Iterative Arrays with Finite Inter-cell Communication

Martin Kutrib and Andreas Malcher(✉)

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

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. One main 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, by reduction of Hilbert's tenth problem it can be shown that all such decidability questions remain undecidable.

1 Introduction

Devices of homogeneous, interconnected, parallel acting automata have widely been investigated from a computational capacity point of view. Multidimensional 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*.

In connection with formal language recognition one-dimensional iterative arrays (IA) have been introduced in [3], where it was shown that the language family accepted by realtime IAs forms a Boolean algebra not closed under concatenation and reversal. In [2] it is shown that for every context-free grammar a two-dimensional lineartime iterative array parser exists. A realtime IA for prime numbers has been constructed in [4]. A characterization of various types of IAs in terms of restricted Turing machines and several results, especially speed-up theorems, are given in [6,7]. Several more results concerning formal languages can be found, for example, in the survey [9].

It is obvious that inter-cell communication is an essential resource for iterative arrays and can be measured qualitatively as well as quantitatively. In the first case, the number of different messages to be communicated by the cells is

A. Castillo-Ramirez and P. P. B. de Oliveira (Eds.): AUTOMATA 2019, LNCS 11525, pp. 35–47, 2019.
https://doi.org/10.1007/978-3-030-20981-0_3

bounded by some fixed constant. IAs with this restricted inter-cell communication have been investigated in [18,19] 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 [10,14,20].

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 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 [12,13]. In [11,13] 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 [15] and comprise again hierarchy results as well as undecidability results for the above questions. Concerning the measure on the maximum communication per cell 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, to be bounded by a constant number.

In this paper, we can answer the latter question negatively. 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 with a constant number of communications per cell. This is done by a reduction of Hilbert's tenth problem and requires a couple of consecutive constructions of IAs with a constant number of communications per cell, whereby the goal is to evaluate a polynomial with integral coefficients given by an instance of Hilbert's tenth problem.

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. For the *reversal of a word* w we write w^R and for its *length* we write $|w|$. For the number of occurrences of a symbol x in w we use the notation $|w|_x$. A language L over some alphabet $\{a_1, a_2, \ldots, a_k\}$ is said to be *letter bounded*, if $L \subseteq a_1^* a_2^* \cdots a_k^*$. 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) is 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*, $b_l, b_r : S \to B \cup \{\perp\}$ 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.*

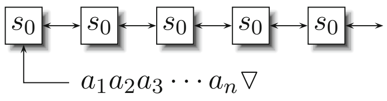

Fig. 1. Initial configuration of an iterative array.

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. The *lineartime* languages $\mathscr{L}_{lt}(X)$ are defined according to $\mathscr{L}_{lt}(X) = \bigcup_{r \in \mathbb{Q},\, r \geq 1} \mathscr{L}_{r \cdot n}(X)$.

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

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

For computations we now 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

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

Let $f : \mathbb{N} \to \mathbb{N}$ be a mapping. If all $w \in L(M)$ are accepted with computations where $\text{mcom}(w) \leq f(|w|)$, 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 $\mathscr{L}_{rt}(\text{MC}(5)\text{-IA})$.

The basic idea is to use the construction given in [17] 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 [17]. 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 [17] 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 2. *The 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. More communications of the communication cell are useless. 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. $\square$

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

Theorem 4. *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 $L_{\mathrm{h}i}$ 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 3, language $L_{\mathrm{h}i}$ is accepted by some MC$(i+1)$-IA in realtime. So, it remains to be shown that $L_{\mathrm{h}i}$ 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 $L_{\mathrm{h}i}$ 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 $w \in L_{\mathrm{h}i}$ 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 $a^{n+p_2} v$ 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 $w = ua^{|S|}b^{|S|^2+|S|}v$. Assume that no communication takes place (on the inter-cell link between the communication cell and its neighbor) while processing this factor. Let

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

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

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

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 $0 \le p_3$, $1 \le p_4 \le |S|$ with $|S| \le p_1 + p_2 + p_3$ and $p_1 + p_2 + p_3 + p_4 \le 2|S|$ such that

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

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

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

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

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

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 $L_{\mathrm{h}i}$ 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 $0 \leq p_1$, $1 \leq p_2$ with $p_1 + p_2 \leq |S|$ 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. $\square$

