Journées Automates Cellulaires 2010 (Turku), pp. 13-30

MEASURING COMMUNICATION IN CELLULAR AUTOMATA

MARTIN KUTRIB AND ANDREAS MALCHER

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

Abstract. Cellular automata and iterative arrays are one-dimensional arrays of interconnected interacting finite automata which work synchronously at discrete time steps. In this paper, the focus lies on the resource of communication which naturally takes place between cells. Communication is measured here with regard to qualitative and quantitative aspects. More detailed, the amount of communication in cellular automata is measured by limiting the bandwidth of the communication links between the cells, as well as limiting the number of messages allowed to be sent between cells. An overview on recent results on the computational capacity of as well as on decidability problems in such restricted cellular automata and iterative arrays is given.

1. Introduction

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

Here, we consider the model of cellular automata with parallel and sequential input mode. The latter model is also known as iterative array. In both models the state of each cell is communicated to its neighbors in every time step. That is, on the one hand the state is sent regardless of whether it is really required, and on the other hand, the number of different messages that may be sent is determined by the number of states. Thus, it seems to be natural to restrict this "unbounded" communication by limiting the number of possible different messages between cells. This brings us to cellular automata and iterative arrays where the bandwidth of the communication links between two cells is bounded by some fixed constant. In the most restricted setting this is one bit bandwidth. However, these automata are

2000 ACM Subject Classification: F.1.1, F.1.2, F.4.3.

Key words and phrases: cellular automata, iterative arrays, message complexity, limited communication, decidability, formal languages.

still powerful enough to accept unary as well as non-unary non-context-free (even non-semilinear) languages in real time. For some classes it is additionally known that almost all of the commonly investigated decidability problems are undecidable. Furthermore, we obtain proper hierarchies with regard to the resources bandwidth, dimension, and time. Finally, we get a complete picture on the relations between cellular automata and iterative arrays in the case of restricted inter-cell bandwidth.

For real-time one-way cellular automata where the communication is quantitatively measured by counting the number of uses of the communication links between cells, the total sum of all communications or the maximal number of communications that may appear between each two cells can be considered. Reducing the number of communications in such a way that each two neighboring cells may communicate constantly often only, leads to devices which also still can accept non-context-free (even non-semilinear) languages. Again, almost all of their decidability questions can be shown to be undecidable. An interesting additional restriction is to consider inputs of a certain form only. For such bounded languages it is known that in other computational models, such as certain variants of multi-head finite automata, undecidable problems become decidable. However, all commonly investigated decidability questions remain undecidable for communication-restricted real-time one-way cellular automata accepting bounded languages.

Finally, both limitations on the communication between cells are combined. It turns out that even this restriction does not lead to positive decidability results. The known undecidability results can be translated to the case of one and two message bandwidth. Thus, the resource communication makes the model in a way inherently complex since even a limitation to a very small amount of communication does not reduce the computational complexity of the model's undecidability problems.

2. Preliminaries and Definitions

We denote the rational numbers by $\mathbb{Q}$, and the non-negative integers 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 $|w|$. The set of words over some alphabet A whose lengths are at most $l \in \mathbb{N}$ is denoted by $A^{\leq l}$. We write $\subseteq$ for set inclusion, and $\subset$ for strict set inclusion. The cardinality of a set M is denoted by $|M|$.

A one-dimensional iterative array is a linear, semi-infinite array of identical deterministic finite automata, sometimes called cells. The finite automata work synchronously at discrete time steps. Except for the leftmost cell each one is connected to its both nearest neighbors (see Figure 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 state transition of all cells but the input cell depends on the current state of the cell itself and the current states of its neighbors. The state transition of the input cell additionally depends on the input symbol to be read next. The head of the one-way input tape is moved at any step to the right. With an eye towards recognition problems the machines have no extra output tape but the states are partitioned into accepting and rejecting states.

In an iterative array with *k-message restricted inter-cell communication*, the state transition depends on the current state of each cell and on the messages that are currently sent by its neighbors, where the possible messages are formalized as a set of possible communication symbols. The messages to be sent by a cell depend on its current state and are determined by so-called communication functions.

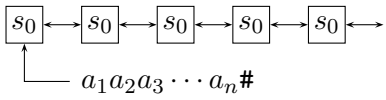

Figure 1: Initial configuration of an iterative array.

Definition 2.1. A *one-dimensional iterative array with k-message restricted inter-cell communication* (IA_k) is a system $\langle S, A, B, F, \#, s_0, b_l, b_r, \delta, \delta_0\rangle$, where

(1) S is the finite, nonempty set of *cell states*,
(2) A is the finite, nonempty set of *input symbols*,
(3) B with $|B| = k$ is the finite set of *communication symbols*,
(4) $F \subseteq S$ is the set of *accepting states*,
(5) $\# \notin A$ is the *end-of-input symbol*,
(6) $s_0 \in S$ is the *quiescent state*,
(7) $b_l, b_r : S \to B \cup \{\perp\}$ are *communication functions* which determine the information *to be sent* to the left and right neighbors, where $\perp$ means *nothing to send*,
(8) $\delta : (B \cup \{\perp\}) \times S \times (B \cup \{\perp\}) \to S$ is the *local transition function for all but the input cells* satisfying $\delta(b_r(s_0), s_0, b_l(s_0)) = s_0$,
(9) $\delta_0 : S \times (A \cup \{\#\}) \times (B \cup \{\perp\}) \to S$ is the *local transition function for the input cell*.

Let $\mathcal{M}$ be an IA_k. A configuration of $\mathcal{M}$ 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: 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 $c_{t+1}(0) = \delta_0(c_t(0), a, b_l(c_t(1)))$ where $a = \#$ 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 $w_t = a_1 \cdots a_n$. Thus, the global transition function Δ is induced by δ and δ_0.

A *two-way cellular automaton with k-message restricted inter-cell communication* is similar to an iterative array. The main difference is that the cell at the origin does not fetch the input but the input is supplied in parallel to the cells. That is, an input $a_1 \cdots a_n$ is fed to the cells $1, \ldots, n$ such that initially cell i is in state a_i. Cells 0 and $n+1$ are initially in a permanent so-called *boundary* state # (see Figure 2). Cell 1 indicates acceptance or rejection, and the array is bounded to the n cells which are initially active.

Definition 2.2. A *cellular automaton with k-message restricted inter-cell communication* (CA_k) is a system $\langle S, A, B, F, \#, b_l, b_r, \delta\rangle$, where

(1) S is the finite, nonempty set of *cell states*,
(2) $A \subseteq S$ is the finite, nonempty set of *input symbols*,
(3) B with $|B| = k$ is the finite set of *communication symbols*,
(4) $F \subseteq S$ is the set of *accepting states*,
(5) $\# \notin S$ is the *boundary state*,
(6) $b_l, b_r : (S \cup \{\#\}) \to (B \cup \{\perp\})$ are *communication functions* which determine the information *to be sent* to the left and right neighbors, where $\perp$ means *nothing to send*,
(7) $\delta : (B \cup \{\perp\}) \times S \times (B \cup \{\perp\}) \to S$ is the *local transition function*.

