consider the maximal number of communications between two cells. Let $c_0, c_1, \ldots, c_{t(|w|)}$ be the sequence of configurations computed on input w by some iterative array with time complexity $t(n)$, that is, the *computation on* w. Then we define

$$\mathrm{mcom}(w) = \max\{ \mathrm{com}(i, t(|w|)) \mid 0 \leq i \leq t(|w|) - 1 \}.$$

Let $f : \mathbb{N} \to \mathbb{N}$ be a mapping. If all $w \in L(M)$ are accepted with computations where $\mathrm{mcom}(w) \leq f(|w|)$, then M is said to be *max communication bounded by* f. We denote the class of IA that are max communication bounded by some function f by MC(f)-IA. In addition, we use the notation *const* for functions from $O(1)$.

To illustrate the definitions we start with an example. In the next section it turns out that the non-regular language $\{ a^n b^n \mid n \geq 1 \}$ is accepted by some realtime iterative array using at most three communications per inter-cell link. The following example reveals that only five communications per inter-cell link are sufficient to accept the non-semilinear and, hence, non-context-free language $L = \{ a^{3n+2\lfloor\sqrt{n}\rfloor} b^{2n} \mid n \geq 1 \}$ in realtime. Moreover, L is a subset of a^*b^* and, hence, a letter-bounded language.

Example 1 The language L belongs to $\mathscr{L}_{rt}(\mathrm{MC}(5)\text{-IA})$.

The basic idea is to use the construction given in Mazoyer and Terrier (1999) where a cellular automaton is described such that the nth cell enters a designated state s exactly at time step $2n + \lfloor\sqrt{n}\rfloor$. We basically implement this construction, but realize it with speed 1/2 in contrast to the construction in Mazoyer and Terrier (1999). Hence, the nth cell enters a designated state exactly at time step $4n + 2\lfloor\sqrt{n}\rfloor$. When the communication cell reads the first b, it sends a signal with maximum speed to the right which arrives in the nth cell exactly at the moment when state s should be entered. In this case, another signal is sent with maximum speed to the left and the input is accepted if the latter signal reaches the communication cell when the end-of-input symbol is read. Thus, L is accepted by a realtime IA. Moreover, the construction given in Mazoyer and Terrier (1999) needs three right-moving signals. Thus, four right-moving signals and one left-moving signal are sufficient to accept L. This shows that the IA constructed is a realtime MC(5)-IA. □

3 $k+1$ communications are better than k

This section is devoted to studying the impact of the precise finite number k of communications between cells. It turns out that this number in fact matters unless it is very small. That is, we will obtain an infinite strict hierarchy for $k \geq 2$, whereas the families of languages accepted with 0, 1, and 2 communications per inter-cell link coincide with the regular languages. We start at the bottom of the hierarchy.

Proposition 1 *The three language families $\mathscr{L}_{rt}(MC(0)\text{-}IA)$, $\mathscr{L}_{rt}(MC(1)\text{-}IA)$, and $\mathscr{L}_{rt}(MC(2)\text{-}IA)$ coincide with the family of regular languages.*

Proof The proof is trivial for MC(0)-IAs. In this case the iterative array has the computational capacity of the communication cell, that is, of a deterministic finite automaton. Similarly, the proof is obvious for MC(1)-IAs. The sole communication on the inter-cell link between the communication cell and its neighbor sends some information to the right, but this information can never come back to the communication cell which, thus, is not affected by the communication at all. We conclude that the computational capacity also for MC(1)-IAs is that of deterministic finite automata.

Let us now consider MC(2)-IAs. Let the first communication on the inter-cell link between the communication cell and its neighbor take place at time step $t \geq 0$. Before, all cells to the right of the communication cell are quiescent. So, the information transmitted by the communication causes the quiescent array to perform some computation and possibly to transmit some information back to the communication cell. The computation performed by the array to the right of the communication cell only depends on the information transmitted during the first communication. However, these finitely many cases can be precomputed. Implementing the transition function of the communication cell so that it simulates the computations of the array in some state register allows to safely remove the first communication. In this way we obtain an equivalent MC(1)-IA that accepts regular languages only. Needless to say that a second communication from left to right is useless and can be omitted as well. □

The witnesses for the hierarchy are languages whose words are repetitions of unary blocks of the same size but with alternating symbols. For $i \geq 2$ define

$$L_{\mathrm{h}i} = \begin{cases} \{\, (a^n b^n)^{\frac{i}{2}} \mid n \geq 1 \,\} & \text{if } i \text{ is even} \\ \{\, (a^n b^n)^{\lfloor \frac{i}{2} \rfloor} a^n \mid n \geq 1 \,\} & \text{if } i \text{ is odd} \end{cases}.$$

Lemma 1 *For $i \geq 2$, the language $L_{\mathrm{h}i}$ is accepted by some MC$(i+1)$-IA.*

Proof Any word from $L_{\mathrm{h}i}$ is the concatenation of i unary blocks of the same size but with alternating symbols. The correct format of an input, namely to be of the form $(a^+b^+)^+a^*$, can be checked by the communication cell. The basic idea to verify the correct lengths of the blocks is as follows (see Fig. 2). Initially, the communication cell sends a signal S_0 with speed 1/3 to the right. That is, the signal resides in the first three configurations in the communication cell, is then sent to the next cell where it resides in the following three configurations, and so on. When the

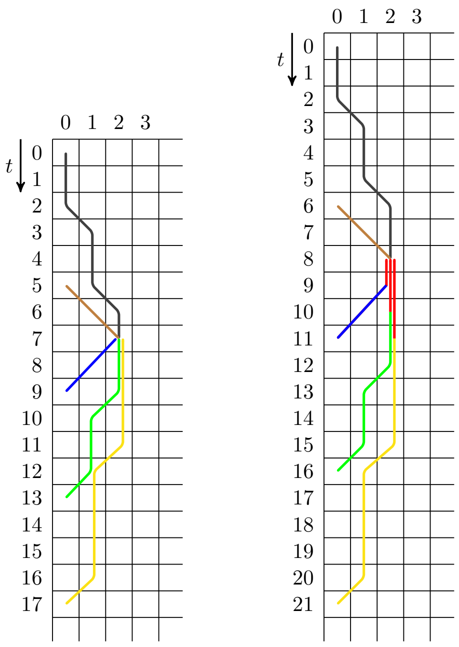

Fig. 2 Two example computations for the constructions in the proof of Lemma 1. For input $a^4b^4a^4b^4$ the block size is even (left) and for input $a^5b^5a^5b^5$ it is odd (right). The red lines on the right indicate the delays of the signals R_j. (Color figure online)

first symbol of the second block has been read, the communication cell sets up another signal S_1 that runs with speed 1 to the right. When the latter signal catches up with the former one, both are canceled and $i-1$ signals R_j with speeds $1/(2j-1)$, for $1 \leq j \leq i-1$, are sent to the left. If these signals arrive in the communication cell at the time steps in which new blocks (or the end-of-input symbol) appear in the input, the lengths of all blocks are correct.

