Theoretical Computer Science 412 (2011) 3917–3931

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Cellular automata with limited inter-cell bandwidth

Martin Kutrib *, Andreas Malcher

Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany

ARTICLE INFO

Keywords:
Cellular automata
Iterative arrays
Restricted communication
Formal languages
Computational capacity
Parallel computing
Closure properties

ABSTRACT

A d-dimensional cellular automaton is a d-dimensional grid of interconnected interacting finite automata. There are models with parallel and sequential input modes. In the latter case, the distinguished automaton at the origin, the communication cell, is connected to the outside world and fetches the input sequentially. Often in the literature this model is referred to as an iterative array. In this paper, d-dimensional iterative arrays and one-dimensional cellular automata are investigated which operate in real and linear time and whose inter-cell communication bandwidth is restricted to some constant number of different messages independent of the number of states. It is known that even one-dimensional two-message iterative arrays accept rather complicated languages such as $\{a^p \mid p \text{ prime}\}$ or $\{a^{2^n} \mid n \in \mathbb{N}\}$ (H. Umeo, N. Kamikawa, Real-time generation of primes by a 1-bit-communication cellular automaton, Fund. Inform. 58 (2003) 421–435). Here, the computational capacity of d-dimensional iterative arrays with restricted communication is investigated and an infinite two-dimensional hierarchy with respect to dimensions and messages is shown. Furthermore, the computational capacity of the one-dimensional devices in question is compared with the power of two-way and one-way cellular automata with restricted communication. It turns out that the relations between iterative arrays and cellular automata are quite different from the relations in the unrestricted case. Additionally, an infinite strict message hierarchy for real-time two-way cellular automata is obtained as well as a very dense time hierarchy for k-message two-way cellular automata. Finally, the closure properties of one-dimensional iterative arrays with restricted communication are investigated and differences to the unrestricted case are shown as well.

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 IAs have been introduced in [5], where it was shown that the language family accepted by real-time IAs forms a Boolean algebra not closed under concatenation and reversal. In [4] it is shown that for every context-free grammar a two-dimensional linear-time IA parser exists. A real-time acceptor for prime numbers has been constructed in [6]. A characterization of various types of IAs in terms of restricted Turing machines and several results, especially speed-up theorems, are given in [7,8]. Several more results concerning formal languages can be found, for example, in [11,16,17].

* Corresponding author.
E-mail addresses: kutrib@informatik.uni-giessen.de (M. Kutrib), malcher@informatik.uni-giessen.de (A. Malcher).

0304-3975/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2011.02.026

When the computational capacity of a device is investigated, there is a particular interest in infinite hierarchies of language families defined by bounding some resources. In [9] a dense IA time hierarchy beyond linear time has been proved. The gap between real time and linear time has been closed in [2]. Further hierarchies depending on the amount of nondeterminism and the number of alternating transitions performed by the communication cell are shown in [1,3]. Descriptional complexity issues are studied in [14].

All these results concern iterative arrays where the states of the neighboring cells are communicated in one time step. That is, the number of messages exchanged is determined by the number of states. A natural and interesting restriction of IAs is to bound the number of messages communicated by some constant being independent of the number of states. Iterative arrays with restricted inter-cell communication have been investigated in [19,20], where algorithmic design techniques for sequence generation are shown. In particular, several important infinite, non-regular sequences such as exponential or polynomial, Fibonacci, and prime sequences can be generated in real time. Further constructions and decidability questions for one-dimensional iterative arrays with restricted communication are studied in [13]. Some of the following results have been shown in [12]. Connectivity recognition problems are dealt with in [18], whereas in [21] the computational capacity of one-way cellular automata with restricted inter-cell communication is considered.

Here, we investigate d-dimensional IAs and one-dimensional cellular automata operating in real and linear time. The inter-cell communication of the array is restricted to some constant number of different messages, in order to determine the power and nature of the communication bandwidth in massively parallel devices. The paper is organized as follows. In Section 2, we define the basic notions and the two models in question. Sections 3 and 4 are devoted to dimension and message hierarchies. We show that there is an infinite strict double hierarchy. That is, for every dimension real-time $(k+1)$-message restricted IAs are strictly more powerful than real-time k-message restricted IAs, and for every k-message restriction real-time $(d+1)$-dimensional k-message restricted IAs are strictly more powerful than real-time d-dimensional k-message restricted IAs. In Section 5, we consider one-dimensional devices. The computational capacity of the devices in question is compared with the power of two-way and one-way cellular automata with restricted communication. It turns out that the relations between iterative arrays and cellular automata are quite different from the relations in the unrestricted case. Moreover, we obtain an infinite strict message hierarchy for real-time two-way cellular automata and a very dense time hierarchy for k-message cellular automata, that is, just one more time step yields a proper superfamily of accepted languages. Additionally, we study the relation to the class of regular and context-free languages and obtain in many cases incomparability results. In the last section, we investigate closure properties of one-dimensional real-time IAs with restricted communication. It turns out that this class is not closed under union, intersection, and inverse homomorphism whereas the unrestricted variant is closed under these operations. On the other hand, we can show that closure properties which do not hold for the unrestricted case, do not hold for the restricted case as well.

2. Preliminaries and definitions

We denote the rational numbers by $\mathbb{Q}$, the integers by $\mathbb{Z}$, the non-negative integers by $\mathbb{N}$, and the positive integers $\{1, 2, \ldots\}$ 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 d-dimensional iterative array is a d-dimensional array ($\mathbb{N}^d$) of finite automata, sometimes called cells, where each of them is connected to its nearest neighbors in every dimension (see Fig. 1). For convenience we identify the cells by their coordinates. Initially they are in the so-called quiescent state. The input is supplied sequentially to the distinguished communication cell at the origin. For this reason, we have different local transition functions. The state transition of all cells but the communication cell depends on the current state of the cell itself and the current states of its neighbors. The state transition of the communication cell additionally depends on the current input symbol (or if the whole input has been consumed on a special end-of-input symbol). 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 (see Fig. 2). The messages to be sent by a cell depend on its current state and are determined by so-called communication functions. The finite automata work synchronously at discrete time steps.

Definition 1. A *d-dimensional iterative array with k-message restricted inter-cell communication* (IA_k^d) is a system $\langle S, A, B, \#, F, s_0, d, b_1, \ldots, b_{2d}, \delta, \delta_0\rangle$, 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, nonempty set of *communication symbols*,
(4) # is the *end-of-input symbol*,
(5) $F \subseteq S$ is the set of *accepting states*,
(6) $s_0 \in S$ is the *quiescent state*,
(7) $d \in \mathbb{N}_+$ is the *dimension*,
(8) $b_i : S \to B$, for $1 \leq i \leq 2d$, are *communication functions* which determine the information *to be sent* to neighbors,

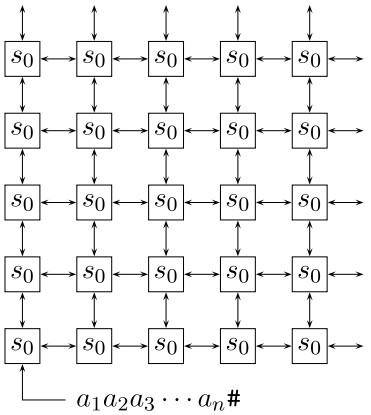

Fig. 1. A two-dimensional iterative array.

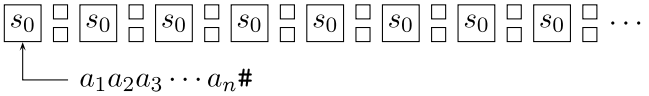

Fig. 2. A one-dimensional one-bit iterative array. The upper box between two cells denotes the communication channel from left to right. The lower box denotes the communication channel from right to left.

(9) $\delta : S \times B^{2d} \to S$ is the *local transition function for non-communication cells* satisfying $\delta(s_0, (b_1(s_0), b_2(s_0), \ldots, b_{2d}(s_0)) = s_0$,

(10) $\delta_0 : S \times (A \cup \{\#\}) \times B^d \to S$ is the *local transition function for the communication cell*.

Let $\mathcal{M}$ be an IA_k^d. 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 $c_t : \mathbb{N}^d \to S$ is a mapping that maps the single cells to their current states. For the sake of simpler notation in connection with cells at a face of $\mathbb{N}^d$, we extend the mappings c_t to arguments from $\mathbb{Z}^d$, and assume that all cells in $\mathbb{Z}^d \setminus \mathbb{N}^d$ are permanently in the quiescent state. The configuration (w_0, c_0) at time 0 is defined by the input word w_0 and the mapping $c_0(i_1, \ldots, i_d) = s_0$, $(i_1, \ldots, i_d) \in \mathbb{N}^d$, 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 $(w_{t+1}, c_{t+1}) = \Delta\big((w_t, c_t)\big)$ is as follows.

$$\begin{aligned} c_{t+1}(i_1, \ldots, i_d) = \delta\big(&c_t(i_1, \ldots, i_d),\\ &b_1(c_t(i_1 - 1, i_2, \ldots, i_d)), b_2(c_t(i_1 + 1, i_2, \ldots, i_d)),\\ &b_3(c_t(i_1, i_2 - 1, \ldots, i_d)), b_4(c_t(i_1, i_2 + 1, \ldots, i_d)), \ldots,\\ &b_{2d-1}(c_t(i_1, i_2, \ldots, i_d - 1)), b_{2d}(c_t(i_1, i_2, \ldots, i_d + 1))\big) \end{aligned}$$

for all $(i_1, \ldots, i_d) \in \mathbb{N}^d \setminus \{(0, \ldots, 0)\}$, and

$$\begin{aligned} c_{t+1}(0, \ldots, 0) = \delta_0\big(&c_t(0, \ldots, 0), a,\\ &b_2(c_t(1, 0, \ldots, 0)), b_4(c_t(0, 1, \ldots, 0)), \ldots,\\ &b_{2d}(c_t(0, 0, \ldots, 1))\big) \end{aligned}$$

where $a = \#$, $w_{t+1} = \lambda$ if $w_t = \lambda$, and $a = a_1$, $w_{t+1} = a_2 \cdots a_n$ if $w_t = a_1 \cdots a_n$. Thus, the global transition function Δ is induced by δ and δ_0.

