Hierarchies and Undecidability Results for Iterative Arrays with Sparse Communication

Andreas Malcher

Hierarchies and Undecidability Results for Iterative Arrays with Sparse Communication

Andreas Malcher

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

Abstract Iterative arrays with restricted internal inter-cell communication are investigated. A quantitative measure for the communication is defined by counting the number of uses of the links between cells and it is differentiated between the sum of all communications of an accepting computation and the maximum number of communications per cell occurring in accepting computations. The computational complexity of both classes of devices is investigated and put into relation. In addition, a strict hierarchy depending on the maximum number of communications per cell is established. Finally, it is shown that almost all commonly studied decidability questions are not semidecidable for iterative arrays with restricted communication and, moreover, it is not semidecidable as well whether a given iterative array belongs to a given class with restricted communication.

1 Introduction

Devices of homogeneous, interconnected, parallel acting automata have extensively been investigated from a computational capacity point of view. The specification of such a system includes the type and specification of the single automata (sometimes called cells), their interconnection scheme (which can imply a dimension to the system), a local and/or global transition function, and the input and output modes. Multidimensional devices with nearest neighbor connections whose cells are finite automata are commonly called *cellular automata* (CA). If the input mode is sequential to a distinguished communication cell, they are called *iterative arrays* (IA). In connection with formal language recognition IA have been introduced in [3], where it was shown that the language family accepted by realtime-IA forms a Boolean algebra not closed under concatenation and reversal. In [1] it is shown that for every context-free grammar a two-dimensional lineartime-IA parser exists. A realtime acceptor for prime numbers has been constructed in [4]. A characterization of various types of IA in terms of restricted Turing machines and several results, especially speed-up theorems, are given in [5,6]. Several more results concerning formal languages can be found, for example, in [16,8].

Communication is an essential resource for cellular automata and can be measured in a qualitative way and a quantitative way. In the first case, the number of different messages to be communicated by an IA is bounded by some fixed

constant. Iterative arrays with this restricted inter-cell communication have been investigated in [17,18] 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 real time. In connection with language recognition and decidability questions multi-dimensional iterative arrays and one-dimensional (one-way) cellular automata with restricted communication are intensively studied in [9,13,19].

To measure the communication in cellular automata in a quantitative way we count the number of uses of the links between cells and we consider, on the one hand, bounds on the sum of all communications of an accepting computation and, on the other hand, bounds on the maximum number of communications per cell that may appear in accepting computations. Many results on this quantitative measure have been obtained for cellular automata in [11,12], and cellular automata that are restricted with respect to the qualitative *and* the quantitative measure are investigated in [10,12] as well. As main results we would like to mention hierarchy results and the undecidability of almost all commonly studied decidability questions such as emptiness, finiteness, equivalence, inclusion, regularity, and context-freeness. It is of particular interest that even a small amount of communication is sufficient to obtain undecidability results.

In this paper, we want to continue the investigation of the quantitative measure by studying iterative arrays with quantitatively restricted communication. In the next section, we present some basic notions and definitions and we introduce the two classes of communication bounded iterative arrays, namely, sum communication bounded IA and max communication bounded IA. Moreover, we discuss several examples whose construction ideas are also helpful for other constructions in the sequel. In Section 3, we study the computational capacity of the introduced devices and obtain proper inclusions in between sum communication bounded IA and max communication bounded IA as well as between both classes. Sections 4 and 5 are devoted to studying decidability questions for sum communication bounded IA and max communication bounded IA. For the former class we obtain the non-semidecidability of emptiness, finiteness, equivalence, inclusion, regularity, and context-freeness for devices that have at most $O(n)$ communications on accepted inputs of length n, whereas for the latter class all questions are not semidecidable as well for devices that have at most $O(\log(n))$ communications per cell on accepted inputs of length n. Moreover, we can show for both classes that it is not semidecidable whether an arbitrary IA belongs to either class. It should be noted that missing proofs are omitted due to space limitations.

