

# ITERATIVE ARRAYS WITH LIMITED NONDETERMINISTIC COMMUNICATION CELL

T. BUCHHOLZ, A. KLEIN AND M. KUTRIB

*Institute of Informatics, University of Giessen*

*Arndtstr. 2, D-35392 Giessen, Germany*

*E-mail: kutrib@informatik.uni-giessen.de*

An iterative array is a line of interconnected interacting finite automata. One distinguished automaton, the communication cell, is connected to the outside world and fetches the input serially symbol by symbol. Sometimes in the literature this model is referred to as cellular automaton with sequential input mode. We are investigating iterative arrays with a nondeterministic communication cell. All the other cells are deterministic ones. The number of nondeterministic state transitions is regarded as a limited resource which depends on the length of the input.

It is shown that the limit can be reduced by a constant factor without affecting the language accepting capabilities, but for sublogarithmic limits there exists an infinite hierarchy of properly included real-time language families. Finally we prove several closure properties of these families.

## 1 Introduction

Devices of interconnected parallel acting automata have extensively been investigated from a language theoretic point of view. The specification of such a system includes the type and specification of the single automata, the interconnection scheme (which sometimes implies a dimension to the system), a local and/or global transition function and the input and output modes. One-dimensional devices with nearest neighbor connections whose cells are deterministic finite automata are commonly called iterative arrays (IA) if the input mode is sequential to a distinguished communication cell.

Especially for practical reasons and for the design of systolic algorithms a sequential input mode is more natural than the parallel input mode of so-called cellular automata. Various other types of acceptors have been investigated under this aspect (e.g. the iterative tree acceptors in [8]).

In connection with formal language recognition IAs have been introduced in [7] where it was shown that the language families accepted by real-time IAs form a Boolean algebra not closed under concatenation and reversal. Moreover, there exists a context-free language that cannot be accepted by any  $d$ -dimensional IA in real-time. On the other hand, in [6] it is shown that for every context-free grammar a 2-dimensional linear-time IA parser exists. In [10] a real-time acceptor for prime numbers has been constructed. Pattern

manipulation is the main aspect in [1]. A characterization of various types of IAs by restricted Turing machines and several results, especially speed-up theorems, are given in [13, 14, 15].

Various generalizations of IAs have been considered. In [20] IAs are studied in which all the finite automata are additionally connected to the communication cell. Several more results concerning formal languages can be found e.g. in [21, 22, 23].

In some cases fully nondeterministic arrays have been studied, but up to now it is not known how the amount of nondeterminism influences the capabilities of the model. In terms of Turing machines bounded nondeterminism has been introduced in [11]. Further results concerning cellular automata, Turing machines, pushdown automata and finite automata can be found e.g. in [3, 5, 16, 17, 18, 19].

Here we introduce IAs with limited nondeterminism. We restrict the ability to perform nondeterministic transformations to the communication cell, all the other automata are deterministic ones. Moreover, we limit the number of allowed nondeterministic transitions dependent on the length of the input.

The paper is organized as follows. In section 2 we define the basic notions and the model in question. Section 3 is devoted to the possibility to reduce the number of nondeterministic transitions by a constant factor. In section 4 by varying the amount of allowed nondeterminism we prove an infinite hierarchy of properly included language families. Due to the results in section 3 we need sublogarithmic limits for the number of nondeterministic transitions in order to obtain the hierarchy. Finally, in section 5 several closure properties of the real-time acceptors with such limits are shown.

## 2 Model and Notions

We denote the rational numbers by  $\mathbb{Q}$ , the integers by  $\mathbb{Z}$ , the positive integers  $\{1, 2, \dots\}$  by  $\mathbb{N}$ , the set  $\mathbb{N} \cup \{0\}$  by  $\mathbb{N}_0$  and the powerset of a set  $S$  by  $2^S$ . The empty word is denoted by  $\varepsilon$  and the reversal of a word  $w$  by  $w^R$ . We use  $\subseteq$  for inclusions and  $\subset$  if the inclusion is strict. For a function  $f$  we denote its  $i$ -fold composition by  $f^{[i]}$ ,  $i \in \mathbb{N}$ , and define the set of mappings that grow strictly less than  $f$  by  $o(f) = \{g : \mathbb{N}_0 \rightarrow \mathbb{N} \mid \lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = 0\}$ . The set  $\Omega(f)$  is defined according to  $\{g : \mathbb{N}_0 \rightarrow \mathbb{N} \mid \liminf_{n \rightarrow \infty} \frac{g(n)}{f(n)} > 0\}$ . The identity function  $n \mapsto n$  is denoted by *id*.

An iterative array with nondeterministic communication cell is an infinite linear array of finite automata, sometimes called cells, where each of them is

connected to its both nearest neighbors (one to the left and one to the right). For convenience we identify the cells by integers. 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 two 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 both 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). The finite automata work synchronously at discrete time steps. More formally:

**Definition 1** An iterative array with nondeterministic communication cell (G-IA) is a system  $(S, \delta, \delta_{nd}, s_0, \#, A, F)$ , where

1.  $S$  is the finite, nonempty set of 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.  $\# \notin A$  is the end-of-input symbol,
6.  $\delta : S^3 \rightarrow S$  is the deterministic local transition function for non-communication cells satisfying  $\delta(s_0, s_0, s_0) = s_0$ ,
7.  $\delta_{nd} : S^3 \times (A \cup \{\#\}) \rightarrow 2^S$  is the nondeterministic local transition function for the communication cell satisfying  $\forall s_1, s_2, s_3 \in S, a \in A \cup \{\#\} : \delta_{nd}(s_1, s_2, s_3, a) \neq \emptyset$ .

