Johann Wolfgang Goethe-Universität
Frankfurt am Main

Institut für Informatik
Fachbereich Biologie und Informatik

On the Descriptional Complexity of Iterative Arrays

Andreas Malcher

Nr. 3/03

Frankfurter Informatik-Berichte

Institut für Informatik • Robert-Mayer-Straße 11–15 • 60054 Frankfurt am Main

ISSN 1616-9107

On the Descriptional Complexity of Iterative Arrays

Andreas Malcher
Institut für Informatik, Johann Wolfgang Goethe-Universität
D-60054 Frankfurt am Main, Germany
E-Mail: malcher@psc.informatik.uni-frankfurt.de

Abstract

The descriptional complexity of iterative arrays (IAs) is studied. Iterative arrays are a parallel computational model with a sequential processing of the input. It is shown that IAs when compared to deterministic finite automata or pushdown automata may provide savings in size which are not bounded by any recursive function, so-called non-recursive trade-offs. Additional non-recursive trade-offs are proven to exist between IAs working in linear time and IAs working in real time. Furthermore, the descriptional complexity of IAs is compared with cellular automata (CAs) and non-recursive trade-offs are proven between two restricted classes. Finally, it is shown that many decidability questions for IAs are undecidable and not semidecidable.

1 Introduction

The descriptional complexity of abstract machines is a field of theoretical computer science where the conciseness of the representation of a formal language in one model is compared with the size of representation in other models. One early result is the exponential trade-off between nondeterministic finite automata (NFAs) and deterministic finite automata (DFAs). On the one hand, it is known that each n-state NFA can be converted to a DFA with at most 2^n states. On the other hand, Meyer and Fischer [9] proved that there is an infinite sequence of languages such that each language can be accepted by an n-state NFA, but every DFA accepting the same language needs at least 2^n states. Apart from this exponential trade-off between two descriptional systems, Meyer and Fischer proved that the trade-off between context-free grammars and DFAs is not bounded by any recursive function. Such a trade-off is said to be non-recursive. Recursive and non-recursive trade-offs have been proven between many language classes. For a summary of results we refer to [3].

In this paper, we continue the investigation of cellular models which started in [7] with the study of cellular automata (CAs). The main results obtained there may be summarized as follows. There are non-recursive trade-offs between CAs and the sequential models DFAs and pushdown automata (PDAs), between lineartime-CAs and realtime-CAs, and between realtime-CAs with one-way communication and realtime-CAs with two-way communication. Here, these results are complemented by studying iterative

arrays (IAs) which are identical to CAs except that the input mode is sequential. In analogy to the approach in [7], which is based on a technique of Hartmanis [4], we are going to show non-recursive trade-offs between IAs and sequential models as well as between lineartime-IAs and realtime-IAs. It is known that the two language classes generated by realtime-IAs and realtime-CAs are incomparable. Here, it is shown that both language classes are incomparable from a descriptional complexity point of view, since there exist non-recursive trade-offs between realtime-IAs and realtime-CAs and vice-versa. In [7] it is shown that many decidability questions for CAs are undecidable and that CA language classes possess no pumping lemma. Identical results can be achieved for IAs.

2 Preliminaries and Definitions

Let Σ^* denote the set of all words over the finite alphabet Σ, $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$. By $|w|$ we denote the length of a string w, and the reversal of a word w is denoted by w^R. Let REG, DCF, CF, RE denote the families of regular, deterministic context-free, context-free and recursively enumerable languages. DCF_ϵ denotes the family of languages which can be accepted by a deterministic pushdown automaton (DPDA) with no ϵ-moves. In this paper we do not distinguish whether a language L contains the empty word ϵ or not. I.e., L is identified with $L \setminus \{\epsilon\}$. We assume that the reader is familiar with the common notions of formal language theory as presented in [5]. Let S be a set of recursively enumerable languages. Then S is said to be a property of the recursively enumerable languages. A set L has the property S, if $L \in S$. Let L_S be the set $\{<M> \mid T(M) \in S\}$ where $<M>$ is an encoding of a Turing machine M. If L_S is recursive, we say the property S is decidable; if L_S is recursively enumerable, we say the property S is semidecidable. Concerning cellular automata and iterative arrays, we largely follow the notations and definitions given in [6].

Definition: A two-way cellular automaton (CA) A is a quintuple $A = (Q, \#, \Sigma, \delta, F)$, where

1. $Q \neq \emptyset$ is the finite set of cell states,
2. $\# \notin Q$ is the boundary state,
3. $\Sigma \subseteq Q$ is the input alphabet,
4. $F \subseteq Q$ is the set of accepting cell states and
5. $\delta : (Q \cup \{\#\}) \times (Q \cup \{\#\}) \times (Q \cup \{\#\}) \to Q$ is the local transition function.

Restricting the flow of information only from the right to the left, we get a one-way cellular automaton (OCA) and the local transition function maps from $(Q \cup \{\#\}) \times (Q \cup \{\#\})$ to Q. To simplify matters we identify the cells by positive integers.

A configuration of a cellular automaton at some time step $t \geq 0$ is a description of its global state, formally a mapping $c_t : \{1, \ldots, n\} \to Q$ for $n \in \mathbb{N}$. The initial configuration at time 0 is defined by the input word $w = x_1 \ldots x_n$: $c_{0,w}(i) = x_i$, $1 \leq i \leq n$. During a computation the O(CA) steps through a sequence of configurations whereby successor configurations are computed according to the global transition function Δ.

