I F I G
RESEARCH
REPORT

INSTITUT FÜR INFORMATIK

ITERATIVE ARRAYS WITH SMALL TIME BOUNDS

Thomas Buchholz Andreas Klein
Martin Kutrib

IFIG RESEARCH REPORT 9906
AUGUST 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 9906, August 1999

Iterative Arrays with Small Time Bounds

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

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

Abstract. 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 investigate deterministic iterative arrays (IA) with small time bounds between real-time and linear-time. It is shown that there exists an infinite dense hierarchy of strictly included complexity classes in that range. The result closes the last gap in the time hierarchy of IAs.

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

Copyright © 1999 by the authors

1 Introduction

Devices of interconnected parallel acting automata have extensively been investigated from a computational complexity point of view. The specification of such a system includes the type and specification of the identical automata, their interconnection scheme (which can imply 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 family accepted by real-time IAs forms a Boolean algebra not closed under concatenation and reversal. In [6] it is shown that for every context-free grammar a 2-dimensional linear-time IA parser exists.

In [9] 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 [10, 11, 12].

Various generalizations of IAs have been considered. In [15] 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 [16, 17, 18]).

In the field of computational complexity there is a particular interest in infinite hierarchies of complexity classes defined by bounding some resources. In order to obtain dense hierarchies one has to show that only a slight increase in the growth rate of the bounding function yields a new complexity class.

Recently in [13] a time hierarchy for IAs has been proved. For time-computable functions t' and t such that $t' \in o(t)$ a strict inclusion between the corresponding complexity classes is shown. Since an IA can be sped-up from $(n+r(n))$-time to $(n+\varepsilon \cdot r(n))$-time for all $\varepsilon > 0$ but real-time is strictly weaker than linear-time this hierarchy is dense in the range above linear-time. Here we close the gap between real-time and linear-time. We show that there exists an infinite dense time hierarchy in between.

Relating our main result for example to Turing machines (TM) it is known that for deterministic one-tape TMs as well as for almost all nondeterministic TMs real-time is as powerful as linear-time. For deterministic real-time machines infinite hierarchies depending on the number of tapes or on the number of heads have been shown. Moreover, for multi- or k-tape TMs ($k \geq 2$) linear-time is known to yield strictly more powerful acceptors than real-time. For any time complexity $n + r(n)$ at most a speed-up to $n + (\varepsilon \cdot r(n))$ ($\varepsilon > 0$), is possible (e.g. see [19] for TM results). The remaining gap between real-time and linear-time was filled with an infinite dense hierarchy in [5].

The basic notions and the model in question are defined in the next section. Section 3 is devoted to the hierarchy theorem and some examples. In Section 4 the proof of the Theorem is presented with the help of two lemmas and three tasks for the construction of a corresponding IA.

2 Preliminaries

We denote the rational numbers by $\mathbb{Q}$, the integers by $\mathbb{Z}$, the positive integers $\{1, 2, ...\}$ by $\mathbb{N}$ and the set $\mathbb{N} \cup \{0\}$ by $\mathbb{N}_0$. The empty word is denoted by ε 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 $f : \mathbb{N}_0 \to \mathbb{N}$ 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 $o(f) = \{g : \mathbb{N}_0 \to \mathbb{N} \mid \lim_{n\to\infty} \frac{g(n)}{f(n)} = 0\}$. If f is strictly increasing then its inverse is defined according to $f^{-1}(n) = \min\{m \in \mathbb{N} \mid f(m) \geq n\}$. The identity function $n \mapsto n$ is denoted by id.

An iterative array is a bi-infinite linear array of finite automata, sometimes called cells, where each of them is connected to its both nearest neighbors (one to the right and one to the left). 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* (IA) *is a system* $(S, \delta, \delta_0, s_0, \#, A, F)$*, 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. $\# \notin A$ *is the* end-of-input symbol,
6. $\delta : S^3 \to S$ *is the* local transition function for non-communication cells *satisfying* $\delta(s_0, s_0, s_0) = s_0$,
   $\delta_0 : S^3 \times (A \cup \{\#\}) \to S$ *is the* local transition function for the communication cell.