2 Definitions and Preliminaries

We denote the positive integers and zero $\{0, 1, 2, ...\}$ by $\mathbb{N}$. The *empty word* is denoted by λ, the *reversal* of a word w by w^R, and for the length of w we write $|w|$. We use $\subseteq$ for *inclusions* and $\subset$ for *strict inclusions*. By $\log(n)$ we

denote the logarithm of n to base 2. Throughout the article two devices are said to be *equivalent* if and only if they accept the same language.

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. 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. A formal definition is:

Definition 1. *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, $A \subseteq S$ *is the finite, nonempty set of* input symbols, B *is the 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 non-communication cells *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.

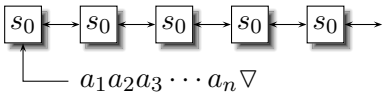

Figure 1. An iterative array.

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 $w_{t+1} = a_2 \cdots a_n$ if $w_t = a_1 \cdots a_n$.

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 $L(M)$ is said to be of time complexity t.

The family of all languages which are accepted by some IA with time complexity t is denoted by $\mathscr{L}_t(\text{IA})$. If t is the function $n+1$, acceptance is said to be in *realtime* and we write $\mathscr{L}_{rt}(\text{IA})$. 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.

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 $s \in S$.

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

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

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

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

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. Similarly, if all $w \in L(M)$ are accepted with computations where $\text{scom}(w) \leq f(|w|)$, then M is said to be *sum communication bounded by* f. In general, it is not expected to have tight bounds on the exact number of communications but tight bounds on their numbers in the order of magnitude. For the sake of readability we denote the class of IA that are max communication bounded by some function $g \in O(f)$ by MC(f)-IA. In addition, we use the notation *const* for functions from $O(1)$. The corresponding notation for sum communication bounded IA is SC(f)-IA.

To illustrate the definitions we start with some examples. Some of the ideas are also necessary for later constructions.

Example 2. The language $\{\, a^n b^{2n} \mid n \geq 1 \,\}$ belongs to $\mathscr{L}_{rt}(\text{MC}(\textit{const})\text{-IA})$.

The idea of the construction is to start a signal with speed 1/2 to the right, that is, at every second time step the signal moves one cell to the right, when

reading the first a. In addition, a signal with maximum speed to the right is started when the first b is read. When both signals meet, another signal with maximum speed is sent to the left and the input is accepted if and only if this signal reaches the communication cell when the last b is read. Since there are two right signals and one left signal used, it is clear that the IA constructed is an MC(*const*)-IA. An example computation on input a^5b^{10} is depicted in Figure 2 (left). ■

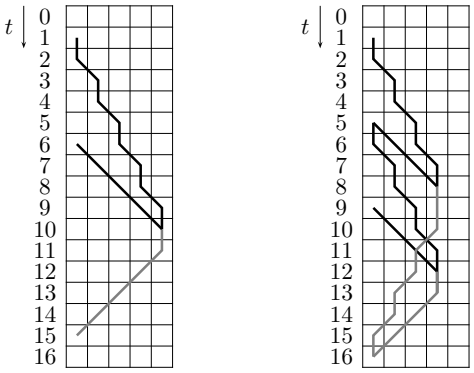

Figure 2. Two example computations for Example 2 (left) on input a^5b^{10} and Example 3 (right) on input $a^4b^4c^8$.

Example 3. The language $\{ a^n b^n c^{2n} \mid n \geq 1 \}$ belongs to $\mathscr{L}_{rt}$(MC(*const*)-IA).

The construction is similar. We start right signals R_1 and R_2 with speed $1/2$ resp. 1, when the first a resp. b is read. Additionally, a signal R_3 with speed $1/2$ is started when reading the first b. Finally, signal R_4 with speed 1 is started when reading the first c. When signals R_1 and R_2 meet, a left signal L_1 with speed $1/2$ is started, and a left signal L_2 with speed 1 is started when signals R_3 and R_4 meet. Finally, the input is accepted if and only if signals L_1 and L_2 meet in the communication cell when the last c is read. Since there are altogether four right signals and two left signals used, the IA constructed is an MC(*const*)-IA. An example computation on input $a^4b^4c^8$ is depicted in Figure 2 (right). ■

The next example shows that a binary counter can already be implemented by an SC(n)-IA.