4 Undecidability Results for Realtime MC(*const*)-IAs

In this section, we first show that the question of emptiness is undecidable for realtime MC(*const*)-IAs by reduction of Hilbert's tenth problem which is known to be undecidable (see, e.g., [8,16]). 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 $p(\alpha_1, \ldots, \alpha_n) = 0$. A reduction of Hilbert's tenth problem to show the undecidability of emptiness for certain two-way counter machines has been used in [5] for the first time. A recent paper that provides a simulation of counter machines by linear-time one-way cellular automata is [1]. The basic idea of Ibarra in [5] is to define a language that encodes all possible values for $x_1, x_2, \ldots, x_n$ and evaluates the polynomial p for that values. Then, a two-counter machine is constructed accepting that language which is empty if and only if Hilbert's tenth problem has no solution in the integers. Such an approach we will in principal use here again. However, since we are concerned with IAs with constant communication and the input has to pass the communication cell to be processed by the IA, the definition of the language that evaluates the polynomial p is much more involved and will be presented in several steps.

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, we continue with $p(x_1, \ldots, x_n)$ multiplied with -1, whose constant summand now has a negative sign and which has the same integral roots as $p(x_1, \ldots, x_n)$. Such a polynomial has the following form:

$$p(x_1, \ldots, x_n) = t_1(x_1, \ldots, x_n) + \cdots + t_r(x_1, \ldots, x_n),$$

where each $t_j(x_1, \ldots, x_n)$, $1 \le j \le r$, is of the form

$$t_j(x_1, \ldots, x_n) = s_j x_1^{i_{j,1}} \cdots x_n^{i_{j,n}} \text{ with } s_j \ne 0 \text{ and } i_{j,1}, \ldots, i_{j,n} \ge 0.$$

Since changing the sequence in which the summands appear does not change the polynomial, we additionally may assume that the summands are ordered according to their sign starting with summands having a positive sign. Moreover, we may assume that a constant term occurs at the end of the sequence only. Thus, $t_r = s_r$, if p contains s_r as constant. Finally, let $i_j = \sum_{k=1}^{n} i_{j,k}$.

Now, we consider a polynomial $p(x_1, \ldots, x_n)$ with integer coefficients that has the above form. Let t_j with $1 \le j \le r$ be a positive term. Then, we define language $L'(t_j)$ over the alphabet $\{b_0, b_1, \ldots, b_{i_j}\}$ as

$$L'(t_j) = \{b_0^{s_j} b_1^{\alpha_1} \cdots b_{i_{j,1}}^{\alpha_1} b_{i_{j,1}+1}^{\alpha_2} \cdots b_{i_{j,1}+i_{j,2}}^{\alpha_2} \cdots b_{i_{j,1}+\cdots+i_{j,n-1}+1}^{\alpha_n} \cdots b_{i_j}^{\alpha_n} \mid \alpha_1, \alpha_2, \ldots, \alpha_n \ge 0\}.$$

Similarly, we define languages $L''(t_j)$ over $\{b_0, b_1, \ldots, b_{i_j}, c_1, \ldots, c_{i_j-1}, \$\}$ where words of $L'(t_j)$ are interleaved with c-blocks and a \$-block at the end, respectively, whose number of symbols is the product of the number of symbols of the two preceding blocks. So, the number of \$-symbols is an evaluation of $t_j(\alpha_1, \alpha_2, \ldots, \alpha_n)$.

$$L''(t_j) = \{ b_0^{s_j} b_1^{\alpha_1} c_1^{s_j\alpha_1} b_2^{\alpha_1} c_2^{s_j\alpha_1^2} b_3^{\alpha_1} c_3^{s_j\alpha_1^3} \cdots b_{i_{j,1}}^{\alpha_1} c_{i_{j,1}}^{s_j\alpha_1^{i_{j,1}}} b_{i_{j,1}+1}^{\alpha_2} c_{i_{j,1}+1}^{s_j\alpha_1^{i_{j,1}}\alpha_2} \cdots b_{i_{j,1}+i_{j,2}}^{\alpha_2} c_{i_{j,1}+i_{j,2}}^{s_j\alpha_1^{i_{j,1}}\alpha_2^{i_{j,2}}} \cdots b_{i_j-1}^{\alpha_n} c_{i_j-1}^{s_j\alpha_1^{i_{j,1}}\alpha_2^{i_{j,2}}\cdots\alpha_n^{i_{j,n}-1}} b_{i_j}^{\alpha_n} \$^{s_j\alpha_1^{i_{j,1}}\alpha_2^{i_{j,2}}\cdots\alpha_n^{i_{j,n}}} \mid \alpha_1, \alpha_2, \ldots, \alpha_n \ge 0 \}$$