Let  $\mathcal{M}$  be a G-IA (G for *guessing*). A configuration of  $\mathcal{M}$  at some time  $t \geq 0$  is a description of its global state which is actually a pair  $(w_t, c_t)$ , where  $w_t \in A^*$  is the remaining input sequence and  $c_t : \mathbb{Z} \rightarrow S$  is a mapping that maps the single cells to their current states. During its course of computation a G-IA steps nondeterministically through a sequence of configurations. The configuration  $(w_0, c_0)$  at time 0 is defined by the input word  $w_0$  and the mapping  $c_0(i) = s_0, i \in \mathbb{Z}$ , while subsequent configurations are chosen according to the global transition function  $\Delta_{nd}$ :

Let  $(w_t, c_t), t \geq 0$ , be a configuration then the possible successor configurations  $(w_{t+1}, c_{t+1})$  are as follows:

$$(w_{t+1}, c_{t+1}) \in \Delta_{nd}((w_t, c_t)) \iff \begin{cases} c_{t+1}(i) = \delta(c_t(i-1), c_t(i), c_t(i+1)) \\ c_{t+1}(0) \in \delta_{nd}(c_t(-1), c_t(0), c_t(+1), a) \end{cases}$$

where  $i \in \mathbb{Z} \setminus \{0\}$ , and  $a = \#$ ,  $w_{t+1} = \varepsilon$  if  $w_t = \varepsilon$ , 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  $\Delta_{nd}$  is induced by  $\delta$

and  $\delta_{nd}$ . The  $i$ -fold composition of  $\Delta_{nd}$  is defined as follows:

$$\Delta_{nd}^{[0]}((w, c)) = \{(w, c)\}, \quad \Delta_{nd}^{[i+1]}((w, c)) = \bigcup_{(w', c') \in \Delta_{nd}^{[i]}((w, c))} \Delta_{nd}((w', c'))$$

If the state set is a Cartesian product of some smaller sets  $S = S_1 \times \dots \times S_k$ , we will use the notion *register* for the single parts of a state. The concatenation of one of the registers of all cells respectively forms a *track*.

A G-IA is deterministic if  $\delta_{nd}(s_1, s_2, s_3, a)$  is a singleton for all states  $s_1, s_2, s_3 \in S$  and all input symbols  $a \in A \cup \{\#\}$ . Deterministic iterative arrays are denoted by IA.

**Definition 2** Let  $\mathcal{M} = (S, \delta, \delta_{nd}, s_0, \#, A, F)$  be a G-IA.

1. A word  $w \in A^*$  is accepted by  $\mathcal{M}$  if there exists a time step  $i \in \mathbb{N}$  such that  $c_i(0) \in F$  for some  $(w_i, c_i) \in \Delta_{nd}^{[i]}((w, c_0))$ .
2.  $L(\mathcal{M}) = \{w \in A^* \mid w \text{ is accepted by } \mathcal{M}\}$  is the language accepted by  $\mathcal{M}$ .
3. Let  $t : \mathbb{N}_0 \rightarrow \mathbb{N}$ ,  $t(n) \geq n + 1$ , be a mapping and  $i_w$  be the minimal time step at which  $\mathcal{M}$  accepts a  $w \in L(\mathcal{M})$  in some computation. If all  $w \in L(\mathcal{M})$  are accepted within  $i_w \leq t(|w|)$  time steps, then  $L$  is said to be of time complexity  $t$ .

The family of all languages which can be accepted by a G-IA with time complexity  $t$  is denoted by  $\mathcal{L}_t(\text{G-IA})$ . In the sequel we will use a corresponding notion for other types of acceptors. If  $t$  equals the function  $n + 1$  acceptance is said to be in *real-time* and we write  $\mathcal{L}_{rt}(\text{G-IA})$ . The *linear-time* languages  $\mathcal{L}_{lt}(\text{G-IA})$  are defined according to  $\mathcal{L}_{lt}(\text{G-IA}) = \bigcup_{k \in \mathbb{Q}, k > 1} \mathcal{L}_{k \cdot n}(\text{G-IA})$ .

There is a natural way to restrict the nondeterminism of the arrays. One can limit the number of allowed nondeterministic state transitions of the communication cell. For this reason a deterministic local transformation  $\delta_d : S^3 \times (A \cup \{\#\}) \rightarrow S$  for the communication cell is provided and the global transformation induced by  $\delta$  and  $\delta_d$  is denoted by  $\Delta_d$ . Let  $g : \mathbb{N}_0 \rightarrow \mathbb{N}_0$  be a mapping that gives the number of allowed nondeterministic transitions dependent on the length of the input. The resulting system  $(S, \delta, \delta_{nd}, \delta_d, s_0, \#, A, F)$  is a  $g$ G-IA ( $g$  guess IA) if starting with the initial configuration  $(w_0, c_0)$  the possible configurations at some time  $i$  are given by the global transformations

as follows:

$$\begin{aligned} & \{(w_0, c_0)\} && \text{if } i = 0 \\ & \Delta_{nd}^{[i]}((w_0, c_0)) && \text{if } i \leq g(|w|) \\ & \bigcup_{(w', c') \in \Delta_{nd}^{[g(|w|)]}((w_0, c_0))} \Delta_d^{[i-g(|w|)]}((w', c')) && \text{otherwise} \end{aligned}$$

Observe that in this definition the nondeterministic transitions have to be applied before the deterministic ones. This is not a serious restriction since nondeterministic transitions for later time steps can be guessed and stored in advance (cf. second part of the proof of Theorem 3). Up to now we have  $g$  not required to be effective. Of course, for almost all applications we will have to do so but some of our general results can be developed without such requirement.

### 3 Guess Reduction

This section is devoted to the reduction of the number of nondeterministic transformations. In the sequel we will make extensively use of the ability of IAs to simulate a pushdown storage [8, 2] or a queue [4] on some track in real-time. The communication cell contains the symbol at the top of the stack or the queue. The left-to-right inclusion in the following theorem is not immediate since there might be computation paths of the  $kg$ G-IA that cannot appear for the  $g$ G-IA. Therefore, the  $kg$ G-IA must be able to verify whether or not its communication cell has performed  $g(n)$  nondeterministic transitions.