Example 4. The language $\{ a^n b^n \mid n \geq 1 \}$ belongs to $\mathscr{L}_{rt}$(SC(n)-IA).

We implement the usual construction of a binary counter for IA (see, for example, [9,13]), where the first $\lceil \log(n) \rceil$ cells store the binary encoding of some number n and the communication cell carries the least significant bit. To increase or decrease the counter we possibly have to send carry-overs in order to update the current encoding. Additionally, we mark the cell carrying the most significant

bit suitably and this mark may move to the right while increasing the counter and may move to the left while decreasing the counter. Thus, to accept an input $a^n b^n$ we start with a counter 0, we increase the counter by one for every read a and decrease the counter by one for every read b. If in the end the counter is 0 again, which can be detected with help of the mark of the most significant bit, and the input format is correct, we accept the input and reject in all other cases. An example computation may be found in Figure 3, where + resp. - denote carry-overs for increasing resp. decreasing the counter. Furthermore, the grey cells mark the cells carrying the most significant bit.

To calculate the number of necessary communications on an accepted input $a^n b^n$ we observe that the only information sent to the right are the carry-overs, while the only information sent to the left is the position of the most significant bit. For the latter we have that there always is exactly one cell carrying the most significant bit which gives $2n$ communications. For the carry-overs it easy to see that the communication cell (cell 0) sends a carry-over in every second time step, while cell 1 sends a carry-over in every fourth time step, cell 2 sends a carry-over in every eighth time step and so on. Altogether, the number of communications for the carry-overs is bounded by

$$\sum_{i=1}^{\lceil \log(n) \rceil} \frac{2n}{2^i} = 2n \sum_{i=1}^{\lceil \log(n) \rceil} \frac{1}{2^i} = 2n \left(1 - \frac{1}{2^{\lceil \log(n) \rceil}}\right)$$
$$\leq 2n \left(1 - \frac{1}{2^{\log(n)+1}}\right) = 2n \left(1 - \frac{1}{2n}\right) = 2n - 1.$$

Hence, the number of communications on input $a^n b^n$ is $2n + 2n - 1 \in O(n)$ and the IA constructed is a realtime-SC(n)-IA. ■

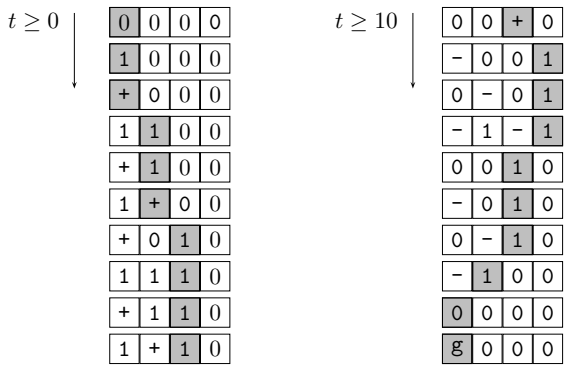

Figure 3. Example computation for the construction given in Example 4 on input $a^9 b^9$. The symbols + resp. - denote carry-overs for increasing resp. decreasing the counter. The cells carrying the most significant bit are marked grey and g denotes an accepting state.

Example 5. The language $\{ a^{2^n} b^{2n} \mid n \geq 1 \}$ belongs to $\mathscr{L}_{rt}(\mathrm{SC}(n)\text{-IA})$.

The rough idea is to implement a binary counter as in Example 4 which is increased for every input symbol a. When the first b is read, a right signal is started which inspects the counter and checks whether all cells are carrying a carry-over except the cell carrying the most significant bit. If so, the number of a's is 2^n for some $n \geq 1$ and the signal is sent back to the left with maximum speed. When it reaches the communication cell exactly when the end-of-input symbol is read, then the input is accepted and in all other cases rejected. ■

3 Separability Results

In this section, we will separate several classes of max communication bounded and sum communication bounded iterative arrays. We start by showing that realtime-SC(n)-IA are less powerful than realtime-SC(n^2)-IA. We remark that a similar result is known between realtime-SC(n)-CA and realtime-SC(n^2)-CA (see, e.g., [11]).