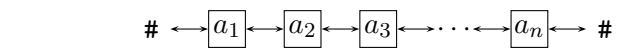

Figure 2: A two-way cellular automaton.

A *one-way cellular automaton* (OCA_k) is a cellular automaton in which each cell receives information from its immediate neighbor to the right only. So, the flow of information is restricted to be from right to left. Formally, δ is a mapping from $S \times (B \cup \{\perp\})$ to S.

A *configuration* of a cellular automaton $\langle S, A, B, F, \#, b_l, b_r, \delta\rangle$ at time $t \geq 0$ is a mapping $c_t : \{1, \ldots, n\} \to S$, for $n \geq 1$. For a given input $w = a_1 \cdots a_n \in A^+$ we set the initial configuration $c_{0,w}(i) = a_i$, for $1 \leq i \leq n$. Successor configurations are computed according to the global transition function Δ:

Let c_t, $t \geq 0$, be a configuration. Then its successor configuration $c_{t+1} = \Delta(c_t)$ is as follows.

$$\begin{aligned} c_{t+1}(1) &= \delta(b_r(\#), c_t(1), b_l(c_t(2))) \\ c_{t+1}(i) &= \delta(b_r(c_t(i-1)), c_t(i), b_l(c_t(i+1))), i \in \{2, \ldots, n-1\} \\ c_{t+1}(n) &= \delta(b_r(c_t(n-1)), c_t(n), b_l(\#)) \end{aligned}$$

for CA_k and

$$\begin{aligned} c_{t+1}(i) &= \delta(c_t(i), b_l(c_t(i+1))), i \in \{1, \ldots, n-1\} \\ c_{t+1}(n) &= \delta(c_t(n), b_l(\#)) \end{aligned}$$

for OCA_k.

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

The family of all languages which are accepted by some device X with time complexity t is denoted by $\mathscr{L}_t(X)$. If t is the function $n+1$, for IA_k, or the function n, for $(\mathrm{O})\mathrm{CA}_k$, acceptance is said to be in *real time* and we write $\mathscr{L}_{rt}(X)$. Since for nontrivial computations an IA_k has to read at least one end-of-input symbol, real time has to be defined as $(n+1)$-time. The *linear-time* languages $\mathscr{L}_{lt}(X)$ are defined according to $\mathscr{L}_{lt}(X) = \bigcup_{r \in \mathbb{Q},\, r \geq 1} \mathscr{L}_{r \cdot n}(X)$.

3. What Arrays with Limited Inter-Cell Bandwidth Can Do

For iterative arrays and cellular automata where the bandwidth of the communication links between two cells is bounded by some fixed constant, the most

restricted setting is one message bandwidth. However, these devices are still powerful enough to solve problems such as the firing squad synchronization problem in optimal time [22]. Moreover, it is known [28, 29] that one-message IA_1 can accept rather complicated unary languages in real time, for example, words whose lengths are powers of two $\{ a^{2^n} \mid n \geq 1 \}$, words whose length are square numbers $\{ a^{n^2} \mid n \geq 1 \}$, words whose length are prime numbers $\{ a^p \mid p \text{ is prime} \}$, or words whose lengths are Fibonacci numbers. Clearly, every deterministic finite automaton can be simulated in the input cell of an iterative array. Thus, we obtain the strict inclusion

$$\mathrm{REG} \subset \mathscr{L}_{rt}(\mathrm{IA}_1)$$

where REG denotes the regular languages.

The next lemma shows that an additive speed-up in IA_k is always possible [17]. The benefit of this lemma is that we do not have to care about additive constant factors which may arise in constructions. The situation is different for cellular automata where we have an infinite, strict, and tight hierarchy depending on the additive constant (see [32] for OCA_k, and Theorem 5.1 for CA_k).

Lemma 3.1. *$\mathscr{L}_{rt+p}(IA_k) = \mathscr{L}_{rt}(IA_k)$, for any constant number $p \geq 1$.*

In [17] it is shown that one-message IA_1 can simulate binary counters in real time. This construction is often a useful tool for the modular design of algorithms. For example, by exploiting counters it is possible to check certain properties of distances between two designated cells, to verify that something happens at a certain time step, to implement clocks and pulses, and much else besides. In general, a specific application of a counter requires some additional mechanisms. For example, it might be of interest to recognize the time step at which a counter overflows or becomes zero, instead of knowing the real counter value. The proofs of some of the next results make extensively use of the ability of one-message IA_1 to simulate binary counters attached to the input cell, which stores the least significant bit. The counter either can be incremented or decremented, and can be tested for zero. Moreover, it is possible to use two counters and to switch from one counter to another with the minimal resources time and communication bandwidth.

The following language is known not to be regular. On the other hand, it can be accepted by pushdown automata making at most one turn.

Lemma 3.2. $\{ a^n b^n \mid n \geq 0 \} \in \mathscr{L}_{rt}(IA_1)$.

The proof of the previous lemma is based on a construction that allows to switch between an increasing counter and a decreasing counter by using some signal. The switching costs four additional time steps. In general, any *constant* number of such switchings can be realized. The resulting device can be made real-time again by Lemma 3.1. Furthermore, we can do more complicated computations based on the same technique. For example, if there are some subroutines sharing the same communication, then the switching signals can be interpreted individually by the routines. As before, the input cell controls the time steps at which the signals are sent. This idea is applied in the construction showing the next result. The language $\{ a^n b^n c^n \mid n \geq 0 \}$ is not accepted by any pushdown automaton, but it is accepted by some one-message IA_1 in real time.

Lemma 3.3. $\{ a^n b^n c^n \mid n \geq 0 \} \in \mathscr{L}_{rt}(IA_1)$.

By similar constructions one can set up real-time one-message IA_1 that recognize languages of the form $\{\, a^n b^n c^n d^n \mid n \geq 0 \,\}$ or $\{\, a^n b^m c^n d^m \mid n, m \geq 0 \,\}$. The important observation is that the number of switching signals to be sent is constant. Glimpsing at the language $\{\, a^n (b^n)^m \mid n, m \geq 0 \,\}$ one receives the impression that the number of switches is about m, which is no longer constant. So, the language might be too hard to be accepted by real-time one-message IA_1. In fact, in order to show the converse we cannot use the techniques developed so far. But instead we can *reuse counter values* in a sense, that once a counter is decremented to zero, it restarts to count from its previous initial value. To this end, a master copy of the original counter value has to be kept in another counter. The construction is sketched in [17].

Lemma 3.4. $\{\, a^n (b^n)^m \mid n, m \geq 0 \,\} \in \mathscr{L}_{rt}(IA_1)$.

Next, we turn to some integer calculations, where the problem instances are suitably encoded and consider the problems of "adding" negative and positive integers as well as "multiplying" non-negative integers [17].