**Theorem 3** *Let  $g : \mathbb{N}_0 \rightarrow \mathbb{N}_0$  be a mapping and  $k \in \mathbb{N}$  be a constant. If  $t : \mathbb{N}_0 \rightarrow \mathbb{N}$ ,  $t(n) \geq n + 1$ , is a mapping such that  $t(n) \geq k \cdot g(n)$  for almost all  $n \in \mathbb{N}$  then*

$$\mathcal{L}_t(g\text{G-IA}) = \mathcal{L}_t(k \cdot g\text{G-IA})$$

**Proof.** The crucial point in proving the inclusion  $\mathcal{L}_t(g\text{G-IA}) \subseteq \mathcal{L}_t(kg\text{G-IA})$  is that a  $kg$ G-IA  $\mathcal{M}'$  which is designated to simulate a given  $g$ G-IA  $\mathcal{M}$  with the same time complexity must not simulate too many nondeterministic transitions of  $\mathcal{M}$ . Therefore, the communication cell of  $\mathcal{M}'$  is equipped with a pushdown storage. During its nondeterministic transitions  $\mathcal{M}'$  either can simulate a nondeterministic step of  $\mathcal{M}$  whereby  $k - 1$  specific symbols are pushed or can simulate a deterministic step of  $\mathcal{M}$  whereby one symbol is popped.

Once  $\mathcal{M}'$  decided to simulate a deterministic transition it has to do so for its remaining nondeterministic steps, whereby again one symbol is popped respectively. In order to accept the input  $\mathcal{M}'$  has to pop the last symbol from the stack exactly at time step  $k \cdot g(n)$  which is its last nondeterministic one.

Let  $m$  be the number of time steps at which symbols are pushed. Then we have  $m \cdot (k - 1) = k \cdot g(n) - m \implies m = g(n)$ .

To see the other inclusion  $\mathcal{L}_t(kg\text{-IA}) \subseteq \mathcal{L}_t(g\text{-IA})$  we use again a push-down storage. The communication cell of a  $g\text{-IA}$   $\mathcal{M}'$  simulating a  $kg\text{-IA}$   $\mathcal{M}$  without any loss of time pushes  $k - 1$  nondeterministically determined functions  $d : S^3 \times (A \cup \{\#\}) \rightarrow S$  satisfying  $d(s_1, s_2, s_3, a) \in \delta_{nd}(s_1, s_2, s_3, a)$  (here  $\delta_{nd}$  denotes the nondeterministic transition function for the communication cell of  $\mathcal{M}$ ) during each of its nondeterministic transitions. Additionally, it simulates a nondeterministic transition of  $\mathcal{M}$ . During the first deterministic transitions such a function is popped and applied to the states of the communication cell and its neighbors and the current input symbol which yields the next state of the communication cell. Hence a nondeterministic transition in  $\mathcal{M}$  is simulated deterministically. Altogether  $\mathcal{M}'$  performs  $g(n) + (k - 1) \cdot g(n) = k \cdot g(n)$  nondeterministic transitions and accepts exactly the same language as  $\mathcal{M}$ .  $\square$

A constant number of nondeterministic transitions does not increase the power of IAs. The principle of the proof is to simulate all finitely many choices on different tracks.

**Theorem 4** *Let  $t : \mathbb{N}_0 \rightarrow \mathbb{N}$ ,  $t(n) \geq n + 1$  be a mapping. If  $k \in \mathbb{N}$  is a constant then*

$$\mathcal{L}_t(k\text{G-IA}) = \mathcal{L}_t(\text{IA})$$

The next corollary extends the previous results.

**Corollary 5** *Let  $g : \mathbb{N}_0 \rightarrow \mathbb{N}_0$  be a mapping and  $q \in \mathbb{Q}$ ,  $0 < q \leq 1$ , be a rational number such that  $g(n) = \lfloor qn \rfloor$  for almost all  $n \in \mathbb{N}$ , then*

$$\mathcal{L}_{rt}(g\text{G-IA}) = \mathcal{L}_{rt}(\text{idG-IA})$$

#### 4 Nondeterministic Hierarchy

**Definition 6** *Let  $L \subseteq A^*$  be a language over an alphabet  $A$  and  $l \in \mathbb{N}_0$  be a constant.*

1. *Two words  $w$  and  $w'$  are  $l$ -equivalent with respect to  $L$  if*

$$ww_l \in L \iff w'w_l \in L \text{ for all } w_l \in A^l$$

2.  $N(n, l, L)$  denotes the number of  $l$ -equivalence classes of words of length  $n$  with respect to  $L$  (i.e.  $|ww_l| = n$ ).

**Lemma 7** Let  $g : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ ,  $g(n) \leq n + 1$ , be a mapping. If  $L \in \mathcal{L}_{rt}(gG\text{-IA})$  then there exist constants  $p, q \in \mathbb{N}$  such that

$$N(n, l, L) \leq p^{l \cdot q^{g(n)}}$$

**Proof.** Let  $\mathcal{M} = (S, \delta, \delta_{nd}, \delta_d, s_0, \#, A, F)$  be a real-time  $gG\text{-IA}$  which accepts  $L$ . We define

$$q = \max \{ |\delta_{nd}(s_1, s_2, s_3, a)| \mid s_1, s_2, s_3 \in S \wedge a \in A \}$$

In order to determine an upper bound to the number of  $l$ -equivalence classes we consider the possible configurations of  $\mathcal{M}$  after reading all but  $l$  input symbols. The remaining computation depends on the last  $l$  input symbols and the states of the cells  $-l-1, \dots, 0, \dots, l+1$ . For the  $2l+3$  states there are  $|S|^{2l+3}$  different possibilities. Let  $p = |S|^5$  then due to  $|S|^{2l+3} = |S|^{2l} \cdot |S|^3 = (|S|^2)^l \cdot |S|^3 \leq (|S|^2)^l \cdot (|S|^3)^l = (|S|^2 \cdot |S|^3)^l \leq p^l$  we have at most  $p^l$  different possibilities for at most  $q^{g(n)}$  different computation paths. Since the number of equivalence classes is not affected by the last  $l$  input symbols in total there are at most  $(p^l)^{q^{g(n)}} = p^{l \cdot q^{g(n)}}$  classes.  $\square$

The following result does not follow for structural reasons since there might be accepting computation paths of the  $fG\text{-IA}$  that cannot appear for the  $gG\text{-IA}$ . Therefore, the  $fG\text{-IA}$  must be able to verify whether or not its communication cell has performed  $g(n)$  nondeterministic transitions.

