I F I G
RESEARCH
REPORT

INSTITUT FÜR INFORMATIK

ITERATIVE ARRAYS WITH LIMITED NONDETERMINISTIC COMMUNICATION CELL

Thomas Buchholz Andreas Klein
Martin Kutrib

IFIG RESEARCH REPORT 9901
JANUARY 1999

Institut für Informatik
JLU Gießen
Arndtstraße 2
D-35392 Giessen, Germany
Tel: +49-641-99-32141
Fax: +49-641-99-32149
mail@informatik.uni-giessen.de
www.informatik.uni-giessen.de

JUSTUS-LIEBIG-
UNIVERSITÄT
GIESSEN

IFIG Research Report
IFIG Research Report 9901, January 1999

Iterative Arrays with Limited Nondeterministic Communication Cell

Thomas Buchholz[1] Andreas Klein
Martin Kutrib[2]

Institute of Informatics, University of Giessen
Arndtstr. 2, D-35392 Giessen, Germany

Abstract. Iterative arrays with restricted nondeterminism are investigated. Nondeterminism is provided for the distinguished communication cell only. All the other cells are deterministic ones. Moreover, the number of allowed nondeterministic state transitions is limited dependent 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 a infinite hierarchy of properly included real-time language families. Finally we prove several closure properties of these families.

CR Subject Classification (1998): F.1, F.4.3, B.6.1, E.4

[1]E-mail: buchholz@informatik.uni-giessen.de
[2]E-mail: kutrib@informatik.uni-giessen.de

1 Introduction

Linear arrays of finite automata can be regarded as models for massively parallel computers. Mainly they differ in how the automata are interconnected and in how the input is supplied. Here we are investigating arrays with a simple interconnection pattern. Each node is connected to its immediate neighbor on the right and on the left. The input mode is sequential: One distinguished communication cell fetches the input symbol by symbol. Such arrays are usually called iterative arrays (IA).

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 [9]).

In connection with formal language recognition IAs have been introduced in [8] 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. Compared with e.g. Turing machines there are essential differences in the recognition power. For example the language of palindromes needs a lower bound of n^2 time steps for Turing machines but is acceptable in real-time by IAs.

Various generalizations of IAs have been considered. In [11] 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 [12, 13, 14].

In [16] 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 [17, 18, 19].

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. Here we introduce arrays with restricted 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 which additionally have to appear at the beginning of the computation.

The paper is organized as follows. In section 2 we define the basic notions and the model in question. The fact that two-way devices can simulate a stack, a queue or counters in real-time is used in the sequel. Therefore we show the principles of such simulations. Section 3 is devoted to the possibility to reduce the number of nondeterministic transitions by a constant factor. In section 4 varying the amount of allowed nondeterminism we prove an infinite hierarchy of properly included language families. Due to the results in section 3 to obtain the hierarchy we need sublogarithmic limits for the number of nondeterministic transformations. Finally, in section 5 several closure properties of the real-time acceptors with such limits are shown.

2 Model and Notions

2.1 Definitions

We denote the positive rational numbers by $\mathbb{Q}_+$, the integers by $\mathbb{Z}$, the positive integers $\{1, 2, \ldots\}$ 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 ε and the reversal of a word w by w^R.

An iterative array with nondeterministic communication cell is an infinite linear array of finite automata, sometimes called cells, each of them is connected to its both nearest neighbors to the left and to the right. For our 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 actual state of the cell itself and the actual states of its both neighbors. The state transition of the communication cell additionally depends on the actual input symbol (or if the whole input has been consumed on a special end-of-input symbol). The finite automata change their states 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*

a) S *is the finite, nonempty set of* states,
b) A *is the finite, nonempty set of* input symbols,
c) $F \subseteq S$ *is the set of* accepting states,
d) $s_0 \in S$ *is the* quiescent state,
e) $\# \notin A$ *is the* end-of-input symbol,
f) $\delta : S^3 \to S$ *is the deterministic* local transition function for non-communication cells *satisfying* $\delta(s_0, s_0, s_0) = s_0$,
g) $\delta_{nd} : S^3 \times (A \cup \{\#\}) \to 2^S$ *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. A configuration of $\mathcal{M}$ at some time $t \geq 0$ is a description of its global state, which is actually a pair (w, c_t), where $w \in A^*$ is the remaining input sequence and $c_t : \mathbb{Z} \to S$ is a mapping that gives the actual states of the single cells. During its course of computation a G-IA steps nondeterministically through a sequence of configurations. The configuration (w, c_0) at time 0 is defined by the input word w and the mapping $c_0(i) := s_0$, $i \in \mathbb{Z}$, while subsequent configurations are chosen according to the global transition Δ_{nd}:

