Fast Cellular Automata with Restricted Inter-Cell Communication: Computational Capacity

Martin Kutrib[1] and Andreas Malcher[2]

[1] Institut für Informatik, Universität Giessen
Arndtstr. 2, D-35392 Giessen, Germany
kutrib@informatik.uni-giessen.de
[2] Institut für Informatik, Johann Wolfgang Goethe Universität
D-60054 Frankfurt am Main, Germany
a.malcher@em.uni-frankfurt.de

Abstract. A d-dimensional cellular automaton with sequential input mode is a d-dimensional grid of interconnected interacting finite automata. 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 iterative array. We investigate d-dimensional iterative arrays and one-dimensional cellular automata operating in real and linear time, whose inter-cell communication is restricted to some constant number of bits independent of the number of states. It is known that even one-dimensional one-bit iterative arrays accept rather complicated languages such as $\{a^p \mid p \text{ prim}\}$ or $\{a^{2^n} \mid n \in \mathbb{N}\}$ [16]. We show that there is an infinite strict double dimension-bit hierarchy. The computational capacity of the one-dimensional devices in question is compared with the power of communication-restricted two-way cellular automata. It turns out that the relations are quite different from the relations in the unrestricted case. On passing, we obtain an infinite strict bit hierarchy for real-time two-way cellular automata and, moreover, a very dense time hierarchy for every k-bit cellular automata, i.e., just one more time step leads to a proper superfamily of accepted languages.

Key words: Cellular automata; Iterative arrays; Restricted communication; Formal languages; Computational capacity; Parallel computing

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 con-

nections whose cells are finite automata are commonly called cellular automata. 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. In [6] a real-time acceptor for prime numbers has been constructed. 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 (e.g., in [12, 13]).

In order to investigate the computational capacity of a device, 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 [10].

All these results concern iterative arrays where the states of the neighboring cells are communicated in one time step. That is, the number of bits exchanged is determined by the number of states. A natural and interesting restriction of IAs is to restrict the number of bits by some constant being independent of the number of states. Iterative arrays with restricted inter-cell communication have been investigated in [15, 16], 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. Connectivity recognition problems are dealt with in [14], whereas in [17] the computational capacity of one-way cellular automata with restricted inter-cell communication is considered.

Here we investigate d-dimensional iterative arrays 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 bits, 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 main model in question, i.e., d-dimensional iterative arrays with restricted inter-cell communication. Section 3 is devoted to dimension and bit hierarchies. We show that there is an infinite strict double hierarchy. That is, for every dimension real-time $(k+1)$-bit restricted iterative arrays are strictly more powerful than real-time k-bit restricted iterative arrays, and for every k-bit restriction real-time $(d+1)$-dimensional k-bit restricted iterative arrays are strictly more powerful than real-time d-dimensional k-bit restricted iterative arrays. In Section 4 we consider one-dimensional devices. The computational capacity of the devices in question is compared with the power of communication-restricted two-way cellular automata. It turns out that the relations are quite different from the relations in the unrestricted case. On passing, we obtain an infinite

strict bit hierarchy for real-time two-way cellular automata and, moreover, a very dense time hierarchy for every k-bit cellular automata, i.e., just one more time step yields to a proper superfamily of accepted languages.

2 Definitions and Preliminaries

We denote the rational numbers by $\mathbb{Q}$, the integers by $\mathbb{Z}$, the non-negative integers by $\mathbb{N}$, and the positive integers $\{1, 2, ...\}$ 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 $A^{\leq l}$. Set inclusion and strict set inclusion are denoted by $\subseteq$ and $\subset$, respectively.

A d-dimensional iterative array is a d-dimensional array (i.e. $\mathbb{N}^d$) of finite automata, sometimes called cells, where each of them is connected to its nearest neighbors in every dimension. 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-bit restricted inter-cell communication, during every time step each cell may communicate only k bit of information to its neighbors. These bits depend on the current state and are determined by so-called bit-functions. The finite automata work synchronously at discrete time steps.