Lemma 3.5. $\{\, a^n b^m c^l \mid n, m, l \geq 0 \text{ and } -n + m = l \,\} \in \mathscr{L}_{rt}(IA_1)$.

In order to construct a real-time one-message IA_1 for the next languages we have to handle counters that do not share the same communication. If we have a constant number of counters to handle, then the *simulation can be time-shared*, that is, the single steps of the counters are simulated cyclically one after the other. Clearly, in general this may violate the real-time constraint. But for our purposes it works fine.

Lemma 3.6. $\{\, a^n b a^m b (ba)^{n \cdot m} \mid n, m \geq 0 \,\} \in \mathscr{L}_{rt}(IA_1)$.

Let $a_1, a_2, \ldots, a_k, \$, a, b$ be $k+3$ different symbols. Obvious generalizations of the construction (using several counters) show that, for example, the languages

$\{\, a_1^n a_2^m \$ (ba)^{n \cdot m} \mid n, m \geq 0 \,\}$,
$\{\, a_1^n a_2^m a_3^l \$ (baa)^{n \cdot m \cdot l} \mid n, m, l \geq 0 \,\}$,
$\{\, a_1^{\alpha_1} a_2^{\alpha_2} \cdots a_k^{\alpha_k} \$ (b a^{k-1})^{\alpha_1 \cdot \alpha_2 \cdots \alpha_k} \mid \alpha_1, \alpha_2, \ldots, \alpha_k \geq 0 \,\}$,
$\{\, a^n b a^m b (baaaa)^{n^2 \cdot m^3} \mid n, m \geq 0 \,\}$,
$\{\, a_1^n a_2^m a_3^l \$ (baaaaaa)^{n^2 \cdot m^3 \cdot l^2} \mid n, m, l \geq 0 \}$ and, given constants $j_1, j_2, \ldots, j_k \geq 0$,
$\{a_1^{\alpha_1} a_2^{\alpha_2} \cdots a_k^{\alpha_k} \$ (b a^{j_1 + j_2 + \cdots + j_k - 1})^{\alpha_1^{j_1} \cdot \alpha_2^{j_2} \cdots \alpha_k^{j_k}} \mid \alpha_1, \alpha_2, \ldots, \alpha_k \geq 0 \,\}$

are accepted by one-message IA_1 in real-time.

4. What Some of the Arrays with Limited Inter-Cell Bandwidth Can Do

To obtain a valuable tool to show that certain languages cannot be accepted by iterative arrays with limited inter-cell bandwidth, we define two equivalence relations and derive upper bounds on the number of equivalence classes which can be distinguished by IA_k. If the number of equivalence classes induced by a certain language exceeds this upper bound, the language is not accepted by any IA_k.

Definition 4.1. Let $L \subseteq A^*$ be a language over an alphabet A and $l \geq 1$ be a constant.

(1) Two words $w \in A^*$ and $w' \in A^*$ are l-right-equivalent with respect to L if for all $y \in A^{\leq l}$: $wy \in L \iff w'y \in L$.
(2) $N_r(l, L)$ denotes the number of l-right-equivalence classes with respect to L.
(3) Two words $w \in A^{\leq l}$ and $w' \in A^{\leq l}$ are l-left-equivalent with respect to L if for all $y \in A^*$: $wy \in L \iff w'y \in L$.
(4) $N_\ell(l, L)$ denotes the number of l-left-equivalence classes with respect to L.

The next lemma provides upper bounds for the number of equivalence classes distinguished by IA_k.

Lemma 4.2. *Let $k \geq 1$ be a constant.*

(1) *If $L \in \mathscr{L}_{rt}(IA_k)$, then there exists a constant $p \geq 0$ such that*

$$N_r(l, L) \leq p^{(l+1)}$$

and

(2) *if $L \in \mathscr{L}_t(IA_k)$, then there exists a constant $p \geq 0$ such that*

$$N_\ell(l, L) \leq p \cdot (k+1)^l$$

for all $l \geq 1$ and all time complexities $t : \mathbb{N} \to \mathbb{N}$.

Now, we fix the time complexity to real time and consider the number of different messages. For any number of messages $k \geq 1$ we define an alphabet $A_k = \{a_0, \ldots, a_k\}$ and a language

$$L_{bit}(k) = \{\, e^x \$ u_1 u_2 \cdots u_m \mid x \geq 1 \text{ and } m \geq 2x-1 \text{ and } u_i \in A_k, 1 \leq i \leq m, \text{ and } u_j = u_{j+2x-1}, 1 \leq j \leq m-(2x-1) \,\}.$$

The languages $L_{bit}(k)$ are witnesses for an infinite, strict, and tight hierarchy dependent on the number of different messages [14].

Theorem 4.3. *Let $k \geq 1$ be a constant. The language $L_{bit}(k+1)$ belongs to the difference $\mathscr{L}_{rt}(IA_{k+1}) \setminus \mathscr{L}_{rt}(IA_k)$. Therefore, $\mathscr{L}_{rt}(IA_k) \subset \mathscr{L}_{rt}(IA_{k+1})$.*

The previous result can be extended to iterative arrays of any higher dimension [14]. Without formal definition we mention that a d-dimensional iterative array IA^d is an iterative array, where the cells are arranged as d-dimensional grid ($\mathbb{N}^d$) as extention of a line ($\mathbb{N}^1$). Each cell is connected to its immediate neighbors in any dimension. Moreover, we fix the time complexity to real time, the number of different messages to constants $k \geq 1$, and consider the dimension. For any dimension $d \geq 2$ we define a language $L_{dim}(d)$ as follows. We start with a series of regular sets:

$$X_1 = \$\{a, b\}^+, \qquad X_{i+1} = \$X_i^+, \text{ for } i \geq 1.$$

Due to the separator symbol \$, every word $u \in X_{i+1}$ can uniquely be decomposed into its subwords from X_i. So, we can define the projection on the jth subword as usual: Let $u = \$u_1 \cdots u_m$, where $u_j \in X_i$, for $1 \leq j \leq m$. Then $u[j]$ is defined to be u_j, if $1 \leq j \leq m$, otherwise $u[j]$ is undefined. Now define the language

$$M(d) = \{\, u ¢ e^{x_d} \$ \cdots \$ e^{x_1} \$ e^{2x} \$ v \mid u \in X_d \text{ and } x_i \geq 1, 1 \leq i \leq d, \text{ and } x = x_1 + \cdots + x_d \text{ and } v = u[x_d][x_{d-1}] \cdots [x_1] \text{ is defined} \,\}.$$

Finally, the language $L_{dim}(d)$ is given as homomorphic image of $M(d)$. More precisely, $L_{dim}(d) = h(M(d))$, where $h : \{a, b, e, \$, ¢\}^* \to \{a, b\}^*$ is defined by: $h(a) = ba$, $h(b) = bb$, $h(e) = b$, $h(\$) = ab$, $h(¢) = aa$.