Let (w, c) be a configuration then the possible successor configurations (w', c') are as follows:

$$(w', c') \in \Delta_{nd}((w, c)) \iff \begin{array}{l} c'(i) = \delta(c(i-1), c(i), c(i+1)), i \in \mathbb{Z} \setminus \{0\}, \\ c'(0) \in \delta_{nd}(c(-1), c(0), c(+1), a) \end{array}$$

where $a = \#$ and $w' = \varepsilon$ if $w = \varepsilon$, and $a = w_1$ and $w' = w_2 \cdots w_n$ if $w = w_1 \cdots w_n$. Thus, the global transformation Δ_{nd} is induced by δ and δ_{nd}. The i-fold composition of Δ_{nd} is defined as follows:

$$\Delta^0_{nd}((w, c)) := \{(w, c)\}, \qquad \Delta^{i+1}_{nd}((w, c)) := \bigcup_{(w',c') \in \Delta^i_{nd}((w,c))} \Delta_{nd}((w', c'))$$

If the state set is a Cartesian product of some smaller sets $S = S_0 \times S_1 \times \cdots \times S_r$, we will use the notion *register* for the single parts of a state. The concatenation of a specific register of all cells 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*.*

a) *A word $w \in A^+$ is* accepted *by $\mathcal{M}$ if there exists a time step $t_w \in \mathbb{N}$ such that there exists a configuration $(w', c_{t_w}) \in \Delta^{t_w}_{nd}((w, c_0))$ where $c_{t_w}(0) \in F$.*

b) *$L(\mathcal{M}) = \{w \in A^+ \mid w$ is accepted by $\mathcal{M}\}$ is the* language accepted *by $\mathcal{M}$.*

c) *Let $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$, be a mapping. If all $w \in L(\mathcal{M})$ are accepted within $t_w \leq t(|w|)$ 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 $\mathscr{L}_{t(n)}(\text{G-IA})$. In the sequel we will use a corresponding notion for other types of acceptors. If t equals the *identity function* $id(n) := n$ acceptance is said to be in *real-time* and we write $\mathscr{L}_{rt}(\text{G-IA})$. The *linear-time* languages $\mathscr{L}_{lt}(\text{G-IA})$ are defined according to

$$\mathscr{L}_{lt}(\text{G-IA}) := \bigcup_{k \in \mathbb{Q},\, k \geq 1} \mathscr{L}_{k \cdot n}(\text{G-IA})$$

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 $\delta_d : S^3 \times (A \cup \{\#\}) \to S$ for the communication cell is provided and the global transformation induced by δ and δ_d is denoted by Δ_d. Let $g : \mathbb{N} \to \mathbb{N}_0$ 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 gG-IA (g guess IA) if starting with the initial configuration (w, c_0) the possible configurations at some time t

are given by the global transformations as follows:

$$\begin{cases} \{(w, c_0)\} & \text{if } t = 0 \\ \Delta_{nd}^{t}((w, c_0)) & \text{if } t \le g(|w|) \\ \displaystyle\bigcup_{(w',c') \in \Delta_{nd}^{g(|w|)}((w,c_0))} \Delta_{d}^{t-g(|w|)}((w', c')) & \text{otherwise} \end{cases}$$

Observe that all nondeterministic transitions have to be applied before the deterministic ones. 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.

2.2 Algorithmic tools

The fact that two-way devices can simulate a stack, a queue or counters on some track in real-time is often a useful tool for the modular design of algorithms. In the sequel we will make extensively use of the ability of IAs to simulate these data structures. Thereby the communication cell contains the symbol at the top of the stack or the queue or the least significant bit of the counter. Moreover, only the right halfline of cells is used (i.e., cells identified by numbers from $\mathbb{N}_0$). The principles of the simulations are shown in the following.

2.2.1 Stacks

Assume without loss of generality that at every time step at most one symbol from a nonempty finite stack alphabet is pushed onto or popped from the stack.

Each cell has three registers that can store one stack symbol respectively. They are numbered 1, 2 and 3 downward the stack. The third register is used as a buffer. The cells preferably fill two of their registers with symbols. In order to reach that charge they behave according to the following rules (cf. Figure 1):

• If all three registers of the left (upward) neighbor of a cell are filled then it stores the symbol from the buffer register of the neighbor in its own first register. The content of the first and second register is shifted to the second resp. third one. Otherwise the content of the buffer is deleted.

• If the buffer of the left neighbor of a cell is empty, its own second register is empty and the first register of its right neighbor is filled then it stores the symbol from the first register of the right neighbor in its own second register.

• If there is filled just one register of the left neighbor of a cell then the symbol in its own first register is deleted, whereby the content of the second register is shifted to the first one.

Eventually, the actions have to be superimposed and it can easily be verified that the simulation works correctly. Thus, as shown by storing two symbols into one cell and using a buffer the delay is avoided which is needed by the lower cells to react to operations applied to the top of the stack.