More precisely, for $x \geq 0$, signal S_0 resides at times $3x$, $3x+1$, and $3x+2$ in cell x. Signal S_1 resides at time $n+1$ in cell 0 and at time $n+1+x$ in cell x. If n is even, signal S_1 arrives at time $n+1+\frac{n}{2} = 3\frac{n}{2}+1$ in cell $\frac{n}{2}$ where S_0 resides at times $3\frac{n}{2}$, $3\frac{n}{2}+1$, and $3\frac{n}{2}+2$. Therefore, signals S_0 and S_1 meet in cell $\lfloor\frac{n}{2}\rfloor$. If n is odd, signal S_1 arrives at time $n+1+\lfloor\frac{n}{2}\rfloor = 3\lfloor\frac{n}{2}\rfloor+2$ in cell $\lfloor\frac{n}{2}\rfloor$ where S_0 resides at times $3\lfloor\frac{n}{2}\rfloor$, $3\lfloor\frac{n}{2}\rfloor+1$, and $3\lfloor\frac{n}{2}\rfloor+2$. Therefore, signals S_0 and S_1 meet in cell $\lfloor\frac{n}{2}\rfloor$ also in this case.

Now, for the sending of the signals R_j the parity of n is distinguished. (The parity of n is straightforwardly checked by the communication cell and can be sent together with signal S_1). If n is even, the signals R_j are set up by cell $\frac{n}{2}$ immediately at time $3\frac{n}{2}+1$. So, signal R_j arrives at the communication cell at time

$$3\frac{n}{2}+1+(2j-1)\frac{n}{2} = 2(j+1)\frac{n}{2}+1 = (j+1)n+1.$$

Since $1 \leq j \leq i-1$, these are the times $2n+1, 3n+1, \ldots, i \cdot n+1$ at which new blocks (or the end-of-input symbol) appear in the input.

If n is odd, the signals R_j are set up by cell $\lfloor \frac{n}{2} \rfloor$ with a delay of j time steps at time $3\lfloor \frac{n}{2} \rfloor + 2 + j$, respectively. So, each signal R_j arrives at the communication cell at time

$$\begin{aligned} &3\left\lfloor \frac{n}{2} \right\rfloor + 2 + j + (2j-1)\left\lfloor \frac{n}{2} \right\rfloor \\ &\quad = 2(j+1)\frac{n}{2} - \frac{2(j+1)}{2} + 2 + j = (j+1)n + 1. \end{aligned}$$

As in the first case, since $1 \leq j \leq i-1$, these are the times $2n+1, 3n+1, \ldots, i \cdot n+1$ at which new blocks (or the end-of-input symbol) appear in the input.

So, the IA constructed accepts language L_{hi} in realtime. Moreover, each of the altogether $i+1$ signals takes at most one communication step per inter-cell link and, thus, the IA is an MC$(i+1)$-IA. □

Theorem 1 *For $i \geq 2$, the family $\mathscr{L}_{rt}(MC(i)\text{-}IA)$ is strictly included in the family $\mathscr{L}_{rt}(MC(i+1)\text{-}IA)$*

Proof For $i \geq 2$, the witness language L_{hi} is used to show the strict inclusion $\mathscr{L}_{rt}(\mathrm{MC}(i)\text{-IA}) \subset \mathscr{L}_{rt}(\mathrm{MC}(i+1)\text{-IA})$. By Lemma 1, language L_{hi} is accepted by some MC$(i+1)$-IA in realtime. So, it remains to be shown that L_{hi} is not accepted by any MC(i)-IA in realtime.

Let $M = \langle S, F, A, B, \triangledown, s_0, b_l, b_r, \delta, \delta_0 \rangle$ be an iterative array accepting L_{hi} in realtime. We consider the communications on the inter-cell link between the communication cell and its neighbor. For clearer writing, we represent a configuration by a pair $(w, s\hat{c})$, where $w \in A^*$ is the remaining input sequence as usual, $s \in S$ is the state of the communication cell, and $\hat{c}$ is a mapping that maps cells $j \geq 1$ to their current states.

A key observation is that on unary inputs long enough the communication cell will run into state cycles unless a communication takes place. So, let $w \in L_{hi}$ be an accepted word whose block length n is long enough.

Assume that no communication takes place while processing the first $|S|$ symbols a from the first block. Recall that by definition we have $b_l(s_0) = b_r(s_0) = \perp$. Let $w = a^n v$ with the first letter in v being b. Then on processing the input prefix $a^{|S|}$ the communication cell necessarily enters some state at least twice. That is, there are $0 \leq p_1$, $1 \leq p_2$ with $p_1 + p_2 \leq |S|$ such that $\Delta^{p_1}(a^n v, s_0\hat{c}_0) = (a^{n-p_1} v, s_1\hat{c}_0)$ and $\Delta^{p_2}(a^{n-p_1} v, s_1\hat{c}_0) = (a^{n-p_1-p_2} v, s_1\hat{c}_0)$. That is, there is a state cycle of length p_2 leading from state s_1 to state s_1. So, the input $a^{n+p_2} v \notin L_{hi}$ is accepted as well. From the contradiction it follows that there is at least one communication during the first $|S|$ time steps.

Next, we consider the sub-computations that process the last $|S|$ symbols of a block and the first $|S|^2 + |S|$ symbols of the following block. Without loss of generality, let this factor be $a^{|S|} b^{|S|^2+|S|}$, and so $w = u a^{|S|} b^{|S|^2+|S|} v$. Assume that no communication takes place (on the inter-cell link between the communication cell and its neighbor) while processing this factor. Let

$$\Delta^{|u|}(u a^{|S|} b^{|S|^2+|S|} v, s_0\hat{c}_0) = (a^{|S|} b^{|S|^2+|S|} v, s\hat{c}).$$

Then there are $0 \leq p_1$, $1 \leq p_2$ with $p_1 + p_2 \leq |S|$ such that

$$\begin{aligned} &\Delta^{p_1}(a^{|S|} b^{|S|^2+|S|} v, s\hat{c}) = (a^{|S|-p_1} b^{|S|^2+|S|} v, s_1\hat{c}_1) \text{ and} \\ &\Delta^{p_2}(a^{|S|-p_1} b^{|S|^2+|S|} v, s_1\hat{c}_1) = (a^{|S|-p_1-p_2} b^{|S|^2+|S|} v, s_1\hat{c}_2). \end{aligned}$$