Let c_t, $t \geq 0$, be a configuration, then its successor configuration is defined as follows.

$$
\begin{aligned}
&c_{t+1} = \Delta(c_t) \Longleftrightarrow \\
&\quad c_{t+1}(1) = \delta(\#, c_t(1), c_t(2)) \\
&\quad c_{t+1}(i) = \delta(c_t(i-1), c_t(i), c_t(i+1)), 2 \leq i \leq n-1 \\
&\quad c_{t+1}(n) = \delta(c_t(n-1), c_t(n), \#)
\end{aligned}
$$

for CAs and

$$
\begin{aligned}
&c_{t+1} = \Delta(c_t) \Longleftrightarrow \\
&\quad c_{t+1}(i) = \delta(c_t(i), c_t(i+1)), 1 \leq i \leq n-1 \\
&\quad c_{t+1}(n) = \delta(c_t(n), \#)
\end{aligned}
$$

for OCAs. Thus, Δ is induced by δ.

Definition: A two-way iterative array (IA) A is a tuple $A = (Q, q_0, \nabla, \Sigma, \delta, \delta_0, F)$, where

1. $Q \neq \emptyset$ is the finite set of cell states,
2. $q_0 \in Q$ is the initial (quiescent) state,
3. $\nabla \notin \Sigma$ is the end-of-input symbol,
4. Σ is the input alphabet,
5. $\delta : Q^3 \rightarrow Q$ is the local transition function for non-communication cells where $\delta(q_0, q_0, q_0) = q_0$,
6. $\delta_0 : Q^3 \times (\Sigma \cup \{\nabla\}) \rightarrow Q$ is the local transition function for the communication cell and
7. $F \subseteq Q$ is the set of accepting cell states.

A configuration of an IA at some time $t \geq 0$ is a pair (w_t, c_t) where $w_t \in \Sigma^*$ is the remaining input sequence and $c_t : \mathbb{Z} \rightarrow Q$ is a function that maps the single cells to their current states. The configuration (c_0, w_0) at time 0 is defined by the input word w_0 and the mapping $c_0(i) = q_0, i \in \mathbb{Z}$. The global transition function Δ is induced by δ and δ_0 as follows. Let $(w_t, c_t), t \geq 0$, be a configuration.

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

where $x = \nabla, w_{t+1} = \epsilon$ if $w_t = \epsilon$, and $x = w_1$, $w_{t+1} = w_2 \ldots w_n$ if $w_t = w_1 \ldots w_n$.

An input string w is accepted by an (O)CA (IA) if at some time step i during its computation the leftmost cell (communication cell) enters an accepting state from the set of accepting states $F \subseteq Q$.

Definition: Let $A = (Q, \#, \Sigma, \delta, F)$ be an (O)CA ($A = (Q, q_0, \nabla, \Sigma, \delta, \delta_0, F)$ be an IA).

1. A word $w \in \Sigma^+$ is accepted by A if there exists a time step $i \in \mathbb{N}$ such that $c_i(1) \in F$ holds for the configuration $c_i = \Delta^i(c_{0,w})$ ($(w_i, c_i) = \Delta^i(w, c_0)$).

2. $T(A) = \{w \in \Sigma^+ \mid w \text{ is accepted by } A\}$ is the language accepted by A.
3. Let $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$, be a mapping and i_w be the minimal time step at which A accepts $w \in T(A)$. If all $w \in T(A)$ are accepted within $i_w \leq t(|w|)$ time steps, then $T(A)$ is said to be of time complexity t.
4. $\mathcal{L}_t((\text{O})\text{CA}) = \{L \mid L \text{ is accepted by an (O)CA with time complexity } t\}$. $\mathcal{L}_t(\text{IA}) = \{L \mid L \text{ is accepted by an IA with time complexity } t\}$.
5. If $t(n) = n$ ($t(n) = n + 1$), we say these languages are accepted in real time. The corresponding language classes are denoted by $\mathcal{L}_{rt}((\text{O})\text{CA})$ ($\mathcal{L}_{rt}(\text{IA})$). The languages accepted in linear time $\mathcal{L}_{lt}((\text{O})\text{CA})$ are defined as $\mathcal{L}_{lt}((\text{O})\text{CA}) = \bigcup_{k \in \mathbb{Q}, k \geq 1} \mathcal{L}_{k \cdot t}((\text{O})\text{CA})$ with $t(n) = n$. $\mathcal{L}_{lt}(\text{IA})$ is defined analogously. The corresponding cellular devices are denoted by realtime-(O)CA, lineartime-(O)CA, realtime-IA, and lineartime-IA.

It is known that $\text{DCF}_\epsilon \subset \mathcal{L}_{rt}(\text{IA})$ and that CF and $\mathcal{L}_{rt}(\text{IA})$ are incomparable [6]. $\mathcal{L}_{rt}(\text{IA})$ is closed under union, intersection, complementation, right concatenation with regular sets, marked iteration, and marked concatenation [10].

In the sequel we will use the set of valid computations of a Turing machine. Details are presented in [4] and [5]. The definition of a Turing machine and of an instantaneous description (ID) of a Turing machine may be found in [5].