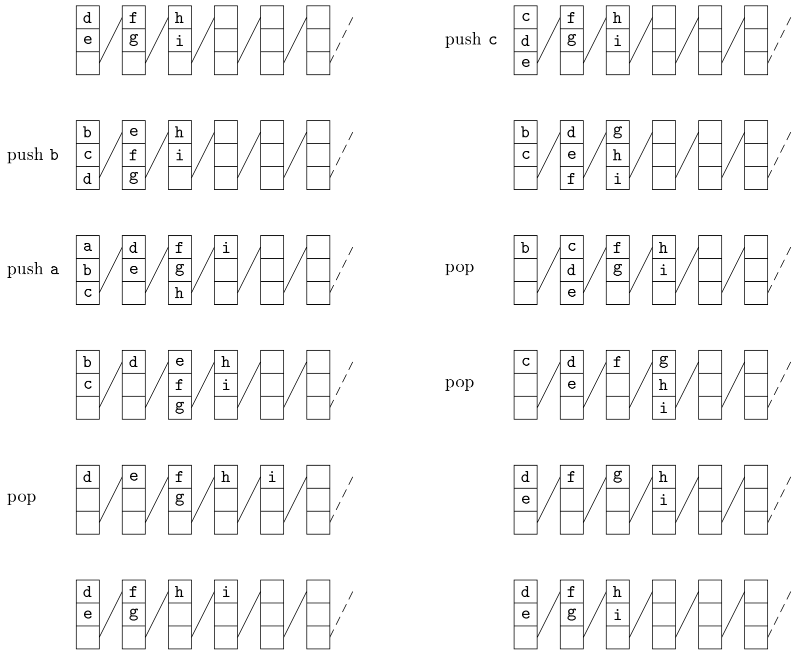

Figure 1: Example for pushdown store simulation by a two-way device.

2.2.2 Queues

By slightly changing the rules two-way devices even can simulate a queue through which symbols can be piped in a first-in-first-out manner.

Therefore, the third register is considered as pipe register through which the symbols are moved from left to right. The other two registers are used for shifting the symbols in the other direction. Similarly, the usage of two registers avoids a delay in the reaction of the lower cells to operations applied to the top of the queue. The rules for a cell are as follows (cf. Figure 2):

• If the pipe register of the left neighbor of a cell is filled with a certain symbol then the cell stores that symbol in its own pipe register if either its own first two registers are filled or the first register of its right neighbor is filled. Otherwise the symbol is stored in its own first empty register.

• If neither the first nor the second register of the left neighbor of a cell is filled then it deletes the content of its own first register. If its own second register is not empty then the content is moved to the first register. If otherwise the first register of its right neighbor is not empty then it stores the content of that register in its own first register.

• If the first and second register of the left neighbor of a cell are filled, its own second register is empty and the first register of its right neighbor is filled with a certain symbol then it stores that symbol in its own second register.

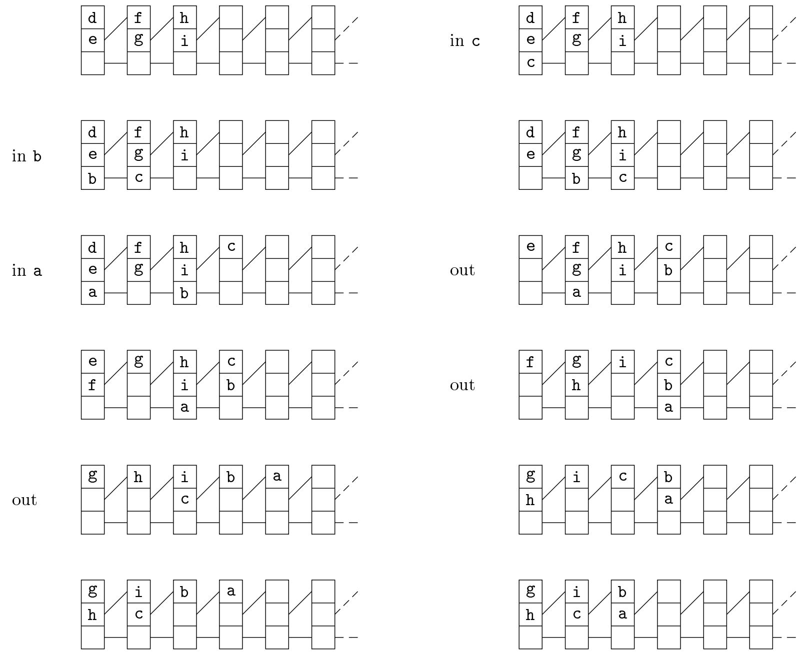

Figure 2: Example for queue simulation by a two-way device.

2.2.3 Dynamic counters

The capability of two-way devices to simulate a stack or queue can easily be exploited to simulate a binary counter which is attached to the cell that forms the top of the stack or the queue. The counter is initialized bit by bit. It can be decremented and incremented by one and be tested for zero.

For the simulation the cells are extended by a forth register through which signals can be sent that are carrying update information.

The counter is initialized from the most to the least significant bit. The initialization is done by pushing the bits onto the stack whereby leading zeros are ignored. If it is more convenient to have a bit stream flowing in the other direction the stack has simply to be replaced by a queue. In order to handle leading zeros in such situations the cells have to notice whether or not a digit one has passed through their pipe register. After the initialization process this information is used to delete the leading zeros. Furthermore, the cell at the top of the queue needs the information immediately after the initialization process in order to test the counter for zero.