Theorem 4.4. *Let $d \geq 1$ and $k \geq 1$ be constants. The language $L_{dim}(d+1)$ belongs to the difference $\mathscr{L}_{rt}(IA_1^{d+1}) \setminus \mathscr{L}_{rt}(IA_k^d)$. Therefore, $\mathscr{L}_{rt}(IA_k^d) \subset \mathscr{L}_{rt}(IA_k^{d+1})$.*

Interestingly, $L_{dim}(d+1)$ belongs to $\mathscr{L}_{lt}(\mathrm{IA}_1^1)$, that is, one can trade all dimensions and messages for a slow-down from real time to linear time.

Theorem 4.5. *Let $d \geq 1$ and $k \geq 1$ be constants. Then $\mathscr{L}_{rt}(IA_k^d) \subset \mathscr{L}_{lt}(IA_k^d)$.*

From Theorem 4.4 and Theorem 4.3 we obtain a double hierarchy concerning messages and dimensions which is depicted in Figure 3.

$$
\begin{array}{ccccccccc}
\vdots & & \vdots & & & & \vdots & & \\
\cup & & \cup & & & & \cup & & \\
\mathscr{L}_{rt}(\mathrm{IA}_1^d) & \subset & \mathscr{L}_{rt}(\mathrm{IA}_2^d) & \subset & \cdots & \subset & \mathscr{L}_{rt}(\mathrm{IA}_k^d) & \subset & \cdots \\
\cup & & \cup & & & & \cup & & \\
\vdots & & \vdots & & & & \vdots & & \\
\cup & & \cup & & & & \cup & & \\
\mathscr{L}_{rt}(\mathrm{IA}_1^2) & \subset & \mathscr{L}_{rt}(\mathrm{IA}_2^2) & \subset & \cdots & \subset & \mathscr{L}_{rt}(\mathrm{IA}_k^2) & \subset & \cdots \\
\cup & & \cup & & & & \cup & & \\
\mathscr{L}_{rt}(\mathrm{IA}_1^1) & \subset & \mathscr{L}_{rt}(\mathrm{IA}_2^1) & \subset & \cdots & \subset & \mathscr{L}_{rt}(\mathrm{IA}_k^1) & \subset & \cdots
\end{array}
$$

Figure 3: Double hierarchy of fast IA with restricted inter-cell communication.

5. What Arrays with Limited Inter-Cell Bandwidth Cannot Do

The main difference between cellular automata and iterative arrays is that the input is processed in parallel by the former model and processed sequentially by the latter model. An interesting variant of cellular automata is the restriction to one-way information flow. The relations between iterative arrays and cellular automata with two-way and one-way information flow are summarized in the left part of Figure 4. Here, we will clarify the relation between the discussed language classes in the case of restricted communication. It turns out that we obtain a finer hierarchy than in the unrestricted case. The results are depicted in the right part of Figure 4.

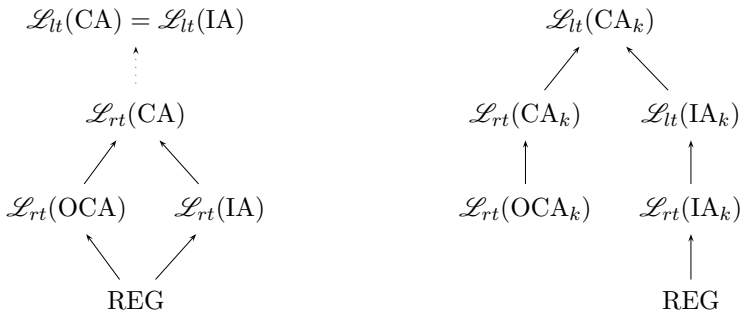

Figure 4: Relations between unrestricted (left) and restricted (right) language families. Solid arrows are strict inclusions and dotted arrows are inclusions. Families which are not connected by any path are incomparable.

The first difference between language classes with and without communication restrictions is that there are regular languages which cannot be accepted by any k-message cellular automaton CA_k even if we add a constant number of time steps to real time, whereas all regular languages are accepted in the unrestricted case. Moreover, the witness languages can be accepted by a real-time CA_{k+1}. Thus, we obtain a strict message hierarchy for two-way real-time cellular automata. Furthermore, the witness languages are accepted by $(n+r+1)$-time CA_k. Thus, we obtain a very dense strict time hierarchy. If we allow just one more time step, we obtain a strictly more powerful device.

Theorem 5.1. *Let $k \geq 1$ and $p \geq 0$ be constants.*

(1) *There is a regular language which is not accepted by any $(n+p)$-time CA_k.*
(2) *Then $\mathscr{L}_{rt+p}(CA_k) \subset \mathscr{L}_{rt+p}(CA_{k+1})$.*
(3) *Then $\mathscr{L}_{rt+p}(CA_k) \subset \mathscr{L}_{rt+p+1}(CA_k)$.*

If the communication channels of the CA have a sufficient capacity, the regular languages are accepted.

Lemma 5.2. *Let $k \geq 1$ be a constant. Then every regular language over a k-letter alphabet is accepted by some real-time CA_k.*

Together we obtain a two-dimensional infinite hierarchy for cellular automata with restricted communication concerning the number of messages communicated and the number of time steps performed, which is depicted in Figure 5.

$$
\begin{array}{ccccccccc}
\vdots & & \vdots & & & & \vdots & & \\
\cup & & \cup & & & & \cup & & \\
\mathscr{L}_{rt+r}(\mathrm{CA}_1) & \subset & \mathscr{L}_{rt+r}(\mathrm{CA}_2) & \subset & \cdots & \subset & \mathscr{L}_{rt+r}(\mathrm{CA}_k) & \subset & \cdots \\
\cup & & \cup & & & & \cup & & \\
\vdots & & \vdots & & & & \vdots & & \\
\cup & & \cup & & & & \cup & & \\
\mathscr{L}_{rt+1}(\mathrm{CA}_1) & \subset & \mathscr{L}_{rt+1}(\mathrm{CA}_2) & \subset & \cdots & \subset & \mathscr{L}_{rt+1}(\mathrm{CA}_k) & \subset & \cdots \\
\cup & & \cup & & & & \cup & & \\
\mathscr{L}_{rt}(\mathrm{CA}_1) & \subset & \mathscr{L}_{rt}(\mathrm{CA}_2) & \subset & \cdots & \subset & \mathscr{L}_{rt}(\mathrm{CA}_k) & \subset & \cdots
\end{array}
$$

Figure 5: Two-dimensional infinite hierarchy of CA with restricted communication.

The next theorem clarifies the relation between iterative arrays and one-way cellular automata.

Theorem 5.3. *Let $k \geq 1$ be a constant. There is a language belonging to the difference $\mathscr{L}_{rt}(OCA_1) \setminus \mathscr{L}_{lt}(IA_k)$.*

Next, we show proper inclusions between language families that are related by inclusions for structural reasons. In [5] an unrestricted real-time iterative array accepting prime numbers in unary has been constructed. In [29] the result has been improved to a one-message iterative array. However, while in the unrestricted case any real-time iterative array can be simulated by a real-time two-way cellular automaton, here we have seen that both devices define incomparable language families. By a different witness language the next result follows.