Finally, $L(t_j)$ is defined over $\{b_0, b_1, \ldots, b_{i_j}, c_1, \ldots, c_{i_j-1}, d, \$\}$ as

$$L(t_j) = \{ wd^{7|w|} \mid w \in L''(t_j) \}.$$

If t_j with $1 \le j \le r$ is a negative and non-constant term, the definition of $L(t_j)$ is identical except for the fact that each symbol \$ is replaced by some symbol # and $b_0^{s_j}$ is replaced by $b_0^{|s_j|}$. If t_r is a constant term, we define $L(t_r) = \{\#^{|s_r|} d^{7|s_r|}\}$.

Example 5. Let $t_j(x_1, x_2, x_3, x_4) = 3x_1^2 x_2 x_4$. Then, $s_j = 3$, $i_{j,1} = 2$, $i_{j,2} = 1$, $i_{j,3} = 0$, $i_{j,4} = 1$, and $i_j = 4$, $L'(t_j) = \{ b_0^3 b_1^{\alpha_1} b_2^{\alpha_1} b_3^{\alpha_2} b_4^{\alpha_4} \mid \alpha_1, \alpha_2, \alpha_4 \ge 0 \}$ and $L''(t_j) = \{ b_0^3 b_1^{\alpha_1} c_1^{3\alpha_1} b_2^{\alpha_1} c_2^{3\alpha_1^2} b_3^{\alpha_2} c_3^{3\alpha_1^2\alpha_2} b_4^{\alpha_4} \$^{3\alpha_1^2\alpha_2\alpha_4} \mid \alpha_1, \alpha_2, \alpha_4 \ge 0 \}$.

For example, to evaluate $t_j(2, 3, x_3, 2) = 3 \cdot 2^2 \cdot 3 \cdot 2 = 72$ we consider the word $b_0^3 b_1^2 c_1^6 b_2^2 c_2^{12} b_3^3 c_3^{36} b_4^2 \$^{72} d^{7\cdot 138} \in L(t_j)$ and to evaluate $t_j(2, 1, x_3, 2) = 3 \cdot 2^2 \cdot 1 \cdot 2 = 24$ we consider the word $b_0^3 b_1^2 c_1^6 b_2^2 c_2^{12} b_3 c_3^{12} b_4^2 \$^{24} d^{7\cdot 64} \in L(t_j)$. ■

Let us start our construction with three lemmas that will be essential in the sequel.

Lemma 6. *An MC(const)-IA can effectively be constructed that shifts an input of the form $w \in a^+b^+c^+$ into its cells within $2|w|$ time steps.*

Lemma 7. *An MC(const)-IA can effectively be constructed that accepts every $w \in \{ a^n b^m c^n \mid n, m \geq 1 \}$ within $5|w|$ time steps.*

Lemma 8. *An MC(const)-IA can effectively be constructed that accepts every $w \in \{ a^n b^m c^{n \cdot m} \mid n, m \geq 1 \}$ within $7|w|$ time steps.*

Lemma 9. *The language $L(t_j)$ belongs to $\mathscr{L}_{rt}(MC(const)\text{-}IA)$ for each term t_j with $1 \leq j \leq r$.*

Proof. We describe the construction of a realtime MC(*const*)-IA accepting $L(t_j)$, where t_j is a positive term. The construction for negative non-constant terms is identical except for exchanging $ by #. If t_j is a negative constant term, $L(t_j)$ is a regular language and can be accepted by a realtime MC(*const*)-IA using its communication cell only.

A realtime MC(*const*)-IA for $L(t_j)$ basically computes four tasks.

1. Check the correct format of the input and check whether there are exactly s_j symbols b_0.
2. Check whether the length of every c-block is the product of the lengths of its two preceding blocks.
3. Check whether the number of d's is equal to seven times the length of the preceding input.
4. Check the equal number of symbols $b_{i_{j,1}+\cdots+i_{j,k-1}+1}, \ldots, b_{i_{j,1}+\cdots+i_{j,k}}$ for every $1 \leq k \leq n$.