That is, there is a state cycle of length p_2 leading from state s_1 to state s_1. Continuing the computation without communication yields the existence of $0 \leq p_3$, $1 \leq p_4 \leq |S|$ with $|S| \leq p_1 + p_2 + p_3$ and $p_1 + p_2 + p_3 + p_4 \leq 2|S|$ such that

$$\begin{aligned} &\Delta^{p_3}(a^{|S|-p_1-p_2} b^{|S|^2+|S|} v, s_1\hat{c}_2) = (b^{|S|^2+2|S|-p_1-p_2-p_3} v, s_2\hat{c}_3), \\ &\Delta^{p_4}(b^{|S|^2+2|S|-p_1-p_2-p_3} v, s_2\hat{c}_3) \\ &\quad = (b^{|S|^2+2|S|-p_1-p_2-p_3-p_4} v, s_2\hat{c}_4), \text{ and} \\ &\Delta^{p_2p_4}(b^{|S|^2+2|S|-p_1-p_2-p_3-p_4} v, s_2\hat{c}_4) \\ &\quad = (b^{|S|^2+2|S|-p_1-p_2-p_3-p_4-p_2p_4} v, s_2\hat{c}_5). \end{aligned}$$

That is, there is a state cycle of length p_4 leading from state s_2 to state s_2. Since there are no communications on the factor $a^{|S|} b^{|S|^2+|S|}$, we can replace this factor by the factor $a^{|S|+p_2p_4} b^{|S|^2+|S|-p_2p_4}$ of the same length, and the configurations to the right of the communication cell develop exactly as before. We obtain the sub-computations

$$\begin{aligned} &\Delta^{p_1}(a^{|S|+p_2p_4} b^{|S|^2+|S|-p_2p_4} v, s\hat{c}) \\ &\quad = (a^{|S|+p_2p_4-p_1} b^{|S|^2+|S|-p_2p_4} v, s_1\hat{c}_1) \text{ and} \\ &\Delta^{p_2}(a^{|S|+p_2p_4-p_1} b^{|S|^2+|S|-p_2p_4} v, s_1\hat{c}_1) \\ &\quad = (a^{|S|+p_2p_4-p_1-p_2} b^{|S|^2+|S|-p_2p_4} v, s_1\hat{c}_2). \end{aligned}$$

Since the communication cell runs through the state cycle of length p_2 leading from state s_1 to state s_1 and there is no communication, the sub-computation continues as

$$\begin{aligned} &\Delta^{p_2p_4}(a^{|S|+p_2p_4-p_1-p_2} b^{|S|^2+|S|-p_2p_4} v, s_1\hat{c}_2) \\ &\quad = (a^{|S|-p_1-p_2} b^{|S|^2+|S|-p_2p_4} v, s_1\hat{c}_6), \\ &\Delta^{p_3}(a^{|S|-p_1-p_2} b^{|S|^2+|S|-p_2p_4} v, s_1\hat{c}_6) \\ &\quad = (b^{|S|^2+2|S|-p_2p_4-p_1-p_2-p_3} v, s_2\hat{c}_7), \\ &\Delta^{p_4}(b^{|S|^2+2|S|-p_2p_4-p_1-p_2-p_3} v, s_2\hat{c}_7) \\ &\quad = (b^{|S|^2+2|S|-p_2p_4-p_1-p_2-p_3-p_4} v, s_2\hat{c}_5). \end{aligned}$$

For the last equation, note that the length of the replaced factor is the same as of the original factor. That, is the number of steps is the same on both factors. Moreover, since there is no communication of the communication cell, during these steps the right part of the configuration develops as before. So, the configuration of the right part is finally $\hat{c}_5$.

Since the configurations after processing the original factor and its replacement are identical, the input word with replaced factor not belonging to L_{hi} is accepted as well. From the contradiction it follows that there is at least one communication while processing the factor $a^{|S|}b^{|S|^2+|S|}$.

Now we turn to the end of the computation. Assume that no communication takes place while processing the last $2|S|$ symbols from the last block, say these are symbols b. Let $w = ub^n$ with the last letter in u being an a. Then on processing the input suffix $b^{2|S|}$ the communication cell necessarily enters some state at least twice. Let $\Delta^{|u|+n-2|S|}(ub^n, s_0\hat{c}_0) = (b^{2|S|}, s\hat{c})$. Then there are $0 \le p_1$, $1 \le p_2$ with $p_1 + p_2 \le |S|$ such that $\Delta^{p_1}(b^{2|S|}, s\hat{c}) = (b^{2|S|-p_1}, s_1\hat{c}_1)$ and $\Delta^{p_2}(b^{2|S|-p_1}, s_1\hat{c}_1) = (b^{2|S|-p_1-p_2}, s_1\hat{c}_2)$. Since the input is accepted, the communication cell enters an accepting state in this state cycle of length p_2 (which cannot be left until the end of the computation) or before. This implies that input ub^{n-p_2} is accepted as well. From the contradiction it follows that there is at least one communication during the last $2|S|$ time steps.

Altogether we have seen that the communication cell of M necessarily communicates with its neighbor, or vice versa, during the first $|S|$ steps, during the last $2|S|$ steps, and on the factors $a^{|S|}b^{|S|^2+|S|}$ and $b^{|S|}a^{|S|^2+|S|}$ that include the block borders. Since there are $i-1$ such block borders, in total, there are at least $i+1$ communications for n long enough. □

4 Undecidability results for realtime $\text{MC}(\text{const}) - \text{IAs}$

It is shown in Kutrib and Malcher (2019) by a reduction of Hilbert's tenth problem that emptiness is undecidable for realtime MC(*const*)-IAs. However, this proof does not provide a precise constant, since the number of communications per cell in the construction depends on the polynomial given in Hilbert's tenth problem. In this section, we will improve this undecidability result in two directions. Namely, we show that emptiness is already undecidable for realtime IAs with at most four communications per cell which, in addition, work on a unary input. The basic idea is to construct a realtime MC(4)-IA that simulates a two-counter machine C on empty input and accepts some input if and only if C halts on empty input. Since it is a well-known fact that it is undecidable whether or not a deterministic counter machine having two counters that are initially zero and starting with empty input will eventually halt (Minsky 1961), the simulation implies the undecidability of the emptiness problem for realtime MC(4)-IAs. In a first step, we will construct a realtime IA in the following lemma which simulates a given two-counter machine and accepts some input if and only if the given counter machine halts on empty input. In a second step, we will refine the construction so that the realtime IA will be a realtime MC(4)-IA which gives the desired undecidability result.

Lemma 2 *Let C be a two-counter machine. Then, a unary realtime IA M can be constructed such that $L(M)$ is not empty if and only if C halts on empty input.*

Proof For the simulation of the two-counter machine by an IA we basically follow the construction given in Carton et al. (2018), where a general construction of a lineartime quasi-acyclic one-way cellular automaton is given that simulates a given k-counter machine on a given input. However, here we have the additional restriction that a two-counter machine on empty input has to be simulated and that we have to ensure that the simulating machine is indeed an MC(4)-IA working in realtime.