Theorem 6. $\mathscr{L}_{rt}(SC(n)\text{-}IA) \subset \mathscr{L}_{rt}(SC(n^2)\text{-}IA)$.

Proof. The inclusion follows from structural reasons. To show the properness of the inclusion we consider the language $L = \{ wcw \mid w \in \{a,b\}^+ \}$, which can be accepted by using a queue store in which the first w part is enqueued. After the separating symbol c the queue store is symbolwise dequeued and matched with the second w part. It is shown in [7] how an IA can simulate such a queue store without any loss of time. Thus, $L \in \mathscr{L}_{rt}(\mathrm{IA})$ which implies that $L \in \mathscr{L}_{rt}(\mathrm{SC}(n^2)\text{-IA})$. Another construction idea for L may be found in [3]. On the other hand, let us assume that L belongs to $\mathscr{L}_{rt}(\mathrm{SC}(n)\text{-IA})$. Then, we will derive a contradiction in two steps. First, it is possible under the above assumption to construct a realtime-SC($n \cdot \sqrt{n}$)-CA accepting $L' = \{ (wc)^{|w|} \mid w \in \{a,b\}^+ \}$. Second, by adapting the proof given in [11] showing that $\{ wcw^R \mid w \in \{a,b\}^+ \}$ does not belong to $\mathscr{L}_{rt}(\mathrm{SC}(f)\text{-CA})$ if $f \in o(n^2/\log(n))$, we obtain that L' does not belong to $\mathscr{L}_{rt}(\mathrm{SC}(n \cdot \sqrt{n})\text{-CA})$ as well which gives the desired contradiction.

Now, let L be accepted by some realtime-SC(n)-IA M. A CA M' accepting L' works as follows on input $wcwc \cdots wc$. M' has six tracks. The original input is kept on track 1 without change. Each cell having as left neighbor a c-cell or the leftmost border cell can identify itself and will act on track 2 as communication cell for the simulation of M. The "input" is fed into the communication cells by shifting the input of the remaining cells on track 3 one cell to the left in every time step, whereby a second c acts as end-of-input symbol. The shifting is stopped before passing the second c-cell. Hence, every communication cell can decide after $2|w|+2$ time steps whether it has recognized the structure wcw and stores this information by entering some state g. On track 4, the rightmost cell starts a signal moving with maximum speed to the left that checks whether the input is correctly formatted and all communication cells have entered state g. Since the checked w-blocks are pairwise overlapping, we can check with this construction whether the input is of the form $(wc)^+$ for some $w \in \{a,b\}^+$. It

remains to check that the number of c's is exactly $|w|$. To this end, we use track 5 on which each c-cell sends a signal with maximum speed to the left. All incoming c-signals are collected on track 6 from left to right starting in the leftmost cell. This means that sending m c-signals leads to the marking of the leftmost m cells on track 6. Hence, the final signal on track 4 has additionally to check whether the leftmost w-block is completely marked on track 6. If this is the case, the input is accepted and in all other cases rejected. Hence, M' accepts L' in realtime. Next, we have to calculate the sum of all communications. The length of an accepted input is $(|w|+1)|w| = |w|^2 + |w| \in O(|w|^2)$. The number of communications on track 1 and track 6 is zero and in $O(|w|)$, respectively. On track 2, we roughly have $|w|$ simulations of M which has $O(|w|)$ communications each by assumption. Hence, track 2 has at most $O(|w|^2)$ communications. On track 3, we shift blocks of length $2|w|+2$ for $O(|w|)$ many time steps which causes at most $O(|w|^2)$ communications in the whole. On track 4 we have $O(|w|^2)$ communications, since the signal passes the complete input. Finally, on track 5 we have $|w|$ signals passing at most $O(|w|^2)$ cells which gives at most $O(|w|^3)$ communications. Altogether, the sum of all communications in M' is in $O(|w|^3)$. Therefore, M' is a realtime-SC($n \cdot \sqrt{n}$)-CA.