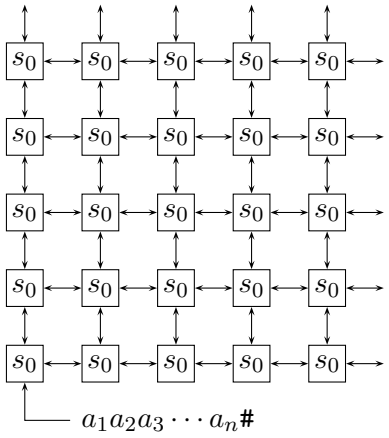

Fig. 1. A two-dimensional iterative array.

Definition 1. *A* d-dimensional iterative array with k-bit restricted inter-cell communication (IA_k^d) *is a system* $\langle S, A, F, s_0, d, k, b_1, \ldots, b_{2d}, \delta, \delta_0 \rangle$, *where*

(1) S *is the finite, nonempty set of* cell states,
(2) A *is the finite, nonempty set of* input symbols,
(3) $F \subseteq S$ *is the set of* accepting states,
(4) $s_0 \in S$ *is the* quiescent state,
(5) $d \in \mathbb{N}_+$ *is the* dimension,
(6) $k \in \mathbb{N}_+$ *is the* number of bits *which can be* communicated *to neighbor cells,*
(7) $b_i : S \to \{0,1\}^k$, *for* $1 \leq i \leq 2d$, *are* bit functions *which determine the bits* to communicate *to neighbors, satisfying* $b_i(s_0) = (0, \ldots, 0)$,
(8) $\delta : S \times (\{0,1\}^k)^{2d} \to S$ *is the* local transition function for non-communication cells *satisfying* $\delta(s_0, (0, \ldots, 0), \ldots, (0, \ldots, 0)) = s_0$,
(9) $\delta_0 : S \times (A \cup \{\#\}) \times (\{0,1\}^k)^d \to S$ *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 $c_t : \mathbb{N}_0^d \to S$ 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 $\mathbb{Z}^d \setminus \mathbb{N}_0^d$ are permanently in the quiescent state sending zeroes. The configuration (w_0, c_0) at time 0 is defined by the input word w_0 and the mapping $c_0(i_1, \ldots, i_d) = s_0$, $(i_1, \ldots, i_d) \in \mathbb{N}_0^d$, 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 $(i_1, \ldots, i_d) \in \mathbb{N}_0^d \setminus \{(0, \ldots, 0)\}$, 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.

A word w is accepted by an IA_k^d if at some time i during its course of computation on input w the communication cell becomes accepting.

Definition 2. *Let* $\mathcal{M} = \langle S, A, F, s_0, d, k, b_1, \ldots, b_{2d}, \delta, \delta_0 \rangle$ *be an* IA_k^d.

(1) A word $w \in A^*$ *is* accepted *by* $\mathcal{M}$, *if there exists a time step* $i \in \mathbb{N}$ *such that* $c_i(0, \ldots, 0) \in F$.

(2) $L(\mathcal{M}) = \{w \in A^* \mid w \text{ is accepted by } \mathcal{M}\}$ *is the* language accepted *by* $\mathcal{M}$.
(3) Let $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n+1$, *be a mapping. If all* $w \in L(\mathcal{M})$ *are accepted with at most* $t(|w|)$ *time steps, then* L *is said to be of* time complexity t.

The family of all languages which can be accepted by an IA_k^d with time complexity t is denoted by $\mathscr{L}_t(\mathrm{IA}_k^d)$. If t equals the function $n+1$, acceptance is said to be in *real time* and we write $\mathscr{L}_{rt}(\mathrm{IA}_k^d)$. The *linear-time* languages $\mathscr{L}_{lt}(\mathrm{IA}_k^d)$ are defined according to $\mathscr{L}_{lt}(\mathrm{IA}_k^d) = \bigcup_{i \in \mathbb{Q}, i \geq 1} \mathscr{L}_{i \cdot n}(\mathrm{IA}_k^d)$.