Since in one-message iterative arrays every cell sends the same and only message at every time step, usable communication is impossible. Therefore, the computational capacity reduces to that of the communication cell, that is, to the capacity of deterministic finite automata. So, the minimal number of nontrivial messages is two. Let us consider an example to illustrate the capability of iterative arrays with restricted communication. We describe how a real-time two-message IA can simulate a counter (cf. [13]).

Example 2. In general, real-time two-message IAs can count by storing the binary encoding of the current value in their cells. Basically, each cell stores one bit, where the communication cell stores the least significant bit. But due to the finite neighborhood it is impossible to obtain configurations that are binary representations literally. Instead, carry-overs are sent from cell to cell until they can be processed. On the other hand, the test for zero requires marking the cell carrying the most significant bit, and this mark may move.

So, the information to be communicated by cells is a carry-over and the position of the most significant bit of the counter. This can be realized by IA_2s. The communication channel from left to right is used to transport carry-overs, and the most significant bit of the counter is marked in the communication channel from right to left.

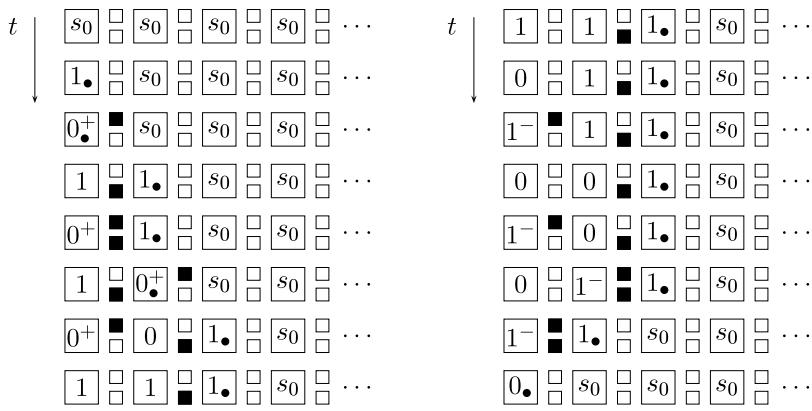

Fig. 3. Two two-message iterative arrays implementing an increasing (left) and a decreasing binary counter (right). The input is omitted. The upper boxes transport carry-overs from left to right, and the lower boxes are used to mark the cell with the most significant bit which, in addition, is marked by •. The exponents + and − denote carry-overs.

An example counting from zero to seven may be found on the left of Fig. 3. The construction of a decreasing counter is a straightforward modification and an example counting from seven to zero may be found on the right of Fig. 3. □

We now introduce cellular automata with restricted communication, since we want to compare their computational capacity with that of iterative arrays with restricted communication. 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 with 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 #. So, cell 1 is the communication cell that indicates acceptance or rejection, and the array is bounded to the n cells which are initially active.

Definition 3. A *cellular automaton with k-message restricted inter-cell communication* (CA_k) is a system $\langle S, \#, A, B, F, b_1, b_2, \delta\rangle$, where

(1) S is the finite, nonempty set of *cell states*,
(2) $\# \notin Q$ is the *boundary state*,
(3) $A \subseteq S$ is the finite, nonempty set of *input symbols*,
(4) B with $|B| = k$ is the finite, nonempty set of *communication symbols*,
(5) $F \subseteq S$ is the set of *accepting states*,
(6) $b_1, b_2 : (S \cup \{\#\}) \to B$ are *communication functions* which determine the information *to be sent* to both neighbors,
(7) $\delta : B \times S \times B \to S$ is the *local transition function*.

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 from right to left. Formally, δ is a mapping from $S \times B$ to S.

A *configuration* of a cellular automaton $\langle S, \#, A, B, F, b_1, b_2, \delta\rangle$ at time $t \geq 0$ is a description of its global state, which is actually a mapping $c_t : \{1, \ldots, n\} \to S$, for $n \geq 1$. The operation starts at time 0 in a so-called *initial configuration*. For a given input $w = a_1 \cdots a_n \in A^+$ we set $c_{0,w}(i) = a_i$, for $1 \leq i \leq n$. During the course of its computation a CA_k steps through a sequence of configurations, whereby successor configurations are computed according to the global transition function Δ.

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

$$c_{t+1}(1) = \delta(b_2(\#), c_t(1), b_1(c_t(2)))$$
$$c_{t+1}(i) = \delta(b_2(c_t(i-1)), c_t(i), b_1(c_t(i+1))), i \in \{2, \ldots, n-1\}$$
$$c_{t+1}(n) = \delta(b_2(c_t(n-1)), c_t(n), b_1(\#))$$

for CA_ks and

$$c_{t+1}(i) = \delta(c_t(i), b_1(c_t(i+1))), i \in \{1, \ldots, n-1\}$$
$$c_{t+1}(n) = \delta(c_t(n), b_1(\#))$$

for OCA_ks.

An input w is accepted by an IA_k^d ((O)CA_k) $\mathcal{M}$ if at some time i during the course of its computation the communication (leftmost) cell 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 (O)CA_ks) 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 languages that are accepted by IA_k^ds (CA_ks, OCA_ks) with time complexity t is denoted by $\mathscr{L}_t(IA_k^d)$ ($\mathscr{L}_t(CA_k)$, $\mathscr{L}_t(OCA_k)$). If t is the function $n+1$ (the function n), acceptance is said to be in *real time* and we write $\mathscr{L}_{rt}(IA_k^d)$ ($\mathscr{L}_{rt}(CA_k)$,

$\mathscr{L}_{rt}(\mathrm{OCA}_k)$). Since for nontrivial computations an IA_k^d has to read at least one end-of-input symbol, real time has to be defined as $(n+1)$-time. The *linear-time* languages $\mathscr{L}_{lt}(\mathrm{IA}_k^d)$ are defined according to $\mathscr{L}_{lt}(\mathrm{IA}_k^d) = \bigcup_{r\in\mathbb{Q},\, r\geq 1} \mathscr{L}_{r\cdot n}(\mathrm{IA}_k^d)$, and similarly for CA_ks and OCA_ks.

Next, we define two equivalence relations and derive upper bounds on the number of induced equivalence classes. This will give us a valuable tool to show that certain languages cannot be accepted by IA_k^ds.

Definition 4. Let $L \subseteq A^*$ be a language over an alphabet A and $l \in \mathbb{N}_+$ be a constant.

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

Lemma 5. *Let $d \geq 1$ and $k \geq 2$ be constants.*

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

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

and

(2) *if $L \in \mathscr{L}_t(IA_k^d)$, then there exists a constant $p \in \mathbb{N}$ such that*

$$N_\ell(l, L) \leq p \cdot k^{d\cdot l}$$

for all $l \in \mathbb{N}_+$ and all time complexities $t : \mathbb{N} \to \mathbb{N}$.

Proof. Let $\mathcal{M} = \langle S, A, B, \#, F, s_0, d, b_1, \ldots, b_{2d}, \delta, \delta_0\rangle$ be a real-time IA_k^d that accepts L. In order to determine an upper bound for the number of l-right-equivalence classes we consider the possible configurations of $\mathcal{M}$ after reading all but $|y| \leq l$ input symbols. The remaining computation depends on the last $|y|$ input symbols, the current state of the communication cell, and the states of the cells which can send information that is received by the communication cell during the last $|y|+1$ time steps. These are at most $(|y|+1)^d$ cells. So, in total there are at most $|S|^{1+(|y|+1)^d} \leq |S|^{2(l+1)^d}$ different possibilities. Setting $p = |S|^2$, we obtain $N_r(l, L) \leq p^{(l+1)^d}$.

Now, let $\mathcal{M}$ be an IA_k^d that accepts L with time complexity t. In order to determine an upper bound for the number of l-left-equivalence classes we consider the possible configurations of $\mathcal{M}$ after reading prefixes w whose lengths are at most l. A computed configuration depends on the information which has been sent to the array by the communication cell, and the current state of the communication cell. So, there are at most $(k^d)^{|w|-1} \cdot |S| \leq |S| \cdot k^{d\cdot l}$ different configurations. Setting $p = |S|$, we obtain $N_\ell(l, L) \leq p \cdot k^{d\cdot l}$. In particular, the number of equivalence classes is independent of the time complexity t. □

3. Dimension hierarchy

In this section, we fix the time complexity to real time, the number of different messages to constants $k \geq 2$, 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

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

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

Theorem 6. *Let $d \geq 1$ and $k \geq 2$ be constants. The language $L_{dim}(d+1)$ belongs to the difference $\mathscr{L}_{rt}(IA_2^{d+1}) \setminus \mathscr{L}_{rt}(IA_k^d)$.*

Proof. For any $m \in \mathbb{N}_+$ we consider sets

$$Y_1 = \$\{a, b\}^m, \qquad Y_{i+1} = \$Y_i^m, \quad \text{for } i \geq 1$$

It follows $Y_i \subset X_i$, for all $i \in \mathbb{N}_+$, and $|Y_i| = 2^{m^i}$. If we choose two different words u and u' from Y_{d+1}, then there is one position at which u has a symbol a and u' has a symbol b or vice versa. We can address this position by $u[x_{d+1}][x_d]\cdots[x_1]$ and obtain

$$h(u)h(\cent e^{x_{d+1}}\$ \cdots \$e^{x_1}\$e^{2x}\$a) \in L_{dim}(d+1) \iff h(u')h(\cent e^{x_{d+1}}\$ \cdots \$e^{x_1}\$e^{2x}\$a) \notin L_{dim}(d+1).$$

There are $2^{m^{d+1}}$ different words in Y_{d+1}, and for the length of the suffix we obtain $|h(\text{¢}e^{x_{d+1}}\$\cdots\$e^{x_1}\$e^{2x}\$a)| \leq 3m(d+1)+2(d+3)+2$ since $x_i \leq m$. This implies a lower bound on the number of induced equivalence classes as follows:

$$N_r(3m(d+1)+2(d+3)+2, L_{dim}(d+1)) \geq 2^{m^{d+1}}.$$

In contrast to the assertion, we now assume $L_{dim}(d+1) \in \mathscr{L}_{rt}(\mathrm{IA}_k^d)$. Then by Lemma 5 there exists a constant $p \in \mathbb{N}_+$ such that $N_r(l, L_{dim}(d+1)) \leq p^{(l+1)^d}$, for all $l \in \mathbb{N}_+$. So, for $l = 3m(d+1)+2(d+3)+2$ we have at most

$$p^{(3m(d+1)+2(d+3)+2+1)^d} \leq p^{(6md+2d+9)^d} \leq p^{(17md)^d} \leq 2^{\lceil \log(p) \rceil (17d)^d m^d}$$

classes. We choose m such that $m > \lceil \log(p) \rceil (17d)^d$, and obtain strictly less than

$$2^{mm^d} = 2^{m^{d+1}}$$

classes. From the contradiction we obtain $L_{dim}(d+1) \notin \mathscr{L}_{rt}(\mathrm{IA}_k^d)$.