For incrementation and decrementation by one the update registers of the cells are used. Depending on its content, say + resp. -, a cell adds resp. subtracts one from the 2-bit number stored in its first and second register and deletes the content of its update register. A cell that recognizes a carry over in its left neighbor stores the corresponding flag in its update register.

The cell containing the most significant bit behaves slightly different. If the counter has to be enlarged after an incrementation then the cell stores a one in its third register. If the counter has to be shortened then it deletes the corresponding register.

The detection whether or not the counter reading is zero can easily be performed since it suffices to check whether the last bit of the counter is deleted.

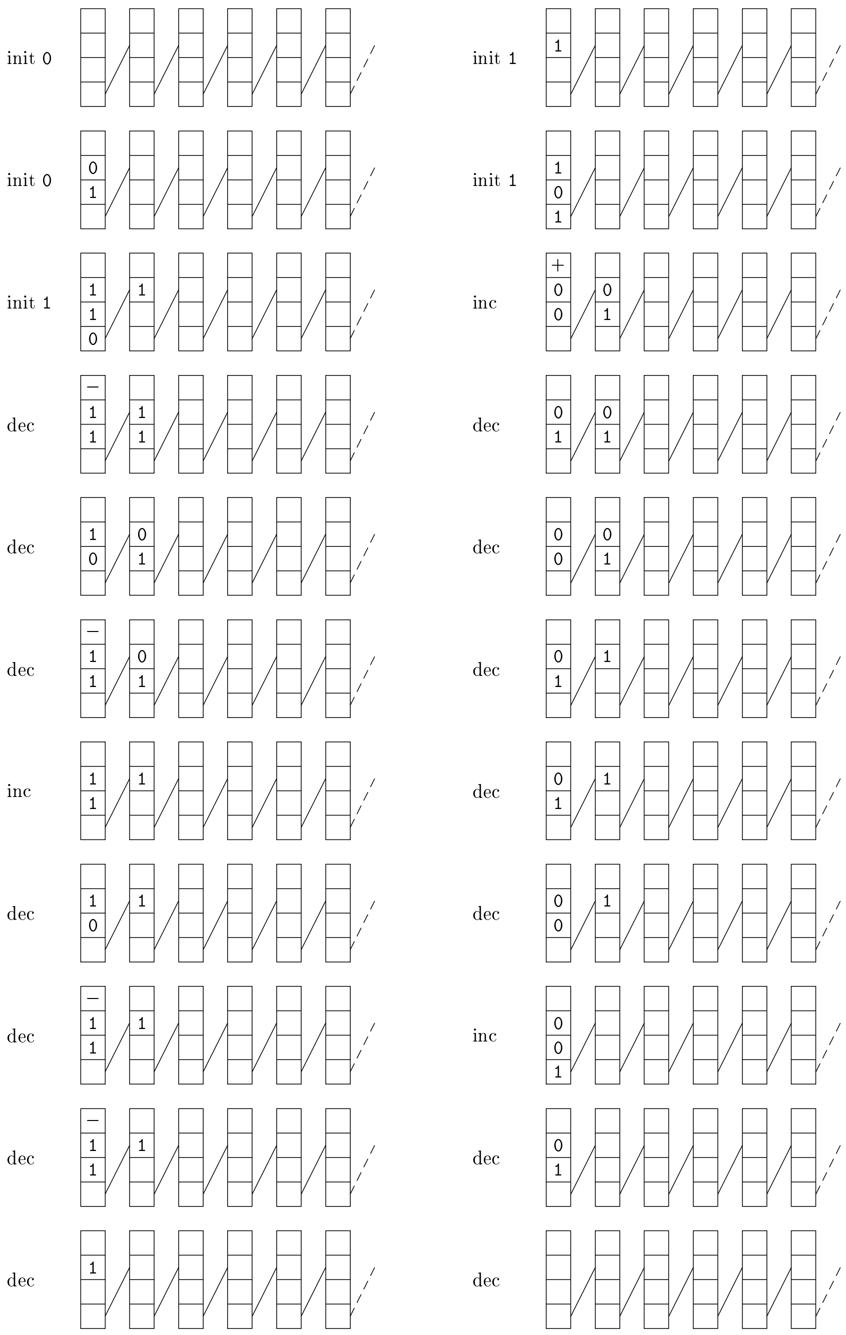

Figure 3: Example for a dynamic counter initialized with the binary number $(01011)_2$ which equals $(11)_{10}$. The leading zero is ignored. The underlying data structure is a stack.

3 Guess Reduction

This section is devoted to the reduction of the number of nondeterministic transformations.

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

$$\mathscr{L}_{t(n)}(g\text{G-IA}) = \mathscr{L}_{t(n)}((k \cdot g)\text{G-IA})$$

Proof. The crucial point in proving the inclusion $\mathscr{L}_{t(n)}(g\text{G-IA}) \subseteq \mathscr{L}_{t(n)}(kg\text{G-IA})$ is that a kgG-IA $\mathcal{M}'$ which is designated to simulate a given gG-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 transformation it has to do so for its remaining nondeterministic steps, whereby again one symbol is popped respectively. 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 in which symbols are pushed. Then we have $m \cdot (k-1) = k \cdot g(n) - m \Longrightarrow m = g(n)$.