Definition 3. *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 4. *Let* $k, d \in \mathbb{N}_+$ *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*

$$N_\ell(l, L) \leq p \cdot 2^{k \cdot d \cdot l}$$

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

Proof. Let $\mathcal{M} = \langle S, A, F, s_0, d, k, b_1, \ldots, b_{2d}, \delta, \delta_0 \rangle$ 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 a IA_k^d that accepts L with time complexity t. In order to determine an upper bound to 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 is sent to the array by the communication cell, and the current state of the communication cell. So, there are at most $(2^{k \cdot d})^{|w|-1} \cdot |S| \leq |S| \cdot 2^{k \cdot d \cdot l}$ different configurations. Setting $p = |S|$, we obtain $N_\ell(l, L) \leq p \cdot 2^{k \cdot d \cdot l}$. In particular, the number of equivalence classes is independent of the time complexity t. □

3 Dimension and Bit Hierarchies

The hierarchies are proved by specific witness languages which are defined dependent on the given resources.

3.1 Dimensions

Here we fix the time complexity to real time, the number of communication bits to any constant $k \in \mathbb{N}_+$, 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

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

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

Theorem 5. *Let $k, d \in \mathbb{N}_+$ be constants. The language $L_{dim}(d+1)$ belongs to the difference $\mathscr{L}_{rt}(IA_1^{d+1}) \setminus \mathscr{L}_{rt}(IA_k^d)$.*

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

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

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 $u[x_{d+1}][x_d] \cdots [x_1]$. Therefore, $h(u)h(\text{¢}e^{x_{d+1}}\$ \cdots \$e^{x_1}\$e^{2x}\$a) \in L_{dim}(d+1) \iff h(u')h(\text{¢}e^{x_{d+1}}\$ \cdots \$e^{x_1}\$e^{2x}\$a) \notin L_{dim}(d+1)$.

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 4 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_1^{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 also 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 $\mathtt{¢}e^{x_{d+1}}\mathtt{\$} \cdots \mathtt{\$}e^{x_1}\mathtt{\$}e^{2x}\mathtt{\$}v$ 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, i.e., $(i_1, \ldots, i_{d+1-j}, 0, \ldots, 0)$. 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 $h(a)$ (resp. $h(b)$), it sends the two bits 10 (resp. 11) along the path until the position is reached. Now the corresponding cell $(i_1, \ldots, i_{d+1})$ stores symbol a (resp. b), activates its neighbor $(i_1, \ldots, i_{d+1} + 1)$ to be the next storage position by sending the bits 01, and extends the current path to the newly activated neighbor.

Whenever the communication cell reads $h(\mathtt{\$})$, it sends the bits 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 $h(\mathtt{¢})$, it sends the bits 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 bit 1 for each read $h(e)$, and the bits 00 for each of the next $d+1$ separators $h(\mathtt{\$})$.

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 1 signals 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 6. *Let $d, k \in \mathbb{N}_+$ be constants, then $\mathscr{L}_{rt}(IA_k^d) \subset \mathscr{L}_{rt}(IA_k^{d+1})$.*

Proof. By Theorem 5, language $L_{dim}(d+1)$ is not accepted by any real-time IA_k^d, but is accepted by some real-time IA_1^{d+1} and, thus, by some IA_k^{d+1}. □

The construction of Theorem 5 can be modified to show that the language $L_{dim}(d+1)$ belongs to $\mathscr{L}_{lt}(\mathrm{IA}_1^d)$, i.e., one can trade one dimension for a slowdown from real time to linear time.