Let $C = \langle S, s_0, \delta \rangle$ be a two-counter machine, where S denotes the state set, s_0 is the initial state, and $\delta : S \times \{0,1\}^2 \to S \times \{+1, 0, -1\}^2$ is the transition function. Here, 0 denotes an empty counter and 1 denotes a positive counter. Without loss of generality we may assume that C accepts by halting and makes at least one move. Now, we construct in a first step a conventional IA on unary input that simulates C and accepts if and only if C halts (see the following Example 2 and its computation depicted in Fig. 3). In a second step, we will later discuss how to refine the construction so that the IA constructed is indeed a realtime MC(4)-IA. The basic idea following (Carton et al. 2018) is to encode the current value of each counter in the first two tracks of the IA in a *"vertical way"*. This means that in cell i the sequence of entries over time in one of those tracks encodes the value of the corresponding counter of C at time $i+1$. In detail, the sequence of states will be a substring of the form $\top 1^* 0 \bot \#^+$, where the number of 1's denotes the current value of the counter. Furthermore, we will store the current state of the counter machine as well as the update information for both counters in additional tracks.

Formally, we define the IA $M = \langle S', \{f'\}, \{a\}, \triangledown, s'_0, \delta', \delta'_0 \rangle$ with state set

t							
0	s'_0	s'_0	s'_0	s'_0	s'_0	s'_0	s'_0
1	s'_1	s'_0	s'_0	s'_0	s'_0	s'_0	s'_0
2	$\top\ \top\ s_1 + +$	s'_0	s'_0	s'_0	s'_0	s'_0	s'_0
3	$1\ 1\ s_1 + +$	s'_0	s'_0	s'_0	s'_0	s'_0	s'_0
4	$0\ 0\ s_1 + +$	$\top\ \top\ s_2 + 0$	s'_0	s'_0	s'_0	s'_0	s'_0
5	$\bot\ \bot\ s_1 + +$	$1\ 1\ s_2 + 0$	s'_0	s'_0	s'_0	s'_0	s'_0
6	$\#\ \#\ s_1 + +$	$1\ 0\ s_2 + 0$	$\top\ \top\ s_3 + -$	s'_0	s'_0	s'_0	s'_0
7	$\#\ \#\ s_1 + +$	$0\ \bot\ s_2 + 0$	$1\ 0\ s_3 + -$	s'_0	s'_0	s'_0	s'_0
8	$\#\ \#\ s_1 + +$	$\bot\ \#\ s_2 + 0$	$1\ \bot\ s_3 + -$	$\top\ \top\ s_3 - 0$	s'_0	s'_0	s'_0
9	$\#\ \#\ s_1 + +$	$\#\ \#\ s_2 + 0$	$1\ \#\ s_3 + -$	$1\ 0\ s_3 - 0$	s'_0	s'_0	s'_0
10	$\#\ \#\ s_1 + +$	$\#\ \#\ s_2 + 0$	$0\ \#\ s_3 + -$	$1\ \bot\ s_3 - 0$	$\top\ \top\ s_3 - 0$	s'_0	s'_0
11	$\#\ \#\ s_1 + +$	$\#\ \#\ s_2 + 0$	$\bot\ \#\ s_3 + -$	$0\ \#\ s_3 - 0$	$1\ 0\ s_3 - 0$	s'_0	s'_0
12	$\#\ \#\ s_1 + +$	$\#\ \#\ s_2 + 0$	$\#\ \#\ s_3 + -$	$\bot\ \#\ s_3 - 0$	$0\ \bot\ s_3 - 0$	$\top\ \top\ s_3 - 0$	s'_0
13	$\#\ \#\ s_1 + +$	$\#\ \#\ s_2 + 0$	$\#\ \#\ s_3 + -$	$\#\ \#\ s_3 - 0$	$\bot\ \#\ s_3 - 0$	$0\ 0\ s_3 - 0$	s'_0
14	$\#\ \#\ s_1 + +$	$\#\ \#\ s_2 + 0$	$\#\ \#\ s_3 + -$	$\#\ \#\ s_3 - 0$	$\#\ \#\ s_3 - 0$	$\bot\ \bot\ s_3 - 0$	f'
15	$\#\ \#\ s_1 + +$	$\#\ \#\ s_2 + 0$	$\#\ \#\ s_3 + -$	$\#\ \#\ s_3 - 0$	$\#\ \#\ s_3 - 0$	f'	s'_2
16	$\#\ \#\ s_1 + +$	$\#\ \#\ s_2 + 0$	$\#\ \#\ s_3 + -$	$\#\ \#\ s_3 - 0$	f'	s'_2	s'_2
17	$\#\ \#\ s_1 + +$	$\#\ \#\ s_2 + 0$	$\#\ \#\ s_3 + -$	f'	s'_2	s'_2	s'_2
18	$\#\ \#\ s_1 + +$	$\#\ \#\ s_2 + 0$	f'	s'_2	s'_2	s'_2	s'_2
19	$\#\ \#\ s_1 + +$	f'	s'_2	s'_2	s'_2	s'_2	s'_2
20	f'	s'_2	s'_2	s'_2	s'_2	s'_2	s'_2

Fig. 3 Example computation on an iterative array simulating the counter machine from Example 2 according to the construction given in Lemma 2

$$S' = \{s'_0, s'_1, s'_2, f'\} \cup \{\top, \bot, \#, 0, 1\}^2 \times S \times \{+1, 0, -1\}^2.$$

For the definition of the transition functions δ'_0 and δ' we again follow the ideas given in Carton et al. (2018). To initialize the computation in the communication cell, we set

$$\delta'_0(a, s'_0, s'_0) = s'_1 \text{ and } \delta'_0(a, s'_1, s'_0) = (\top, \top, s_1, m_1, m_2),$$

if $\delta(s_0, 0, 0) = (s_1, m_1, m_2)$ with $s_1 \in S$ and $m_1, m_2 \in \{+1, 0, -1\}$. To update the counters in the communication cell, we add the following transitions where $s \in S$, $q \in S'$, $c_1, c_2 \in \{0, 1\}$, and $m_1, m_2 \in \{+1, 0, -1\}$. For $i \in \{1, 2\}$ we define c'_i as follows.

$$\delta'_0(a, (\top, \top, s, m_1, m_2), q) = (c_1, c_2, s, m_1, m_2)$$
$$\text{with } c_i = \begin{cases} 1, \text{if } m_i = 1, \\ 0, \text{otherwise,} \end{cases}$$

and

$$\delta'_0(a, (c_1, c_2, s, m_1, m_2), q) = (c'_1, c'_2, s, m_1, m_2)$$
$$\text{with } c'_i = \begin{cases} 0, \text{if } c_i = 1, \\ \bot, \text{if } c_i = 0, \\ \#, \text{if } c_i \in \{\bot, \#\}. \end{cases}$$

The transitions for the other cells are as follows. To start the computation, we first implement a delay using the transitions

$$\delta'(q, s'_0, s'_0) = s'_0, \quad \text{for } q \in \{s'_0, s'_1\} \cup \{\top\}^2 \times S \times \{+1, 0, -1\}^2,$$

and initialize the computation when the left neighbor initializes its counter values by changing from $\top$ to 1 or 0. Let us first assume that the computation is initialized with a non-halting configuration. For $s, s' \in S$, $c_1, c_2 \in \{0, 1\}$, and $m_1, m_2, m'_1, m'_2 \in \{+1, 0, -1\}$ we set

$$\delta'((c_1, c_2, s, m_1, m_2), s'_0, s'_0) = (\top, \top, s', m'_1, m'_2),$$
$$\text{if } \delta(s, c_1, c_2) = (s', m'_1, m'_2).$$

To update the counters we use the transitions

$$\delta'((c_1, c_2, s, m_1, m_2), (c'_1, c'_2, s', m'_1, m'_2), q)$$
$$= (c''_1, c''_2, s', m'_1, m'_2),$$

with $c_1, c_2, c'_1, c'_2 \in \{\top, 0, 1, \bot, \#\}$, $m_1, m_2, m'_1, m'_2 \in \{+1, 0, -1\}$, $s, s' \in S$, and $q \in S' \setminus \{f'\}$. For $i \in \{1, 2\}$ we define c''_i as follows.