To see the other inclusion $\mathscr{L}_{t(n)}(kg\text{G-IA}) \subseteq \mathscr{L}_{t(n)}(g\text{G-IA})$ we use again a pushdown storage. The communication cell of a gG-IA $\mathcal{M}'$ simulating a kgG-IA $\mathcal{M}$ without any loss of time pushes $k-1$ nondeterministically determined functions $d : S^3 \times (A \cup \{\#\}) \to S$ satisfying $d(s_1, s_2, s_3, a) \in \delta_{nd}(s_1, s_2, s_3, a)$ (here δ_{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 $\mathcal{M}$ respectively. During the first deterministic transitions such a function is popped and applied to the states of the communication cell and its neighbors and the actual 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 $\mathcal{M}$. □

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

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

$$\mathscr{L}_{t(n)}(k\text{G-IA}) = \mathscr{L}_{t(n)}(\text{IA})$$

Proof. By theorem 3 it suffices to prove $\mathscr{L}_{t(n)}(1\text{G-IA}) = \mathscr{L}_{t(n)}(\text{IA})$. For structural reasons it follows $\mathscr{L}_{t(n)}(\text{IA}) \subseteq \mathscr{L}_{t(n)}(1\text{G-IA})$ since the nondeterministic transition function of the communication cell can chosen to be identical to the deterministic one. So it remains to show that $\mathscr{L}_{t(n)}(1\text{G-IA})$ is contained in $\mathscr{L}_{t(n)}(\text{IA})$.

Let therefore $\mathcal{M}$ be some 1G-IA. We are going to construct an iterative array $\mathcal{M}'$ which simulates $\mathcal{M}$ without any loss of time. Denote by r the maximum number of choices in a nondeterministic transition of the communication cell of $\mathcal{M}$. $\mathcal{M}'$ now uses r tracks on which it simulates the possible behaviors of $\mathcal{M}$ in parallel. These are completely determined by the transition that the communication cell might perform during the first time step. If $\mathcal{M}'$ is arranged to accept its input iff at least in one of these tracks an accepting computation of $\mathcal{M}$ has been simulated it consequently accepts the same language as $\mathcal{M}$ with the same time complexity. □

The next corollary extends the previous results.

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

$$\mathscr{L}_{rt}(g\text{G-IA}) = \mathscr{L}_{rt}(id\text{G-IA})$$

Proof. By using the technique of pushing nondeterministically determined deterministic transitions for later application that has been introduced in the proof of Theorem 3 a real-time gG-IA can trivially simulate a real-time idG-IA.

On the other hand, since q is a rational number the communication cell of a real-time idG-IA can determine the time step $\lfloor qn \rfloor$ of its own by using the corresponding regularity. Hence, a simulation of a real-time gG-IA by a real-time idG-IA is possible. □

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 $g : \mathbb{N} \to \mathbb{N}_0$, $g(n) \leq n$, be a mapping. If $L \in \mathscr{L}_{rt}(g\text{G-IA})$ 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-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 $-l, \dots, 0, \dots, l$. For the $2l + 1$ states there are $|S|^{2l+1}$ different possibilities. Let $p := |S|^3$ then due to $|S|^{2l+1} = |S|^{2l} \cdot |S| = (|S|^2)^l \cdot |S| \leq (|S|^2)^l \cdot |S|^l = (|S|^2 \cdot |S|)^l \leq p^l$ 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. □

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

Theorem 8 *Let $f : \mathbb{N} \to \mathbb{N}_0$, $f(n) \leq \frac{n}{2}$, and $g : \mathbb{N} \to \mathbb{N}_0$, $g(n) \leq f(n)$, 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 $\mathscr{L}_{lt}(\text{IA})$ then*

$$\mathscr{L}_{rt}(g\text{G-IA}) \subseteq \mathscr{L}_{rt}(f\text{G-IA})$$

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

Since $f \geq g$ $\mathcal{M}'$ can guess the time step g 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 [13]. Thus, L_v belongs to $\mathscr{L}_{2id}(\text{IA})$. Now $\mathcal{M}'$ simulates such an acceptor $\mathcal{M}''$ on an additional track where each cell simulates two cells of $\mathcal{M}''$. Due to this compressed simulation it can take place at double speed. During the first g time steps $\mathcal{M}'$ simulates two steps of $\mathcal{M}''$ at every transformation under the assumption it fetches input symbols a. From the guessed time step g up to the last nondeterministic step f $\mathcal{M}'$ simulates two steps of $\mathcal{M}''$ under the assumption it fetches input symbols b respectively, and during the last $n - f(n)$ time steps $\mathcal{M}'$ simulates $\mathcal{M}''$ without input at double speed.