The proof that L' does not belong to $\mathscr{L}_{rt}$(SC(f)-CA) if $f \in o(n^2/\log(n))$ is an adaption of a proof given in [11] and omitted here. □

For separating results in between the classes of max communication bounded iterative arrays we will use in the following the notion of time constructability. We say that a function $f : \mathbb{N} \to \mathbb{N}$ is *time-constructible* by an MC-IA M, if the communication cell of M enters, on empty input, a certain state exactly at the time steps $f(n)$ for all $n \geq 1$. For more information on time-constructible functions we refer to [8,15].

Lemma 7. *The function 2^n can be time constructed by an MC($\log(n)$)-IA. The function n^2 can be time constructed by an MC($\sqrt{n}$)-IA.*

Now, we can state the hierarchy of max communication bounded iterative arrays.

Theorem 8. *1. REG $\subset \mathscr{L}_{rt}$(MC(const)-IA),*
2. $\mathscr{L}_{rt}$(MC(const)-IA) $\subset \mathscr{L}_{rt}$(MC($\log(n)$)-IA),
3. $\mathscr{L}_{rt}$(MC($\log(n)$)-IA) $\subset \mathscr{L}_{rt}$(MC($\sqrt{n}$)-IA),
4. $\mathscr{L}_{rt}$(MC($\sqrt{n}$)-IA) $\subset \mathscr{L}_{rt}$(MC(n)-IA).

Proof. The inclusion claimed in 1. is clear, since the communication cell of an IA can simulate a deterministic finite automaton accepting a given regular language without using any communication. The inclusion is proper, since Example 2 provides a non-regular language accepted by a realtime-MC(*const*)-IA.

The inclusions claimed in 2.–4. follow from structural reasons. Hence, it remains for us to show each properness. We start with the inclusion claimed in 2. and consider language $L = \{ c^{2^{|w|}} wcw \mid w \in \{a,b\}^+ \}$ for which we show that it belongs to $\mathscr{L}_{rt}$(MC($\log(n)$)-IA), but not to $\mathscr{L}_{rt}$(MC(*const*)-IA). To construct a

realtime-MC($\log(n)$)-IA M for L we implement on track 1 the construction of 2^n given in Lemma 7. Additionally, we simulate on track 2 a queue store, where the top of the queue is located in the communication cell, and enter some symbol \$ into the queue at every time step at which the communication cell recognizes a time step 2^n for $n \geq 1$. Finally, we simulate on track 3 another queue store. As soon as the communication cell processes the first input symbol a or b from w, we check whether a time step 2^n has been identified in the last time step and we stop the computation on track 1. Additionally, we start to enter w to the queue on track 3 while for every symbol of w a symbol \$ from the queue on track 2 is removed. When the separating symbol c is processed by the communication cell, we check whether the queue on track 2 is empty. If so, the number of initial c's has exactly been $2^{|w|}$ and we can continue to check the remaining input against the contents of the queue on track 3. Finally, the input is accepted if the check is positive and in all other cases the input is rejected. Next, we want to estimate the maximum number of communications per cell. An accepted input has a length of $2^{|w|} + 2|w| + 1$. Due to Lemma 7 we know that at most $O(|w|)$ communications take place on the first track. To enter $|w|$ symbols into the queue on track 2 needs at most $|w|$ communications per cell. Finally, at most $2|w| + 1$ communications per cell can take place on track 2 and track 3 while processing the input suffix wcw. Altogether, at most $O(|w|)$ communications take place per cell. Thus, L can be accepted by a realtime-MC($\log(n)$)-IA.