**Theorem 8** Let  $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ ,  $f(n) \leq \frac{n}{2}$ , and  $g : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ ,  $g(n) \leq f(n)$ , be two increasing mappings such that  $\forall m, n \in \mathbb{N} : f(m) = f(n) \implies g(m) = g(n)$ . If  $L_v = \{ a^{g(n)} b^{f(n)-g(n)} \mid n \in \mathbb{N} \}$  belongs to the family  $\mathcal{L}_{tt}(\text{IA})$  then

$$\mathcal{L}_{rt}(gG\text{-IA}) \subseteq \mathcal{L}_{rt}(fG\text{-IA})$$

**Proof.** Let  $\mathcal{M}$  be a real-time  $gG\text{-IA}$  that accepts the language  $L$ . A real-time  $fG\text{-IA}$   $\mathcal{M}'$  which simulates  $\mathcal{M}$  works as follows.

Since  $f \geq g$   $\mathcal{M}'$  can guess the time step  $g(n)$  and therefore simulate  $\mathcal{M}$  directly. Additionally,  $\mathcal{M}'$  has to verify that its guess was correct. Otherwise the computation must not be accepting.

It is known that deterministic linear-time IAs can be sped-up to  $(2 \cdot id)$ -time [14]. Thus,  $L_v$  belongs to  $\mathcal{L}_{2id}(\text{IA})$ . Now  $\mathcal{M}'$  simulates such an acceptor  $\mathcal{M}''$  on an additional track. During the first  $g(n)$  time steps  $\mathcal{M}'$  simulates  $\mathcal{M}''$  under the assumption that  $\mathcal{M}''$  fetches input symbols  $a$ . From the guessed

time step  $g(n)$  up to the last nondeterministic step  $f(n)$   $\mathcal{M}'$  simulates  $\mathcal{M}''$  under the assumption that  $\mathcal{M}''$  fetches input symbols  $b$ , respectively, and during the last  $n - f(n)$  time steps  $\mathcal{M}'$  simulates  $\mathcal{M}''$  without input.

Due to the condition  $f(n) \leq \frac{n}{2}$  altogether  $\mathcal{M}'$  simulates at least  $2 \cdot id$  time steps of  $\mathcal{M}''$ . If  $\mathcal{M}'$  guessed  $g(n)$  correctly it simulates  $\mathcal{M}''$  for the input  $a^{g(n)}b^{f(n)-g(n)}$  and, hence, an accepting computation. On the other hand, if  $\mathcal{M}'$  simulates an accepting computation then it guessed a time step  $t$  such that the input  $a^tb^{f(n)-t}$  belongs to  $L_v$ . It follows  $t \in \{g(m) \mid f(m) = f(n)\}$  and due to the assumption  $\forall m, n, \in \mathbb{N} : f(m) = f(n) \implies g(m) = g(n)$  it holds  $t = g(n)$ . Therefore,  $\mathcal{M}'$  can verify whether its guess was correct and, thus, accepts  $L$  in real-time.  $\square$

The following situation may clarify the necessity of the condition  $\forall m, n, \in \mathbb{N} : f(m) = f(n) \implies g(m) = g(n)$ . Let  $m < n$  and  $f(m) = f(n)$  and  $g(m) < g(n)$ . Since  $a^{g(n)}b^{f(n)-g(n)}$  belongs to  $L_v$  the word  $a^{g(n)}b^{f(m)-g(n)}$  does. Consequently, for an input of length  $m$  the word  $a^{g(n)}b^{f(m)-g(n)}$  would lead to an accepting computation but since  $g(m) < g(n)$  the time step  $g$  might be guessed wrong.

Now we are going to extend the previous result to a hierarchy of properly included language families.

**Theorem 9** *Let  $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$  and  $g : \mathbb{N}_0 \rightarrow \mathbb{N}_0$  be two mappings which meet the conditions of Theorem 8. If additionally  $f \in o(\log)$  and  $g \in o(f)$  then*

$$\mathcal{L}_{rt}(gG\text{-IA}) \subset \mathcal{L}_{rt}(fG\text{-IA})$$

**Proof.** We define a mapping  $h : \mathbb{N}_0 \rightarrow \mathbb{N}$  by  $h(n) = 2^{f(n)}$ .  $h$  is increasing since  $f$  is. Moreover, since  $f \in o(\log)$  for all  $k \in \mathbb{Q}$ ,  $k \geq 0$ , it holds  $\lim_{n \rightarrow \infty} \frac{h(n)}{n^k} = \lim_{n \rightarrow \infty} \frac{2^{f(n)}}{2^{\log(n) \cdot k}} = 0$  and therefore  $h \in o(n^k)$ . Especially for  $k = \frac{1}{2}$  it follows that the mapping  $m(n) = \max \{n' \in \mathbb{N}_0 \mid (h(n)+1) \cdot (n'+1) \leq n\}$  is unbounded, and for large  $n$  we obtain  $m(n) > h(n)$ . Now we define a language  $L$  that belongs to  $\mathcal{L}_{rt}(fG\text{-IA})$  but does not belong to  $\mathcal{L}_{rt}(gG\text{-IA})$ .

$$\begin{aligned} L = \{ & \$^r w_1 \$ w_2 \$ \cdots \$ w_j \# y \# \mid \exists n \in \mathbb{N} : j = h(n) \wedge w_i \in \{0, 1\}^{m(n)}, 1 \leq i \leq j, \\ & \wedge r = n - (h(n) + 1) \cdot (m(n) + 1) \\ & \wedge \exists 1 \leq i' \leq j : w_{i'} = y^R \} \end{aligned}$$

It follows that  $L$  is not empty (cf. Example 10). Assume now  $L \in \mathcal{L}_{rt}(gG\text{-IA})$ . Then by Lemma 7 there exist constants  $p, q \in \mathbb{N}$  such that  $N(n, m(n) + 1, L) \leq p^{(m(n)+1) \cdot q^{g(n)}}$ . Since  $g \in o(f)$  for all  $k \in \mathbb{Q}$ ,  $k \geq 0$ , it holds  $\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = \lim_{n \rightarrow \infty} \frac{k \cdot g(n)}{f(n)} = \lim_{n \rightarrow \infty} \frac{2^{k \cdot g(n)}}{2^{f(n)}} = 0$ . Thus, we obtain

$2^{k \cdot g} \in o(2^f) = o(h)$ . Therefore, for large  $n$  the number of equivalence classes is bounded as follows:

$$\begin{aligned} N(n, m(n) + 1, L) &\leq p^{(m(n)+1) \cdot q^{g(n)}} \leq p^{2 \cdot m(n) \cdot q^{g(n)}} \\ &= 2^{\log(p) \cdot 2 \cdot m(n) \cdot 2^{\log(q) \cdot g(n)}} \end{aligned}$$

Let  $k$  be  $\log(q)$ , then  $2^{\log(q) \cdot g(n)} = 2^{k \cdot g(n)} \in o(h)$ . Now we can find a constant  $n_0$  such that for all  $n \geq n_0$ :  $2 \cdot \log(p) \cdot 2^{k \cdot g(n)} < \frac{1}{4}h(n)$ .