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 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 $a^t b^{f(n)-t}$ belongs to L_v. It follows $t \in \{g(m) \mid f(m) = f(n)\}$ and due to the assumption $\forall\, m,n, \in \mathbb{N} : f(m) = f(n) \Longrightarrow g(m) = g(n)$ it holds $t = g(n)$. Therefore $\mathcal{M}'$ can verify whether its guess was correct and, thus, accept L in real-time. $\square$

The following situation may clarify the necessity of the condition

$$\forall\, m,n, \in \mathbb{N} : f(m) = f(n) \Longrightarrow g(m) = g(n).$$

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, too. 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 wrongly.

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

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

$$\mathscr{L}_{rt}(g\text{G-IA}) \subset \mathscr{L}_{rt}(f\text{G-IA})$$

Proof. We define a mapping $h : \mathbb{N} \to \mathbb{N}$ by $h(n) := 2^{f(n)}$. h is increasing since f is. Moreover since $f \in o(\log)$ for all $k \in \mathbb{Q}_+$ it holds $\lim_{n\to\infty} \frac{h(n)}{n^k} = \lim_{n\to\infty} \frac{2^{f(n)}}{2^{\log(n)\cdot k}} = 0$ 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 $\mathscr{L}_{rt}(f\text{G-IA})$ but does not belong to $\mathscr{L}_{rt}(g\text{G-IA})$.

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

From the definition it follows that L is not empty (see Example 10). Assume now $L \in \mathscr{L}_{rt}(g\text{G-IA})$. Then by Lemma 7 there exist constants $p, q \in \mathbb{N}$ such that $N(n, m(n)+1, L) \leq p^{(m(n)+1)\cdot q^{g(n)}}$. Obviously, there exists a constant $k \in \mathbb{Q}_+$ for which $2 \cdot \log(p) \cdot q \cdot 2^k \leq \frac{1}{8}$ holds. Since $g \in o(f)$ we can find a constant n_0 such that for all $n \geq n_0 : g(n) < k \cdot f(n)$ and $m(n) \geq 1$. Therefore, the number of equivalence classes is bounded as follows:

$$\begin{aligned} N(n, m(n)+1, L) &\leq p^{(m(n)+1)\cdot q^{g(n)}} \leq p^{2\cdot m(n)\cdot q^{g(n)}} \\ &= 2^{\log(p)\cdot 2\cdot m(n)\cdot 2^{\log(q)\cdot g(n)}} \\ &\leq 2^{\log(p)\cdot 2\cdot m(n)\cdot 2^{\log(q)\cdot k\cdot f(n)}} \\ &= 2^{m(n)\cdot h(n)\cdot 2\cdot\log(p)\cdot q\cdot 2^k} \\ &\leq 2^{m(n)\cdot h(n)\cdot\frac{1}{8}} \end{aligned}$$

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 $u := \$^r w_1 \$ \cdots \$ w_{h(n)} ¢$ 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 $2^{m(n)}$ 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 $V \setminus U$. It follows $uw_i^R ¢ \in L \iff vw_i^R ¢ \notin L$ and, hence,

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

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

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

From the contradiction we obtain $L \notin \mathscr{L}_{rt}(g\text{G-IA})$.

It remains to show $L \in \mathscr{L}_{rt}(f\text{G-IA})$. A fG-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_is 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. $\square$

On a 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

$$f(n) := \log^i(n) \quad \text{and} \quad g(n) := \log^{i+1}(n)$$

($\log^i$ denotes the i-fold composition).

Then by Theorem 9 we have $\mathscr{L}_{rt}(g\text{G-IA}) \subset \mathscr{L}_{rt}(f\text{G-IA})$. Since $\mathscr{L}_{lt}(\text{IA})$ is identical to the linear-time cellular automata languages [18] and

$$\{a^n b^{2^n - n} \mid n \in \mathbb{N}\}$$

is acceptable by such devices

$$\{a^{g(n)} b^{f(n)-g(n)} \mid n \in \mathbb{N}\} \in \mathscr{L}_{lt}(\text{IA})$$

holds. Moreover, from $g \in \log(f)$ follows

$$\forall\, m, n \in \mathbb{N} : f(m) = f(n) \Longrightarrow g(m) = g(n).$$

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 $\$01\$11\$10\$00¢11¢ \in L$. $\square$

5 Closure Properties

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

Theorem 11 *Let $g : \mathbb{N} \to \mathbb{N}_0$ and $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$, be two mappings. Then the family $\mathscr{L}_{t(n)}(g\text{G-IA})$ is closed under union and intersection and trivially contains $\mathscr{L}_{t(n)}(\text{IA})$.*

Proof. Using the same two channel technique of [10] and [18] 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 $\mathscr{L}_{rt}(g\text{G-IA})$ are closed under concatenation and for some others they are not. At first we consider the closure under marked concatenation.

Lemma 12 *Let $g : \mathbb{N} \to \mathbb{N}_0$ be an increasing mapping such that the language $\{\mathtt{a}^{g(m)}\mathtt{b}^{m-g(m)} \mid m \in \mathbb{N}\}$ belongs to $\mathscr{L}_{rt}(\text{IA})$. Then the family $\mathscr{L}_{rt}(g\text{G-IA})$ 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 gG-IAs $\mathcal{M}_1$ resp. $\mathcal{M}_2$. Let L denote the marked concatenation of L_1 and L_2: i.e.,

$$L := \{w_1 \mathtt{c} w_2 \mid w_1 \in L_1 \text{ and } w_2 \in L_2\}$$

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

A gG-IA $\mathcal{M}$ that accepts L in real-time works as follows.