Let $\mathcal{M}$ be an IA. 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 $c_t : \mathbb{Z} \to S$ is a mapping that maps the single cells to their current states. The configuration (w_0, c_0) at time 0 is defined by the input word w_0 and the mapping $c_0(i) = s_0$, $i \in \mathbb{Z}$, 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}) is as follows:

$$
\begin{aligned}
(w_{t+1}, c_{t+1}) = \Delta\big((w_t, c_t)\big) \iff& \\
& c_{t+1}(i) = \delta\big(c_t(i-1), c_t(i), c_t(i+1)\big), i \in \mathbb{Z} \setminus \{0\}, \\
& c_{t+1}(0) = \delta_0\big(c_t(-1), c_t(0), c_t(1), a\big)
\end{aligned}
$$

where $a = \#$, $w_{t+1} = \varepsilon$ 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 Δ is induced by δ and δ_0.

If the state set is a Cartesian product of some smaller sets $S = S_1 \times S_2 \times \cdots \times S_k$ 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 word w is accepted by an IA if at some time i during its course of computation on input w the communication cell becomes accepting.

Definition 2 *Let $\mathcal{M} = (S, \delta, \delta_0, s_0, \#, A, F)$ be an* 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 $(w_i, c_i) = \Delta^{[i]}((w, c_0))$.*
2. *$L(\mathcal{M}) = \{w \in A^* \mid w$ is accepted by $\mathcal{M}\}$ is the* language accepted *by $\mathcal{M}$.*
3. *Let $t : \mathbb{N}_0 \to \mathbb{N}$, $t(n) \geq n+1$, be a mapping and i_w be the minimal time step at which $\mathcal{M}$ accepts a $w \in L(\mathcal{M})$. 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 an IA with time complexity t is denoted by $\mathscr{L}_t(\text{IA})$. If t equals the function $n+1$ acceptance is said to be in *real-time* and we write $\mathscr{L}_{rt}(\text{IA})$. The *linear-time* languages $\mathscr{L}_{lt}(\text{IA})$ are defined according to $\mathscr{L}_{lt}(\text{IA}) = \bigcup_{k \in \mathbb{Q}, k \geq 1} \mathscr{L}_{k \cdot n}(\text{IA})$.

In order to prove infinite dense time hierarchies in almost all cases honest time bounding functions are required. Usually the notion "honest" is concretized in terms of the computability or constructibility of the function with respect to the device in question. Here we will use time-constructible functions in IAs.

Definition 3

1. *A strictly increasing function $f : \mathbb{N} \to \mathbb{N}$ is* IA-time-constructible *iff there exists an* IA *$(S, \delta, \delta_0, s_0, \#, A, F)$ such that for all $i \in \mathbb{N}$*

$$c_i(0) \in F \iff \exists\, n \in \mathbb{N} : i = f(n)$$

   where $(\varepsilon, c_i) = \Delta^{[i]}\big((\varepsilon, c_0)\big)$.
2. *The set of all* IA*-time constructible functions is denoted by $\mathscr{F}(\text{IA})$.*
3. *The set of their inverses is $\mathscr{F}^{-1}(\text{IA}) = \{f^{-1} \mid f \in \mathscr{F}(\text{IA})\}$.*

Thus, an iterative array that time constructs a function f becomes accepting on an empty input exactly at the time steps $f(1), f(2), \ldots$ Note that since an $f \in \mathscr{F}(\text{IA})$ is strictly increasing it holds $f^{-1}(n) \leq n$ for all $n \geq 1$.

The family $\mathscr{F}(\text{IA})$ is very rich. It includes $id!$, k^{id}, id^k, $id + \lfloor\sqrt{id}\rfloor$, $id + \lfloor\log\rfloor$, etc., where $k \geq 1$ is an integer. It is closed under the operations addition of constants, addition, iterated addition, multiplication, composition, minimum, maximum etc. [14]. In [9] the function $n \mapsto p_n$ where p_n denotes the nth prime number has shown to be IA-time constructible. In [13] the time-constructibility is related to the time-computability in Turing machines. Further results can be found in [3, 4].

3 The hierarchy between real-time and linear-time

The hierarchy is proved by a specific witness language which is defined dependent on a mapping $h : \mathbb{N} \to \mathbb{N}$. Later on h is related to the time complexity in question.

$$
\begin{aligned}
L_h = \{\$^{h(m)-(m+1)^2+1} w_1 \$ w_2 \$ \cdots \$ w_m ¢ y \mid\ & m \in \mathbb{N} \\
& \wedge\ y, w_i \in \{\mathtt{0}, \mathtt{1}\}^m, 1 \le i \le m \\
& \wedge\ \exists\, 1 \le j \le m : y = w_j \}
\end{aligned}
$$

The following theorem is the main result of the paper.