Definition: Let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ be a deterministic Turing machine.

$$\begin{aligned}\text{VALC}[M] = \{&ID_0(x)\#ID_1(x)^R\#ID_2(x)\#ID_3(x)^R\#\ldots \mid \\ &x \in \Sigma^*, ID_0(x) \in q_0\Sigma^* \text{ is an initial ID, } ID_n(x) \in \Gamma^*F\Gamma^* \text{ is an accepting ID,} \\ &ID_{i+1}(x) \in \Gamma^*Q\Gamma^* \text{results from } ID_i(x), \text{i.e., } ID_i(x) \vdash^M ID_{i+1}(x)\}\end{aligned}$$

The invalid computations INVALC[M] are defined as the complement of VALC[M] with respect to a suitable coding alphabet.

Remark: Let M be an arbitrary Turing machine and $Q_1 \subset Q$ denote the set of states being assumed in M's first computation, i.e., $Q_1 = \{q \in Q \mid \exists \gamma_1, \gamma_2 \in \Gamma, S \in \{L, R\} : \delta(q_0, \gamma_1) = (q, \gamma_2, S)\}$. Then, M can be modified such that all $q \in Q_1$ are entered only in the first computation. This can be achieved by copying the state set Q to Q' and by replacing each transition $\delta(q, \gamma_1) = (p, \gamma_2, S)$ with $\delta(q, \gamma_1) = (p', \gamma_2, S)$ for $q \neq q_0$ and by adding transitions $\delta(q', \gamma_1) = (p', \gamma_2, S)$ for each transition $\delta(q, \gamma_1) = (p, \gamma_2, S)$.

Concerning the notations and definitions of descriptional complexity we follow the presentation in [3]. A descriptional system K is a set of finite descriptors (e.g. automata or grammars) relating each $M \in K$ to a language $T(M)$. The language class being described by K is $T(K) = \{T(M) \mid M \in K\}$. For every language L we define $K(L) = \{M \in K \mid T(M) = L\}$. A complexity measure for K is a total function $|\cdot| : K \to \mathbb{N}$. Comparing two descriptional systems K_1 and K_2, we assume that $T(K_1) \cap T(K_2)$ is not finite. We say that a function $f : \mathbb{N} \to \mathbb{N}$, $f(n) \geq n$ is an *upper bound* for the trade-off when changing from a minimal description in K_1 for an arbitrary language to an equivalent minimal description in K_2, if for all $L \in T(K_1) \cap T(K_2)$

the following holds.

$$\min\{|M| \mid M \in K_2(L)\} \leq f(\min\{|M| \mid M \in K_1(L)\})$$

If no recursive function is an upper bound for the trade-off between two descriptional systems K_1 and K_2, we say the trade-off is non-recursive and write $K_1 \overset{nonrec}{\longrightarrow} K_2$.

3 Non-Recursive Trade-Offs

Theorem 1 *Let M be a Turing machine. Then two realtime-IAs A_1 and A_2 can be constructed such that $T(A_1) = \text{VALC}[M]$ and $T(A_2) = \text{INVALC}[M]$.*

Proof: It is known [6] that $\text{DCF}_\epsilon \subset \mathcal{L}_{rt}(\text{IA})$. We show that VALC$[M]$ is the intersection of two languages $L_1, L_2 \in \text{DCF}_\epsilon$. Since $\mathcal{L}_{rt}(\text{IA})$ is effectively closed under intersection and complementation, we then can construct two realtime-IAs accepting VALC$[M]$ and INVALC$[M]$, respectively. We first observe that DCF_ϵ is closed under marked concatenation and marked iteration, since both operations do not introduce ϵ-moves. It is shown in [5] that VALC$[M] = L_1 \cap L_2$ where

$$\begin{aligned}
L_1 &= (L_3\{\#\})^*(\{\epsilon\} \cup \Gamma^* F \Gamma^* \{\#\}), \\
L_2 &= \{q_0\}\Sigma^*\{\#\}(L_4\{\#\})^*(\{\epsilon\} \cup \Gamma^* F \Gamma^* \{\#\}), \\
L_3 &= \{y\#z^R \mid y \overset{M}{\vdash} z\}, \\
L_4 &= \{y^R\#z \mid y \overset{M}{\vdash} z\}.
\end{aligned}$$

In [5] it is described how two pushdown automata can be constructed to accept L_3 and L_4, respectively. It is not difficult to modify this construction such that L_3 and L_4 are accepted by DPDAs with accepting states and no ϵ-moves. Thus, $L_3, L_4 \in \text{DCF}_\epsilon$. Then, $(L_3\{\#\})^*, (L_4\{\#\})^* \in \text{DCF}_\epsilon$, since DCF_ϵ is closed under marked iteration and every second # acts as a marking symbol. Since DCF_ϵ is closed under marked concatenation, we obtain that $\{q_0\}\Sigma^*\{\#\}(L_4\{\#\})^*$ is in DCF_ϵ. It remains to be shown that the right concatenation with the regular set $(\{\epsilon\} \cup \Gamma^* F \Gamma^* \{\#\})$ does not introduce ϵ-moves. This can be realized with a second component in the state sets of the DPDAs accepting $(L_3\{\#\})^*$ and $(L_4\{\#\})^*$, respectively. This component checks after every second # whether the remaining input is of the form $(\{\epsilon\} \cup \Gamma^* F \Gamma^* \{\#\})$. If so, the input is accepted. Obviously, no ϵ-moves are necessary. Hence, two DPDAs without ϵ-moves can be constructed accepting L_1 and L_2, respectively. □

The next theorem provides a criterion for the existence of non-recursive trade-offs. The proof uses a technique given by Hartmanis [4] and may be found in [7].