Now we turn to the construction of a real-time IA_2^{d+1} which accepts $L_{dim}(d+1)$. First we observe that the structure of accepted words is regular. Therefore, the communication cell can check it and, moreover, can decode the checked input over $\{a, b\}$ uniquely to a word from $M(d+1)$. For convenience, we explain the acceptance in terms of these words. Basically, the idea is to store the prefix u in such a way that the symbol $u[x_{d+1}]\cdots[x_1]$ is stored in cell $(x_{d+1}-1, x_d-1, \ldots, x_1-1)$. While subsequently reading the suffix $\text{¢}e^{x_{d+1}}\$\cdots\$e^{x_1}\$e^{2x}\$v$ symbol $u[x_{d+1}]\cdots[x_1]$ is addressed and sent to the communication cell where it is compared with v. Accordingly, we call the first phase the storage and the second phase the retrieval phase.

We name cells dependent on their coordinates. A cell is said to be of level j, if its last j coordinates are 0, that is, $(i_1, \ldots, i_{d+1-j}, 0, \ldots, 0)$. Note that a level j cell is also of level $j' < j$, and the communication cell is the sole level $d+1$ cell. A cell with maximal level j activates its neighbors $(i_1, \ldots, i_{d+1-j}, 0, \ldots, 0, 1)$, $(i_1, \ldots, i_{d+1-j}, 0, \ldots, 1, 0)$, $\ldots$, $(i_1, \ldots, i_{d+1-j}, 1, \ldots, 0, 0)$, and $(i_1, \ldots, i_{d+1-j}+1, 0, \ldots, 0)$, i.e., sends a non-zero signal for the first time. Therefore, each cell is uniquely activated by one of its neighbors and, moreover, can determine its maximal level by this neighbor. A cell with maximal level $j \leq d$ may activate at most $j+1$ neighbors.

Activation takes place during the storage phase, in which cells mark a path to the current storage position by state components. When the communication cell reads $h(a)$ $(h(b))$, it sends the two messages 10 (11) along the path until the position is reached. Now the corresponding cell $(i_1, \ldots, i_{d+1})$ stores symbol a (b), activates its neighbor $(i_1, \ldots, i_{d+1}+1)$ to be the next storage position by sending the messages 01, and extends the current path to the newly activated neighbor.

Whenever the communication cell reads $h(\$)$, it sends the messages 01 along the path. In this situation the cells on the path count the number of at most d consecutive 01 signals, and possibly reroute the path as follows. A cell lets pass $p-1$ signals, where p is the number of already activated neighbors. If there is another signal, it activates the next neighbor according to the above given ordering, and reroutes the path to it. Clearly, there cannot be more signals than the number of activated neighbors minus one, since the next predecessor cell of higher level does not let pass so many of them.

When the communication cell reads $h(\text{¢})$, it sends the messages 00 to the array. This signal is distributed to all activated cells recursively. It is the beginning of the retrieval phase. During this phase a path to the addressed symbol is set up. To this end, the communication cell sends along the path a message 1 for each $h(e)$ read, and the messages 00 for each of the next $d+1$ separators $h(\$)$.

A cell remembers whether it is on the path or not, and whether it is the end of the path. Initially, only the communication cell is on the path. If a cell is on the path but not at the end, it simply routes the signals along the path. The end of the path, say $(i_1, \ldots, i_j, 0, \ldots, 0)$ sends the signal 1 to its neighbor $(i_1, \ldots, i_j+1, 0, \ldots, 0)$ which in turn deletes it and becomes the new end of path. The end of path cell $(i_1, \ldots, i_j, 0, \ldots, 0)$ deletes a 00 signal and sends the next signals 1 to its neighbor $(i_1, \ldots, i_j, 1, 0, \ldots, 0)$. So, on input $e^{x_{d+1}}\$\cdots\$e^{x_1}\$$ a path to cell $(x_{d+1}, x_d, \ldots, x_1)$ is established. The $(d+1)$st signal 00 causes cell $(x_{d+1}, x_d, \ldots, x_1)$ to send the information which it has stored during the storage phase back to the communication cell. The $(d+1)$st 00 signal takes $x_{d+1}+x_d+\cdots+x_1$ time steps to reach the end of path. Subsequently, the same number of time steps is necessary to send the information back to the communication cell. Altogether, these are $2x$ time steps. Therefore, the information can be compared with input symbol v by the communication cell. It remains to be mentioned that, in fact, symbol v has to be compared with the information stored in cell $(x_{d+1}-1, x_d-1, \ldots, x_1-1)$ instead of of $(x_{d+1}, x_d, \ldots, x_1)$. But the construction can be modified appropriately in a straightforward manner. □

Corollary 7. *Let $d \geq 1$ and $k \geq 2$ be constants. Then $\mathscr{L}_{rt}(IA_k^d) \subset \mathscr{L}_{rt}(IA_k^{d+1})$.*

Proof. By Theorem 6, language $L_{dim}(d+1)$ cannot be accepted by any real-time IA_k^d, but is accepted by some real-time IA_2^{d+1} and, thus, by some IA_k^{d+1}. □

Next, we want to present another construction for accepting the language $L_{dim}(d+1)$ which shows that $L_{dim}(d+1)$ belongs to $\mathscr{L}_{lt}(\mathrm{IA}_2^1)$, that is, one can trade all dimensions and messages for a slow-down from real time to linear time.

Theorem 8. *Let $d \geq 1$ and $k \geq 2$ be constants. Then $\mathscr{L}_{rt}(IA_k^d) \subset \mathscr{L}_{lt}(IA_k^d)$.*

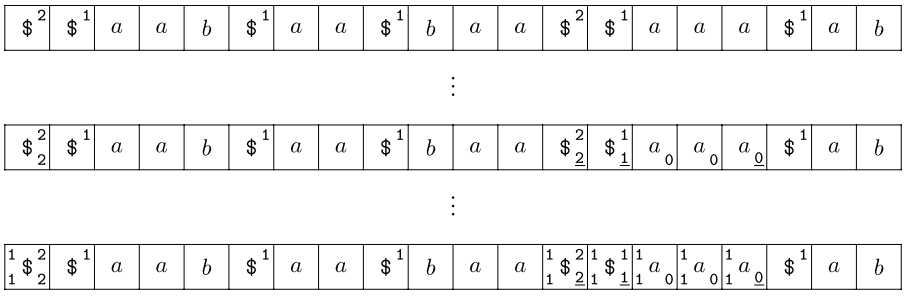

Fig. 4. Three snapshots of an IA_2^1 $\mathcal{M}$ working on input $h(w)$ with $w = u_{¢}e^2\$e\$e^3\$e^9\$a$ and $u = \$\$\$aab\$aa\$baa\$\$aaa\$ab$. The first snapshot shows the first 20 cells of $\mathcal{M}$ after the storage phase of input $h(u)$. The upper right corner of each cell denotes register R_1. The second snapshot takes a picture of the addressing in the retrieval phase after having processed $h(¢e^2\$e\$e^3\$)$. Here, the lower right corner of each cell denotes register R_2. Finally, the last snapshot shows the end of the retrieval phase after having processed $h(e^9\$a)$. Here, stopped incoming signals 1 are stored in registers which are depicted in the upper and lower left corner of each cell.

Proof. The inclusion is obvious. Since $L_{dim}(d+1)$ cannot be accepted by any real-time IA_k^d owing to Theorem 6, it remains to be shown that $L_{dim}(d+1)$ can be accepted by an IA_2^1 $\mathcal{M}$ working in linear time.

The idea of the construction is to read the input prefix $h(u)$ and to store u into the cells $0, 1, \ldots, |u|-1$ such that the first symbol of u is stored in cell 0 and the last symbol of u is stored in cell $|u|-1$ (cf. Fig. 4). The principle of the construction follows the real-time simulation of a queue shown in [10]. Then, while reading the input part $h(e^{x_{d+1}}\$\cdots\$e^{x_1}\$)$ a position in u is addressed and its content is moved to the communication cell in subsequent time steps, where it can be compared to the input $h(v)$. Finally, while reading the input part $h(e^{2x}\$)$ the equality $x = x_{d+1} + \cdots + x_1$ has to be checked.