Theorem 7. *Let $k, d \in \mathbb{N}_+$ be constants, then $\mathscr{L}_{rt}(IA_k^d) \subset \mathscr{L}_{lt}(IA_k^d)$.*

3.2 Bits

Here we fix the time complexity to real time, the dimension to any constant $d \in \mathbb{N}_+$, and consider the number of communication bits. For any number of communication bits $k \in \mathbb{N}_+$ we define an alphabet $A_{d,k} = \{a_0, \ldots, a_{2^{d \cdot k}-2}\}$ and a language $L_{bit}(d, k)$.

$$L_{bit}(d,k) = \{u_1 \cdots u_m \$ e^{2m+4} \$ e^x \$ e^{2x} \$ v \mid m, x \in \mathbb{N}_+ \text{ and } x \leq m \text{ and } u_i \in A_{d,k}, 1 \leq i \leq m, \text{ and } v = u_x\}$$

Theorem 8. *Let $k, d \in \mathbb{N}_+$ 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 $L_{bit}(d, k+1) \in \mathscr{L}_{rt}(\mathrm{IA}_k^d)$. Then by Lemma 4 there exists a constant $p \in \mathbb{N}_+$ such that $N_\ell(l, L_{bit}(d, k+1)) \leq p \cdot 2^{k \cdot d \cdot l}$, 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 an x such that $u_x \neq u'_x$. Therefore, $we^{2l+4}\$e^x\$e^{2x}\$u_x \in L_{bit}(d, k+1) \iff w'e^{2l+4}\$e^x\$e^{2x}\$u_x \notin L_{bit}(d, k+1)$. For all $d, k \in \mathbb{N}_+$, there are

$$(2^{d\cdot(k+1)} - 1)^l = (2^d 2^{d\cdot k} - 1)^l \geq (2 \cdot 2^{d\cdot k} - 1)^l \geq$$
$$(2^{d\cdot k} + 1)^l > (2^{d\cdot k} + \frac{1}{2})^l = ((1 + \frac{1}{2^{d\cdot k+1}})2^{d\cdot k})^l$$

different words of this form. Since $\frac{1}{2^{d\cdot k+1}} > 0$, we may choose l in such a way that $(1 + \frac{1}{2^{d\cdot k+1}})^l > p$. This implies the following lower bound on the number of induced equivalence classes:

$$N_\ell(l, L_{bit}(d, k+1)) > p \cdot 2^{d\cdot k\cdot l}$$

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

It remains to be shown that $L_{bit}(d, k+1) \in \mathscr{L}_{rt}(\mathrm{IA}_{k+1}^d)$. As in the proof of Theorem 5, a corresponding iterative array stores the symbols u_i in a storage phase, and in a retrieval phase symbol u_x is addressed and sent back to the communication cell that compares it with v. The input symbols are binary encoded by $(k+1) \cdot d$ bits, respectively, such that the code of a_i is $i+1$.

First, we present the construction for $d = 1$, which is generalized subsequently. During the storage phase, a symbol u_i is read and its code is sent to the array. It is stored in cell i at time step $2i$. At time $2i+1$ cell $i+1$ is activated by cell i. So, cell $i+1$ can store symbol u_{i+1} at time $2(i+1)$. The following behavior stops the storage phase. It is constructed with an eye towards generalizations to higher dimensions. When the communication cell reads the symbol \$ it sends a 0 to the array. When this 0 is to be stored in cell $m+1$ at time $2(m+1)$, the cell recognizes the end of the storage phase, waits for three time steps, and sends a signal from right to left that informs all cells passed through about the end of the phase. The signal arrives at the communication cell at time step $2(m+1) + 3 + (m+1) = 3m+6$, i.e., when the input prefix $u_1 \cdots u_m\$e^{2m+4}\$$ has been read.

Now the retrieval phase starts. To this end, the communication cell sends signals 1 to the array as long as it reads the next input part e^x. When it reads the following \$ it sends a 2. Each cell which receives a 1 for the first time deletes the 1 from the stream. The unique cell that receives the 2 immediately after receiving a 1 for the first time, identifies itself to be the addressed cell x. It sends its stored symbol u_x to the left. The symbol arrives at the communication cell at time $3x$ after the beginning of the retrieval phase, i.e., before the v appears in the input.