Theorem 2 *Let K_1 and K_2 be two descriptional systems. If for every Turing machine M a language $L_M \in T(K_1)$ can be effectively constructed such that $L_M \in T(K_2) \Leftrightarrow T(M)$ is finite, then the trade-off between K_1 and K_2 is non-recursive.*

Remark: Let A be a realtime-IA. In [6] it is explained that the result of the computation of a remaining input of length m in A depends on the states of the cells

$-m-1,\ldots,0,\ldots,m+1$. These cells will be denoted by the term m-window. It is observed in [6] that there are at most $n^{2(m+1)+1}$ different m-windows, where n denotes the number of states in A.

Remark: Every $x \in \mathrm{VALC}[M]^R$ is ending with a string $\#yq_0$ where $\#$ is a separating symbol, $y \in \Sigma^*$ the reversal of the input and q_0 the initial state of M. The mapping $\pi : \mathrm{VALC}[M]^R \to \Sigma^*$ is defined as $\pi(x) = y$ and extracts the input.

Lemma 1 *Let A be a realtime-IA accepting $\mathrm{VALC}[M]^R$ where M is a Turing machine accepting an infinite language $L \subseteq \Sigma^*$. Let $U = \{u_1, u_2, \ldots\}$ be an infinite subset of $\mathrm{VALC}[M]^R$ which has the property that $|\pi(u_j)| > |\pi(u_i)| + 3$ and $|u_i| < |u_j|$ for $i \geq 1$ and $j > i$. Then there exists a natural number m such that there exist $u_i, u_j \in U$ with $j > i$, $|\pi(u_i)| > m$ and the m-window of the configuration at time $|u_i| - m$ when processing u_i occurs in at least one m-window of the configurations up to time $|u_j| - m - 3$ when processing u_j.*

Proof: By way of contradiction we assume that the above statement does not hold. Then, for all natural numbers m there exist no such words u_i and u_j. Or, in other words, for arbitrary m holds: each $u_i \in U$ with $|\pi(u_i)| > m$ has an m-window at time $|u_i| - m$ which occurs in no other computation of words $u_j \in U$ with $j > i$ up to time $|u_j| - m - 3$. We now consider the words $w_1 = u_{i+1}, w_2 = u_{i+2}, \ldots, w_{|Q|^{2(m+1)+1}+1} = u_{i+|Q|^{2(m+1)+1}+1}$. Due to our assumption, we know that, for $1 \leq j \leq |Q|^{2(m+1)+1} + 1$, w_j has an m-window c_j at time $|w_j| - m$ that does not occur in any m-window of computations of $w_{j+1}, \ldots, w_{|Q|^{2(m+1)+1}+1}$ up to time $|w_l| - m - 3$ for $j + 1 \leq l \leq |Q|^{2(m+1)+1} + 1$. Hence the set $\{c_1, c_2, \ldots, c_{|Q|^{2(m+1)+1}+1}\}$ is pairwise distinct which is a contradiction to the fact that there are at most $|Q|^{2(m+1)+1}$ different m-windows. □

Lemma 2 *Let M be a Turing machine and* $\mathrm{L}[M] = \{w^{|w|!} \mid w \in \{\#_0\}\mathrm{VALC}[M]\{\#_1\}\}$.

(1) $\mathrm{INVALC}[M] \in REG \Leftrightarrow T(M)$ *is finite*
(2) $\mathrm{VALC}[M] \in CF \Leftrightarrow T(M)$ *is finite*
(3) $\mathrm{VALC}[M]^R \in \mathcal{L}_{rt}(IA) \Leftrightarrow T(M)$ *is finite*
(4) $\mathrm{INVALC}[M]^R \in \mathcal{L}_{rt}(IA) \Leftrightarrow T(M)$ *is finite*
(5) $\mathrm{L}[M] \in \mathcal{L}_{rt}(OCA) \Leftrightarrow T(M)$ *is finite*
(6) $\mathrm{L}[M] \in \mathcal{L}_{rt}(IA)$

Proof: (2) is proven in [5] and (1) is then easy to show. The "if" portion of (3) is obvious, since REG is a subset of $\mathcal{L}_{rt}$(IA). The "only if" portion is proven by using Lemma 1. We show that $\mathrm{VALC}[M]^R \notin \mathcal{L}_{rt}(\mathrm{IA})$ if $T(M)$ is infinite: we assume that $\mathrm{VALC}[M]^R \in \mathcal{L}_{rt}(\mathrm{IA})$. Let $m \in \mathbb{N}$ be the integer from Lemma 1 which can be applied, since $T(M)$ is infinite. Then there exist two words $u, u' \in \mathrm{VALC}[M]^R$ with $|u'| > |u|$ and $|\pi(u)| > m$. Let $u = u_1u_2$ with $|u_2| = m$ and $u' = u'_1u'_2$ where u'_1 is the shortest prefix of u' such that the m-window c after processing u'_1 is identical to that after processing u_1. It is easily observed that u has a suffix $ID_1(\pi(u))^R\#\pi(u)q_0$ and that