If $c'_i \in \{\top, 1\}$, then we set

$$c''_i = \begin{cases} 1, & \text{if } c_i = 1, \\ 1, & \text{if } c_i = 0 \text{ and } m'_i \geq 0, \\ 1, & \text{if } c_i = \bot \text{ and } m'_i = 1, \\ 0, & \text{otherwise.} \end{cases}$$

If $c'_i \in \{0, \bot, \#\}$, we set

$$c''_i = \begin{cases} \bot, & \text{if } c'_i = 0, \\ \#, & \text{if } c'_i \in \{\bot, \#\}. \end{cases}$$

As soon as the computation would be initialized with a halting configuration of the counter machine, the accepting state f' is entered and is sent with maximum speed to the left. Hence, we set for $s \in S$, $c_1, c_2 \in \{0,1\}$, and $m_1, m_2 \in \{+1, 0, -1\}$

$$\delta'((c_1, c_2, s, m_1, m_2), s'_0, s'_0) = f', \text{ if } \delta(s, c_1, c_2) \text{ is undefined.}$$

Furthermore, $\delta'(q, q', f') = f'$ for all $q, q' \in S'$ with $q' \neq f'$, and $\delta_0(\nabla, q, f') = f'$, for all $q \in S' \setminus \{s'_2\}$. Finally, $\delta'(q, f', q') = s'_2$, for all $q, q' \in S'$, and $\delta_0(\nabla, q, q') = s'_2$, for all $q \in S'$ and $q' \in S' \setminus \{f'\}$. By this construction, M is a unary realtime IA and accepts a non-empty language if and only if C is halting on empty input. □

Example 2 As an example for the construction we consider the following two-counter machine $C = \langle\{s_0, s_1, s_2, s_3\}, s_0, \delta\rangle$ with the transitions

$$\begin{aligned}\delta(s_0, 0, 0) &= (s_1, +1, +1),\\ \delta(s_1, 1, 1) &= (s_2, +1, 0),\\ \delta(s_2, 1, 1) &= (s_3, +1, -1), \text{ and}\\ \delta(s_3, 1, 0) &= (s_3, -1, 0).\end{aligned}$$

The remaining transitions are undefined which means that C halts and accepts if a configuration is entered where no transition is defined. The accepting computation of C on empty input is then as follows:

$$\begin{aligned}&(s_0, 0, 0) \vdash (s_1, 1, 1) \vdash (s_2, 2, 1) \vdash (s_3, 3, 0)\\ &\quad \vdash (s_3, 2, 0) \vdash (s_3, 1, 0) \vdash (s_3, 0, 0).\end{aligned}$$

The simulation of this computation by a realtime IA M is depicted in Fig. 3. Since the computation of C on empty input is halting, M accepts some input, in this case a^{19}. Apart from the quiescent state s'_0, the accepting state f', and the auxiliary states s'_1 and s'_2, the states of M are quintuples, where in the first (second) track the first (second) counter of C is simulated. The third track contains the current state of C and in track four (five) the next action on counter one (two) is stored. It can be observed that the configuration at time $i+1$ is stored in cell i. For example, the configuration $(s_2, 2, 1)$ at time two is stored in cell 1: track 3 contains state s_2, track 1 contains (vertically read) the substring $\top 110 \bot$# which is an encoding of two, and track 2 contains the substring $\top 10 \bot$# which is an encoding of one. □

Our next task is to refine the construction of Lemma 2 to obtain a realtime MC(4)-IA. To this end, we consider again the example computation in Fig. 3 and first analyze the flow of information from left to right. It can be observed that the simulation in cell i is started at time step $2(i+1)$ for $i \geq 1$. This starting point is defined by the first time when tracks 1 and 2 of the left neighboring cell of cell i change from $\top$ to 1 or 0. This leads to one communication per cell. The corresponding cells sending this information are depicted with solid gray background in Fig. 3. Next, in tracks 1 and 2 the following holds: whenever state 0 is entered at time t, the state at time $t+1$ is $\bot$, and the state at time $t' > t+1$ is #. This means that no communication from left to right is necessary to compute the next states in some track as soon as the information that track 1 or track 2 enters 0 is communicated. In addition, it can be assumed without further communication that, starting with the initialization, track 1 and track 2 of the left neighbor are in state 1 up to the moment when state 0 is communicated. Hence, the transition function can suitably be modified such that the only communication needed from the left neighbor for the state changes in track 1 or track 2 is the information when track 1 or track 2 enter state 0. This gives at most two more communications per cell and the corresponding cells are depicted with a crosshatched background. Finally, we have one additional communication per cell from right to left for communicating the accepting state f' to the communication cell in case of a halting computation of the counter machine. Such cells are depicted with a lined background. Altogether, we have at most four communications per cell. These considerations and the fact that the halting problem is undecidable for two-counter machines on empty input (Minsky 1961), immediately yield the following undecidability result.

Theorem 2 *Emptiness is undecidable for unary realtime MC(i)-IAs, if $i \geq 4$.*

The undecidability of the emptiness problem and its proof can be used to show the undecidability of the following problems.

Theorem 3 *Finiteness, infiniteness, inclusion, equivalence are undecidable for unary realtime MC(i)-IAs, if $i \geq 4$. Regularity is undecidable for realtime MC(i)-IAs, if $i \geq 4$.*

Proof In the proof of Lemma 2 a unary realtime IA M is constructed that accepts a non-empty language if and only if the simulated counter machine is halting. In addition, if M is non-empty, it accepts exactly one input a^n. The construction can be modified in a straightforward way to a unary realtime IA M' so that M' accepts all unary inputs a^m with $m \geq n$, where a^n is accepted by M. Hence, M' accepts an infinite language if and only if the simulated counter machine is halting. Similar to the considerations of Theorem 2 we may assume that M' is a realtime MC(4)-IA. Since the halting problem on empty input is undecidable for counter machines, we therefore obtain that finiteness and infiniteness are undecidable for unary realtime MC(i)-IAs, if $i \geq 4$.

The fact that the empty set is accepted by some realtime MC(4)-IA and the result that emptiness is undecidable for realtime MC(4)-IAs by Theorem 2 immediately imply the undecidability of inclusion and equivalence for unary realtime MC(i)-IAs, if $i \geq 4$.