It follows

$$2^{\log(p) \cdot 2 \cdot m(n) \cdot 2^{\log(q) \cdot g(n)}} < 2^{m(n) \cdot h(n) \cdot \frac{1}{4}}$$

On the other hand, let for all  $n \in \mathbb{N}$  and for every subset  $U = \{w_1, \dots, w_{h(n)}\}$  of  $\{0, 1\}^{m(n)}$  a word  $u$  be defined according to  $u = \$^r w_1 \$ \dots \$ w_{h(n)} \#$  where  $r = n - (h(n) + 1) \cdot (m(n) + 1)$ . Then for all  $y \in \{0, 1\}^{m(n)}$ :

$$y \in U \iff uy^R \# \in L$$

Since there exist at least  $2^{m(n)}$  different words  $w_i$  there are  $\binom{2^{m(n)}}{h(n)}$  different subsets  $U$ . For every pair  $U, V$  of subsets one can find a  $w_i$  belonging to  $U \setminus V$  or to  $V \setminus U$ . It follows  $uw_i^R \# \in L \iff vw_i^R \# \notin L$  and, hence,

$$\begin{aligned} N(n, m(n) + 1, L) &\geq \binom{2^{m(n)}}{h(n)} = \frac{2^{m(n)} \cdot (2^{m(n)} - 1) \cdot \dots \cdot (2^{m(n)} - h(n) + 1)}{h(n)!} \\ &\geq \frac{(2^{m(n)} - h(n))^{h(n)}}{h(n)^{h(n)}} \end{aligned}$$

From  $m(n) > h(n)$  for large  $n$  it follows  $2^{m(n)} - h(n) \geq 2^{m(n) \cdot \frac{1}{2}}$ . Thus

$$\begin{aligned} \frac{(2^{m(n)} - h(n))^{h(n)}}{h(n)^{h(n)}} &\geq \left( \frac{2^{m(n) \cdot \frac{1}{2}}}{h(n)} \right)^{h(n)} \geq \left( \frac{2^{m(n) \cdot \frac{1}{2}}}{m(n)} \right)^{h(n)} \\ &= \left( \frac{2^{m(n) \cdot \frac{1}{2}}}{2^{\log(m(n))}} \right)^{h(n)} = 2^{(m(n) \cdot \frac{1}{2} - \log(m(n))) \cdot h(n)} \\ &\geq 2^{m(n) \cdot h(n) \cdot \frac{1}{4}} \end{aligned}$$

From the contradiction we obtain  $L \notin \mathcal{L}_{rt}(gG\text{-IA})$ .

It remains to show  $L \in \mathcal{L}_{rt}(fG\text{-IA})$ . An  $fG\text{-IA}$   $\mathcal{M}$  which accepts  $L$  has to check whether  $j = h(n)$ , whether all the  $w_i$  are of the same length, whether  $r < h(n)$  (from which now follows that  $|w_i| = m(n)$ ), and whether there exists an  $i'$  such that  $w_{i'} = y^R$ . Accordingly  $\mathcal{M}$  performs four tasks in parallel.

For the first task  $\mathcal{M}$  simulates a stack and pushes a symbol 1 at every nondeterministic transformation. After the last nondeterministic transformation the pushed string is handled as a counter which is decremented every time step a new  $w_i$  appears in the input. The decrementation starts for  $w_2$ . The number of  $w_i$ s is accepted if the counter is 0 after reading the input because  $1^{f(n)}$  is the binary number  $2^{f(n)} - 1 = h(n) - 1$ .

For the second task  $\mathcal{M}$  uses two more stacks. The subword  $w_1$  is pushed onto one of them. When  $\mathcal{M}$  fetches  $w_2$  it pushes  $w_2$  to the second stack and pops  $w_1$  from the first stack whereby their lengths are compared symbol by symbol. This task is repeated up to  $w_j$ .

The third task uses another stack on which the first  $r$  symbols  $\$$  of the input are pushed. Subsequently for each subword  $w_i$  one of them is popped.

The last task is to find an  $i'$  such that  $w_{i'} = y^R$ . Here the nondeterminism is used. During the first  $f(n)$  nondeterministic steps a binary string is guessed bit by bit and pushed onto a stack. From time  $f(n)$  on it is handled as a counter which is decremented for every subword  $w_i$ . If it is 0 the next word is pushed onto another stack. It will be popped and compared symbol by symbol when the word  $y$  appears in the input. Thus, the  $i'$  is guessed during the nondeterministic transformations.  $\square$

At first glance the witness  $L$  for the proper inclusion seems to be rather complicated. But here is a natural example for a hierarchy:

**Example 10** Let  $i > 1$  be a constant and  $f(n) = \log^{[i]}(n)$  and  $g(n) = \log^{[i+1]}(n)$ . Then by Theorem 9 we have  $\mathcal{L}_{rt}(g\text{G-IA}) \subset \mathcal{L}_{rt}(f\text{G-IA})$ . Since  $\mathcal{L}_{lt}(\text{IA})$  is identical to the linear-time cellular automata languages [22] and  $\{a^n b^{2^n - n} \mid n \in \mathbb{N}\}$  is acceptable by such devices  $\{a^{g(n)} b^{f(n) - g(n)} \mid n \in \mathbb{N}\} \in \mathcal{L}_{lt}(\text{IA})$  holds. Moreover, from  $g \in \log(f)$  follows  $\forall m, n \in \mathbb{N} : f(m) = f(n) \implies g(m) = g(n)$ . Thus, the conditions of Theorem 8 are met. Trivially,  $g$  is of order  $o(f)$ . E.g. for  $i = 2$  we obtain  $m(4) = 0$ ,  $m(8) = 1$ ,  $m(16) = 2$ ,  $m(32) = 4$ , and  $\$01\$11\$10\$00\text{\textcircled{c}}11\text{\textcircled{c}} \in L$ .