To show that L is not accepted by any realtime-MC(*const*)-IA we combine two techniques which have successfully been applied for SC-CA [11] and for IA with a bounded constant number of different messages to be communicated [13]. First, we derive an upper bound for the number of different communications that the communication cell can perform while processing an input of length n and performing ℓ communications. We have to take into account the information to be communicated and the time steps at which the communication takes place. Since there are $\binom{n}{\ell}$ possibilities to choose time steps and $|B|$ different messages to be sent, we obtain, for some constant $k_0 \geq 1$, at most

$$\begin{aligned}\binom{n}{\ell}|B|^\ell &\leq \frac{n^\ell}{(\ell/2)^{\ell/2}} 2^{\log(|B|)\ell} = \frac{n^\ell 2^{\ell/2}}{\ell^{\ell/2}} 2^{\log(|B|)\ell} \\ &= 2^{\log(n)\ell + \ell/2 + \log(|B|)\ell - \log(\ell)\ell/2} \leq 2^{k_0 \log(n)\ell} = n^{k_0 \ell}\end{aligned}$$

possibilities. Now, we assume that L is accepted by realtime-MC(*const*)-IA M and we denote by k the constant number of maximal communications per cell. Moreover, let c_p be the configuration after processing the complete c-prefix of the input. Next, we want to calculate the number of different configurations of M starting in c_p and processing the first w part of an input. Such a configuration depends on the information that has been sent to the IA via the communication cell and the current state of the communication cell. Hence, there are at most $|w|^{k_0 \cdot k} \cdot |S|$ different configurations, where S denotes the state set of M. On the other hand, there are $2^{|w|}$ different words w. Now, we choose $|w|$ large enough such that $2^{|w|}$ is larger than the polynomial $|w|^{k_0 \cdot k} \cdot |S|$. Then, there are two different words $w \neq w'$ such that $|w| = |w'|$ and M enters the same

configuration after processing $c^{2^{|w|}}w$ as well as after processing $c^{2^{|w|}}w'$. Since the input $c^{2^{|w|}}wcw$ is accepted, input $c^{2^{|w|}}w'cw$ is accepted as well which is a contradiction.

The proof of claim 3. is similar. We consider $L' = \{\, c^{|w|^2}wcw \mid w \in \{a,b\}^+ \,\}$ and can construct in a similar way as above an MC($\sqrt{n}$)-IA accepting L' in realtime by taking into account that Lemma 7 shows that at most $O(|w|)$ communications per cell are necessary to time construct $|w|^2$. To show with the above technique that L' is not accepted by any realtime-MC($\log(n)$)-IA it is sufficient to choose $|w|$ large enough such that $2^{|w|} > |w|^{k_0 \log(|w|)} \cdot |S|$. This is possible since the latter inequality is equivalent to $|w| > k_0 \cdot |S| \cdot \log(|w|) \cdot \log(|w|)$ which holds for $|w|$ large enough.

Finally, we show 4. by considering $L' = \{\, wcw \mid w \in \{a,b\}^+ \,\}$ which is obviously accepted by a realtime-MC(n)-IA. On the other hand, we have to choose $|w|$ large enough such that $2^{|w|} > |w|^{k_0\sqrt{|w|}} \cdot |S|$ which is equivalent to $|w| > k_0 \cdot |S| \cdot \log(|w|) \cdot \sqrt{|w|}$ and holds for $|w|$ large enough. □

4 Undecidability Results for SC(n)-IA

In this section, we will show that almost all commonly studied decidability questions such as emptiness, finiteness, equivalence, inclusion, regularity, and context-freeness are not semidecidable for realtime-SC(n)-IA. Here, we say that a decision problem is *decidable (undecidable)* if the set of all instances for which the answer is "yes" is recursive (not recursive). A decision problem is said to be *semidecidable* if the set of all instances for which the answer is "yes" is recursively enumerable. It is known that the above-mentioned decidability questions are not semidecidable for realtime-IA [14]. Thus, the basic idea in the following is to find suitable languages that relate realtime-IA with realtime-SC(n)-IA. Let M be a realtime-IA over some alphabet A and a, b be symbols such that $A \cap \{a,b\} = \emptyset$. Then, we define language