The IA_2^1 $\mathcal{M}$ stores u in $2\cdot|h(u)|$ time steps. The symbols $h(a)$, $h(b)$, $h(\$)$, and $h(¢)$ are encoded by signals 10, 11, 01, and 00. For technical reasons the first two letters $h(\$)$ are ignored and not stored in the cells. Next, we consider blocks of adjacent symbols \$ and add a register R_1 to every cell which stores \$. While storing the input $h(u)$ every such register R_1 is updated and the number of remaining symbols \$ in the current block is remembered. Let n be the length of such a block. Then the first cell of the block has the information n in R_1 and the last cell of the block carries the information 1 in R_1.

When reading the separating symbol $h(¢)$ the signal 00 is sent to the array. For further symbols $h(e)$ and $h(\$)$ the signals 1 and 00 are sent to the array. Now we have to address the position given by the input. Therefore, every $h(e)$ from the block $e^{x_{d+1}}\$$ marks the next cell which carries d in R_1 with the information d in some register R_2. Finally, $h(\$)$ marks the last marked cell with the information $\underline{d}$ in R_2. Consequently, every $h(e)$ from the block $e^{x_d}\$$ marks the next cell which carries $d-1$ in R_1 with the information $d-1$ in register R_2 and $h(\$)$ marks the last marked cell with the information $\underline{d-1}$ in R_2. This behavior is iterated until every cell marked with $\underline{1}$ in register R_2 is followed by some block from $\{a,b\}^+$. Every $h(e)$ from the block $e^{x_1}\$$ marks the next cells which carry a or b with the information 0 in R_2 and $h(\$)$ marks the lastly marked cell with $\underline{0}$. This cell is the cell addressed by $e^{x_{d+1}}\$\cdots\$e^{x_1}\$$. Thus, its content is sent to the communication cell to be compared there with the input $h(v)$.

Now $\mathcal{M}$ has to read the suffix $h(e^l\$v)$ and to check whether $l = 2\cdot(x_{d+1}+\cdots+x_1)$. To this end, for every $h(e)$ and $h(\$)$, signals 1 and 00 are sent to the array. Every cell which is marked in its register R_2 stops the first two incoming signals 1. Further incoming signals 1 are forwarded to the right. Since there are $x_{d+1}+\cdots+x_1$ cells which carry information in R_2, it is sufficient to check, by using the signal 00, whether the cell already marked with $\underline{0}$ is the last cell which has stopped two signals 1. If this is true, an accepting signal can be sent to the communication cell.

Altogether, we can construct $\mathcal{M}$ such that the input is accepted if it is formatted correctly, $h(v)$ complies with the content of the position addressed, and the number of $h(e)$ in the last block complies with $2\cdot(x_{d+1}+\cdots+x_1)$. Obviously, $\mathcal{M}$ works in linear time. □

4. Bit hierarchy

In this section, we fix the time complexity to real time, the dimension to be a constant $d \in \mathbb{N}_+$, and consider the number of different messages. For any number of messages $k \geq 2$ we define an alphabet $A_{d,k} = \{a_0, \ldots, a_{k^d-1}\}$ and a language $L_{bit}(d,k)$.

$$L_{bit}(d,k) = \{\, e^x\$u_1u_2\cdots u_m \mid x \in \mathbb{N}_+ \text{ and } m \geq 2x-1 \text{ and } u_i \in A_{d,k},\ 1 \leq i \leq m,\ \text{and } u_j = u_{j+2x-1},\ 1 \leq j \leq m-(2x-1)\,\}$$

Theorem 9. *Let $d \geq 1$ and $k \geq 2$ be constants. The language $L_{bit}(d,k+1)$ belongs to the difference $\mathscr{L}_{rt}(IA_{k+1}^d) \setminus \mathscr{L}_{rt}(IA_k^d)$.*

Proof. Contrarily, assume $L_{bit}(d,k+1)$ is accepted by a real-time IA_k^d $\mathcal{M}$. Let $\mathcal{M}$ be in configuration c_x after processing the input prefix $e^x\$$. Now we consider the possible configurations of $\mathcal{M}$ after further reading $u_1u_2\cdots u_{2x-1}$. Starting in c_x such

$$
\begin{array}{ccccccccc}
\vdots & & \vdots & & & & \vdots & & \\
\cup & & \cup & & & & \cup & & \\
\mathscr{L}_{rt}(\mathrm{IA}_2^d) & \subset & \mathscr{L}_{rt}(\mathrm{IA}_3^d) & \subset & \cdots & \subset & \mathscr{L}_{rt}(\mathrm{IA}_k^d) & \subset & \cdots \\
\cup & & \cup & & & & \cup & & \\
\vdots & & \vdots & & & & \vdots & & \\
\cup & & \cup & & & & \cup & & \\
\mathscr{L}_{rt}(\mathrm{IA}_2^2) & \subset & \mathscr{L}_{rt}(\mathrm{IA}_3^2) & \subset & \cdots & \subset & \mathscr{L}_{rt}(\mathrm{IA}_k^2) & \subset & \cdots \\
\cup & & \cup & & & & \cup & & \\
\mathscr{L}_{rt}(\mathrm{IA}_2^1) & \subset & \mathscr{L}_{rt}(\mathrm{IA}_3^1) & \subset & \cdots & \subset & \mathscr{L}_{rt}(\mathrm{IA}_k^1) & \subset & \cdots
\end{array}
$$

Fig. 5. Double hierarchy of fast IAs with restricted inter-cell communication.

a configuration depends on the information which has been sent to the array by the communication cell, and the current state of the communication cell. So, there are at most $(k^d)^{2x-1} \cdot |S|$ different configurations. On the other hand, there are

$$(k+1)^{d\cdot(2x-1)} > k^{d\cdot(2x-1)} \cdot d \cdot (2x-1) \cdot k^{d\cdot(2x-1)-1}$$

different words $u_1u_2\cdots u_{2x-1}$. Choosing x such that $d \cdot (2x-1) \geq |S| \cdot k$ we obtain at least $(k^d)^{2x-1} \cdot (|S|+1)$ different words.

So, for at least two different words $u_1\cdots u_{2x-1}$ and $u'_1\cdots u'_{2x-1}$ $\mathcal{M}$ has computed the same configuration and, thus $e^x\$u_1\cdots u_{2x-1}u_1\cdots u_{2x-1}$ and $e^x\$u'_1\cdots u'_{2x-1}u_1\cdots u_{2x-1}$ are both accepted. But since they are different, there is an i such that $u_i \neq u'_i$ and, therefore, the second input has to be rejected. From the contradiction we obtain $L_{bit}(d, k+1) \notin \mathscr{L}_{rt}(\mathrm{IA}_k^d)$.

It remains to be shown that $L_{bit}(d, k+1) \in \mathscr{L}_{rt}(\mathrm{IA}_{k+1}^d)$. The construction below is applied to all dimensions in parallel. First, we present the construction for $d = 1$, which is generalized subsequently. A corresponding iterative array sends a message b_1 to the right as long as it reads the input prefix e^x. When a cell in the quiescent state (which emits messages b_0) receives b_1 it is activated. Subsequently, an activated cell sends a message b_1 to the right for every b_1 received from the left. In this way exactly x cells including the communication cell are activated. When the communication cell reads the \$ it interrupts the continuous sending of b_1 by sending b_0. This signal switches the activated cells successively to the second phase. During this phase the communication cell sends the symbols read into the array (there are as many messages as symbols). These symbols are shifted to the right through the activated cells. The rightmost of these cells (which can identify itself at the end of the first phase) sends the symbols received back to the left. In this way every input symbol sent by the communication cell to the right is back at the communication cell $2x-1$ time steps later. So, it can be compared with the current input symbol in order to check whether the input can still be accepted. There is one detail missing so far. The communication cell has to detect when it receives the first symbol back from the array. Since until that time step it receives only messages b_0 from the right, any message different from b_0 does the job. But to this end the first letter u_1 must not be shifted as message b_0 through the array. In order to avoid this situation the communication cell may choose one of the mappings from input symbols to messages. It chooses one depending on u_1 and remembers it until the end of the computation.

For higher dimensions, the encodings of the input symbols u_i are split into d messages. So, by k different messages we can encode k^d symbols. These d messages are distributed to the d neighbors of the communication cell. □

Corollary 10. *Let $d \geq 1$ and $k \geq 2$ be constants. Then $\mathscr{L}_{rt}(IA_k^d) \subset \mathscr{L}_{rt}(IA_{k+1}^d)$.*

From Corollaries 7 and 10 we obtain a double hierarchy concerning messages and dimensions which is depicted in Fig. 5.

5. Relations to cellular automata with restricted communication

In this section, we consider one-dimensional devices in order to compare them to cellular automata with restricted communication. 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 Fig. 6. 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 Fig. 6.

The first difference between language classes with and without communication restrictions is that there are regular languages which cannot be accepted by cellular automata with restricted communication even if we add a constant number of time steps to real-time, whereas all regular languages are accepted in the unrestricted case.

Lemma 11. *Let $k \geq 2$ and $r \geq 0$ be constants. There is a regular language which is not accepted by any $(n+r)$-time CA_k.*

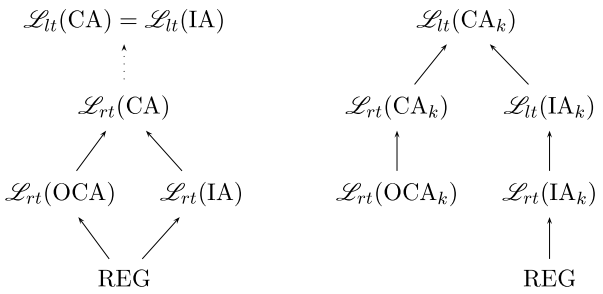

Fig. 6. Relations between unrestricted (left) and restricted (right) language families. REG denotes the family of regular languages. Solid arrows are strict inclusions and dotted arrows are inclusions. Families which are not connected by any path are incomparable.