The first task can be done by the communication cell which rejects the input in case of a wrong format or a wrong number of b_0's. For the second task we use the construction described in Lemma 8 multiple times. Having stored the complete input other than the d's in the array, the communication cell sends a signal with maximum speed to the right which starts in the b_0-block as well as in every c-block an instance of the construction given in Lemma 8 which checks whether the length of every c-block and of the last $-block is the product of the lengths of its two preceding blocks. Whenever an error is encountered an error signal is sent to the left which rejects the input. Since there is only a finite number of c-blocks, we have to keep track of a finite number of instances of the construction of Lemma 8 which in addition are distributed over the array and at most two instances are overlapping. Since one instance can be realized by an MC(*const*)-IA, we obtain that the finite number of such instances can be realized by an MC(*const*)-IA as well. It has been shown in Lemma 8 that the total time for one instance is bounded by seven times the length of the input. Hence, we can conclude that, for some input wd^* with $w \in L''(t_j)$, time $7|w|$ is sufficient to accomplish the second task. This time is provided by the number of d's whose correct number is checked in the third task as follows. The communication cell sends a signal C with speed $1/6$ to the right which is reflected at the cell carrying the last $ and is sent back to the left with maximum speed. The third task is successful if and only if C arrives again at the communication cell when the end-of-input symbol $\triangledown$ is read for the first time.

Finally, we have to accomplish the fourth task. Here, we have to check for every $1 \leq i \leq n$ whether each of a finite number, say n_i, of different b-symbols, say $\{b_{l_i}, b_{l_i+1}, \ldots, b_{l_i+n_i-1}\}$ is exactly α_i. This can basically be realized by implementing $n_i - 1$ instances of the construction given in Lemma 7. The first instance checks that the number of symbols b_{l_i} is equal to the number of symbols b_{l_i+1}, the second instance checks that the number of symbols b_{l_i+1} is equal to the number of symbols b_{l_i+2}, and so on. If all checks are positive for every $1 \leq i \leq n$, the fourth task is accomplished and requires at most $t = \sum_{i=1}^{n} 5\alpha_i(n_i - 1)$ time steps due to Lemma 7. Let $wd^{7|w|}$ with $w \in L''(t_j)$ be the input. Then, $\sum_{i=1}^{n} \alpha_i(n_i - 1)$ is bounded by $|w|$. Hence, $t \leq 5|w|$ and the number of d's provided gives enough time to accomplish the fourth task in realtime. Since n and all n_i for $1 \leq i \leq n$ are fixed numbers depending on t_j, we need only a finite number of instances of the construction given in Lemma 7 each of which can be realized by an MC(const)-IA. Hence, we obtain that the fourth task can be realized by an MC(const)-IA.

Altogether, by implementing the different tasks in parallel in different tracks we obtain a realtime MC(const)-IA which accepts its input if and only if all four tasks have been accomplished successfully. $\square$

In Lemma 9 we have described how an evaluation of a term t_j can be simulated. Our next step is to simulate an evaluation of the given polynomial p. To this end, we have to put the evaluations of the single terms together. This will be done by concatenating certain regular languages around each language $L(t_j)$. Finally, the intersection of all these sets is considered which leads to an evaluation of all terms on the same input and thus to an evaluation of p.

Let us consider the following regular languages R_k depending on the sign of the term t_k. We set $R_k = b_0^* b_1^* c_1^* \ldots b_{i_k-1}^* c_{i_k-1}^* b_{i_k}^* \$^* d^*$ if $s_k > 0$, $R_k = \#^* d^*$ if t_k is the negative constant, and $R_k = b_0^* b_1^* c_1^* \ldots b_{i_k-1}^* c_{i_k-1}^* b_{i_k}^* \#^* d^*$ otherwise. Then, we define