Theorem 4 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ and $r' : \mathbb{N}_0 \to \mathbb{N}$ be two functions. If $r \in \mathscr{F}^{-1}(\mathrm{IA})$ and $r' \in o(r)$ then*

$$\mathscr{L}_{id+r'}(\mathrm{IA}) \subset \mathscr{L}_{id+r}(\mathrm{IA})$$

The theorem is proved by Lemma 6 and Lemma 7 where it is shown that a specific instance of L_h is not acceptable with time complexity $id + r'$ but is acceptable with time complexity $id + r$. The inclusion $\mathscr{L}_{id+r'}(\mathrm{IA}) \subseteq \mathscr{L}_{id+r}(\mathrm{IA})$ follows immediately from the definitions.

The following example for hierarchies are based on natural functions.

Example 5 Since $\mathscr{F}(\mathrm{IA})$ is closed under composition and contains 2^{id} and id^k, $k \ge 1$, the functions $\log^i$, $i \ge 1$, and $\sqrt[k]{id}$ are belonging to $\mathscr{F}^{-1}(\mathrm{IA})$[§]. Therefore, an application to the hierarchy theorem yields

$$\mathscr{L}_{rt}(\mathrm{IA}) \subset \cdots \subset \mathscr{L}_{id+\log^{[i+1]}}(\mathrm{IA}) \subset \mathscr{L}_{id+\log^{[i]}}(\mathrm{IA}) \subset \cdots \subset \mathscr{L}_{lt}(\mathrm{IA})$$

and

$$\mathscr{L}_{rt}(\mathrm{IA}) \subset \cdots \subset \mathscr{L}_{id+id^{\frac{1}{i+1}}}(\mathrm{IA}) \subset \mathscr{L}_{id+id^{\frac{1}{i}}}(\mathrm{IA}) \subset \cdots \subset \mathscr{L}_{lt}(\mathrm{IA})$$

or in combinations e.g.,

$$
\begin{aligned}
\mathscr{L}_{rt}(\mathrm{IA}) \subset \cdots \mathscr{L}_{id+(\log^{[j+1]})^{\frac{1}{i+1}}}(\mathrm{IA}) \subset \mathscr{L}_{id+(\log^{[j+1]})^{\frac{1}{i}}}(\mathrm{IA}) \subset \cdots \\
\cdots \subset \mathscr{L}_{id+(\log^{[j]})^{\frac{1}{i+1}}}(\mathrm{IA}) \subset \mathscr{L}_{id+(\log^{[j]})^{\frac{1}{i}}}(\mathrm{IA}) \subset \cdots \subset \mathscr{L}_{lt}(\mathrm{IA})
\end{aligned}
$$

[§] Actually, the inverses of 2^{id} and id^k are $\lceil \log \rceil$ and $\lceil id^{\frac{1}{k}} \rceil$ respectively but for convenience we simplify the notation.

4 Proof of the hierarchy

The object of the present section is to prove the lemmas from which the hierarchy follows. At first we show that a specific instance of the language L_h cannot be accepted in $(id + r')$-time by any deterministic IA. Therefore let r be a function in $\mathscr{F}^{-1}(\mathrm{IA})$. Then there exists a function $f_r \in \mathscr{F}(\mathrm{IA})$ such that $r = f_r^{-1}$.

Now let $h_r : \mathbb{N}_0 \to \mathbb{N}$ be defined as

$$h_r(n) = f_r((n+1)^2)$$

Lemma 6 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ and $r' : \mathbb{N}_0 \to \mathbb{N}$ be two functions. If $r \in \mathscr{F}^{-1}(\mathrm{IA})$ and $r' \in o(r)$ then*

$$L_{h_r} \notin \mathscr{L}_{id+r'}(IA)$$

Proof. Contrarily, assume $L_{h_r} = \{\$^{h_r(m)-(m+1)^2+1}w_1\$\cdots\$w_m¢y \mid m \in \mathbb{N} \wedge y, w_i \in \{0,1\}^m, 1 \le i \le m \wedge \exists\, 1 \le j \le m : y = w_j\}$ is acceptable by $\mathcal{M} = (S, \delta, \delta_0, s_0, \#, A, F)$ with time complexity $id + r'$.

Now we consider the situation at time $n - m$ where n denotes the length of the input and $m < n$. The remaining computation of the communication cell depends on the last m input symbols and the states of the cells $-m - r'(n), \ldots, 0, \ldots, m + r'(n)$ only. Due to their distance from the communication cell all the other cells cannot influence the overall computation result.