We turn to higher dimensions. Roughly, the idea is to split the encodings of the input symbols u_i into d blocks of length k bits, respectively. These blocks are distributed to the d neighbors of the communication cell. This would lead to a straightforward generalization. But the problem arises that we cannot stop the storage phase since signal 0 (and any other signal) may appear as a block

in encodings. So, we have to provide more sophisticated mechanisms. The communication cell still sends the d blocks to its neighbors. But the blocks, e.g., of symbol u_i are stored in the cells $(i, 0, \ldots, 0)$, $(i, 1, 0, \ldots, 0)$, $(i, 0, 1, 0, \ldots, 0)$,..., $(i, 0, \ldots, 0, 1)$. For example, the block sent to neighbor $(0, \ldots, 0, 1, 0, \ldots, 0)$ of the communication cell is rerouted to the cells $(i, 0, \ldots, 0, 1, 0, \ldots, 0)$ by this neighbor. The communication cell itself sends blocks to cells $(i, 0, \ldots, 0)$ with one time step delay. Therefore, all blocks of symbol u_i reach their destinations at time $2i + 1$. In order to stop the storage phase, the symbol u_i is reconstructed in cell $(i, 0, \ldots, 0)$ at time $2i + 2$. To this end, all cells storing blocks of u_i send their blocks to their common neighbor $(i, 0, \ldots, 0)$. If cell $(m + 1, 0, \ldots, 0)$ reconstructs the signal 0, it sends stop signals back to its neighbors at time $2(m + 1) + 3$. In turn, these neighbors send stop signals back to the communication cell. So, the storage phase ends at time $2(m+1)+3+(m+1) = 3m+6$, i.e., when the input prefix $u_1 \cdots u_m \$ e^{2m+4} \$$ has been read. The retrieval phase is a straightforward generalization of the one-dimensional case. □

Corollary 9. *Let $k, d \in \mathbb{N}_+$ be constants, then $\mathscr{L}_{rt}(IA_k^d) \subset \mathscr{L}_{rt}(IA_{k+1}^d)$.*

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

Fig. 2. Double hierarchy of fast IAs with restricted inter-cell communication.

4 Relations with Restricted Cellular Automata

In this section we consider one-dimensional devices in order to compare their computational capacity with communication restricted cellular automata. A *two-way cellular automaton with k-bit restricted inter-cell communication* (CA_k) 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. I.e., 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. Real time is defined to be n time steps. 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. The relations between these devices and iterative arrays in general are depicted in the left part of Figure 3. Now we turn to explore the relations for restricted devices.

Theorem 10. *For all $k \in \mathbb{N}_+$, there is a regular language which is not accepted by any real-time k-bit CA.*

Proof. Let $L_k = \{xvx \mid v \in \{a\}^* \text{ and } x \in \{a_0, \ldots, a_{2^{2k}}\}\}$ be the regular witness language. Assume contrarily, L_k is accepted by some real-time CA_k with state set S, bit functions $b_1, b_2 : S \to \{0,1\}^k$ giving the bits communicated to the left and to the right, and local transition function $\delta : \{0,1\}^k \times S \times \{0,1\}^k \to S$.

First we partition the input states $\{a_0, \ldots, a_{2^{2k}}\}$ according to b_1, i.e., two states s_1 and s_2 are in the same class if and only if $b_1(s_1) = b_1(s_2)$. Since there are $2^{2k}+1$ input states and the range of b_1 has 2^k elements, there is at least one class S_1 with at least 2^k+1 states. Next, S_1 is partitioned according to $b_1(\delta(b_2(a), s, b_1(\#)))$. Therefore, there is at least one subclass of S_1 that has at least two states, say a_i and a_j.