## 5 Closure Properties

Besides that closure properties are interesting of their own they are a powerful tool for relating families of languages. Our first results in this sections deal with Boolean operations.

**Lemma 11** Let  $g : \mathbb{N}_0 \rightarrow \mathbb{N}_0$  and  $t : \mathbb{N}_0 \rightarrow \mathbb{N}$ ,  $t(n) \geq n + 1$ , be two mappings. Then the family  $\mathcal{L}_t(g\text{G-IA})$  is closed under union and intersection and trivially contains  $\mathcal{L}_t(\text{IA})$ .

**Proof.** Using the same two channel technique of [9] and [22] the assertion can easily be seen. Each cell consists of two registers in which acceptors for both languages are simulated in parallel.  $\square$

Now we turn to more language specific closure properties. For some functions  $g$  the families  $\mathcal{L}_{rt}(g\text{G-IA})$  are closed under concatenation and for some others are not. At first we consider the closure under marked concatenation.

**Lemma 12** *Let  $g : \mathbb{N}_0 \rightarrow \mathbb{N}_0$  be an increasing mapping such that the language  $\{\mathbf{a}^{g(m)}\mathbf{b}^{m-g(m)} \mid m \in \mathbb{N}\}$  belongs to  $\mathcal{L}_{rt}(\text{IA})$ . Then the family  $\mathcal{L}_{rt}(g\text{G-IA})$  is closed under marked concatenation.*

**Proof.** Let  $L_1$  resp.  $L_2$  be formal languages over the alphabets  $A_1$  resp.  $A_2$  which are acceptable in real-time by the  $g\text{G-IAs}$   $\mathcal{M}_1$  resp.  $\mathcal{M}_2$ . Let  $L$  denote the marked concatenation of  $L_1$  and  $L_2$ : i.e.,

$$L = \{w_1cw_2 \mid w_1 \in L_1 \text{ and } w_2 \in L_2\}$$

where  $c \notin A_1 \cup A_2$  is a *marking* symbol.

A  $g\text{G-IA}$   $\mathcal{M}$  that accepts  $L$  in real-time works as follows.  $A_1^*cA_2^*$  is a regular language and, therefore, belongs trivially to  $\mathcal{L}_{rt}(g\text{G-IA})$ . Since  $\mathcal{L}_{rt}(g\text{G-IA})$  is closed under intersection (cf. Lemma 11) it is sufficient to consider inputs of the form  $A_1^*cA_2^*$  only. Let  $w = w_1cw_2$  with  $w_1 \in A_1^*$ ,  $w_2 \in A_2^*$ , and  $n_1 = |w_1|$ ,  $n_2 = |w_2|$ .

Now the idea is as follows: On input  $w$  the array  $\mathcal{M}$  simulates the behavior of  $\mathcal{M}_1$  (on input  $w_1$ ) until reading the marking symbol  $c$  and subsequently the behavior of  $\mathcal{M}_2$  (on input  $w_2$ ).  $\mathcal{M}$  accepts  $w$  iff both simulations are accepting.

The simulation of  $\mathcal{M}_1$  can be performed directly since  $g$  is monotonically increasing and therefore  $g(n) \geq g(n_1)$ . But the time step  $g(n_1)$  has to be guessed and verified. In order to perform this task an acceptor for the language  $L' = \{\mathbf{a}^{g(m)}\mathbf{b}^{m-g(m)} \mid m \in \mathbb{N}\}$  is simulated on an additional track in parallel. Thereby an input symbol  $\mathbf{a}$  is assumed for each nondeterministic step (up to the guessed time  $g(n_1)$ ) and an input symbol  $\mathbf{b}$  for each deterministic step (up to the end of simulation at time  $n_1$ ).

So the number  $x$  resp.  $y$  of simulated nondeterministic resp. deterministic transitions corresponds to a word  $\mathbf{a}^x\mathbf{b}^y$  belonging to  $L'$  iff there exists an  $m \in \mathbb{N}$  such that  $x = g(m)$  and  $y = m - g(m)$ . Thus, iff  $n_1 = x + y = g(m) + m - g(m) = m$ .

The simulation of  $\mathcal{M}_2$  is performed similarly. However, a problem would arise with the nondeterministic transitions if  $g(n) < n_1 + 1 + g(n_2)$ . Therefore, during its nondeterministic transitions  $\mathcal{M}$  uses a queue into which it

pipes nondeterministically chosen local transition functions corresponding to a possible nondeterministic transition of  $\mathcal{M}_2$  (cf. the proof of Theorem 3). During the simulation of the nondeterministic transitions of  $\mathcal{M}_2$  these functions are successively extracted from the queue and applied to the communication cell.  $\square$

The assertions of the lemma can essentially be weakened. Let  $h$  be a homomorphism such that  $h(x) = \mathbf{a}$  for  $x \neq \mathbf{b}$  and  $h(\mathbf{b}) = \mathbf{b}$ . Then instead of requiring  $L = \{\mathbf{a}^{g(m)}\mathbf{b}^{m-g(m)} \mid m \in \mathbb{N}\}$  to be acceptable in real-time by some iterative array it is sufficient to require that some language  $L'$  with  $h(L') = L$  belongs to  $\mathcal{L}_{rt}(\text{IA})$ .

By  $\mathcal{G}$  we denote the set of functions  $g : \mathbb{N} \rightarrow \mathbb{N}_0$ ,  $g(n) \leq n$ , such that there exists a language  $L' \in \mathcal{L}_{rt}(\text{IA})$  whose image under  $h$  is  $\{\mathbf{a}^{g(m)}\mathbf{b}^{m-g(m)} \mid m \in \mathbb{N}\}$ .