So we are concerned with at most $|S|^{2\cdot(m+r'(n))+1}$ distinct situations. Related to L_{h_r} where m is the length of the subword y we obtain

$$|S|^{2\cdot(m+r'(h_r(m)))+1} \le (|S|^3)^{m+r'(h_r(m))}.$$

Since $r = f_r^{-1}$ and $h_r(n) = f_r((n+1)^2)$ we have

$$r(h_r(m)) = f_r^{-1}(f_r((m+1)^2)) = (m+1)^2.$$

Moreover, from $r' \in o(r)$ it follows

$$r'(h_r(m)) \in o(m^2).$$

Now choose $k \in \mathbb{N}$ such that $2^{\frac{k}{8}} \ge |S|^3$. For all sufficiently large m it holds

$$r'(h_r(m)) < \frac{m^2}{k}$$

and further

$$(|S|^3)^{m+r'(h_r(m))} \le (|S|^3)^{m+\frac{m^2}{k}} \le (|S|^3)^{\frac{2\cdot m^2}{k}} \le 2^{\frac{m^2}{4}}.$$

On the other hand, for all m-subsets $U = \{w_1, \ldots, w_m\}$ of $\{0,1\}^m$, $m \in \mathbb{N}$, there exists a word w_U such that $y \in U \iff w_U y \in L_{h_r}$. (E.g. $w_U =$

$\$^{h(m)-(m+1)^2+1}w_1\$\cdots\$w_m¢)$. Moreover, for each two distinct m-subsets U and V of $\{0,1\}^m$ there exists a word $y \in (U \setminus V) \cup (V \setminus U)$. Thus

$$w_U y \in L_{h_r} \iff w_V y \notin L_{h_r}.$$

The number of m-subsets U of $\{0,1\}^m$ is

$$\binom{2^m}{m} > \left(\frac{2^m - m}{m}\right)^m \geq \left(\frac{2^{\frac{m}{2}}}{m}\right)^m = \left(2^{\frac{m}{2}-\log(m)}\right)^m > 2^{\frac{m^2}{4}}$$

for all sufficiently large m.

Since $\mathcal{M}$ can distinguish at most $2^{\frac{m^2}{4}}$ different situations at time $n-m$ there must exist two distinct m-subsets U and V such that the situation after processing w_U and w_V is identical. We conclude for $y \in (U \setminus V) \cup (V \setminus U)$ that $\mathcal{M}$ accepts $w_U y$ iff it accepts $w_V y$. But exactly one of the words belongs to L_{h_r} what is a contradiction. □

It remains to show that L_{h_r} is acceptable with time complexity $id + r$. The following results are devoted to the construction of an appropriate IA.

Lemma 7 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be a function such that $r \in \mathscr{F}^{-1}(\mathrm{IA})$ then*

$$L_{h_r} \in \mathscr{L}_{id+r}(\mathrm{IA})$$

Proof. In what follows we construct an IA $\mathcal{M}$ that accepts L_{h_r} with time complexity $id + r$. Since $\mathscr{L}_{id+r}(\mathrm{IA})$ is closed under intersection and contains the regular languages we may restrict our considerations to input words of the form $\$^*\{0,1\}^*\$\{0,1\}^*\$\cdots\$\{0,1\}^+¢\{0,1\}^+$, say $w = \$^p u¢y$ where $u = w_1\$w_2\$\cdots\$w_q$ for $p \geq 0$, $q \geq 1$ and $w_1, \cdots, w_q, y \in \{0,1\}^*$.

$\mathcal{M}$ is designed to perform three tasks in parallel. The first one is to check whether $|w| = h_r(|y|)$, and similarly the second one to ensure that $|u¢y| = |y|^2 + 2|y|$. Finally, the third task is to verify that $|w_1| = \cdots = |w_q| = |y|$ and that $w_j = y$ for some $1 \leq j \leq q$. The input is accepted if and only if all tasks succeed.

In this case if we set $m = |y|$ we have $|u¢| = qm + q$ and thus $qm + q + m = |u¢y| = m^2 + 2m$ from which $q = m$ follows. Further it holds $p = |w| - |u¢y| = h_r(m) - m^2 - 2m = h_r(m) - (m+1)^2 + 1$ and hence $w \in L_{h_r}$.