For an accepting computation on input $a_i a^n a_i$, for some $n \in \mathbb{N}_+$, we consider the relevant states of the cells $n-1$, n, $n+1$ at time steps 0, 1, 2. In particular, $c_0(n-1) = a$, $c_0(n) = a_i$, $c_0(n+1) = \#$, $c_1(n-1) = a'$, $c_1(n) = a'_i$, $c_2(n-1) = a''$. Due to the real-time restriction, states $c_1(n+1)$, $c_2(n)$, and $c_2(n+1)$ cannot affect the overall computation result. Since a_i and a_j are in the same class S_1, for input $a_i a^n a_j$ we obtain $c_0(n-1) = a$, $c_0(n) = a_j$, $c_0(n+1) = \#$, $c_1(n-1) = a'$, $c_1(n) = a'_j$. Since a_i and a_j are in the same subclass we obtain $c_2(n-1) = a''$. Therefore, input $a_i a^n a_j$ not belonging to L_k would be accepted. □

It is not hard to see that language L_k is accepted by a real-time CA_{k+1} as well as by a CA_k in time $n+1$. So, we obtain a strict bit hierarchy for two-way real-time cellular automata.

Theorem 11. *Let $k \in \mathbb{N}_+$ be a constant, then $\mathscr{L}_{rt}(CA_k) \subset \mathscr{L}_{rt}(CA_{k+1})$.*

Moreover, by modification of the witness language, i.e., by increasing the underlying alphabet, we obtain a very dense strict time hierarchy. That is, if we allow just one more time step, we obtain a strictly more powerful device.

Theorem 12. *Let $k \in \mathbb{N}_+$, $r \in \mathbb{N}$ be constants, then*

$$\mathscr{L}_{rt+r}(CA_k) \subset \mathscr{L}_{rt+r+1}(CA_k).$$

Since, trivially, any regular language is accepted by some real-time IA_1, the next theorem completes the incomparability results.

Theorem 13. *Let $k \in \mathbb{N}_+$ be a constant. There is a language belonging to the difference $\mathscr{L}_{rt}(OCA_1) \setminus \mathscr{L}_{lt}(IA_k)$.*

Proof. First we give the sketch of a construction of a one-bit real-time OCA that accepts the witness language $L_k = \{u_1 \cdots u_m e^x v \mid m \in \mathbb{N}_+ \text{ and } u_i \in \{a_0, \ldots, a_{2^k-1}\},\ 1 \leq i \leq m,\ \text{and } v \in \{e, a_0, \ldots, a_{2^k-1}\}^* \text{ and } x \text{ is greater than or equal to the number represented by the } 2^k\text{-ary interpretation of } u_1 \cdots u_m\}$.

Initially, all non-boundary states send bit 1 to the left. This identifies the rightmost cell uniquely. Next all cells with input e send a 1 and all cells in a state u_i send a 0. This identifies cells in state u_i with an e-neighbor to the right, and vice versa. Now all cells e with right neighbor u_i or in boundary state send a 0-signal to the left. All other cells e send bits 1 to the left until they receive a 0-signal from the right. The cells in states u_i form a 2^k-ary counter. The cells in state u_m with e-neighbor start to decrease the counter by one in every time step until they receive a 0-signal. A counter cell accepts when it generates the first carryover to the left.

In order to show that L_{k+1} is not accepted by any IA_k we adapt the proof of Theorem 8, and obtain $N_\ell(m, L_{k+1}) > p \cdot 2^{k \cdot m}$ induced equivalence classes, and $N_\ell(m, L_{k+1}) \leq p \cdot 2^{k \cdot m}$ distinguished equivalence classes. □

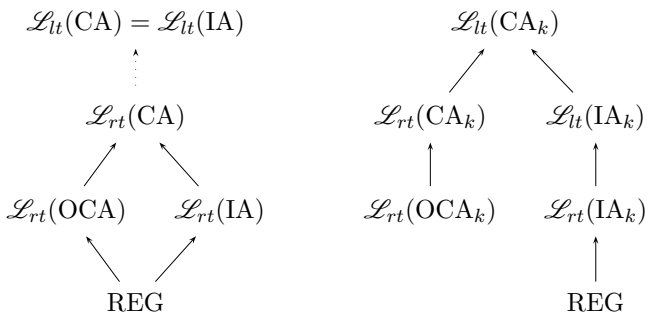

Fig. 3. Relations between unrestricted and restricted language families, respectively. Solid lines are strict inclusions, dotted lines are inclusions. Families which are not connected by any path are incomparable.