Proof. We consider the alphabet $A_{k,r} = \{a_0, \ldots, a_{k^{r+2}-1}\}$ and the regular witness language $L_{k,r} = \{ xa^{r+1}xv \mid x \in A_{k,r} \text{ and } v \in (A_{k,r} \cup \{a\})^* \}$. Assume contrarily, that $L_{k,r}$ is accepted by some $(n+r)$-time CA_k. We consider the first $r+2$ messages sent to the left by the right x-cell on inputs of the form $xa^{r+1}x$ that belong to $L_{k,r}$ and are accepted at latest at time $2r+3$. (Messages sent afterwards do not influence the overall computation result.) These messages depend on the input symbol x as well as on the information which is received from the left and right. The information received is independent of the leftmost input symbol. Therefore, the messages sent by the rightmost cell solely depend on x. In total there are k^{r+2} possibilities to send different messages. We obtain a contradiction if two different symbols x send the same messages. So, all k^{r+2} symbols send different messages. Now we consider inputs of the form $xa^{r+2}x$ not belonging to $L_{k,r}$. Whatever first $r+2$ messages are sent by the cell initially carrying the last but one input symbol, the same messages are send in some accepting computation on an input of the form $xa^{r+1}x$. So, input $xa^{r+2}x$ not belonging to $L_{k,r}$ would also be accepted. □

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

Lemma 12. *Let $k \geq 2$ be a constant. Then every regular language over a $(k-1)$-letter alphabet is accepted by a real-time CA_k.*

Proof. The idea of the construction is to simulate a deterministic finite automaton accepting the regular language in the leftmost cell. The input of the CA_k is continuously shifted to the left in order to feed the deterministic finite automaton. To this end $(k-1)$ different messages are sufficient. In addition, one message is emitted by the right boundary cell that signals the end of the input. □

Next, we show that language $L_{k,r}$ of the proof of Lemma 11 can be accepted by a real-time CA_{k+1}. Thus, we obtain a strict message hierarchy for two-way real-time cellular automata.

Theorem 13. *Let $k \geq 2$ and $r \geq 0$ be constants. Then $\mathscr{L}_{rt+r}(CA_k) \subset \mathscr{L}_{rt+r}(CA_{k+1})$.*

Proof. The inclusion is obvious. For the properness of the inclusion we consider the language $L_{k,r}$ of the proof of Lemma 11. We know $L_{k,r} \notin \mathscr{L}_{rt+r}(\mathrm{CA}_k)$. Thus, it remains to be shown how an $(n+r)$-time CA_{k+1} can accept $L_{k,r}$. The idea of the construction is as follows. Each input symbol is encoded by $r+2$ messages for which k different messages are necessary. In addition one special message, say e, is used. So, in total $k+1$ different messages are sufficient. The boundary cells send continuously the special message. Initially, every a-cell sends the special message to the left and some other fixed message to the right. Subsequently, they transmit the information received from the right to the left. Every cell carrying an input symbol from $A_{k,r}$ sends the same fixed message to the right as the a-cells, and consecutively the $r+2$ parts of the encoding of its input to the left.

Now, the leftmost cell can identify itself since it is the only cell receiving the special message e from the left. Only a leftmost non-a-cell may accept. It starts to count up to $r+1$ as long as it receives an e from the right. In this way the correct number of as in between the symbols x is verified. Next the leftmost cell expects the $r+2$ messages from the right that encode the left symbol x. It decodes the messages and compares the right x with its own input symbol. If both are identical and the number of as was correct it accepts, otherwise it rejects. So, we obtain $L_{k,r} \in \mathscr{L}_{rt}(\mathrm{CA}_{k+1})$. □

Moreover, we show that language $L_{k,r}$ of the proof of Lemma 11 is accepted by an $(n+r+1)$-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 14. *Let $k \geq 2$ and $r \geq 0$ be constants. Then $\mathscr{L}_{rt+r}(CA_k) \subset \mathscr{L}_{rt+r+1}(CA_k)$.*

Proof. The inclusion is obvious. For the properness of the inclusion we consider again the language $L_{k,r}$ of the proof of Lemma 11. We know $L_{k,r} \notin \mathscr{L}_{rt+r}(\mathrm{CA}_k)$. Thus, it remains to be shown how an $(n+r+1)$-time CA_k can accept $L_{k,r}$. The construction is similar to the construction given in the proof of Theorem 13. The main observation is that we can encode the alphabet $A_{k,r}$ with $r+3$ parts of a k-ary alphabet such that none of the encodings starts with the special symbol e, since $(k-1)k^{r+2} \geq k^{r+2}$, for all $k \geq 2$. Therefore, symbol e can be used in the same way as in the construction of the proof of Theorem 13. □

$$
\begin{array}{ccccccccc}
\vdots & & \vdots & & & & \vdots & & \\
\cup & & \cup & & & & \cup & & \\
\mathscr{L}_{rt+r}(\mathrm{CA}_2) & \subset & \mathscr{L}_{rt+r}(\mathrm{CA}_3) & \subset & \cdots & \subset & \mathscr{L}_{rt+r}(\mathrm{CA}_k) & \subset & \cdots \\
\cup & & \cup & & & & \cup & & \\
\vdots & & \vdots & & & & \vdots & & \\
\cup & & \cup & & & & \cup & & \\
\mathscr{L}_{rt+1}(\mathrm{CA}_2) & \subset & \mathscr{L}_{rt+1}(\mathrm{CA}_3) & \subset & \cdots & \subset & \mathscr{L}_{rt+1}(\mathrm{CA}_k) & \subset & \cdots \\
\cup & & \cup & & & & \cup & & \\
\mathscr{L}_{rt}(\mathrm{CA}_2) & \subset & \mathscr{L}_{rt}(\mathrm{CA}_3) & \subset & \cdots & \subset & \mathscr{L}_{rt}(\mathrm{CA}_k) & \subset & \cdots
\end{array}
$$

Fig. 7. Two-dimensional infinite hierarchy of CAs with restricted communication.

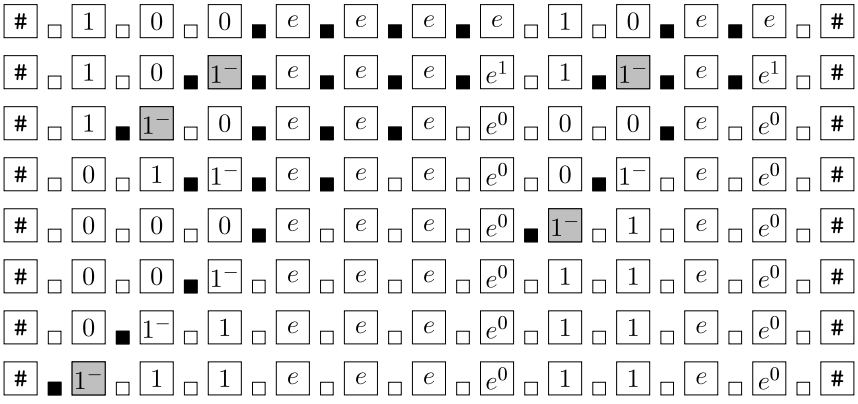

Fig. 8. Schematic computation of a real-time OCA_2 in the construction of Theorem 15 on input 100*eeee*10*ee* and $k = 2$. Accepting states are gray shaded, e^1 and e^0 represent e-cells having no e-cells as right neighbor and sending signals 1 and signals 0 to the left.

From Theorem 13 and Theorem 14 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 Fig. 7.

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

Theorem 15. *Let $k \geq 2$ be a constant. There is a language belonging to the difference $\mathscr{L}_{rt}(OCA_2) \setminus \mathscr{L}_{lt}(IA_k)$.*

Proof. First we give the sketch of the construction of a two-message real-time OCA that accepts the witness language $L_k = \{\, u_1 \cdots u_m e^x v \mid m \geq 1 \text{ and } u_i \in \{a_0, \ldots, a_{k-1}\},\ 1 \leq i \leq m,\ v \in \{e, a_0, \ldots, a_{k-1}\}^*,$ and x is greater than or equal to the number represented by the k-ary interpretation of $u_1 \cdots u_m \,\}$.

Initially, all cells with input e send a 1 and all the other cells send a 0. This identifies cells having input e and a u_i-neighbor or boundary neighbor to the right. Now, all these cells e send one signal 1 to the left and then send signal 0 to the left in all subsequent time steps. All the other cells e send a 1 to the left until they receive a signal 0 from the right. Each block of adjacent cells with input u_i forms a k-ary counter representing the least significant bit in the rightmost cell. The counters are decreased by one in every time step until the rightmost counter cell receives a signal 0. Carry-overs are sent to the left using signal 1. A counter cell accepts when it generates the first carry-over to the left. Clearly, one-way information flow suffices. Moreover, every block e^x sends exactly $x+1$ signals 1 to the left. Thus, the leftmost cell of the block $u_1 \ldots u_m e^x$ generates the first carry-over at time step $x + m$ if x is equal to the k-ary interpretation of $u_1 \ldots u_m$. An example computation is depicted in Fig. 8. So, L_k is accepted by a real-time OCA_2. In particular, L_{k+1} is accepted by a real-time OCA_2 as well.

In order to show that L_{k+1} is not accepted by any IA_k, we assume $L_{k+1} \in \mathscr{L}_{rt}(\mathrm{IA}_k)$. By Lemma 5 there exists a constant $p \in \mathbb{N}_+$ such that $N_\ell(m, L_{k+1}) \leq p \cdot k^m$. On the other hand, consider two different prefixes $u_1 \ldots u_m$ and $v_1 \ldots v_m$ and let l_1 and l_2 be the $(k+1)$-ary interpretation of $u_1 \ldots u_m$ and $v_1 \ldots v_m$. Without loss of generality, we may assume $l_1 < l_2$ and, thus, obtain $u_1 \ldots u_m e^{l_1} \in L_{k+1}$ and $v_1 \ldots v_m e^{l_1} \notin L_{k+1}$. Since there are $(k+1)^m$ such prefixes, $(k+1)^m > k^m + mk^{m-1}$ is a lower bound on the number of induced equivalence classes. Choosing $m \geq p \cdot k$ we obtain $k^m + pk^m = (p+1)k^m$ and, thus, $N_\ell(m, L_{k+1}) > p \cdot k^m$. This is a contradiction and shows the assertion. □

Next, we show proper inclusions between language families that are related by inclusions for structural reasons. In [6] an unrestricted real-time iterative array accepting prime numbers in unary has been constructed. In [20] the result has been improved to a two-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 shown that both devices define incomparable language families. Therefore, we use a different witness language to prove the next result.