Now we consider the time requirements of the three tasks mentioned above what proves the lemma. □

Task 1. First we need to derive two constructibility properties of the functions in question. Since $\mathscr{F}(\mathrm{IA})$ contains the functions f_r, id and id^2 and is closed under addition of constants and under composition, the function $h_r = f_r((id+1)^2)$ belongs to $\mathscr{F}(\mathrm{IA})$. Moreover, $h' = h_r + 1$ and $g' = h_r - id$ are IA-time-constructible functions which is obvious for h' and can be seen for g' as follows.

Let $m \geq 1$. Since f_r is strictly increasing it holds $f_r(m+k) \geq f_r(m)+k \geq m+k$ for all $k \in \mathbb{N}$. Observe that g' is strictly increasing, since $g'(m+1) = f_r((m+$

$2)^2)-(m+1) \geq f_r((m+1)^2)+m+2 > f_r((m+1)^2)-m = g'(m) \geq (m+1)^2-m = m^2+m+1 > m$.

Thus, $g' + id$ $(= h_r)$ is IA-time-constructible. The following result has been shown in [2] (in terms of cellular spaces). Let $f \geq id$ be a strictly increasing function, then

$$f + id \in \mathscr{F}(\mathrm{IA}) \Longrightarrow f \in \mathscr{F}(\mathrm{IA})$$

Therefore, we obtain that g' is IA-time-constructible.

In order to realize the first task $\mathcal{M}$ basically simulates a time constructor for g' and another one for h' on two different tracks. The task succeeds if the time constructor for g' marks the communication cell at the time step it fetches the ¢ symbol from the input and the next marking is by the time constructor for h' at the time step it fetches the first end-of-input symbol.

In such a case we have $|w| = h'(m) - 1 = h_r(m)$ for some $m \in \mathbb{N}$. Moreover $|y| = h'(m) - g'(m) - 1 = m$. It remains to show that the constructors do not interfere: Suppose $\mathcal{M}$ fetches ¢ at time step $g'(m)$ for some $m \in \mathbb{N}$. For all $m' \in \mathbb{N}$ we have $g'(m') < h'(m') = f_r((m'+1)^2)+1 < f_r((m'+1)^2)+m'+2 \leq f_r((m'+2)^2) - m' - 1 = g'(m'+1) < h'(m'+1)$. It follows that after being marked by g' the communication cell is next marked by h', then next by g', and so on alternating.

Task 2. This task is realized analogously to the previous task. We just have to use time constructors for $id^2 + id$ and $(id+1)^2$ which are started as soon as the first symbol different from \$ appears in the input. It follows $|u¢y| = m^2 + 2m$ for some $m \in \mathbb{N}$ and $|y| = (m+1)^2 - (m^2+m) - 1 = m$.

Task 3. For the last task an acceptor for $L = \{w_1\$w_2\$\cdots\$w_q¢y \mid q \in \mathbb{N} \wedge \forall\, 1 \leq i \leq q : w_i, y \in \{0,1\}^* \wedge |w_1| = \cdots = |w_q| = |y| \wedge \exists\, 1 \leq j \leq q : w_j = y\}$ is simulated which accepts L with time complexity $2id$ (see Lemma 8). Whereby the simulation is started when the first symbol different from \$ is fetched.

Hence the last task requires at most $p + 2|u\$y|$ time steps.

If all tasks succeed w is accepted at time step $p+2|u\$y| = h_r(m)-(m+1)^2+1+2(m^2+2m) = h_r(m)+m^2+2m < h_r(m)+(m+1)^2 = h_r(m)+r(h_r(m)) = |w|+r(|w|)$. It follows $L_{h_r} \in \mathscr{L}_{id+r}(\mathrm{IA})$.

Lemma 8 $L = \{w_1\$w_2\$\cdots\$w_q¢y \mid q \in \mathbb{N} \wedge \forall\, 1 \leq i \leq q : w_i, y \in \{0,1\}^* \wedge |w_1| = \cdots = |w_q| = |y| \wedge \exists\, 1 \leq j \leq q : w_j = y\} \in \mathscr{L}_{2\cdot id}(\mathrm{IA})$.

Proof. Since $\mathscr{L}_{lt}(\mathrm{IA}) = \mathscr{L}_{2\cdot id}(\mathrm{IA})$ [11] it suffices to construct an acceptor $\mathcal{M}$ for L with time complexity $3 \cdot id$. W.l.o.g. we may assume that the input w is of the form $\{0,1\}^*\$\{0,1\}^*\$\cdots\$\{0,1\}^+¢\{0,1\}^+$, say $w = u¢y$ with $u = w_1\$w_2\$\cdots\$w_q$ and $w_1,\ldots,w_q,y \in \{0,1\}^*$, $q \geq 1$.