$A_1^* \mathtt{c} A_2^*$ is a regular language and, therefore, belongs trivially to $\mathscr{L}_{rt}(g\text{G-IA})$. Since $\mathscr{L}_{rt}(g\text{G-IA})$ is closed under intersection (cf. Lemma 11) it is sufficient to consider inputs of the form $A_1^* \mathtt{c} A_2^*$ only. Let $w := w_1 \mathtt{c} w_2$ 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 $\mathtt{c}$ 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 $L' := \{\mathtt{a}^{g(m)}\mathtt{b}^{m-g(m)} \mid m \in \mathbb{N}\}$ is simulated on an additional track in parallel. Thereby an input symbol $\mathtt{a}$ is assumed for each nondeterministic step (up to the guessed time $g(n_1)$) and an input symbol $\mathtt{b}$ 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 $\mathtt{a}^x\mathtt{b}^y$ 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 corresponing 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. □

The preconditions of the Lemma can essentially be weakened. Let h be a homomorphism such that $h(x) = \mathtt{a}$ for $x \neq \mathtt{b}$ and $h(\mathtt{b}) = \mathtt{b}$. Then instead of requiring $L := \{\mathtt{a}^{g(m)}\mathtt{b}^{m-g(m)} \mid m \in \mathbb{N}\}$ 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 $\mathscr{L}_{rt}(\text{IA})$.

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

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

For the following we need, furthermore, the notion of time constructibility that has been investigated e.g. in [2, 3, 4].

A strictly increasing function $f : \mathbb{N} \to \mathbb{N}$ is said to be *time-constructible* iff there exists an iterative array such that on input ε its communication cell enters an accepting state exactly at each time step $f(n)$, $n \in \mathbb{N}$. The set of all time-constructible functions is denoted by $\mathscr{F}$.

Let $f : \mathbb{N} \to \mathbb{N}$ be a strictly increasing function. Its complement function $f^{-1} : \mathbb{N} \to \mathbb{N}_0$ is defined by $f^{-1}(m) = \max(\{n \in \mathbb{N} \mid f(n) \leq m\} \cup \{0\})$.

The next lemma shows that the family $\mathscr{G}$ is very rich since $\mathscr{F}$ is.

Lemma 13 $f \in \mathscr{F} \Longrightarrow f^{-1} \in \mathscr{G}$.

Proof. It suffices to show that $L = \{\mathtt{a}^{f^{-1}(n)}\mathtt{b}^{n-f^{-1}(n)} \mid n \in \mathbb{N}\}$ belongs to $\mathscr{L}_{rt}(\text{IA})$. W.l.o.g. we may assume that the input to an iterative array $\mathcal{M}$ which accepts L in real-time is of the form $w = \mathtt{a}^x\mathtt{b}^y$. Further let $n := |w|$.

One task of $\mathcal{M}$ is now simply to simulate a time constructor of f at one of its tracks. Using a pushdown storage which is attached to the communication cell $\mathcal{M}$ can check whether or not $f(x) \leq n$. Time step n occurs at the end of the input. For recognizing time step $f(x)$ $\mathcal{M}$ has to push all its input symbols $\mathtt{a}$ and to pop one symbol at each time step an accepting state is entered in the time construction of f (here always $f(0)$ is assumed to be 0).

Similarly $\mathcal{M}$ can check whether or not $n < f(x+1)$ holds.

Together we have $x \leq f^{-1}(n) < x + 1$, i.e. $x = f^{-1}(n)$ and $y = n - x = n - f^{-1}(n)$. □

At the proof of the closure under marked concatenation the necessity of a marking symbol can be relaxed if the mapping g allows a gG-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 14 *Let $g : \mathbb{N} \to \mathbb{N}_0$, $g \in \Omega(\log)$ be a mapping. If $\mathscr{L}_{rt}(g\text{G-IA})$ is closed under marked concatenation then it is closed under concatenation.*

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

Theorem 15 *Let $g : \mathbb{N} \to \mathbb{N}_0$, $g \in o(\log\log)$, be a mapping. Then $\mathscr{L}_{rt}(g\text{G-IA})$ is not closed under concatenation.*

Proof. Let A be the alphabet consisting of the four symbols $\mathtt{0}$, $\mathtt{1}$, $\mathtt{a}$, and $\mathtt{b}$. 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 $\mathscr{L}_{rt}(\text{IA})$ and thus to $\mathscr{L}_{rt}(g\text{G-IA})$.