Theorem 5.4. *Let $k \geq 1$ be a constant. Then $\mathscr{L}_{rt}(OCA_k) \subset \mathscr{L}_{rt}(CA_k)$.*

The next result says that for cellular automata with restricted communication a parallel processing of the input is more powerful than a sequential processing under linear time conditions. This is in contrast to the unrestricted case where both input modes imply the same computational capacity.

Theorem 5.5. *Let $k \geq 1$ be a constant. Then $\mathscr{L}_{lt}(IA_k) \subset \mathscr{L}_{lt}(CA_k)$.*

We complement our considerations with the following theorem.

Theorem 5.6. *Let $k \geq 1$ be a constant. Then the language families $\mathscr{L}_{rt}(OCA_k)$ and $\mathscr{L}_{rt}(CA_k)$ are incomparable with REG, $\mathscr{L}_{rt}(IA_k)$, and $\mathscr{L}_{lt}(IA_k)$.*

It is a long-standing open problem whether linear-time cellular automata are more powerful than real-time cellular automata. Since linear-time CA_k can accept all regular languages whereas real-time CA_k cannot, we can answer this question in the affirmative for the case of restricted communication. On the other hand, Theorem 5.5 says that in the case of restricted communication parallel input mode implies a more powerful model than sequential input mode whereas both input modes are equally powerful in the unrestricted case.

In Figure 6 we summarize the relations between cellular automata with restricted and unrestricted communication [13].

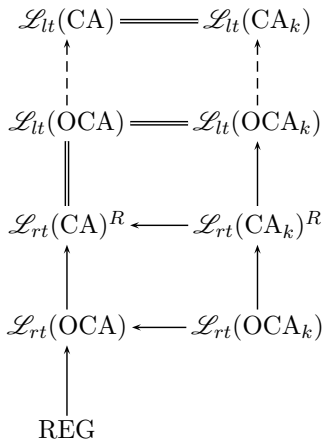

Figure 6: Relations between unrestricted and restricted language families. Solid arrows are strict inclusions, dashed arrows are inclusions, and double lines denote equivalence.

Decidability Questions

For real-time IA with unrestricted communication it is known that many decidability questions are undecidable [21, 27]. One may ask under which additional conditions undecidable questions become decidable. Since the recognizing power of real-time IA_k is weaker than general real-time IA, it is a natural question whether undecidable questions for real-time IA are decidable for IA_k. But this question has been answered negatively [17]. In fact, several of the common decidability questions are even *non-semidecidable* for real-time IA_1. A formal problem is *semidecidable*, if the algorithm halts on all instances for which the answer is *yes*. Thus, non-semidecidability results for real-time IA_1 show that even real-time IA with a minimum amount of communication are very powerful devices.

A well-known non-semidecidable problem is emptiness for Turing machines (or algorithms) [4, 26]. The proof of the non-semidecidability results relies on a reduction of the emptiness problem for Turing machines. To this end, Turing machine computations are encoded into small grammars [7]. Roughly speaking, *valid computations of Turing machines* are histories of accepting Turing machine computations. It suffices to consider deterministic Turing machines with a single tape and a single read-write head. Without loss of generality and for technical reasons, one can assume that any accepting computation has at least three and, in general, an odd number of steps. Therefore, it is represented by an even number of configurations. Moreover, it is assumed that the Turing machine cannot print blanks, and that a configuration is halting if and only if it is accepting.

Let S be the state set of some Turing machine $\mathcal{M}$, where s_0 is the initial state, $T \cap S = \emptyset$ is the tape alphabet containing the blank symbol, $A \subset T$ is the set of input symbols, and $F \subseteq S$ is the set of accepting states. Then a configuration of $\mathcal{M}$ can be written as string of the form T^*ST^* such that $t_1 \cdots t_i s t_{i+1} \cdots t_n$ is used to express that $\mathcal{M}$ is in state s, scanning tape symbol t_{i+1}, and t_1 to t_n is the non-blank part of the tape inscription. The set of *valid computations* $\mathrm{VALC}(\mathcal{M})$ is now defined to be the set of patterns of the form $w_1\$w_3\$\cdots\$w_{2k-1}\¢w_{2k}^R\$\cdots\$w_4^R\$w_2^R$, where w_i are configurations, $\$$ and $¢$ are symbols not appearing in w_i, w_1 is an initial configuration of the form s_0A^*, w_{2k} is an accepting configuration of the form T^*FT^*, and w_{i+1} is the successor configuration of w_i, for $1 \leq i \leq 2k$. The set of *invalid computations* $\mathrm{INVALC}(\mathcal{M})$ is the complement of $\mathrm{VALC}(\mathcal{M})$ with respect to the coding alphabet $\{\$, ¢\} \cup T \cup S$.

The simple but nevertheless important and fruitful observation is that the set $\mathrm{VALC}(\mathcal{M})$ is empty if and only if the set accepted by the Turing machine $\mathcal{M}$ is empty. The following lemma is the starting point of the reductions. It has been shown in [21], see also [12].

Lemma 5.7. *Given a Turing machine $\mathcal{M}$, (general) real-time IA can effectively be constructed that accept VALC($\mathcal{M}$) and INVALC($\mathcal{M}$).*

To establish non-semidecidability results for real-time IA_1 we need some variation of the set of valid computations which still has the property that the set is empty if and only if the corresponding Turing machine accepts the empty set. Thus, we proceed as follows. Given a set of symbols $A = \{a_1, a_2, \ldots, a_n\}$, for all $k \geq 1$ we define a homomorphism h_k by $h_k(a_i) = a_i^k$, $1 \leq i \leq n$.

Lemma 5.8. *Given a Turing machine $\mathcal{M}$, there exists a number $k \geq 1$ such that real-time IA_1 can effectively be constructed that accept $h_k(VALC(\mathcal{M}))$ and $h_k(INVALC(\mathcal{M}))$.*

Clearly, the application of the homomorphism preserves the property mentioned before: The set $h_k(\mathrm{VALC}(\mathcal{M}))$ is empty if and only if the set accepted by the Turing machine $\mathcal{M}$ is empty. So, we obtain the next result immediately.

Theorem 5.9. *Emptiness, universality, finiteness, infiniteness, equivalence, and inclusion are undecidable for real-time IA_1.*

With respect to language theory there are more relations between (in)valid computations and non-semidecidable problems. The following reasonings originate from [7], see also [12, 21].

Let $\mathcal{M}$ be some Turing machine, and assume that $\mathcal{M}$ accepts a finite set. Then $\mathrm{VALC}(\mathcal{M})$ is finite and, clearly, context free. If conversely $\mathcal{M}$ accepts an infinite

set, then an application of the pumping lemma shows that VALC($\mathcal{M}$) is not context free. Therefore, $\mathcal{M}$ accepts a finite set if and only if VALC($\mathcal{M}$) is context free.