Theorem 16. *Let $k \geq 2$ be a constant. Then $\mathscr{L}_{rt}(OCA_k) \subset \mathscr{L}_{rt}(CA_k)$.*

Proof. It is well known that all unary languages belonging to $\mathscr{L}_{rt}(\mathrm{OCA})$ are regular [15]. Therefore, it suffices to show that the non-regular language $L = \{ a^{2^x+2x} \mid x \geq 1 \}$ belongs to $\mathscr{L}_{rt}(\mathrm{CA}_2)$.

A corresponding CA_2 works as follows. We use the construction given in Example 2 of a binary counter whose least significant bit is stored in the leftmost cell. We observe that the counter is extended by one digit (cell) to the right at time steps $2^x + x$, for $x \geq 0$. In particular, at time steps $2^x - 1$ all counter cells store bit 1. Subsequently, it takes $x + 1$ time steps until the carry-overs reach the new cell that extends the counter.

In addition, at time step 1 the rightmost cell sends a signal 1 to the left. The input is accepted if and only if this signal appears in a cell exactly at a time step at which this cell becomes the new most significant bit of the counter, that is, at time steps $2^x + x$. In this case the signal 1 is passed through the counter in order to cause the leftmost cell to accept. Since the previous counter length was x, the total time is $2^x + x + x$. □

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 17. *Let $k \geq 2$ be a constant. Then $\mathscr{L}_{lt}(IA_k) \subset \mathscr{L}_{lt}(CA_k)$.*

Proof. First, we show the equivalence $\mathscr{L}_{lt}(\mathrm{CA}_k) = \mathscr{L}_{lt}(\mathrm{CA})$ which means for cellular automata working in linear time that restricted communication does not affect their computational capacity.

Since the inclusion $\mathscr{L}_{lt}(\mathrm{CA}_k) \subseteq \mathscr{L}_{lt}(\mathrm{CA})$ is obvious, we have to show $\mathscr{L}_{lt}(\mathrm{CA}) \subseteq \mathscr{L}_{lt}(\mathrm{CA}_k)$. Let $\mathcal{M}$ be a linear-time CA having state set S. Then every state from S can be encoded by $m = \lceil \log_2 |S| \rceil$ bits. Now, we construct an equivalent linear-time CA_2 $\mathcal{M}'$ by simulating one time step of $\mathcal{M}$ in m time steps of $\mathcal{M}'$. To this end, every cell in $\mathcal{M}'$ sends in one time step one bit of the encoding of states from S to its neighbors. After m time steps the original state from S can be decoded. $\mathcal{M}'$ accepts the input, if an accepting state of $\mathcal{M}$ is decoded in the leftmost cell. Obviously, $\mathcal{M}'$ is a linear-time CA_2 being equivalent to $\mathcal{M}$.

The equivalence shown implies $\mathscr{L}_{lt}(\mathrm{IA}_k) \subseteq \mathscr{L}_{lt}(\mathrm{IA}) = \mathscr{L}_{lt}(\mathrm{CA}) = \mathscr{L}_{lt}(\mathrm{CA}_k)$. The properness of the inclusion is obtained by using the language L_k from Theorem 15. □

We complement our considerations with the following theorem.

Theorem 18. *Let $k \geq 2$ 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 $\mathscr{L}_{lt}(IA_k)$.*

Proof. We consider the language $L_{k,0}$ from Lemma 11. $L_{k,0}$ is regular and, therefore, contained in $\mathscr{L}_{rt}(\mathrm{IA}_k)$ and $\mathscr{L}_{lt}(\mathrm{IA}_k)$. On the other hand, L_k is not contained in $\mathscr{L}_{rt}(\mathrm{CA}_k)$ and clearly not in $\mathscr{L}_{rt}(\mathrm{OCA}_k)$.

Conversely, the language L_k from Theorem 15 belongs to $\mathscr{L}_{rt}(\mathrm{OCA}_2)$ and, therefore, to $\mathscr{L}_{rt}(\mathrm{OCA}_k)$ and $\mathscr{L}_{rt}(\mathrm{CA}_k)$ as well. On the other hand, L_k is not in $\mathscr{L}_{lt}(\mathrm{IA}_k)$ and, thus, not in $\mathscr{L}_{rt}(\mathrm{IA}_k)$ and REG. □

Clearly, every deterministic finite automaton can be simulated in the communication cell of an iterative array. Thus, we obtain the inclusion $\mathrm{REG} \subset \mathscr{L}_{rt}(\mathrm{IA}_2)$ which is proper due to the fact that the non-regular language $\{ a^n b^n \mid n \geq 1 \}$ can be accepted by some real-time IA_2 [13]. The relations between cellular automata with unrestricted and restricted communication are summarized in Fig. 6. It turns out that we can give more precise results for models with restricted communication. It is a long-standing open problem whether linear-time cellular automata are more powerful than real-time cellular automata. Since linear-time CA_ks can accept all regular languages whereas real-time CA_ks cannot, we can answer this question in the affirmative for the case of restricted communication. On the other hand, Theorem 17 says that in the case of restricted communication a parallel input mode implies a more powerful model than sequential input mode whereas both input modes are equally powerful in the unrestricted case.

In Fig. 9 we summarize the relations between cellular automata with restricted and unrestricted communication. In the proof of Theorem 17 we have shown the equivalence $\mathscr{L}_{lt}(\mathrm{CA}) = \mathscr{L}_{lt}(\mathrm{CA}_k)$. The equivalence $\mathscr{L}_{lt}(\mathrm{OCA}) = \mathscr{L}_{lt}(\mathrm{OCA}_k)$ is proved similarly. The proper inclusions $\mathscr{L}_{rt}(\mathrm{CA}_k)^R \subset \mathscr{L}_{rt}(\mathrm{CA})^R$, $\mathscr{L}_{rt}(\mathrm{CA}_k)^R \subset \mathscr{L}_{lt}(\mathrm{OCA}_k)$, and $\mathscr{L}_{rt}(\mathrm{OCA}_k) \subset \mathscr{L}_{rt}(\mathrm{OCA})$ as well as the incomparability to REG results from the fact that all regular languages can be accepted by the unrestricted models, but not by the restricted models real-time CA_k and OCA_k. The remaining proper inclusion $\mathscr{L}_{rt}(\mathrm{OCA}_k) \subset \mathscr{L}_{rt}(\mathrm{CA}_k)^R$ is obtained by using the language $L = \{ a^{2^x+2x} \mid x \geq 1 \}$ from the proof of Theorem 16.

Finally, we mention the following incomparability result.

Theorem 19. *Let $k \geq 2$ be a constant. Then $\mathscr{L}_{rt}(IA_k)$ is incomparable to the class of context-free languages.*

Proof. It is shown in [5] that there is a context-free language L which cannot be accepted by a real-time IA. Thus, $L \notin \mathscr{L}_{rt}(\mathrm{IA}_k)$. On the other hand, the language $L' = \{ a^n b^n c^n \mid n \geq 0 \}$ belongs to $\mathscr{L}_{rt}(\mathrm{IA}_k)$ due to the construction given in [13], but L' is not context free. □

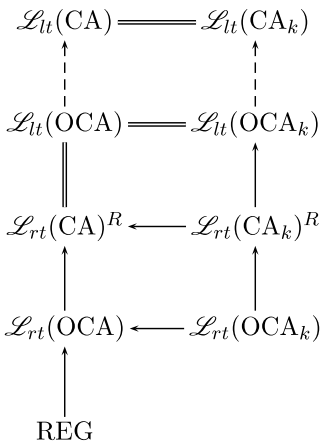

Fig. 9. Relations between unrestricted and restricted language families. Solid arrows are strict inclusions, dashed arrows are inclusions, and double lines denote equivalence.

6. Closure properties

In this section we investigate the closure properties of the language classes accepted by one-dimensional real-time IA_ks and start with positive closure results.

Lemma 20. *Let $k \geq 2$ be a constant. Then the family $\mathscr{L}_{rt}(IA_k)$ is closed under complementation, intersection with regular languages, union with regular languages, and right concatenation with regular languages.*

Proof. All constructions can be realized in the communication cell and thus are identical to the general case of unrestricted communication [15]. □

It will be shown in Lemma 23 that real-time IA_ks are not closed under arbitrary inverse homomorphisms. Nevertheless, $\mathscr{L}_{rt}(\mathrm{IA}_k)$ is closed under inverse letter-to-letter homomorphisms. A homomorphism $h : A \to A'$ is called *letter-to-letter* if $|h(a)| = 1$ for all $a \in A$.

Lemma 21. *Let $k \geq 2$ be a constant. Then the family $\mathscr{L}_{rt}(IA_k)$ is closed under inverse letter-to-letter homomorphisms.*

Proof. Let $L \in \mathscr{L}_{rt}(\mathrm{IA}_k)$ be accepted by some real-time IA_k $\mathcal{M}$ and h be a letter-to-letter homomorphism. Basically, the communication cell of a real-time IA_k $\mathcal{M}'$ accepting the inverse image $h^{-1}(L) = \{ w \mid h(w) \in L \}$ simulates the behavior of the communication cell of $\mathcal{M}$ on $h(a)$ for every input symbol a. The remaining cells are not affected by the construction. So, $\mathcal{M}'$ accepts an input w if and only if $\mathcal{M}$ accepts $h(w)$. Since h is a letter-to-letter homomorphism, we obtain $|w| = |h(w)|$. Therefore, $\mathcal{M}'$ accepts $h^{-1}(L)$ in real time. □

We now turn to non-closure results and first show the non-closure under union and intersection. In case of unrestricted communication these operations can be realized using the Cartesian product construction whereas such constructions are not possible in case of restricted communication. In the sequel we use the alphabet $A_k = \{a_0, \ldots, a_{k-1}\}$ for $k \geq 2$.