So in fact any family  $\mathcal{L}_{rt}(g\text{G-IA})$  where  $g \in \mathcal{G}$  is closed under marked concatenation.

The usage of a marking symbol can be omitted if the limiting function  $g$  allows a  $g\text{G-IA}$  to determine a possible concatenation point by its own (for instance nondeterministically by using a  $b$ -ary counter). Hence, we obtain the following corollary.

**Corollary 13** *Let  $g : \mathbb{N}_0 \rightarrow \mathbb{N}_0$  be a mapping with  $g \in \Omega(\log)$ . If  $\mathcal{L}_{rt}(g\text{G-IA})$  is closed under marked concatenation then it is closed under concatenation.*

On the other hand, there exist functions  $g$  for which  $g\text{G-IA}$  is not closed under concatenation. The proof follows essentially an idea presented in [7] to show that the family  $\mathcal{L}_{rt}(\text{IA})$  is not closed under concatenation.

**Theorem 14** *Let  $g : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ ,  $g \in o(\log \log)$ , be a mapping. Then  $\mathcal{L}_{rt}(g\text{G-IA})$  is not closed under concatenation.*

**Proof.** Let  $A$  be the alphabet consisting of the four symbols 0, 1,  $\mathbf{a}$ , and  $\mathbf{b}$ . Further let  $L_1 = A^*$  and denote by  $L_2$  the language of palindromes over  $A$ , i.e. the set of all words  $w$  over  $A$  which are identical to their reversals  $w^R$ . As it has been shown in [7]  $L_1$  as well as  $L_2$  are belonging to  $\mathcal{L}_{rt}(\text{IA})$  and thus to  $\mathcal{L}_{rt}(g\text{G-IA})$ .

Consider now the concatenation  $L = L_1L_2$  and assume contrarily that  $L$  belongs to  $\mathcal{L}_{rt}(g\text{G-IA})$ , too. Then let  $W_n = \{0w1 \mid w \in \{\mathbf{a}, \mathbf{b}\}^n\}$  for  $n \in \mathbb{N}$  and define for each subset  $U = \{w_1, \dots, w_k\}$  of  $W_n$  the word  $u$  as

$$u = \begin{cases} \varepsilon & \text{if } U = \emptyset \\ u_k & \text{otherwise} \end{cases}$$

where the  $u_1, \dots, u_k$  are recursively defined by

$$u_0 = \varepsilon, \quad u_{i+1} = w_{i+1}^R w_i^R w_i, \quad 1 \leq i \leq m-1.$$

One easily sees that  $|u| = n(2^k - 1)$  and that for all  $w \in W_n$  it holds  $w \in U$  iff  $uw \in L$ . Therefore (choosing  $k = n$ ) there are especially at least  $\binom{2^n}{n}$  different  $n$ -equivalence classes with respect to  $L$  in the set of words of length  $n2^n$  over  $A$ . Hence using the assumption on  $g$  we can work out a contradiction to Lemma 7 for a sufficiently large  $n$ . So  $L$  is not acceptable in real-time by a  $gG$ -IA, i.e.  $\mathcal{L}_{rt}(gG\text{-IA})$  is not closed under concatenation.  $\square$

Note that one can additionally show that for  $g \in o(\log \log)$  the corresponding family  $\mathcal{L}_{rt}(gG\text{-IA})$  is not closed under marked iteration although it might be closed under marked concatenation.

**Theorem 15** *Let  $g : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ ,  $g \in o(\log)$ , be a mapping. Then the family  $\mathcal{L}_{rt}(gG\text{-IA})$  is not closed under reversal.*

**Proof.** Consider the language  $L$  consisting of all marked concatenations of binary sequences of equal length where the first sequence occurs at least twice, i.e

$$L = \{w_1\$ \dots w_k\$ \mid k \geq 2 \wedge \exists m \in \mathbb{N} : w_i \in \{0, 1\}^m, 1 \leq i \leq k, \\ \wedge \exists 2 \leq j \leq k : w_1 = w_j\}.$$

We are going to show that  $L$  belongs to  $\mathcal{L}_{rt}(\text{IA}) \subseteq \mathcal{L}_{rt}(gG\text{-IA})$ , but  $L^R \notin \mathcal{L}_{rt}(gG\text{-IA})$ .

An iterative array  $\mathcal{M}$  that accepts  $L$  in real-time works as follows. The communication cell is equipped with a queue through which symbols can be piped in first-in-first-out manner. At the beginning of the computation  $\mathcal{M}$  stores its input symbols to the queue until the first symbol  $\$$  appears.

Afterwards at every time step one symbol is extracted from the queue and compared to the current input symbol. At the same time step it is stored in the queue again. Thus, the symbols of  $w_1$  circulate through the queue and  $w_1$  is compared with all the  $w_i$ ,  $2 \leq i \leq k$ , serially.

It remains to show that  $L^R$  does not belong to  $\mathcal{L}_{rt}(gG\text{-IA})$ . Let us assume that  $L^R$  is acceptable by some  $gG$ -IA in real-time. Let us consider the equivalence classes  $N((m+1)^2, (m+1), L^R)$ . For every pair of different subsets  $\{x_1, \dots, x_m\}$  and  $\{y_1, \dots, y_m\}$  of the set  $\{0, 1\}^m$  there are words  $\$x_1 \dots \$x_m$  and  $\$y_1 \dots \$y_m$  which belong to different such  $(m+1)$ -equivalence classes. W.l.o.g. let  $x_1 \notin \{y_1, \dots, y_m\}$ . Then  $\$x_1 \dots \$x_m \$x_1$  belongs to  $L^R$

whereas  $y_1 \cdots y_m x_1$  does not. Hence, there are at least  $\binom{2^m}{m}$  such  $(m+1)$ -equivalence classes. Since  $f \in o(\log)$  we obtain a contradiction to Lemma 7 for a sufficiently large  $m$  which concludes the proof.  $\square$

The last two results deal with the closure under homomorphisms.