$$L_M = \left\{\, wa^{2^{|w|}}b^{2^{|w|}} \mid w \in L(M) \,\right\}.$$

Lemma 9. *Let M be a realtime-IA. Then, $L_M \in \mathscr{L}_{rt}$(SC(n)-IA).*

Proof. We sketch the construction of a realtime-SC(n)-IA M' accepting L_M. The IA M' uses three tracks. Track 1 is used to simulate M where two cells of M are grouped into one cell of M'. Track 2 is used to store in a queue for every input symbol from A a certain symbol `$`. When the first a-symbol is read, the first $|w|$ cells of track 2 are marked with `$`. At this moment, we stop the simulation of M on track 1 and we start to increase a binary counter on track 3 as long as the input symbols are a. If the first b is read, we send a signal which checks whether the number of a's has been $2^{|w|}$. This can be done by inspecting the counter and checking whether exactly all cells marked with `$` have been used (see, e.g, Example 5). Additionally, we start to decrease the counter for every input symbol b. Finally, we accept the input if the counter has been decreased to zero and reject in all other cases.

The number of communications for the simulation of M on track 1 is bounded by $O(|w|^2)$. The number of communications to mark the first $|w|$ cells by \$ and to stop the simulation is bounded by $O(|w|)$. By observing that binary counters can be realized by SC(n)-IA, cf. Example 4, we know that the number of communications to increase the binary counter is bounded by $O(2^{|w|})$. The number of communications to check that the number of a's has been $2^{|w|}$ is bounded by $O(|w|)$. Finally, the number of communications to decrease the binary counter is bounded by $O(2^{|w|})$ as well. Altogether, the number of all communications is bounded by in $O(2^{|w|+1})$. Since the input length is $|w| + 2^{|w|+1}$, we obtain that the IA constructed is an SC(n)-IA. □

Now, the non-semidecidable property of realtime-IA M to accept the empty or a finite language, respectively, is reflected in properties of language L_M which enable us in the next theorem to obtain the desired non-semidecidability results for realtime-SC(n)-IA.

Lemma 10. *Let M be a realtime-IA.*

1. *L_M is empty if and only if $L(M)$ is empty.*
2. *L_M is finite if and only if $L(M)$ is finite.*
3. *L_M is regular if and only if $L(M)$ is finite.*
4. *L_M is context-free if and only if $L(M)$ is finite.*

Proof. Claim 1 and claim 2 are obvious. It can be shown by a standard application of the pumping lemma that the language L_M is not context-free, if $L(M)$ is infinite. On the other hand, if $L(M)$ is finite, L_M is finite as well. This shows claim 3. and claim 4. □

Theorem 11. *Emptiness, finiteness, infiniteness, equivalence, inclusion, regularity, and context-freeness are not semidecidable for realtime-SC(n)-IA.*

Proof. It is known that all above-mentioned questions are not semidecidable for realtime-IA due to the results given in [14]. By applying Lemma 9 and Lemma 10 we can immediately translate the non-semidecidability results to realtime-SC(n)-IA. □

Moreover, we cannot even semidecide the property of being sum communication bounded by n.

Theorem 12. *It is not semidecidable for an arbitrary realtime-IA M whether or not M is a realtime-SC(n)-IA.*

Finally, we can apply similar construction ideas as in Lemma 9 to separate the classes of realtime-MC(*const*)-IA and realtime-SC(n)-IA.

Theorem 13. $\mathscr{L}_{rt}(MC(const)\text{-}IA) \subset \mathscr{L}_{rt}(SC(n)\text{-}IA)$.

5 Undecidability Results for MC(log(n))-IA

In this section, we prove similar non-semidecidability results for MC(log(n))-IA by using similar methods as in the previous section. Let M be a realtime-IA over some alphabet A and c be a symbol such that $A \cap \{c\} = \emptyset$. Then, we define language

$$L_M = \left\{ c^{2^{|w|}} w \mid w \in L(M) \right\}.$$

Lemma 14. *Let M be a realtime-IA. Then, $L_M \in \mathscr{L}_{rt}(MC(\log(n))\text{-}IA)$.*

The proof of the following lemma and the following theorem is nearly identical to the proof of Lemma 10 and of Theorem 11, respectively.