To show the undecidability of the regularity problem we will construct a realtime MC(4)-IA M' that accepts the language

$$L_M = \{\, b^n c^n w a^2 \mid w \in L(M) \text{ and } n \geq 2 \text{ is even} \,\},$$

where M is the realtime MC(4)-IA from Theorem 2 that accepts a non-empty language if and only if the simulated counter machine is halting. It is clear that $L_M = L(M')$ is a regular language if and only if $L(M)$ is empty. The latter holds if and only if the simulated counter machine is not halting. Hence, if we could decide the regularity problem for M', we could decide the halting problem for counter machines which is a contradiction to (Minsky 1961). Thus, it remains for us to construct the realtime MC(4)-IA M'. First, we implement the construction given in Lemma 1 for the language $\{\, b^n c^n \mid n \geq 1 \,\}$ for which we know that at most three communications per cell are needed due to two signals from left to right and one signal from right to left (see Fig. 4). The signal from right to left is started when both signals from left to right meet. This happens for even n at time $3n/2 + 1$ in cell $n/2$. In the next time step, we start in the right neighboring cell $n/2 + 1$ the simulation of M where cell $n/2 + 1$ is considered as the communication cell of M. Whenever cell $n/2 + 1$ enters an accepting state of M a new signal g is sent with maximum speed to the left which brings the communication cell of M' (cell 0) to accept an input x after reading the end-of-input symbol $\triangledown$ exactly when signal g arrives, if x has the correct format $b^+c^+a^+$ and a maximal prefix from b^+c^+ that belongs to $\{\, b^n c^n \mid n \geq 2 \text{ is even} \,\}$. The property that n is even can be checked in the communication cell. From this construction we know that the simulation of M is started at time $3n/2 + 2$ in cell $n/2 + 1$. Hence, this cell enters state f' at time $3n/2 + 2 + |w|$, if $w \in L(M)$. Thus, the signal g reaches the communication cell at time $3n/2 + 2 + |w| + n/2 + 1 = 2n + |w| + 3$. We can conclude that M' accepts L_M. To show that M' performs at most four communications per cell we observe that the active cells in M' either simulate M or check whether the input prefix from b^+c^+ in fact belongs to $\{\, b^n c^n \mid n \geq 2 \text{ is even} \,\}$. In the latter case, we know from the construction given in Lemma 1 (see Fig. 4) that at most three communications are performed per cell. In addition, the rightmost active cell in Fig. 4 performs one communication only, namely, the signal to the left is started. Here, this cell $n/2$ performs two more communications, namely, it initiates that cell $n/2 + 1$ starts the simulation of M and it transports the signal g to the left. Hence, cell $n/2$ performs three communications, whereas cells $0, 1, \ldots, n/2 - 1$ perform at most four communications taking into account signal g. For cell $n/2 + 1$ we obtain from Fig. 3 that two communications are sent to the right. Here, we have three communications in cell $n/2 + 1$ considering the additional signal g. The remaining cells $j > n/2 + 1$ perform at most four communications by construction. Altogether, we obtain that M' is a realtime MC(4)-IA. □

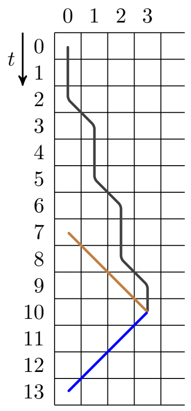

Fig. 4 Example computation on input b^6c^6 of a realtime MC(3)-IA accepting $\{\, b^n c^n \mid n \geq 1 \,\}$ according to the construction given in Lemma 1

Finally, we can show that it is neither possible to decide whether an arbitrary realtime IA performs a constant number of communications per cell at all nor whether an arbitrary realtime IA performs a fixed number of communications per cell for a given number $i \geq 4$.

Theorem 4 *It is undecidable for an arbitrary realtime* IA *M whether or not M is a realtime MC(const)-IA. It is undecidable for an arbitrary realtime IA M and some $i \geq 4$ whether or not M is a realtime MC(i)-IA.*

Proof Let M' be a realtime MC(4)-IA. We define the language

$$L_{M'} = \left\{\, w c^{|w|} \mid w \in L(M') \,\right\},$$

where c is some new alphabet symbol. A realtime IA M can accept $L_{M'}$ by using two tracks as follows. The correct number of c- and non-c-symbols can be checked in one track in the same way as in Lemma 1 for the language $\{\, a^n b^n \mid n \geq 1 \,\}$. This check needs three communications per cell. Additionally, the IA M' is simulated on the other track. As soon as M reads the first c and the simulated IA M' would enter an accepting state, the communication cell sends in every of the remaining $|w|$ time steps two alternating signals with maximum speed into the array. Finally, the input is accepted if the check on track 1 has been successful. So, the maximum communication per cell is in $O(|w|)$, if $w \in L_{M'}$. Thus, M is a realtime MC($const$)-IA if and only if $L_{M'}$ is finite. Since finiteness is undecidable for realtime MC(4)-IAs due to Theorem 3, we obtain that the question of whether M is a realtime MC($const$)-IA is undecidable.

Now, we consider language $L_{M',i} = \{\, c^{i+1}w \mid w \in L(M')\,\}$ for a fixed $i \geq 4$. A realtime IA M can accept $L_{M',i}$ as follows. First, we use the communication cell to check that exactly $i+1$ c's are read. For every c the communication cell performs some communication. When the first non-c is read, the simulation of the given realtime MC(4)-IA M' is started. Finally, the IA M accepts if the simulated IA M' would accept and the number of c's has been $i+1$. Thus, we can conclude that M is a realtime MC(i)-IA if and only if $L(M')$ is empty. Since emptiness is undecidable for realtime MC(4)-IAs due to Theorem 2, we obtain that the question of whether M is a realtime MC(i)-IA is undecidable. □

We would like to note that the results obtained in this section are almost optimal. All decidability questions discussed are decidable for realtime MC(i)-IAs with $i \leq 2$, since the languages accepted by such automata are regular owing to Proposition 1. The only remaining question is the decidability status of the problems for realtime MC(3)-IAs.

5 Undecidability results for realtime MC(*const*) – OCAs and realtime MC(*const*) – CAs

In this section, we will use the ideas and constructions presented in the preceding section to obtain similar undecidability results for max communication bounded realtime one-way cellular automata (realtime MC(f)-OCA) and realtime two-way cellular automata (realtime MC(f)-CA) which process their input in contrast to IAs a parallel way. For a precise definition of such automata we refer to (Kutrib and Malcher 2010a). The best results known so far are as follows. It is shown in Kutrib and Malcher (2010a) that all usually studied decidability questions are undecidable for realtime MC($const$)-OCAs and hence are undecidable for realtime MC($const$)-CAs as well. The proof is by a reduction of Hilbert's tenth problem. Thus, the precise number of communications per cell is not known. The special case of letter-bounded languages has been investigated in Kutrib and Malcher (2010b), where it was possible to show undecidability results for realtime MC($\log(n)$)-OCAs accepting letter-bounded languages, again by a reduction of Hilbert's tenth problem. However, the number of different symbols in the letter-bounded language depends on the given polynomial of the instance of Hilbert's tenth problem and is therefore not a fixed number.