**Theorem 16** *Let  $g : \mathbb{N}_0 \rightarrow \mathbb{N}_0$  be a mapping. If  $\mathcal{L}_{rt}(gG\text{-IA}) \subseteq \mathcal{L}_{rt}(idG\text{-IA})$  then  $\mathcal{L}_{rt}(gG\text{-IA})$  is closed under  $\varepsilon$ -free homomorphism iff  $\mathcal{L}_{rt}(gG\text{-IA}) = \mathcal{L}_{rt}(idG\text{-IA})$ .*

**Proof.** One can show that the family  $\mathcal{L}_{rt}(idG\text{-IA})$  coincides with the closure of  $\mathcal{L}_{rt}(IA)$  under  $\varepsilon$ -free homomorphisms and forms an AFL which is closed under intersection and reversal. Consequently  $\mathcal{L}_{rt}(idG\text{-IA})$  is closed under  $\varepsilon$ -free homomorphisms, too, implying the closure of  $\mathcal{L}_{rt}(gG\text{-IA})$  under  $\varepsilon$ -free homomorphism if  $\mathcal{L}_{rt}(gG\text{-IA}) = \mathcal{L}_{rt}(idG\text{-IA})$  holds.

On the other hand, since  $\mathcal{L}_{rt}(IA) \subseteq \mathcal{L}_{rt}(gG\text{-IA})$  it follows that the closure of  $\mathcal{L}_{rt}(IA)$  under  $\varepsilon$ -free homomorphisms (which is  $\mathcal{L}_{rt}(idG\text{-IA})$ ) is contained in the closure of  $\mathcal{L}_{rt}(gG\text{-IA})$ . If the latter family is  $\mathcal{L}_{rt}(gG\text{-IA})$  itself then it follows  $\mathcal{L}_{rt}(idG\text{-IA}) \subseteq \mathcal{L}_{rt}(gG\text{-IA}) \subseteq \mathcal{L}_{rt}(idG\text{-IA})$ , i.e.  $\mathcal{L}_{rt}(gG\text{-IA}) = \mathcal{L}_{rt}(idG\text{-IA})$   $\square$

**Corollary 17** *Let  $g : \mathbb{N}_0 \rightarrow \mathbb{N}_0$  be a mapping. If  $\mathcal{L}_{rt}(gG\text{-IA}) \subset \mathcal{L}_{rt}(idG\text{-IA})$  then  $\mathcal{L}_{rt}(gG\text{-IA})$  is not closed under  $\varepsilon$ -free homomorphism, homomorphism and  $\varepsilon$ -free substitution and substitution.*

By Theorem 9 such functions exist.

## References

1. Beyer, W. T. *Recognition of topological invariants by iterative arrays*. Technical Report TR-66, MIT, Cambridge, Proj. MAC, 1969.
2. Buchholz, Th. and Kutrib, M. *Some relations between massively parallel arrays*. *Parallel Comput.* 23 (1997), 1643–1662.
3. Buchholz, Th., Klein, A., and Kutrib, M. *One guess one-way cellular arrays*. In: *Proc. Int. Sym. on Mathematical Foundations of Computer Science (MFCS)*. LNCS 1450, Springer, 1998, 807–815.
4. Buchholz, Th., Klein, A., and Kutrib, M. *Iterative arrays with limited nondeterministic communication cell*. IFIG Research Report 9901, Institute of Informatics, University of Giessen, Giessen, 1999.
5. Buss, J. and Goldsmith, J. *Nondeterminism within P*. *SIAM J. Comput.* 22 (1993), 560–572.

6. Chang, J. H., Ibarra, O. H., and Palis, M. A. *Parallel parsing on a one-way array of finite-state machines*. IEEE Trans. Comput. C-36 (1987), 64–75.
7. Cole, S. N. *Real-time computation by  $n$ -dimensional iterative arrays of finite-state machines*. IEEE Trans. Comput. C-18 (1969), 349–365.
8. Čulik II, K. and Yu, S. *Iterative tree automata*. Theoret. Comput. Sci. 32 (1984), 227–247.
9. Dyer, C. R. *One-way bounded cellular automata*. Inform. Control 44 (1980), 261–281.
10. Fischer, P. C. *Generation of primes by a one-dimensional real-time iterative array*. J. Assoc. Comput. Mach. 12 (1965), 388–394.
11. Fischer, P. C. and Kintala, C. M. R. *Real-time computations with restricted nondeterminism*. Math. Systems Theory 12 (1979), 219–231.
12. Hromkovic, J. et al. *Measures of nondeterminism in finite automata*. In: *Proc. Int. Conf. on Automata, Languages, and Programming (ICALP)*. LNCS 1853, Springer, 2000, 199–210.
13. Ibarra, O. H. and Jiang, T. *On one-way cellular arrays*. SIAM J. Comput. 16 (1987), 1135–1154.
14. Ibarra, O. H. and Palis, M. A. *Some results concerning linear iterative (systolic) arrays*. J. Parallel and Distributed Comput. 2 (1985), 182–218.
15. Ibarra, O. H. and Palis, M. A. *Two-dimensional iterative arrays: Characterizations and applications*. Theoret. Comput. Sci. 57 (1988), 47–86.
16. Kintala, C. M. and Fischer, P. C. *Refining nondeterminism in relativized complexity classes*. SIAM J. Comput. 13 (1984), 329–337.
17. Kintala, C. M. and Wotschke, D. *Amounts of nondeterminism in finite automata*. Acta Inf. 13 (1980), 199–204.
18. Salomaa, K. and Yu, S. *Limited nondeterminism for pushdown automata*. Bulletin of the EATCS 50 (1993), 186–193.
19. Salomaa, K. and Yu, S. *Measures of nondeterminism for pushdown automata*. J. Comput. System Sci. 49 (1994), 362–374.
20. Seiferas, J. I. *Iterative arrays with direct central control*. Acta Inf. 8 (1977), 177–192.
21. Seiferas, J. I. *Linear-time computation by nondeterministic multidimensional iterative arrays*. SIAM J. Comput. 6 (1977), 487–504.
22. Smith III, A. R. *Real-time language recognition by one-dimensional cellular automata*. J. Comput. System Sci. 6 (1972), 233–253.
23. Terrier, V. *On real time one-way cellular array*. Theoret. Comput. Sci. 141 (1995), 331–335.