u' has a suffix $ID_1(\pi(u'))^R\#\pi(u')q_0$. We now consider the string $w = u_1'u_2$ and have to differentiate three cases. At first, w may have the wrong format of an ID. Then $w \notin \text{VALC}[M]^R$. If w has the correct format, then w has a suffix $\#ID_l(\pi(u'))\#xu_2$, if l is even, or $\#ID_l(\pi(u'))^R\#xu_2$ otherwise. Since u_2 is a suffix of $\pi(u)q_0 \in \Sigma^+\{q_0\}$, we can assume that $x \in \Sigma^*$. If $l > 1$, then $w \notin \text{VALC}[M]^R$, since M can be modified such that, due to an above remark, certain states are assumed only in the first computation. If $l = 1$, then w has a suffix $ID_1(\pi(u'))^R\#xu_2$ and $|xu_2| \leq |\pi(u')q_0| - 3 < |\pi(u')q_0| - 2$, since the identical m-window c is assumed after at most $|u'| - m - 3$ time steps. We can observe that an ID changes its length when compared to its preceding ID by at most 1. I.e., $|ID_{k+1}(y)| = |ID_k(y)| + p$ with $p \in \{-1, 0, 1\}$ and $k \geq 0$, $y \in \Sigma^*$. Hence, $|ID_1(\pi(u'))| = |xu_2| + p \leq |xu_2| + 1 < |\pi(u')q_0| - 1 = |ID_0(\pi(u'))| - 1$. This implies that $w \notin \text{VALC}[M]^R$. So we obtain in all three cases that $w \notin \text{VALC}[M]^R$. On the other hand, since the m-window c leads to acceptance when processing u_2, we obtain $w = u_1'u_2 \in \text{VALC}[M]^R$ which is a contradiction. This proves (3). Since $\text{INVALC}[M]^R = \overline{\text{VALC}[M]^R}$ and $\mathcal{L}_{rt}(\text{IA})$ is closed under complementation, (4) follows from (3). A proof of (5) may be found in [7]. To prove (6) we show how to construct a realtime-IA recognizing L[M]. L[M] is the intersection of the following three languages L_1, L_2, L_3. Let VALC[M] $\subset \Sigma^*$ and $\#_0, \#_1$ be new symbols with $\{\#_0, \#_1\} \cap \Sigma = \emptyset$.

$$L_1 = \{w \mid w \in (\{\#_0\}\text{VALC}[M]\{\#_1\})^*\}$$
$$L_2 = \{w^n \mid w \in \{\#_0\}\Sigma^*\{\#_1\}, n \geq 2, n \textit{ is even}\}$$
$$L_3 = \{wx \mid w \in \{\#_0\}\Sigma^*\{\#_1\}, x \in (\{\#_0\}\Sigma^*\{\#_1\})^*, |wx|_{\#_0} = |w|!\}$$

Since $\mathcal{L}_{rt}(\text{IA})$ is closed under intersection, it remains to be shown that $L_i \in \mathcal{L}_{rt}(\text{IA})$ for $1 \leq i \leq 3$. $L_1 \in \mathcal{L}_{rt}(\text{IA})$, since $\{\#_0\}\text{VALC}[M] \in \mathcal{L}_{rt}(\text{IA})$ and $\mathcal{L}_{rt}(\text{IA})$ is closed under marked concatenation where $\#_1$ acts as a marking symbol.

In [1] it is shown that $\{ww \mid w \in \Sigma^*\} \in \mathcal{L}_{rt}(\text{IA})$. Thus, $L = \{ww \mid w \in \{\#_0\}\Sigma^*\{\#_1\}\} \in \mathcal{L}_{rt}(\text{IA})$. We can observe that $L^* \in \mathcal{L}_{rt}(\text{IA})$, because the iteration of languages L is in a way a marked iteration: after the second $\#_1$, the next L starts. By a similar argument and the fact that $\mathcal{L}_{rt}(\text{IA})$ is closed under right concatenation with regular sets, we can see that $\{\#_0\}\Sigma^*\{\#_1\}L^*\{\#_0\}\Sigma^*\{\#_1\} \in \mathcal{L}_{rt}(\text{IA})$. Hence, $L_2 = L^* \cap \{\#_0\}\Sigma^*\{\#_1\}L^*\{\#_0\}\Sigma^*\{\#_1\} \in \mathcal{L}_{rt}(\text{IA})$.

Now it remains for us to show that $L_3 \in \mathcal{L}_{rt}(\text{IA})$. We sketch the construction. We use an iterative array where each cell is split into four subcells, so we can speak of four tracks. On the first track we are checking whether the input string has the correct format $(\{\#_0\}\Sigma^*\{\#_1\})^*$. The second track computes the factorials according to the construction presented in [8]. I.e., the communication cell assumes a designated state whenever a factorial has been computed. On the third and fourth track we install binary counters (Counter1, Counter2) starting in the communication cell. The number of cells used for storing is varying with the size of the number that has to be stored. The rightmost cell used is marked with a special symbol. A construction of such counters is possible in realtime-IAs due to their two-way communication. While reading the input up to the first $\#_1$, Counter1 is incremented in every step. Whenever a factorial has been computed on the second track, Counter1 is decremented. When Counter1 is decremented to zero, we know that $|w|!$ has been computed. Up to this moment, Counter2 is incremented in every time step. Whenever $\#_0$ is read, Counter2 is decremented. When Counter2 is decremented to zero, we know that $|w|!$ $\#_0$'s have

been read. If the remaining input is a string in $\Sigma^*\#_1$, the input is accepted, otherwise the input is rejected. Hence, a realtime-IA accepting L_3 can be constructed. Thus, $\mathrm{L}[M] \in \mathcal{L}_{rt}(\mathrm{IA})$. □

Now, we can show the following non-recursive trade-offs by combining Theorem 2 and Lemma 2.