Lemma 22. *Let $k \geq 2$ be a constant. Then $\mathscr{L}_{rt}(IA_k)$ is not closed under intersection and union.*

Proof. First, we show the non-closure under union. Let $A'_k = A_k \cup \{b\}$ where $b \notin A_k$, and let $M_k = \{M_{1,k}, \ldots, M_{n,k}\}$ be an enumeration of all subsets of A'_k of size k.

For a given $j \in \mathbb{N}_+$ so that $1 \leq j \leq |M_k|$, we consider the languages $L(j, k) = L_{bit}(1, k)\$w_{j,k}$ where $w_{j,k}$ is some fixed word from $M_{j,k}^*$ which enumerates all elements from $M_{j,k}$ and thus identifies the set $M_{j,k}$.

Since $L_{bit}(1, k)$ belongs to $\mathscr{L}_{rt}(\mathrm{IA}_k)$ due to Theorem 9, $L(j, k)$ belongs to $\mathscr{L}_{rt}(\mathrm{IA}_k)$ as well (the suffix $\$w_{j,k}$ can be checked in the communication cell).

Next, we consider a letter-to-letter homomorphism $h_{j,k} : A'_k \cup \{e, \$\} \to M_{j,k} \cup \{e, \$\}$, where $h_{j,k}(u) = u$, for $u \in M_{j,k} \cup \{e, \$\}$, define

$$L'(j, k) = h_{j,k}^{-1}(L(j, k)) \cap e^*\$(A'_k)^*\$w_{j,k},$$

and obtain

$$L'(j, k) = \{ e^x\$u_1u_2 \cdots u_m\$w_{j,k} \mid x \in \mathbb{N}_+ \text{ and } m \geq 2x - 1 \text{ and } u_i \in A'_k,\ 1 \leq i \leq m,$$
$$\text{and } h_{j,k}(u_t) = h_{j,k}(u_{t+2x-1}),\ 1 \leq t \leq m - (2x - 1) \}$$

Since $\mathscr{L}_{rt}(\mathrm{IA}_k)$ is closed under inverse letter-to-letter homomorphism and intersection with regular sets, we obtain $L'(j, k) \in \mathscr{L}_{rt}(\mathrm{IA}_k)$. For each pair $u_1, u_2 \in A'_k$ with $u_1 \neq u_2$ we now choose some homomorphism $h_{j,k}$, and thus a set

$L'(j,k)$, such that $h_{j,k}(u_1) = u_1$ and $h_{j,k}(u_1) \neq h_{j,k}(u_2)$. Since $|A'_k|$ and $|M_{j,k}|$, for every $1 \leq j \leq |M_k|$, differ by one, it is always possible to find a set $M_{j,k}$ and a homomorphism $h_{j,k}$ such that $u_1 \in M_{j,k}$, $u_2 \notin M_{j,k}$, and $h_{j,k}(u_1) \neq h_{j,k}(u_2)$.

Now, let L_k be the union of all such sets. Next, we show that $L_k \notin \mathscr{L}_{rt}(\mathrm{IA}_k)$ which implies that $\mathscr{L}_{rt}(\mathrm{IA}_k)$ is not closed under union. By way of contradiction, we assume that L_k is accepted by a real-time IA_k $\mathcal{M}$. By the same reasoning as in the proof of Theorem 9, we obtain that there are at least two different words $u_1 \cdots u_{2x-1}$ and $u'_1 \cdots u'_{2x-1}$ such that $u_i \neq u'_i$ for some i, and for which $\mathcal{M}$ has computed the same configuration. Furthermore, there is a homomorphism $h_{j,k}$ such that $h_{j,k}(u_i) \neq h_{j,k}(u'_i)$ and $h_{j,k}(u_i) = u_i$. We can conclude that $w = e^x\$u_1 \cdots u_{2x-1}u_1 \cdots u_{i-1}h_{j,k}(u_i)u_{i+1} \cdots u_{2x-1}\$w_{j,k}$ and $w' = e^x\$u'_1 \cdots u'_{2x-1}u_1 \cdots u_{i-1}h_{j,k}(u_i)u_{i+1} \cdots u_{2x-1}\$w_{j,k}$ are both accepted by $\mathcal{M}$, since $w = e^x\$u_1 \cdots u_{2x-1}u_1 \cdots u_{i-1}u_iu_{i+1} \cdots u_{2x-1}\$w_{j,k} \in L'(j,k)$ and thus $w \in L_k$. On the other hand, $w' = e^x\$u'_1 \cdots u'_{2x-1}u_1 \cdots u_{i-1}u_iu_{i+1} \cdots u_{2x-1}\$w_{j,k} \notin L'(j,k)$, since $h_{j,k}(u'_i) \neq h_{j,k}(u_i) = u_i$. Additionally, w' does not belong to any other set $L'(j',k)$ due to the suffix $\$w_{j,k}$. Altogether, $w' \notin L_k$ which is a contradiction. Thus, $L_k \notin \mathscr{L}_{rt}(\mathrm{IA}_k)$ and $\mathscr{L}_{rt}(\mathrm{IA}_k)$ is not closed under union. Since $\mathscr{L}_{rt}(\mathrm{IA}_k)$ is closed under complementation, $\mathscr{L}_{rt}(\mathrm{IA}_k)$ is not closed under intersection as well. □

The next result shows non-closure under arbitrary inverse homomorphisms, whereas closure under inverse letter-to-letter homomorphisms is known by Lemma 21.

Lemma 23. *Let $k \geq 2$ be a constant. Then the family $\mathscr{L}_{rt}(IA_k)$ is not closed under inverse homomorphism.*

Proof. Let language L_k be defined as the restriction of $L_{bit}(1,k)$ to words starting with an even number of es, that is,

$$L_k = \{\, e^{2x}\$u_1u_2 \cdots u_m \mid x \in \mathbb{N}_+ \text{ and } m \geq 4x-1 \text{ and } u_i \in A_k,\ 1 \leq i \leq m, \text{ and } u_j = u_{j+4x-1},\ 1 \leq j \leq m-(4x-1) \,\}.$$

Clearly, $L_k \in \mathscr{L}_{rt}(\mathrm{IA}_k)$: in addition to the construction given in Theorem 9, it is checked in the communication cell whether the number of es is even. Let $A'_k = \{b_1, \ldots, b_n\}$ be an alphabet of size $n = k^2$ and consider a homomorphism $h : A'_k \cup \{\$, e\} \to A_kA_k \cup \{\$, e\}$ so that $h(\$) = \$$, $h(e) = ee$, $h(b_1) = a_0a_0$, $h(b_2) = a_0a_1$, $h(b_3) = a_0a_2$, $\ldots$, $h(b_n) = a_{k-1}a_{k-1}$. Then we have

$$h^{-1}(L_k) = \{\, e^x\$u_1u_2 \cdots u_m \mid x \in \mathbb{N}_+ \text{ and } m \geq 2x-1 \text{ and } u_i \in A'_k,\ 1 \leq i \leq m, \text{ and } u_j = u_{j+2x-1},\ 1 \leq j \leq m-(2x-1) \,\}.$$

By way of contradiction, assume that $\mathscr{L}_{rt}(\mathrm{IA}_k)$ is closed under h^{-1}. Then $h^{-1}(L_k)$ belongs to $\mathscr{L}_{rt}(\mathrm{IA}_k)$. Since $|A'_k| = k^2 \geq k+1$ for $k \geq 2$, we can show that $h^{-1}(L_k) \notin \mathscr{L}_{rt}(\mathrm{IA}_k)$ analogously to the proof of Theorem 9. This is a contradiction which concludes the proof. □

In order to show the next negative closure results we define two languages L'_k and L''_k where alphabet A'_k is a primed copy of A_k.

$$L'_k = \{\, u_1 \cdots u_m\$a_0^{j-1}u_{m-j+1} \mid j, m \in \mathbb{N}_+,\ u_i \in A_k,\ 1 \leq i \leq m,\ 1 \leq j \leq m \,\}$$
$$L''_k = \{\, u_1 \cdots u_{m-j}u'_{m-j+1}u_{m-j+2} \cdots u_m\$a_0^{j-1}u_{m-j+1} \mid j, m \in \mathbb{N}_+,\ u_i \in A_k,\ u'_i \in A'_k,\ 1 \leq i \leq m,\ 1 \leq j \leq m \,\}.$$

Lemma 24. *Let $k \geq 2$ be a constant. Then $L'_{k+1} \notin \mathscr{L}_{rt}(IA_k)$ and $L''_{k+1} \in \mathscr{L}_{rt}(IA_k)$.*

Proof. By way of contradiction, we assume that $L'_{k+1} \in \mathscr{L}_{rt}(\mathrm{IA}_k)$. Then, by Lemma 5 there exists a constant $p \in \mathbb{N}_+$ such that $N_\ell(l, L'_{k+1}) \leq p \cdot k^l$ for all $l \in \mathbb{N}_+$. On the other hand, consider two different prefixes $w = u_1 \cdots u_l\$$ and $w' = u'_1 \cdots u'_l\$$. Since they are different, there is a j such that $u_j \neq u'_j$, and we obtain

$$wa_0^{(l-j+1)-1}u_{l-(l-j+1)+1} = wa_0^{l-j}u_j \in L'_{k+1} \iff w'a_0^{l-j}u_j \notin L'_{k+1}.$$

There are $(k+1)^l$ such prefixes. Since $\frac{k+1}{k} > 1$, we may choose l in such a way that $\left(\frac{k+1}{k}\right)^l > p$. This implies $N_\ell(l, L'_{k+1}) > p \cdot k^l$ as a lower bound on the number of induced equivalence classes. From the contradiction we obtain $L'_{k+1} \notin \mathscr{L}_{rt}(\mathrm{IA}_k)$.