Consider now the concatenation $L = L_1L_2$ and assume contrarily that L belongs to $\mathscr{L}_{rt}(g\text{G-IA})$, too. Then let $W_n := \{\mathtt{0}w\mathtt{1} \mid w \in \{\mathtt{a},\mathtt{b}\}^n\}$ for $n \in \mathbb{N}$ and define for each subset $U = \{w_1, \ldots w_k\}$ of W_n the word u as

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

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

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

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. $\mathscr{L}_{rt}(g\text{G-IA})$ is not closed under concatenation. □

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

Theorem 16 *Let $g : \mathbb{N} \to \mathbb{N}_0$, $g \in o(\log)$, be a mapping. Then the family $\mathscr{L}_{rt}(g\text{G-IA})$ 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

$$\begin{aligned} L := \{w_1\$ \ldots w_k\$ \mid \quad & k \geq 2 \wedge \exists\, m \in \mathbb{N} : w_i \in \{\mathtt{0},\mathtt{1}\}^m, 1 \leq i \leq k, \\ & \wedge \exists\, 2 \leq j \leq k : w_1 = w_j\}\,. \end{aligned}$$

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

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 a 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 actual 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 $\mathscr{L}_{rt}(g\text{G-IA})$. 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 $\{x_1, \ldots, x_m\}$ and $\{y_1, \ldots, y_m\}$ of the set $\{\mathtt{0},\mathtt{1}\}^m$ there are words $\$x_1 \cdots \$x_m$ and $\$y_1 \cdots \$y_m$ which belong to different such $(m+1)$-equivalence classes. W.l.o.g. let $x_1 \notin$

$\{y_1, \ldots, y_m\}$. Then $\$x_1 \cdots \$x_m\$x_1$ 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 17 *Let $g : \mathbb{N} \to \mathbb{N}_0$ be a mapping.*
If $\mathscr{L}_{rt}(g\text{G-IA}) \subseteq \mathscr{L}_{rt}(id\text{G-IA})$ then $\mathscr{L}_{rt}(g\text{G-IA})$ is closed under ε-free homomorphism if and only if $\mathscr{L}_{rt}(g\text{G-IA}) = \mathscr{L}_{rt}(id\text{G-IA})$.

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

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

Corollary 18 *Let $g : \mathbb{N} \to \mathbb{N}_0$ be a mapping. If $\mathscr{L}_{rt}(g\text{G-IA}) \subset \mathscr{L}_{rt}(id\text{G-IA})$ then $\mathscr{L}_{rt}(g\text{G-IA})$ is not closed under ε-free homomorphism, homomorphism and ε-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. *On the power of one-way bounded cellular time computers*. Developments in Language Theory III, 1997, pp. 365–375.

[3] Buchholz, Th. and Kutrib, M. *Some relations between massively parallel arrays*. Parallel Computing 23 (1997), 1643–1662.

[4] Buchholz, Th. and Kutrib, M. *On time computability of functions in one-way cellular automata*. Acta Informatica 35 (1998), 329–352.

[5] Buchholz, Th., Klein, A., and Kutrib, M. *One guess one-way cellular arrays*. Mathematical Foundations in Computer Science 1998, LNCS 1450, 1998, pp. 807–815.

[6] Chang, J. H., Ibarra, O. H., and Palis, M. A. *Parallel parsing on a one-way array of finite-state machines*. IEEE Transactions on Computers C-36 (1987), 64–75.

[7] Cole, S. N. *Real-time computation by n-dimensional iterative arrays of finite-state machines*. 7^{th} Ann. IEEE Symposium on Switching and Automata Theory, 1966, pp. 53–77.

[8] Cole, S. N. *Real-time computation by n-dimensional iterative arrays of finite-state machines*. IEEE Transactions on Computers C-18 (1969), 349–365.

[9] Čulik II, K. and Yu, S. *Iterative tree automata*. Theoretical Computer Science 32 (1984), 227–247.

[10] Dyer, C. R. *One-way bounded cellular automata*. Information and Control 44 (1980), 261–281.

[11] Fischer, P. C. *Generation of primes by a one-dimensional real-time iterative array*. Journal of the ACM 12 (1965), 388–394.

[12] Ibarra, O. H. and Jiang, T. *On one-way cellular arrays*. SIAM Journal on Computing 16 (1987), 1135–1154.

[13] Ibarra, O. H. and Palis, M. A. *Some results concerning linear iterative (systolic) arrays*. Journal of Parallel and Distributed Computing 2 (1985), 182–218.

[14] Ibarra, O. H. and Palis, M. A. *Two-dimensional iterative arrays: Characterizations and applications*. Theoretical Computer Science 57 (1988), 47–86.

[15] Mazoyer, J. and Terrier, V. *Signals in one dimensional cellular automata*. Research Report RR 94-50, Ecole Normale Supérieure de Lyon, Lyon, 1994.

[16] Seiferas, J. I. *Iterative arrays with direct central control*. Acta Informatica 8 (1977), 177–192.

[17] Seiferas, J. I. *Linear-time computation by nondeterministic multidimensional iterative arrays*. SIAM Journal on Computing 6 (1977), 487–504.

[18] Smith III, A. R. *Real-time language recognition by one-dimensional cellular automata*. Journal of Computer and System Sciences 6 (1972), 233–253.

[19] Terrier, V. *On real time one-way cellular array*. Theoretical Computer Science 141 (1995), 331–335.