Theorem 3 *(1) realtime-IA* $\overset{nonrec}{\longrightarrow}$ *DFA*

(2) realtime-IA $\overset{nonrec}{\longrightarrow}$ *PDA*

(3) PDA $\overset{nonrec}{\longrightarrow}$ *realtime-IA*

(4) lineartime-IA $\overset{nonrec}{\longrightarrow}$ *realtime-IA*

(5) realtime-OCA $\overset{nonrec}{\longrightarrow}$ *realtime-IA*

(6) realtime-IA $\overset{nonrec}{\longrightarrow}$ *realtime-OCA*

Proof: (1) and (2) can be shown with $L_M = \mathrm{INVALC}[M]$ and $L_M = \mathrm{VALC}[M]$, respectively. To prove (3) we set $L_M = \mathrm{INVALC}[M]^R$. $L_M \in \mathrm{CF}$, since $\mathrm{INVALC}[M] \in \mathrm{CF}$ [5] and CF is closed under reversal. Now, we consider $L_M = \mathrm{VALC}[M]^R$. It is shown in [7] that $\mathrm{VALC}[M] \in \mathcal{L}_{rt}(\mathrm{OCA})$. Since $\mathcal{L}_{rt}(\mathrm{OCA})$ is closed under reversal, we obtain that $L_M \in \mathcal{L}_{rt}(\mathrm{OCA}) \subset \mathcal{L}_{lt}(\mathrm{CA}) = \mathcal{L}_{lt}(\mathrm{IA})$ [2]. This proves (4) and (5). Finally, (6) can be shown with $L_M = \mathrm{L}[M]$. □

4 Decidability Questions

Due to the theorem of Rice [5], all non-trivial decidability questions for Turing machines are undecidable. Furthermore, it is known that certain decidability questions such as emptiness, inclusion, equivalence, finiteness, infiniteness, regularity, and context-freedom are not semidecidable for Turing machines [5]. The fact that $\mathrm{VALC}[M]$ is in $\mathcal{L}_{rt}(\mathrm{IA})$ and the results of Lemma 2 imply that many decidability questions for IAs can be reduced to decidability questions for Turing machines. Hence, the above-mentioned decidability questions are undecidable and not semidecidable for IAs. A detailed discussion may be found in [7] where similar results are shown for cellular automata. It should be noted that some of the undecidability results for realtime-IAs were first proven by Seidel in [10] using reductions of the Post Correspondence Problem. The approach discussed here provides simpler proofs and shows the non-semidecidability of the questions.

Theorem 4 *It is not semidecidable for arbitrary realtime-IAs* A, A' *whether*

- $T(A) = \emptyset$, $T(A) = \Sigma^*$
- $T(A)$ *is finite,* $T(A)$ *is infinite*
- $T(A) = T(A')$, $T(A) \subseteq T(A')$

- $T(A) \in REG$, $T(A) \in CF$

In [7] it is shown that there exists no pumping lemma and no minimization algorithm for cellular automata. The proofs rely on the fact that for cellular automata infiniteness is not semidecidable and emptiness is undecidable. Since both statements are valid for IAs, we obtain that there exists no pumping lemma and no minimization algorithm for IAs.

Theorem 5 *$\mathcal{L}_{rt}(IA)$ and each language class containing $\mathcal{L}_{rt}(IA)$ does not possess a pumping lemma.*

Theorem 6 *For realtime-IAs there is no minimization algorithm converting an arbitrary realtime-IA A to a realtime-IA A' which accepts $T(A)$ and has a minimal number of states.*

5 Conclusion

We studied descriptional complexity aspects of iterative arrays. The results known for CAs were complemented by similar results for IAs. Moreover, non-recursive trade-offs were shown to exist between CAs and IAs operating in real time.

References

[1] S.N. Cole: "Real-time computation by n-dimensional iterative arrays of finite-state machines," IEEE Transaction on Computers, C-18: 349–365, 1969

[2] M. Delorme, J. Mazoyer: "Cellular automata as language recognizers," In M. Delorme, J. Mazoyer (eds.): "Cellular automata," 153–179, Kluwer Academic Publishers, Dordrecht, 1999

[3] J. Goldstine, M. Kappes, C.M.R. Kintala, H. Leung, A. Malcher, D. Wotschke: "Descriptional complexity of machines with limited resources," Journal of Universal Computer Science, 8(2): 193–234, 2002

[4] J. Hartmanis: "On the succinctness of different representations of languages," In H. Maurer (ed.): International Colloquium on Automata, Languages and Programming 1979 (ICALP'79), LNCS 71, 282–288, Springer-Verlag, Berlin, 1979

[5] J.E. Hopcroft, J.D. Ullman: Introduction to Automata Theory, Languages and Computation, Addison-Wesley, Reading MA, 1979

[6] M. Kutrib: "Automata arrays and context-free languages," In C. Martín-Vide, V. Mitrana (eds.): "Where Mathematics, Computer Science, Linguistics and Biology Meet," 139–148, Kluwer Academic Publishers, Dordrecht, 2001

[7] A. Malcher: "Descriptional complexity of cellular automata and decidability questions," Journal of Automata, Languages and Combinatorics, 7(4): 549–560, 2002

[8] J. Mazoyer, V. Terrier: "Signals in one-dimensional cellular automata," Theoretical Computer Science, 217: 53–80, 1999

[9] A.R. Meyer, M.J. Fischer: "Economy of descriptions by automata, grammars, and formal systems," IEEE Symposium on Foundations of Computer Science, 188–191, 1971

[10] S.R. Seidel: "Language recognition and the synchronization of cellular automata," Technical Report 79-02, Department of Computer Science, University of Iowa, Iowa City, 1979

Interne Berichte am Fachbereich Informatik
Johann Wolfgang Goethe-Universität Frankfurt