$$\begin{aligned}\tilde{L}(t_j) = \{\, & a_1^{\alpha_1} \cdots a_n^{\alpha_n} w_1 w_2 \cdots w_r \ \mid \alpha_1, \ldots, \alpha_n \geq 0, w_i \in R_i, 1 \leq i \leq r, i \neq j, \\ & w_j = w'_j d^{7|w'_j|}, \text{ and } w'_j = b_0^{s_j} b_1^{\alpha_1} c_1^{s_j \alpha_1} b_2^{\alpha_1} c_2^{s_j \alpha_1^2} b_3^{\alpha_1} c_3^{s_j \alpha_1^3} \cdots \\ & b_{i_{j,1}}^{\alpha_1} c_{i_{j,1}}^{s_j \alpha_1^{i_{j,1}}} b_{i_{j,1}+1}^{\alpha_2} c_{i_{j,1}+1}^{s_j \alpha_1^{i_{j,1}} \alpha_2} \cdots b_{i_{j,1}+i_{j,2}}^{\alpha_2} c_{i_{j,1}+i_{j,2}}^{s_j \alpha_1^{i_{j,1}} \alpha_2^{i_{j,2}}} \cdots \\ & b_{i_j-1}^{\alpha_n} c_{i_j-1}^{s_j \alpha_1^{i_{j,1}} \alpha_2^{i_{j,2}} \cdots \alpha_n^{i_{j,n}-1}} b_{i_j}^{\alpha_n} \$^{s_j \alpha_1^{i_{j,1}} \alpha_2^{i_{j,2}} \cdots \alpha_n^{i_{j,n}}} \,\}\end{aligned}$$

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

We can observe that each language $\tilde{L}(t_j)$ consists of the language $L(t_j)$, which evaluates $t_j(x_1, x_2, \ldots, x_n)$, and the concatenation of the regular sets $R_1 R_2 \cdots R_{j-1}$ to the left and $R_{j+1} \cdots R_r$ to the right. Additionally, each word of $\tilde{L}(t_j)$ starts with the substring $a_1^{\alpha_1} a_2^{\alpha_2} \cdots a_n^{\alpha_n}$ which determines the variables $x_1, x_2, \ldots, x_n$ to be evaluated. Since $\tilde{L}(p)$ is the intersection over all terms of the polynomial p, $\tilde{L}(p)$ simulates all those terms on the same input $\alpha_1, \alpha_2, \ldots, \alpha_n$. Thus, $\tilde{L}(p)$ denotes an evaluation of the given polynomial p whereby evaluations of positive and negative terms are encoded by $-symbols and #-symbols, respectively.

Lemma 10. *The language $\tilde{L}(p)$ belongs to $\mathscr{L}_{rt}(MC(const)\text{-}IA)$.*

Finally, we set $L(p) = \{\, we^{4|w|} \mid w \in \tilde{L}(p), |w|_\$ = |w|_\# \,\}$ for some new alphabet symbol e.

Lemma 11. *The language $L(p)$ belongs to $\mathscr{L}_{rt}(MC(const)\text{-}IA)$.*

Proof. To accept $L(p)$ three checks are performed in parallel. First, it is verified that the prefix w belongs to $\tilde{L}(p)$ using the construction given in Lemma 10, which says in addition that the construction can be realized by an MC(*const*)-IA in realtime. Second, to check the correct number of e's the communication cell starts in the first time step a signal E_1 with speed $1/2$ to the right and starts another signal E_2 when the first e is read with maximum speed to the right. When both signals meet, some signal E_3 is sent back to the left with speed $1/3$ and has to arrive at the communication cell when the end-of-input symbol $\triangledown$ is read for the first time. Clearly, the second task can be done by a realtime MC(*const*)-IA. The third task is to check the equal number of $- and #-symbols. The basic idea here is to implement again the construction given in Lemma 7. Since the terms of the given polynomial p are ordered with respect to their sign, we know that blocks of $-symbols are followed by blocks of #-symbols. In between these blocks there are blocks of other alphabet symbols which have to be ignored. In a straightforward modification of the constructions given in Lemmas 6 and 7 we first enqueue a finite number of $-blocks in one queue and then a finite number of #-blocks in a different queue. If both queues have an equal length, the third task is accomplished. According to Lemmas 6 and 7 we know that the third task can be done by an MC(*const*)-IA and needs at most $5|w|$ time steps. Due to the e-symbols provided the overall length of the input is $5|w|$ which implies that the third task can be accomplished in realtime.

Altogether, the input is accepted when all three task have successfully been done and we obtain that $L(p)$ is accepted by a realtime MC(*const*)-IA. □

Now, all prerequisites are done and we obtain the following undecidability results for realtime MC(*const*)-IAs.