Moreover, if $\mathcal{M}$ accepts a finite set, then VALC($\mathcal{M}$) is finite and its complement INVALC($\mathcal{M}$) is regular. Conversely, if INVALC($\mathcal{M}$) is regular, then VALC($\mathcal{M}$) is regular since the regular languages are closed under complementation. Therefore, VALC($\mathcal{M}$) is context free which implies that $\mathcal{M}$ accepts a finite set. Therefore, $\mathcal{M}$ accepts a finite set if and only if INVALC($\mathcal{M}$) is regular.

Corollary 5.10. *Regularity and context-freeness are undecidable for real-time IA_1.*

Now, the previous results can be applied to show that there is no general pumping lemma and no minimization algorithm for real-time IA_1. In general, a family of languages possesses a *pumping lemma in the narrow sense* if for each language L from the family there exists a constant $n \geq 1$ computable from L such that each $z \in L$ with $|z| > n$ admits a factorization $z = uvw$, where $|v| \geq 1$ and $u'v^iw' \in L$, for infinitely many $i \geq 0$. The prefix u' and the suffix w' depend on u, w and i.

Theorem 5.11.

(1) *The family of languages accepted by real-time IA_1 does not possess a pumping lemma (in the narrow sense).*
(2) *There is no minimization algorithm converting some real-time IA_1 to an equivalent real-time IA_1 having a minimal number of states.*

6. Limiting the Number of Messages

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

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

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

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

Let $f : \mathbb{N} \to \mathbb{N}$ be a mapping. If all $w \in L(\mathcal{M})$ are accepted with computations where $\mathrm{mcom}(w) \leq f(|w|)$, then $\mathcal{M}$ is said to be *max communication bounded by f*. Similarly, if all $w \in L(\mathcal{M})$ are accepted with computations where $\mathrm{scom}(w) \leq f(|w|)$, then $\mathcal{M}$ is said to be *sum communication bounded by f*. In general, it is not expected to have tight bounds on the exact number of communications but tight bounds on their numbers in the order of magnitude. For the sake of readability we denote the class of CA that are max communication bounded by some function $g \in O(f)$ by MC(f)-CA, where it is understood that f gives the order of magnitude. In addition,

we use the notation *const* for functions from $O(1)$. Corresponding notations are used for OCA and sum communication bounded CA and OCA. (SC(f)-CA and SC(f)-OCA).

Computational Capacity

In order to identify the computational power of communication bounded real-time devices we begin by describing the relationship to previous works. In [30, 31] two-way cellular automata are considered where the number of proper state changes is bounded. Similar as in the present paper the sum of all state changes or the maximal number of the state changes of single cells are bounded. By applying the technique of saving communication steps by storing the last signal received in the state and to interpret an arriving $\perp$ suitably [19], it is not hard to see, that such a device can be simulated by the corresponding communication bounded device. Whether or not state change bounded devices are strictly weaker than communication bounded ones is an open problem. However, we adapt some of the results shown in connection with state changes in the next theorem.

Theorem 6.1 ([30, 31])**.**
(1) $\mathscr{L}_{rt}(MC(const)\text{-}CA) \subset \mathscr{L}_{rt}(SC(n)\text{-}CA)$.
(2) $REG \subset \mathscr{L}_{rt}(MC(const)\text{-}CA) \subset \mathscr{L}_{rt}(MC(\sqrt{n})\text{-}CA) \subset \mathscr{L}_{rt}(MC(n)\text{-}CA)$.
(3) $\mathscr{L}_{rt}(MC(const)\text{-}CA) \subset NL$.

In order to clarify our notion and to show the power of the devices in question, we give some examples of languages which can be accepted by MC(*const*)-OCA. It has been shown in [19] that the family MC(*const*)-OCA contains the non-context-free languages $\{\, a_1^n a_2^n \cdots a_k^n \mid n \geq 1 \,\}$ for $k \geq 2$, $\{\, a^n b^m c^n d^m \mid n, m \geq 1 \,\}$, as well as the languages $\{\, a^n w \mid n \geq 1 \wedge w \in (b^*c^*)^k b^* \wedge |w|_b = n \,\}$, for all constants $k \geq 0$. All of these languages are either semilinear or non-bounded. But in contrast to many other computational devices, for example certain multi-head finite automata, parallel communicating finite automata, and certain parallel communicating grammar systems, MC(*const*)-OCA can accept non-semilinear bounded languages. This is shown in the next example.

Example 6.2. The language $L = \{\, a^n b^{n+\lfloor\sqrt{n}\rfloor} \mid n \geq 1 \,\}$ belongs to the family $\mathscr{L}_{rt}$(MC(*const*)-OCA).

In [23] a CA is constructed such that its cell n enters a designated state exactly at time step $2n+\lfloor\sqrt{n}\rfloor$, and at most n cells are used for the computation. In fact, the CA constructed is actually an OCA. Additionally, each cell performs only a finite number of communication steps. Thus, the CA constructed is an MC(*const*)-OCA.

An MC(*const*)-OCA accepting L implements the mirror of the above construction on the a-cells of the input $a^n b^m$. Thus, the leftmost cell enters the designated state q at time step $2n + \lfloor\sqrt{n}\rfloor$. Additionally, in the rightmost cell a signal s with maximum speed is sent to the left. When this signal arrives in an a-cell exactly at a time step at which the cell would enter the designated state q, the cell changes to an accepting state instead. So, if $m = n + \lfloor\sqrt{n}\rfloor$, then s arrives at time $2n + \lfloor\sqrt{n}\rfloor$ at the leftmost cell and the input is accepted. In all other cases the input is rejected. Clearly, the OCA constructed is an MC(*const*)-OCA. ∎

Next, we turn to an infinite strict hierarchy of real-time SC(f)-CA families [19]. The top of the hierarchy is given by the next theorem.

Theorem 6.3. *Let $f : \mathbb{N} \to \mathbb{N}$ be a function. If $f \in o(n^2/\log(n))$, then language $L = \{ wcw^R \mid w \in \{a, b\}^+ \}$ is not accepted by any real-time SC(f)-CA.*

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

Lemma 6.4. *Let $i \geq 1$ be an integer and $f : \mathbb{N} \to \mathbb{N}$ be a function.*

1. *If $f \in o((n \log^{[i]}(n))/\log^{[i+1]}(n))$ then language L_i is not accepted by any real-time SC(f)-CA.*
2. *Language L_i is accepted by some real-time SC($n \log^{[i]}(n)$)-CA.*

So, we can derive the infinite hierarchy.

Theorem 6.5. *Let $i \geq 0$ be an integer. Then $\mathscr{L}_{rt}$(SC($n \log^{[i+1]}(n)$)-CA) is properly included in $\mathscr{L}_{rt}$(SC($n \log^{[i]}(n)$)-CA).*

Decidability Questions

As is the case for devices with limited inter-cell bandwidth, various problems are undecidable even for the weakest non-trivial device with limited messages, that is, for real-time MC(*const*)-OCA.