1/1987 Risse, Thomas:
On the number of multiplications needed to evaluate the reliability of k-out-of-n systems

2/1987 Roll, Georg [u.a.]:
Ein Assoziativprozessor auf der Basis eines modularen vollparallelen Assoziativspeicherfeldes

3/1987 Waldschmidt, Klaus ; Roll, Georg:
Entwicklung von modularen Betriebssystemkernen für das ASSKO-Multi-Mikroprozessorsystem

4/1987 Workshop über Komplexitätstheorie, effiziente Algorithmen und Datenstrukturen:
3.2.1987, Universität Frankfurt/Main

5/1987 Seidl, Helmut:
Parameter-reduction of higher level grammars

6/1987 Kemp, Rainer:
On systems of additive weights of trees

7/1987 Kemp, Rainer:
Further results on leftist trees

8/1987 Seidl, Helmut:
The construction of minimal models

9/1987 Weber, Andreas ; Seidl, Helmut:
On finitely generated monoids of matrices with entries in N

10/1987 Seidl, Helmut:
Ambiguity for finite tree automata

1/1988 Weber, Andreas:
A decomposition theorem for finite-valued transducers and an application to the equivalence problem

2/1988 Roth, Peter:
A note on word chains and regular languages

3/1988 Kemp, Rainer:
Binary search trees for d-dimensional keys

4/1988 Dal Cin, Mario:
On explicit fault-tolerant, parallel programming

5/1988 Mayr, Ernst W.:
Parallel approximation algorithms

6/1988 Mayr, Ernst W.:
Membership in polynomial ideals over Q is expotential space complete

1/1989 Lutz, Joachim [u.a.]:
Parallelisierungskonzepte für ATTEMPO-2

2/1989 Lutz, Joachim [u.a.]:
Die Erweiterung der ATTEMPO-2 Laufzeitbibliothek

3/1989 Kemp, Rainer:
A One-to-one Correspondence between Two Classes of Ordered Trees

4/1989 Mayr, Ernst W. ; Plaxton, C. Greg:
Pipelined Parallel Prefix Computations, and Sorting on a Pipelined Hypercube

5/1989 Brause, Rüdiger:
Performance and Storage Requirements of Topology-conserving Maps for Robot Manipulator Control

6/1989 Roth, Peter:
Every Binary Pattern of Length Six is Avoidable on the Two-Letter Alphabet

7/1989 Mayr, Ernst W.:
Basic Parallel Algorithms in Graph Theory

8/1989 Brauer, Johannes:
A Memory Device for Sorting

1/1990 Vollmer, Heribert:
Subpolynomial Degrees in P and Minimal Pairs for L

2/1990 Lenz, Katja:
The Complexity of Boolean Functions in Bound Depth Circuits over Basis $\{\wedge, \oplus\}$

3/1990 Becker, Bernd ; Hahn R. ; Krieger, R. ; Sparmann, U.:
Structure Based Methods for Parallel Pattern Fault Simulation in Combinational Circuits

4/1990 Goldstine, J. ; Kintala, C.M.R. ; Wotschke D.:
On Measuring Nondeterminism in Regular Languages

5/1990 Goldstein, J. ; Leung, H. ; Wotschke, D.:
On the Relation between Ambiguity and Nondeterminism in Finite Automata

1/1991 Brause, Rüdiger:
Approximator Networks and the Principles of Optimal Information Distribution

2/1991 Brauer, Johannes ; Stuchly, Jürgen:
HyperEDIF: Ein Hypertext-System für VLSI Entwurfsdaten

3/1991 Brauer, Johannes:
Repräsentation von Entwurfsdaten als symbolische Ausdrücke

4/1991 Trier, Uwe:
Additive Weights of a Special Class of Nonuniformly Distributed Backtrack Trees

5/1991 Dömel, P. [u.a.]:
Concepts for the Reuse of Communication Software

6/1991 Heistermann, Jochen:
Zur Theorie genetischer Algorithmen

7/1991 Wang, Alexander [u.a.]:
Embedding complete binary trees in faulty hypercubes

1/1992 Brause, Rüdiger:
The Minimum Entropy Network

2/1992 Trier, Uwe:
Additive Weights Under the Balanced Probability Model

3/1992 Trier, Uwe:
(Un)expected path lengths of asymetric binary search trees

4/1992 Coen Alberto ; Lavazza, Luigi ; Zicari, Roberto:
Assuring type-safety of object oriented languages

5/1992 Coen, Alberto ; Lavazza, Luigi ; Zicari, Roberto:
Static type checking of an object-oriented database schema

6/1992 Coen, Alberto ; Lavazza, Luigi ; Zicari, Roberto:
Overview and progress report of the ESSE project : Supporting object-oriented database schema analysis and evolution

7/1992 Schmidt-Schauß, Manfred:
Some results for unification in distributive equational theories

8/1992 Mayr, Ernst W. ; Werchner, Ralph:
Divide-and-conquer algorithms on the hypercube

1/1993 Becker, Bernd ; Drechsler, Rolf ; Hengster, Harry:
Local circuit transformations preserving robust path-delay-fault testability

2/1993 Krieger, Rolf ; Becker, Bernd ; Sinković, Robert:
A BDD-based algorithmen for computation of exact fault detection probabilities

3/1993 Mayr, Ernst W. ; Werchner, Ralph:
Optimal routing of parentheses on the hypercube

4/1993 Drechsler, Rolf ; Becker, Bernd:
Rapid prototyping of fully testable multi-level AND/EXOR networks