Here, we can improve these known results. First, we show that the emptiness problem is undecidable for realtime MC(4)-CAs working on a unary alphabet. Second, we obtain the undecidability of emptiness for realtime MC(4)-OCAs that accept letter-bounded languages that are subsets of a^*b^*. Then the undecidability of finiteness, infiniteness, inclusion, equivalence, and regularity can be shown as well. These results are in a certain sense the best possible results, since it is known that unary languages accepted by realtime OCAs are regular (Seidel 1979). Hence, all questions discussed become decidable. Let us start with some undecidability results for unary realtime MC(i)-CAs with $i \geq 4$.

Theorem 5 *Emptiness, finiteness, infiniteness, inclusion, and equivalence are undecidable for unary realtime MC(i)-CAs, if $i \geq 4$.*

Proof To show the undecidability of the emptiness problem we implement the construction of a unary realtime IA M given in the proof of Lemma 2 in a realtime CA M', where the leftmost cell of M' is simulating the communication cell of M and the remaining cells of M' are used the same way as the other cells in M. In addition, the rightmost cell of M' starts in the first time step a new signal g which is sent with maximum speed to the left. M' will only accept if g arrives at the leftmost cell. This can only happen if g arrives in some cell exactly at the moment when the accepting signal f' is started. Then g is forwarded to the left instead of f'. If g arrives in the rightmost cell currently representing a counter machine state at any other time, then g is deleted and M' will never accept. Hence, M' accepts one input if and only if the underlying counter machine is halting. According to the considerations of Theorem 2 and the fact that signal g either replaces signal f' or concerns cells that are not involved in the simulation of M, we can conclude that M' is a realtime MC(4)-CA accepting $L(M)$ which gives the undecidability of emptiness.

If we slightly adapt the above construction by never forwarding signal f' and only forwarding an accepting signal g when it has met some cell carrying f' or s'_2, then we obtain a realtime MC(4)-CA that accepts an infinite language if and only if the underlying counter machine is halting. Hence, the problems of finiteness and infiniteness are undecidable as well. Finally, the undecidability of inclusion as well as equivalence can be shown the same way as it is done in the proof of Theorem 3. □

Now, we turn to one-way cellular automata and show the undecidability of the emptiness problem if at most four communications per cell are performed and the given OCA accepts a letter-bounded language that is a subset of a^*b^*.

Theorem 6 *Emptiness is undecidable for realtime MC(i)-OCAs, if $i \geq 4$.*

Proof We consider an input from a^+b^+ and we will implement on the a-part of the input the construction given in the proof of Lemma 2 in a reversed way. This means that the communication cell is simulated in the rightmost a-cell and the flow of information is from right to left. In addition, the accepting signal f', which is initiated whenever a halting configuration of the simulated counter machine is entered, is here not started, but the information is kept in some cell. The b-part of the input is provided to have enough time for the simulation of the counter machine in the a-part. Hence, the rightmost cell of the b-part starts a signal g with maximum speed to the left which checks whether the input has the format a^+b^+ and which changes to an accepting signal g' if a state f' is met in the a-part which means that the counter machine has entered a halting configuration. If a wrong format is detected or the state f' is never met, the input is rejected. As an example an accepting computation may be found in Fig. 5.

So, the OCA constructed accepts a non-empty and infinite language if and only if the counter machine halts. Thus, the emptiness problem is undecidable due to the undecidability of the halting problem for counter machines (Minsky 1961). According to the considerations of Theorem 2 we know that the IA constructed there is an MC(4)-IA. Hence, we can conclude that the OCA constructed here is an MC(4)-OCA, since we have on the one hand at most three communications per cell on the a-part for the simulation of the counter machine. We note that the signal f' is not started in our modified construction. On the other hand, we have one additional communication per cell for the signals g and g'. □

The undecidability of the emptiness problem can now be used to show further undecidability results.

Theorem 7 *Finiteness, infiniteness, inclusion, equivalence, and regularity are undecidable for realtime MC(i)-OCAs, if $i \geq 4$.*

Proof In the proof of Theorem 6 a realtime MC(4)-OCA is constructed that accepts an infinite language if and only if the counter machine halts. Thus, the finiteness problem as well as the infiniteness problem are undecidable for realtime MC(i)-OCAs, if $i \geq 4$.

The fact that the empty set is accepted by some realtime MC(4)-OCA and the result that emptiness is undecidable for realtime MC(4)-OCAs by Theorem 6 immediately imply the undecidability of inclusion and equivalence for realtime MC(i)-OCAs, if $i \geq 4$.

To show the undecidability of the regularity problem we use a similar approach as in the proof of Theorem 3 and construct a realtime MC(4)-OCA M' that accepts the language $L_M = \{\, c^n d^n w \mid w \in L(M) \text{ and } n \geq 1 \,\}$, where M is the realtime MC(4)-OCA from Theorem 6. Again, $L_M = L(M')$ is a regular language if and only if $L(M)$ is empty which leads to the undecidability of regularity.

For the construction of the realtime MC(4)-OCA M' we implement the construction of a realtime MC(2)-OCA that, on the input part from c^+d^+, accepts $\{\, c^n d^n \mid n \geq 1 \,\}$ as given in Example 2 of Kutrib and Malcher (2010a). On the input part from a^+b^+ we implement the construction given in the proof of Theorem 6 which implies that every cell performs at most four communications. The signal g' is forwarded from the leftmost a-cell through the c^+d^+ part to the leftmost cell of M'. The signal g' checks additionally that the remaining input has the format c^+d^+, and stops and changes to the sole accepting state g'' if an accepting state of the construction given in Example 2 of Kutrib and Malcher (2010a) is found. By this construction, every cell in the c^+d^+ part performs at most three communications.