Two of the common techniques to show undecidability results are reductions of Post's Correspondence Problem or reductions of the emptiness and finiteness problem on Turing machines using the set of valid computations. Both techniques have been used successfully to obtain results for variants of cellular automata [15, 20, 21, 27]. Taking a closer look at these known techniques, it is not clear yet whether they can be applied to MC(*const*)-OCA. In [19] it is shown that emptiness is undecidable for real-time MC(*const*)-OCA by reduction of Hilbert's tenth problem which is known to be undecidable. The problem is to decide whether a given polynomial $p(x_1, \ldots, x_n)$ with integer coefficients has an integral root. That is, to decide whether there are integers $\alpha_1, \ldots, \alpha_n$ such that $p(\alpha_1, \ldots, \alpha_n) = 0$. In [10] Hilbert's tenth problem has been used to show that emptiness is undecidable for certain multi-counter machines. As is remarked in [10], 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 then 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 $t_j(x_1, \ldots, x_n)$ $(1 \leq j \leq r)$ is of the form $t_j(x_1, \ldots, x_n) = s_j x_1^{i_{j,1}} \cdots x_n^{i_{j,n}}$ with $i_{j,1}, \ldots, i_{j,n} \geq 0$. If $|s_j| > 1$ we replace $s_j x_1^{i_{j,1}} \cdots x_n^{i_{j,n}}$ by s_j copies of $x_1^{i_{j,1}} \cdots x_n^{i_{j,n}}$. So, we may assume without loss of generality that all constant factors are either 1 or -1.

Sketchy speaking, the reduction is as follows. For a polynomial $p(x_1, \ldots, x_n)$ with integer coefficients that has the above form, languages $L(t_j)$ for every term

t_j are defined that evaluate the terms t_j. The next step is to simulate an evaluation of the given polynomial p. To this end, the evaluations of the single terms have to be put together, which is done by concatenating certain regular languages around each language $L(t_j)$ resulting in $\tilde{L}(t_j)$, and intersecting the reversals of all these languages. This gives a language $\tilde{L}(p) = \bigcap_{j=1}^{r} \tilde{L}(t_j)^R$. From $\tilde{L}(p)$ another language $L(p)$ is constructed which is empty if and only if $p(x_1, \ldots, x_n)$ has no solution in the non-negative integers. In a long proof it is shown that $L(p)$ belongs to $\mathscr{L}_{rt}$(MC(*const*)-OCA). So, emptiness is undecidable for $\mathscr{L}_{rt}$(MC(*const*)-OCA) and we obtain the next theorem.

Theorem 6.6. *Emptiness, finiteness, infiniteness, equivalence, inclusion, regularity, and context-freeness are undecidable for real-time MC(const)-OCA.*

Moreover, even the restrictions itself are not decidable:

Theorem 6.7. *It is undecidable for an arbitrary real-time OCA whether it is a real-time MC(const)-OCA.*

So, even for the weakest non-trivial device with limited messages, that is, for real-time MC(*const*)-OCA all the mentioned properties are undecidable.

An approach often investigated and widely accepted is to consider a given type of device for special purposes only, for example, for the acceptance of languages having a certain structure or form. From this point of view it is natural to start with unary languages (for example, [1, 2, 11, 24, 25]). For general real-time one-way cellular automata it is known that they accept only regular unary languages [27]. Since the proof is constructive, we derive that the borderline between decidability and undecidability has been crossed. So, we generalize unary languages to bounded languages. For several devices it is known that they accept non-semilinear languages in general, but only semilinear bounded languages. Since for semilinear sets several properties are decidable [6], constructive proofs lead to decidable properties for these devices in connection with bounded languages [3, 8, 9, 10]. Example 6.2 witnesses the following theorem.

Theorem 6.8. *The language family $\mathscr{L}_{rt}$(MC(const)-OCA) contains non-semilinear bounded languages.*

So, it is natural to consider decidability problems for the devices under consideration accepting *bounded* languages. In [16] it has been shown that also with this additional restriction none of the problems becomes decidable as long as the number of messages allowed is not too small.

Theorem 6.9. *Emptiness, finiteness, infiniteness, equivalence, inclusion, regularity, and context-freeness are undecidable for arbitrary real-time SC(n)-OCA and MC($\log n$)-OCA accepting* bounded *languages.*

7. Both Restrictions At Once

Since it turned out that neither limiting the bandwidth nor limiting the number of messages, even for bounded languages, reduces the computational capacity of the devices such that certain properties become decidable, we next combine both approaches and study real-time one-way cellular automata which are allowed to communicate only a fixed, finite number of messages per time step. Additionally, the

communication links between two cells are allowed to be used constantly often only. In the most restricted case, we consider real-time one-way cellular automata which can communicate one message between two cells exactly once. The next example reveals that real-time MC(*const*)-OCA$_1$ accept non-regular context-free languages. More precisely, only two messages per cell are used.

Example 7.1. We describe the computation of an MC(*const*)-OCA$_1$ $\mathcal{M}$ on inputs of the form a^nbc^m. The acceptance of the language is governed by two signals. During the first time step the rightmost cell receives the message 1 associated with the boundary symbol and identifies itself to be the rightmost cell. During the second time step the cell with input symbol b (b-cell) sends a message. In this way, the unique a-cell with right neighboring b-cell can identify itself. Subsequently, the a-cell sends the message 1 with speed 1/2, and the rightmost cell sends the message 1 with maximal speed to the left. So, the slow signal starts at time step 2 in the rightmost a-cell, takes $2n-2$ further time steps to reach the leftmost cell, and thus stays at time steps $2n$ and $2n+1$ in the leftmost cell. The fast signal is set up at time step 1 and takes $n+m$ further time steps to reach the leftmost cell. When both signals meet in a cell, that is, $n+m+1 = 2n+1$, an accepting state is entered. Therefore, the leftmost cell accepts if and only if $n = m$. Moreover, each cell sends at most two messages, and can be set up to send no more messages even for inputs of another form.

Now assume that the language accepted by $\mathcal{M}$ is regular. Since regular languages are closed under intersection, so is the language $L(\mathcal{M}) \cap a^*bc^* = a^nbc^n$, which is non-regular but context free. ∎

Turning to our key question, we present a lemma which relates languages accepted by real-time one-way cellular automata with a constant number of communications with languages accepted by real-time one-way cellular automata with a constant number of communications and one-message communication [18]. In this way, undecidability results can be derived from the undecidability results known.

Lemma 7.2. *Let $\mathcal{M} = \langle S, F, A, B, \#, b_l, \delta\rangle$ be a real-time MC(const)-OCA and $\$ \notin A$ be a new symbol. Then a real-time MC(const)-OCA$_1$ $\mathcal{M}'$ accepting the language $\{\, w\$^{(|B|+2)(|w|+1)}v \mid v \in \{\$, A(A \cup \{\$\})^*\}, w \in L(\mathcal{M}) \,\}$ can effectively be constructed.*

It is straightforward to generalize this construction for cellular automata with two-way communication. Furthermore, the construction increases the number of communication steps per cell only linearly. Therefore, the construction also works fine for SC(f)-CA and MC(f)-CA where f is not necessarily a constant function. Now, we can use Lemma 7.2 in order to reduce the undecidability problems for MC(*const*)-OCA to MC(*const*)-OCA$_1$.