Basically $\mathcal{M}$ stores the input symbols into successive cells to the right whereby the ith fetched symbol is piped through the cells such that it is stored into cell $i-1$. Additionally, during the pipe process y is compared with w_i, $1 \leq i \leq q$. The result of the comparison can verify that $|y| = |w_i|$ for all $1 \leq i \leq q$ and that at least one w_i matches y.

The piped symbol ¢ plays a distinguished role for initiating the comparison. The cells storing \$ are used to remember partial comparison results. If the first end-of-input symbol passes through the cell storing the symbol ¢ a leftward moving signal is generated which collects the partial comparison results.

More precisely, the cells of $\mathcal{M}$ are equipped with a pipe and a store register which are initially empty. If a cell detects that the pipe register of its left neighbor cell is filled then it takes over the content and stores it into its own store register if it is empty and into its own pipe register otherwise. If a cell is passed through by the symbol ¢ its subsequent behavior depends on the content of its store register as follows.

If its store register contains 0 or 1 it waits for the first unmarked symbol 0 or 1 in the pipe register of its left neighbor. If this symbol is equal to the content of its own store register then it marks its both register contents by $+$ and otherwise by $-$. In any case it takes over the symbol into its own pipe register.

If its store register contains \$ it waits for the first end-of-input symbol in the pipe register of its left neighbor. Meanwhile it behaves as follows: If it detects an unmarked symbol 0 or 1 (of y) in the pipe register of its left neighbor then it marks its own store register content by $-$. This corresponds to the situation that at least one symbol of y could not be compared to a symbol in the subword w_i which proceeds the occurance of that \$ in w, i.e., $|y| > |w_i|$. Additionally, it copies the symbol in the pipe register of its left neighbor into its own pipe register, whereby existing marks are removed. Thus subsequently the symbols of y can be compared to the symbols of the next subword w_{i+1}.

Further, after the first end-of-input symbol has reached the cell which has stored the symbol ¢ a signal is generated which moves leftward with maximal speed. Each cell which is reached by this signal is turned into an accepting state iff none of the cells already passed through contains an marked (by $-$) symbol \$ or an unmarked symbol 0 or 1 in its store register and, additionally, at least one sequence of cells (between two consecutive \$-cells) has been passed through in which all store registers are marked by $+$.

Thus the communication cell of $\mathcal{M}$ becomes accepting iff $|w_i| \leq |y|$ and $|w_i| \geq |y|$ for all $1 \leq i \leq q$ and there is at least one word w_j which matches y (i.e. $w \in L$).

Moreover it takes $2|u|+1$ time steps to move the symbol ¢ into the store register of cell $|u|$. Since the signal reaches the communication cell after at most $|u|+1$ additional time steps, in total w is accepted in $2|u|+1+|u|+1 < 3|u¢y| = 3|w|$ time steps. Hence , L belongs to $\mathscr{L}_{3\cdot id}(\mathrm{IA})$. □

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, 1997, pp. 365–375.

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

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

[5] Buchholz, Th., Klein, A., and Kutrib, M. *Deterministic turing machines in the range between real-time and linear-time*. To appear.

[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] Fischer, P. C. *Generation of primes by a one-dimensional real-time iterative array*. J. Assoc. Comput. Mach. 12 (1965), 388–394.

[10] Ibarra, O. H. and Jiang, T. *On one-way cellular arrays*. SIAM J. Comput. 16 (1987), 1135–1154.

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

[12] Ibarra, O. H. and Palis, M. A. *Two-dimensional iterative arrays: Characterizations and applications*. Theoret. Comput. Sci. 57 (1988), 47–86.

[13] Iwamoto, C., Hatsuyama, T., Morita, K., and Imai, K. *On time-constructible functions in one-dimensional cellular automata*. Fundamentals of Computation Theory 1999, LNCS 1684, 1999, pp. 317–326.

[14] Mazoyer, J. and Terrier, V. *Signals in one dimensional cellular automata*. Theoret. Comput. Sci. 217 (1999), 53–80.

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

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

[17] Smith III, A. R. *Real-time language recognition by one-dimensional cellular automata*. J. Comput. System Sci. 6 (1972), 233–253.

[18] Terrier, V. *On real time one-way cellular array*. Theoret. Comput. Sci. 141 (1995), 331–335.

[19] Wagner, K. and Wechsung, G. *Computational Complexity*. Reidel Publishing, Dordrecht, 1986.