Lemma 15. *Let M be a realtime-IA.*

1. *L_M is empty if and only if $L(M)$ is empty.*
2. *L_M is finite if and only if $L(M)$ is finite.*
3. *L_M is regular if and only if $L(M)$ is finite.*
4. *L_M is context-free if and only if $L(M)$ is finite.*

Theorem 16. *Emptiness, finiteness, infiniteness, equivalence, inclusion, regularity, and context-freeness are not semidecidable for realtime-MC(log(n))-IA.*

Finally, we cannot even semidecide the property of being max communication bounded by log(n).

Theorem 17. *It is not semidecidable for an arbitrary realtime-IA M whether or not M is a realtime-MC(log(n))-IA.*

We remark that it is currently an open question whether or not all discussed decidability questions are not semidecidable for realtime-MC(*const*)-IA as well.

Acknowledgment

Thanks are given to Victor Roussanaly for several discussions on the topic while his internship at our institute in 2014.

References

1. 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)
2. Choffrut, C., II, K.C.: On real-time cellular automata and trellis automata. Acta Inf. 21, 393–407 (1984)
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., Palis, M.A.: Some results concerning linear iterative (systolic) arrays. J. Parallel Distributed Comput. 2, 182–218 (1985)
6. Ibarra, O.H., Palis, M.A.: Two-dimensional iterative arrays: Characterizations and applications. Theoret. Comput. Sci. 57, 47–86 (1988)
7. Kutrib, M.: Cellular automata – a computational point of view. In: Bel-Enguix, G., Jiménez-López, M.D., Martín-Vide, C. (eds.) New Developments in Formal Languages and Applications, chap. 6, pp. 183–227. Springer (2008)
8. Kutrib, M.: Cellular automata and language theory. In: Meyers, R.A. (ed.) Encyclopedia of Complexity and Systems Science, pp. 800–823. Springer (2009)
9. Kutrib, M., Malcher, A.: Computations and decidability of iterative arrays with restricted communication. Parallel Processing Letters 19(2), 247–264 (2009)
10. Kutrib, M., Malcher, A.: On one-way one-bit $O(1)$-message cellular automata. Electr. Notes Theor. Comput. Sci. 252, 77–91 (2009)
11. Kutrib, M., Malcher, A.: Cellular automata with sparse communication. Theor. Comput. Sci. 411(38-39), 3516–3526 (2010)
12. Kutrib, M., Malcher, A.: One-way cellular automata, bounded languages, and minimal communication. J. Autom. Lang. Comb. 15(1/2), 135–153 (2010)
13. Kutrib, M., Malcher, A.: Cellular automata with limited inter-cell bandwidth. Theor. Comput. Sci. 412(30), 3917–3931 (2011)
14. Malcher, A.: On the descriptional complexity of iterative arrays. IEICE Trans. Inf. Syst. E87-D, 721–725 (2004)
15. Mazoyer, J., Terrier, V.: Signals in one-dimensional cellular automata. Theor. Comput. Sci. 217(1), 53–80 (1999)
16. Smith III, A.R.: Real-time language recognition by one-dimensional cellular automata. J. Comput. Syst. Sci. 6(3), 233–253 (1972)
17. Umeo, H., Kamikawa, N.: A design of real-time non-regular sequence generation algorithms and their implementations on cellular automata with 1-bit inter-cell communications. Fund. Inform. 52, 257–275 (2002)
18. Umeo, H., Kamikawa, N.: Real-time generation of primes by a 1-bit-communication cellular automaton. Fund. Inform. 58, 421–435 (2003)
19. Worsch, T.: Linear time language recognition on cellular automata with restricted communication. In: Gonnet, G.H., Panario, D., Viola, A. (eds.) Theoretical Informatics (LATIN 2000). LNCS, vol. 1776, pp. 417–426. Springer (2000)