Theorem 7.3. *Emptiness, finiteness, infiniteness, equivalence, inclusion, regularity, and context-freeness are undecidable for arbitrary real-time MC(const)-OCA$_1$.*

Clearly, the undecidability carries over to, for example, MC(*const*)-CA$_k$ with two-way communication and the models SC(n)-OCA$_k$ and SC(n)-CA$_k$. The undecidability results for real-time SC(n)-OCA and MC($\log n$)-OCA accepting bounded languages cannot be translated directly to the corresponding automata with one communication symbol by using the construction of Lemma 7.2, since the construction does not preserve the boundedness of the languages. However, if we allow an

additional communication symbol, then we obtain undecidability results also for bounded languages accepted by real-time SC(n)-OCA and MC($\log n$)-OCA with restricted communication alphabet of size two.

Lemma 7.4. *Let $\mathcal{M} = \langle S, F, A, B, \#, b_l, \delta\rangle$ be a real-time MC(const)-OCA and $\$ \notin A$ be a new symbol. Then a real-time MC(const)-OCA$_2$ $\mathcal{M}'$ accepting the language $\{w\$^{(|B|+2)(|w|+1)}\$ \mid w \in L(\mathcal{M})\}$ can effectively be constructed.*

It is obvious that $L(\mathcal{M}')$ is a bounded language if $L(\mathcal{M})$ is. Thus, we obtain the following undecidability results.

Theorem 7.5. *Emptiness, finiteness, infiniteness, inclusion, equivalence, regularity, and context-freeness are undecidable for arbitrary real-time SC(n)-OCA$_2$ and MC($\log n$)-OCA$_2$ accepting bounded languages.*

References

[1] Book, R.V.: Tally languages and complexity classes. Inform. Control **26** (1974) 186–193
[2] Chrobak, M.: Finite automata and unary languages. Theoret. Comput. Sci. **47** (1986) 149–158
[3] Csuhaj-Varjú, E., Dassow, J., Kelemen, J., Păun, G.: Grammar Systems: A Grammatical Approach to Distribution and Cooperation. Gordon and Breach, Yverdon (1984)
[4] Cudia, D.F., Singletary, W.E.: Degrees of unsolvability in formal grammars. J. ACM **15** (1968) 680–692
[5] Fischer, P.C.: Generation of primes by a one-dimensional real-time iterative array. J. ACM **12** (1965) 388–394
[6] Ginsburg, S.: The Mathematical Theory of Context-Free Languages. McGraw Hill, New York (1966)
[7] Hartmanis, J.: Context-free languages and Turing machine computations. Proc. Symposia in Applied Mathematics **19** (1967) 42–51
[8] Ibarra, O.H.: Simple matrix languages. Inform. Control **17** (1970) 359–394
[9] Ibarra, O.H.: A note on semilinear sets and bounded-reversal multihead pushdown automata. Inform. Process. Lett. **3** (1974) 25–28
[10] Ibarra, O.H.: Reversal-bounded multicounter machines and their decision problems. J. ACM **25** (1978) 116–133
[11] Klein, A., Kutrib, M.: Cellular devices and unary languages. Fund. Inform. **78** (2007) 343–368
[12] Kutrib, M.: Cellular automata and language theory. In: Encyclopedia of Complexity and System Science. Springer (2009) 800–823
[13] Kutrib, M., Malcher, A.: Cellular automata with limited inter-cell bandwidth. Theoret. Comput. Sci., to appear
[14] Kutrib, M., Malcher, A.: Fast cellular automata with restricted inter-cell communication: Computational capacity. In: Theoretical Computer Science (IFIP TCS2006). Volume 209 of IFIP, Springer (2006) 151–164
[15] Kutrib, M., Malcher, A.: Fast reversible language recognition using cellular automata. Inform. Comput. **206** (2008) 1142–1151
[16] Kutrib, M., Malcher, A.: Bounded languages meet cellular automata with sparse communication. In: Descriptional Complexity of Formal Systems (DCFS 2009), Otto-von-Guericke-Universität Magdeburg (2009) 211–222
[17] Kutrib, M., Malcher, A.: Computations and decidability of iterative arrays with restricted communication. Parallel Process. Lett. **19** (2009) 247–264
[18] Kutrib, M., Malcher, A.: On one-way one-bit O(one)-message cellular automata. Electron. Notes Theor. Comput. Sci. **252** (2009) 77–91
[19] Kutrib, M., Malcher, A.: Cellular automata with sparse communication. Theoret. Comput. Sci. **411** (2010) 3516–3526
[20] Malcher, A.: Descriptional complexity of cellular automata and decidability questions. J. Autom., Lang. Comb. **7** (2002) 549–560

[21] Malcher, A.: On the descriptional complexity of iterative arrays. IEICE Trans. Inf. Syst. **E87-D** (2004) 721–725
[22] Mazoyer, J.: A minimal time solution to the firing squad synchronization problem with only one bit of information exchanged. Technical Report TR 89-03, Ecole Normale Supérieure de Lyon (1989)
[23] Mazoyer, J., Terrier, V.: Signals in one-dimensional cellular automata. Theoret. Comput. Sci. **217** (1999) 53–80
[24] Mereghetti, C., Pighizzini, G.: Optimal simulations between unary automata. SIAM J. Comput. **30** (2001) 1976–1992
[25] Pighizzini, G., Shallit, J.O.: Unary language operations, state complexity and Jacobsthal's function. Int. J. Found. Comput. Sci. **13** (2002) 145–159
[26] Rogers, H.: Theory of Recursive Functions and Effective Computability. McGraw-Hill, New York (1967)
[27] Seidel, S.R.: Language recognition and the synchronization of cellular automata. Technical Report 79-02, Department of Computer Science, University of Iowa (1979)
[28] Umeo, H., Kamikawa, N.: A design of real-time non-regular sequence generation algorithms and their implementations on cellular automata with 1-bit inter-cell communications. Fund. Inform. **52** (2002) 257–275
[29] Umeo, H., Kamikawa, N.: Real-time generation of primes by a 1-bit-communication cellular automaton. Fund. Inform. **58** (2003) 421–435
[30] Vollmar, R.: On cellular automata with a finite number of state changes. Computing **3** (1981) 181–191
[31] Vollmar, R.: Some remarks about the 'efficiency' of polyautomata. Internat. J. Theoret. Phys. **21** (1982) 1007–1015
[32] Worsch, T.: Linear time language recognition on cellular automata with restricted communication. In: LATIN 2000: Theoretical Informatics. Volume 1776 of LNCS, Springer (2000) 417–426