The construction of a real-time IA_k accepting L''_{k+1} may be sketched as follows. At first, the communication cell checks whether the input satisfies the correct format of L''_{k+1}. Additionally, the input is read until the first input symbol from A'_k occurs which is stored in the finite control of the communication cell. It has been shown in [13] how a two-message real-time IA can accept the language $\{\, a^nb^n \mid n \geq 1 \,\}$. This construction consists of increasing a binary counter while reading as and decreasing the counter while reading bs. Finally, it has to be checked that the counter has been decreased to zero. A similar construction can be used here: A binary counter is increased for every input symbol read until the input \$ occurs. The counter is then decreased for every input symbol. If the counter has been decreased to zero, the next input symbol is compared to the symbol stored. If both are equal, the input is accepted and otherwise rejected. □

Lemma 25. *Let $k \geq 2$ be a constant. Then the family $\mathscr{L}_{rt}(IA_k)$ is not closed under homomorphism, λ-free homomorphism, and reversal.*

Proof. It is shown in Lemma 24 that L''_{k+1} belongs to $\mathscr{L}_{rt}(\mathrm{IA}_k)$. Assume that $\mathscr{L}_{rt}(\mathrm{IA}_k)$ is closed under λ-free homomorphism. For the λ-free homomorphism $h : A_k \cup A'_k \to A_k$ with $h(a') = h(a) = a$, for all $a \in A_k$, we have $h(L''_{k+1}) = L'_{k+1} \in \mathscr{L}_{rt}(\mathrm{IA}_k)$, which is a contradiction to Lemma 24. The non-closure under arbitrary homomorphism is obvious. Next, consider the

Table 1
Closure properties of $\mathscr{L}_{rt}(\mathrm{IA}_k)$. $\cap R$ and $\cup R$ denote intersection and union with regular sets. $\cdot R$ and $R\cdot$ denote right and left concatenation with regular sets.

	$\overline{\ \ }$	$\cap$	$\cup$	$\cap R$	$\cup R$	$\cdot$	$*$	h	h_λ	h^{-1}	$h^{-1}_{1\text{-}1}$	$\cdot R$	$R\cdot$	^{R}
$\mathscr{L}_{rt}(\mathrm{IA})$	+	+	+	+	+	−	−	−	−	+	+	+	−	−
$\mathscr{L}_{rt}(\mathrm{IA}_k)$	+	−	−	+	+	−	−	−	−	−	+	+	−	−
CFL	−	−	+	+	+	+	+	+	+	+	+	+	+	+

reversal of L'_{k+1}. It is not difficult to see that $(L'_{k+1})^R \in \mathscr{L}_{rt}(\mathrm{IA}_k)$. Since $((L'_{k+1})^R)^R = L'_{k+1}$, we obtain that $\mathscr{L}_{rt}(\mathrm{IA}_k)$ is not closed under reversal. □

Lemma 26. *Let $k \geq 2$ be a constant. Then the family $\mathscr{L}_{rt}(IA_k)$ is not closed under left concatenation with regular languages, concatenation, and Kleene star.*

Proof. The language

$$L'''_{k+1} = \{\, u_1u_2\cdots u_m\$a_0^{m-1}u_1 \mid m \in \mathbb{N}_+ \text{ and } u_i \in A_{k+1},\ 1 \leq i \leq m \,\}$$

belongs to $\mathscr{L}_{rt}(\mathrm{IA}_k)$. The first input symbol u_1 is stored in the communication cell, and a binary counter is implemented, which is incremented until a \$ appears in the input. The counter is then decremented. If it is decreased to one, the next input symbol is compared with the symbol u_1 which is stored in the communication cell.

Concatenating the regular language A^*_{k+1} from left, we obtain $A^*_{k+1}L'''_{k+1} = L'_{k+1}$. The assumption that $\mathscr{L}_{rt}(\mathrm{IA}_k)$ is closed under left concatenation with regular languages leads to a contradiction. So, the non-closure under concatenation follows immediately.

To show non-closure under Kleene star we consider the language $(A^*_{k+1} \cup L'''_{k+1}\$)^* \cap (A^*_{k+1}\$a_0^*A_{k+1}\$)$ which is identical to $L'_{k+1}\$$. Since $\mathscr{L}_{rt}(\mathrm{IA}_k)$ is closed under union, intersection, and right concatenation with regular languages, the assumption that $\mathscr{L}_{rt}(\mathrm{IA}_k)$ is closed under Kleene star leads to a contradiction. □

The closure properties of $\mathscr{L}_{rt}(\mathrm{IA}_k)$ are summarized in Table 1. The main differences to real-time IAs with unrestricted communication are the non-closure under union, intersection, and inverse homomorphism. The remaining closure properties are identical. In particular, restricted communication does not lead to additional positive closure properties.

7. Conclusions

We have investigated d-dimensional IAs and one-dimensional cellular automata operating in real and linear time, whose inter-cell communication is restricted to some constant number of different messages. We obtained an infinite strict and dense double hierarchy over the dimension and the number of messages. The computational capacity of one-dimensional devices has been compared with the power of two-way and one-way cellular automata with restricted communication. It turned out that the relations between iterative arrays and cellular automata are quite different from the relations in the unrestricted case. Moreover, we obtained an infinite strict message hierarchy for real-time two-way cellular automata and a dense time hierarchy for k-message cellular automata. Additionally, we studied the relation to the family of regular and context-free languages and obtained in many cases incomparability results. We investigated closure properties of one-dimensional real-time IAs with restricted communication. It turned out that this class is not closed under union, intersection, and inverse homomorphism whereas the unrestricted variant is closed under these operations. On the other hand, we could show that closure properties which do not hold for the unrestricted case, do not hold for the restricted case as well.

For several results the alphabet of witness languages has been chosen dependent on structural and computational resources of the devices in question, in particular, on the communication bandwidth or on the time complexity beyond real time. While it is often easy to increase the input alphabet such that a language cannot be accepted by some device with limited resources, it is not that easy to design a language in that way so that it becomes acceptable again when the resources are slightly increased. That is, to obtain strict and dense hierarchies. Nevertheless, the challenging question whether the number of symbols can be decreased in the order of magnitude or even to constants independent of the resources remains unanswered.

References

[1] T. Buchholz, A. Klein, M. Kutrib, Iterative arrays with a wee bit alternation, in: Fundamentals of Computation Theory, FCT 1999, in: LNCS, vol. 1684, Springer, 1999.
[2] T. Buchholz, A. Klein, M. Kutrib, Iterative arrays with small time bounds, in: Mathematical Foundations of Computer Science, MFCS 1998, in: LNCS, vol. 1893, Springer, 2000.
[3] T. Buchholz, A. Klein, M. Kutrib, Iterative arrays with limited nondeterministic communication cell, in: Words, Languages and Combinatorics III, World Scientific Publishing, 2003.
[4] J. H. Chang, O. H. Ibarra, M. A. Palis, Parallel parsing on a one-way array of finite-state machines, IEEE Trans. Comput. C-36 (1987) 64–75.
[5] S. N. Cole, Real-time computation by n-dimensional iterative arrays of finite-state machines, IEEE Trans. Comput. C-18 (1969) 349–365.

[6] P.C. Fischer, Generation of primes by a one-dimensional real-time iterative array, J. ACM 12 (1965) 388–394.
[7] O. H. Ibarra, M. A. Palis, Some results concerning linear iterative (systolic) arrays, J. Parallel Distributed Comput. 2 (1985) 182–218.
[8] O.H. Ibarra, M.A. Palis, Two-dimensional iterative arrays: Characterizations and applications, Theoret. Comput. Sci. 57 (1988) 47–86.
[9] C. Iwamoto, T. Hatsuyama, K. Morita, K. Imai, On time-constructible functions in one-dimensional cellular automata, in: Fundamentals of Computation Theory, FCT 1999, in: LNCS, vol. 1684, Springer, 1999.
[10] M. Kutrib, Cellular automata — a computational point of view, in: New Developments in Formal Languages and Applications, Springer, 2008, pp. 183–227 (chapter 6).
[11] M. Kutrib, Cellular automata and language theory, in: Encyclopedia of Complexity and System Science, Springer, 2009, pp. 800–823.
[12] M. Kutrib, A. Malcher, Fast cellular automata with restricted inter-cell communication: Computational capacity, in: Theoretical Computer Science, IFIP TCS2006, in: IFIP, vol. 209, Springer, 2006.
[13] M. Kutrib, A. Malcher, Computations and decidability of iterative arrays with restricted communication, Parallel Process. Lett. 19 (2009) 247–264.
[14] A. Malcher, On the descriptional complexity of iterative arrays, IEICE Trans. Inf. Syst. E87-D (2004) 721–725.
[15] S.R. Seidel, Language recognition and the synchronization of cellular automata, Tech. Rep. 79-02, Department of Computer Science, University of Iowa, 1979.
[16] A. R. Smith III, Real-time language recognition by one-dimensional cellular automata, J. Comput. System Sci. 6 (1972) 233–253.
[17] V. Terrier, On real time one-way cellular array, Theoret. Comput. Sci. 141 (1995) 331–335.
[18] H. Umeo, Linear-time recognition of connectivity of binary images on 1-bit inter-cell communication cellular automaton, Parallel Comput. 27 (2001) 587–599.
[19] H. Umeo, N. Kamikawa, 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.
[20] H. Umeo, N. Kamikawa, Real-time generation of primes by a 1-bit-communication cellular automaton, Fund. Inform. 58 (2003) 421–435.
[21] T. Worsch, Linear time language recognition on cellular automata with restricted communication, in: Theoretical Informatics (LATIN 2000), in: LNCS, vol. 1776, Springer, 2000.