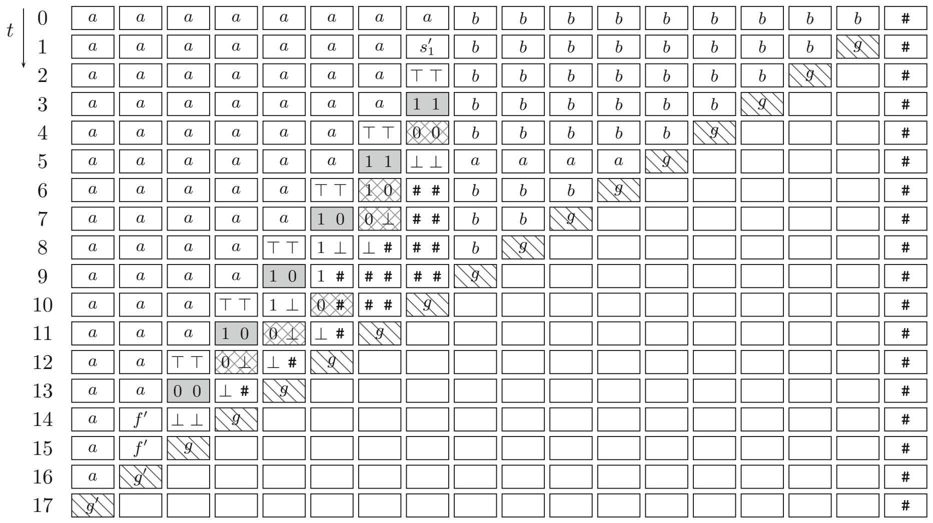

Fig. 5 Example computation of a one-way cellular automaton simulating the counter machine from Example 2 according to the construction given in Theorem 6. In contrast to Fig. 3 only the first two tracks of the cells are depicted. Cells that perform some communication are depicted with a non-empty background

Altogether, we obtain that M' is a realtime MC(4)-OCA accepting L_M. □

Since OCAs are in particular CAs, we immediately obtain the undecidability of regularity in the two-way case.

Corollary 1 *Regularity is undecidable for realtime MC(i)-CAs, if $i \geq 4$.*

Finally, we show that it is again undecidable to check whether or not a given realtime OCA performs a finite number of communications per cell.

Theorem 8 *It is undecidable for an arbitrary realtime OCA M whether or not M is a realtime MC(const)-OCA. It is undecidable for an arbitrary realtime OCA M and some $i \geq 4$ whether or not M is a realtime MC(i)-OCA.*

Proof Let M' be an arbitrary realtime MC(4)-OCA. From M' we construct an equivalent realtime OCA M that has basically the same transitions as M', but is additionally modified such that every cell performs a communication in every time step. Hence, M is a realtime MC(*const*)-OCA if and only if $L(M')$ is finite. Since finiteness is undecidable for realtime MC(4)-OCAs by Theorem 7, we obtain the first statement.

Next, let M' be the realtime MC(4)-OCA constructed in the proof of Theorem 6. We define the language $L_{M',i} = \{ c^n w \mid w \in L(M') \text{ and } n \geq i+1 \}$, where c is some new alphabet symbol and $i \geq 4$ is fixed. A realtime OCA M can accept $L_{M',i}$ by simulating M' on non-c cells. Additionally, it is ensured that all c-cells perform a communication in every time step. As soon as the rightmost c-cell obtains the signal g' from its right neighbor, this signal is forwarded to the left whereby it is checked that at least $i+1$ c's are provided. If so, the input is accepted and otherwise rejected. We claim that M is an MC(i)-OCA if and only if $L(M')$ is empty. If $L(M')$ is empty, then M accepts the empty set and hence is in particular an MC(i)-OCA. If $L(M')$ is not empty, then $L(M')$ contains a word w that is at least of length 2. Hence, $L_{M',i}$ contains a word $c^{i+1}w$. By construction, this already implies more than i communications in the leftmost cell. Hence, M is not an MC(i)-OCA. Since the emptiness problem is undecidable for M' by Theorem 6, we obtain the claim of the theorem. □

Again, these results hold in case of two-way communication as well.

Corollary 2 *It is undecidable for an arbitrary realtime CA M whether or not M is a realtime MC(const)-CA. It is undecidable for an arbitrary realtime CA M and some $i \geq 4$ whether or not M is a realtime MC(i)-CA.*

Acknowledgements The authors would like to thank one anonymous reviewer of the conference version (Kutrib and Malcher 2019) who gave us the valuable hint to consider the simulation of counter machines by cellular automata given in Carton et al. (2018). Additionally, the authors gratefully acknowledge the comments and suggestions given by the anonymous referees of the journal submission which improved the presentation of the paper.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Carton O, Guillon B, Reiter F (2018) Counter machines and distributed automata: a story about exchanging space and time. In: Baetens JM, Kutrib M (eds) Cellular automata and discrete complex systems (AUTOMATA 2018), *LNCS*, vol 10875. Springer, Belrin, pp 13–28

Chang JH, Ibarra OH, Palis MA (1987) Parallel parsing on a one-way array of finite-state machines. IEEE Trans Comput C–36:64–75

Cole SN (1969) Real-time computation by n-dimensional iterative arrays of finite-state machines. IEEE Trans Comput C–18(4):349–365

Fischer PC (1965) Generation of primes by a one-dimensional real-time iterative array. J ACM 12:388–394

Ibarra OH, Palis MA (1985) Some results concerning linear iterative (systolic) arrays. J Parallel Distrib Comput 2:182–218

Ibarra OH, Palis MA (1988) Two-dimensional iterative arrays: characterizations and applications. Theor Comput Sci 57:47–86

Kutrib M (2009) Cellular automata and language theory. In: Meyers RA (ed) Encyclopedia of complexity and systems science. Springer, Berlin, pp 800–823

Kutrib M, Malcher A (2009) Computations and decidability of iterative arrays with restricted communication. Parallel Process. Lett. 19(2):247–264

Kutrib M, Malcher A (2009) On one-way one-bit $O(1)$-message cellular automata. Electron Notes Theor Comput Sci 252:77–91

Kutrib M, Malcher A (2010) Cellular automata with sparse communication. Theor Comput Sci 411(38–39):3516–3526

Kutrib M, Malcher A (2010) One-way cellular automata, bounded languages, and minimal communication. J Autom Lang Comb 15(1/2):135–153

Kutrib M, Malcher A (2011) Cellular automata with limited inter-cell bandwidth. Theor Comput Sci 412(30):3917–3931

Kutrib M, Malcher A (2019) Iterative arrays with finite inter-cell communication. In: Castillo-Ramirez A, de Oliveira PPB (eds) Cellular automata and discrete complex systems (AUTOMATA 2019), *LNCS*, vol 11525. Springer, Berlin, pp 35–47

Malcher A (2018) Hierarchies and undecidability results for iterative arrays with sparse communication. In: Baetens JM, Kutrib M (eds) Cellular automata and discrete complex systems (AUTOMATA 2018), *LNCS*, vol 10875. Springer, Berlin, pp 100–112

Mazoyer J, Terrier V (1999) Signals in one-dimensional cellular automata. Theor Comput Sci 217(1):53–80

Minsky ML (1961) Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. Ann Math 2(74):437–455

Seidel SR (1979) Language recognition and the synchronization of cellular automata. Tech. Rep. 79-02, Department of Computer Science, University of Iowa

Umeo H, Kamikawa N (2002) A design of real-time non-regular sequence generation algorithms and their implementations on cellular automata with 1-bit inter-cell communications. Fund Inform 52:257–275

Umeo H, Kamikawa N (2003) Real-time generation of primes by a 1-bit-communication cellular automaton. Fund Inform 58:421–435

Worsch T (2000) Linear time language recognition on cellular automata with restricted communication. In: Gonnet GH, Panario D, Viola A (eds) Theoretical Informatics (LATIN 2000), *LNCS*, vol 1776. Springer, Berlin, pp 417–426