Theorem 12. *Emptiness is undecidable for realtime MC(const)-IAs.*

Proof. By applying Lemma 11 we can construct a realtime MC(*const*)-IA M accepting language $L(p)$ given a polynomial $p(x_1, \ldots, x_n)$. Moreover, the language $L(M) = L(p)$ is empty if and only if the polynomial $p(x_1, \ldots, x_n)$ has no solution in the non-negative integers Since Hilbert's tenth problem is undecidable, the emptiness problem for realtime MC(*const*)-IAs is undecidable as well. □

Corollary 13. *The questions of finiteness, infiniteness, equivalence, and inclusion are undecidable for realtime MC(const)-IAs.*

References

1. Carton, O., Guillon, B., Reiter, F.: Counter machines and distributed automata a story about exchanging space and time. In: Baetens, J.M., Kutrib, M. (eds.) AUTOMATA 2018. LNCS, vol. 10875, pp. 13–28. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92675-9_2
2. Chang, J.H., Ibarra, O.H., Palis, M.A.: Parallel parsing on a one-way array of finite-state machines. IEEE Trans. Comput. **C–36**, 64–75 (1987)
3. Cole, S.N.: Real-time computation by n-dimensional iterative arrays of finite-state machines. IEEE Trans. Comput. **C–18**(4), 349–365 (1969)
4. Fischer, P.C.: Generation of primes by a one-dimensional real-time iterative array. J. ACM **12**, 388–394 (1965)
5. Ibarra, O.H.: Reversal-bounded multicounter machines and their decision problems. J. ACM **25**(1), 116–133 (1978)
6. Ibarra, O.H., Palis, M.A.: Some results concerning linear iterative (systolic) arrays. J. Parallel Distrib. Comput. **2**, 182–218 (1985)
7. Ibarra, O.H., Palis, M.A.: Two-dimensional iterative arrays: characterizations and applications. Theor. Comput. Sci. **57**, 47–86 (1988)
8. Jones, J.P., Matijasevič, Y.V.: Proof of recursive unsolvability of Hilbert's tenth problem. Am. Math. Mon. **98**, 689–709 (1991)
9. Kutrib, M.: Cellular automata and language theory. In: Meyers, R.A. (ed.) Encyclopedia of Complexity and Systems Science, pp. 800–823. Springer, Berlin (2009). https://doi.org/10.1007/978-0-387-30440-3
10. Kutrib, M., Malcher, A.: Computations and decidability of iterative arrays with restricted communication. Parallel Process. Lett. **19**(2), 247–264 (2009)
11. Kutrib, M., Malcher, A.: On one-way one-bit $O(1)$-message cellular automata. Electr. Notes Theor. Comput. Sci. **252**, 77–91 (2009)
12. Kutrib, M., Malcher, A.: Cellular automata with sparse communication. Theor. Comput. Sci. **411**(38–39), 3516–3526 (2010)
13. Kutrib, M., Malcher, A.: One-way cellular automata, bounded languages, and minimal communication. J. Autom. Lang. Comb. **15**(1/2), 135–153 (2010)
14. Kutrib, M., Malcher, A.: Cellular automata with limited inter-cell bandwidth. Theor. Comput. Sci. **412**(30), 3917–3931 (2011)
15. Malcher, A.: Hierarchies and undecidability results for iterative arrays with sparse communication. In: Baetens, J.M., Kutrib, M. (eds.) AUTOMATA 2018. LNCS, vol. 10875, pp. 100–112. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92675-9_8
16. Matijasevič, Y.V.: On recursive unsolvability of Hilbert's tenth problem. In: Logic, Methodology and Philosophy of Science, IV (Proceedings of the Fourth International Congress, Bucharest, 1971), North-Holland, pp. 89–110 (1973)
17. Mazoyer, J., Terrier, V.: Signals in one-dimensional cellular automata. Theor. Comput. Sci. **217**(1), 53–80 (1999)
18. 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. Fundam. Inf. **52**, 257–275 (2002)
19. Umeo, H., Kamikawa, N.: Real-time generation of primes by a 1-bit-communication cellular automaton. Fundam. Inf. **58**, 421–435 (2003)
20. Worsch, T.: Linear time language recognition on cellular automata with restricted communication. In: Gonnet, G.H., Viola, A. (eds.) LATIN 2000. LNCS, vol. 1776, pp. 417–426. Springer, Heidelberg (2000). https://doi.org/10.1007/10719839_41