Finally, we show the proper inclusions between language families that are related by inclusions for structural reasons.

Theorem 14. *Let $k \in \mathbb{N}_+$ 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 [11] languages. Therefore, it suffices to show that the non-regular language $L = \{a^{2^x+2x} \mid x \in \mathbb{N}_+\}$ belongs to $\mathscr{L}_{rt}(\mathrm{CA}_1)$.

A corresponding CA_1 works as follows. It sets up 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 $x \in \mathbb{N}$. In particular, at time steps $2^x - 1$ all counter cells store bit 1. Subsequently, it

takes $x + 1$ time steps until the carryovers 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 to be 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, i.e., 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 input length is $2^x + x + x$. □

For the sake of completeness, the following theorem is presented without proof.

Theorem 15. *Let $k \in \mathbb{N}_+$ be a constant, then $\mathscr{L}_{lt}(IA_k) \subset \mathscr{L}_{lt}(CA_k)$.*

References

1. Buchholz T, Klein A, Kutrib M (1999) Iterative arrays with a wee bit alternation. In: Fundamentals of Computation Theory 1999, LNCS 1684, pp 173–184
2. Buchholz T, Klein A, Kutrib M (2000) Iterative arrays with small time bounds. In: Mathematical Foundations of Computer Science 1998, LNCS 1893, pp 243–252
3. Buchholz T, Klein A, Kutrib M (1999) Iterative arrays with limited nondeterministic communication cell. In: Words, Languages and Combinatorics III, pp 73–87
4. Chang JH, Ibarra OH, Palis MA (1987) Parallel parsing on a one-way array of finite-state machines. IEEE Trans Comput C-36:64–75
5. Cole SN (1969) Real-time computation by n-dimensional iterative arrays of finite-state machines. IEEE Trans Comput C-18:349–365
6. Fischer PC (1965) Generation of primes by a one-dimensional real-time iterative array. J ACM 12:388–394
7. Ibarra OH, Palis MA (1985) Some results concerning linear iterative (systolic) arrays. J Parallel Distributed Comput 2:182–218
8. Ibarra OH, Palis MA (1988) Two-dimensional iterative arrays: Characterizations and applications. Theoret Comput Sci 57:47–86
9. Iwamoto C, Hatsuyama T, Morita K, Imai K (1999) On time-constructible functions in one-dimensional cellular automata. In: Fundamentals of Computation Theory 1999, LNCS 1684, pp 317–326
10. Malcher A (2004) On the descriptional complexity of iterative arrays. IEICE Transactions on Information and Systems E87-D:721–725
11. Seidel SR (1979) Language recognition and the synchronization of cellular automata. Technical Report 79-02, Department of Computer Science, University of Iowa, Iowa City
12. Smith III AR (1972) Real-time language recognition by one-dimensional cellular automata. J Comput System Sci 6:233–253
13. Terrier V (1995) On real time one-way cellular array. Theoret Comput Sci 141:331–335
14. Umeo H (2001) Linear-time recognition of connectivity of binary images on 1-bit inter-cell communication cellular automaton. Parallel Comput 27:587–599

15. Umeo H, Kamikawa N (2002) A design of real-time non-regular sequence generation algorithms and their implementations on cellular automata with 1-bit inter-cell communications. Fund Inform 52:257–275
16. Umeo H, Kamikawa N (2003) Real-time generation of primes by a 1-bit-communication cellular automaton. Fund Inform 58:421–435
17. Worsch T (2000) Linear time language recognition on cellular automata with restricted communication. In: Latin 2000: Theoretical Informatics, LNCS 1776, pp 417–426