5/1993 Becker, Bernd ; Drechsler, Rolf:
On the computational power of functional decision diagrams

6/1993 Berghoff, P. ; Dömel, P. ; Drobnik, O. [u.a.]:
Development and management of communication software systems

7/1993 Krieger, Rolf ; Hahn, Ralf ; Becker Bernd:
test_circ : Ein abstrakter Datentyp zur Repräsentation von hierarchischen Schaltkreisen (Benutzeranleitung)

8/1993 Krieger, Rolf ; Becker, Bernd ; Hengster, Harry:.
lgc++ : Ein Werkzeug zur Implementierung von Logiken als abstrakte Datentypen in C++ (Benutzeranleitung)

9/1993 Becker, Bernd ; Drechsler, Rolf ; Meinel, Christoph:
On the testability of circuits derived from binary decision diagrams

10/1993 Liu, Ling ; Zicari, Roberto ; Liebherr, Karl ; Hürsch, Walter:
Polymorphic reuse mechanism for object-oriented database specifications

11/1993 Ferrandina, Fabrizio ; Zicari, Roberto:
Object-oriented database schema evolution: are lazy updates always equivalent to immediate updates ?

12/1993 Becker, Bernd ; Drechsler, Rolf ; Werchner, Ralph:
On the Relation Between BDDs and FDDs

13/1993 Becker, Bernd ; Drechsler, Rolf:
Testability of circuits derived from functional decision diagrams

14/1993 Drechsler, R. ; Sarabi, A. ; Theobald, M. ; Becker, B. ; Perkowski, M.A.:
Efficient repersentation and manipulation of switching functions based on ordered Kronecker functional decision diagrams

15/1993 Drechsler, Rolf ; Theobald, Michael ; Becker, Bernd:
Fast FDD based Minimization of Generalized Reed-Muller Forms

1/1994 Ferrandina, Fabrizio ; Meyer, Thorsten ; Zicari, Roberto:
Implementing lazy database updates for an object database system

2/1994 Liu, Ling ; Zicari, Roberto ; Hürsch, Walter ; Liebherr, Karl:
The Role of Polymorhic Reuse mechanism in Schema Evolution in an Object-oriented Database System

3/1994 Becker, Bernd ; Drechsler, Rolf ; Theobald, Michael:
Minimization of 2-level AND/XOR Expressions using Ordered Kronecker Functional Decision Diagrams

4/1994 Drechsler, R. ; Becker, B. ; Theobald, M. ; Sarabi, A. ; Perkowski, M.A.:
On the computational power of Ordered Kronecker Functional Decision Diagrams

5/1994 Even, Susan ; Sakkinen, Marku:
The safe use of polymorphism in the O2C database language

6/1994 GI/ITG-Workshop:
Anwendungen formaler Methoden im Systementwurf : 21. und 22. März 1994

7/1994 Zimmermann, M. ; Mönch, Ch. [u.a.]:
Die Telematik-Klassenbibliothek zur Programmierung verteilter Anwendungen in C++

8/1994 Zimmermann, M. ; Krause, G.:
Eine konstruktive Beschreibungsmethodik für verteilte Anwendungen

9/1994 Becker, Bernd ; Drechsler, Rolf:
How many Decomposition Types do we need ?

10/1994 Becker, Bernd ; Drechsler, Rolf:
Sympathy: Fast Exact Minimization of Fixed Polarity Reed-Muller Expression for Symmetric Functions

11/1994 Drechsler, Rolf ; Becker, Bernd ; Jahnke, Andrea:
On Variable Ordering and Decompostion Type Choice in OKFDDs

12/1994 Schmidt-Schauß:
Unification of Stratified Second-Order Terms

13/1994 Schmidt-Schauß:
An Algorithmen for Distributive Unification

14/1994 Becker, Bernd ; Drechsler, Rolf:
Synthesis for Testability: Circuit Derived from ordered Kronecker Functional Decision Diagrams

15/1994 Bär, Brigitte:
Konformität von Objekten in offenen verteilten Systemen

16/1994 Seidel, T. ; Puder, A. ; Geihs, K. ; Gründer, H.:
Global object space: Modell and Implementation

17/1994 Drechsler, Rolf ; Esbensen, Henrik ; Becker, Bernd:
Genetic algorithms in computer aided design of integrated circuits

1/1995 Schütz, Marko:
The $G^{\#}$-Machine: efficient strictness analysis in Haskell

2/1995 Henning, Susanne ; Becker, Bernd:
GAFAP: A Linear Time Scheduling Approach for High-Level-Synthesis

3/1995 Drechsler, Rolf ; Becker, Bernd ; Göckel, Nicole:
A Genetic Algorithm for variable Ordering of OBDDs

4/1995 Nebel, Markus E.:
Exchange Trees, eine Klasse Binärer Suchbäume mit Worst Case Höhe von $\log(n)$

5/1995 Drechsler, Rolf ; Becker, Bernd:
Dynamic Minimization of OKFDDs

6/1995 Breché, Philippe ; Ferrandina, Fabrizio ; Kuklok, Martin:
Simulation of Schema and Database Modification using Views

7/1995 Breché, Philippe ; Wörner, Martin:
Schema Update Primitives for ODB Design

8/1995 Schmidt-Schauß, Manfred:
On the Sematics and Interpretation of Rule Based Programs with Static Global Variables

9/1995 Rußmann, Arnd:
Adding Dynamic Actions to LL(k) Parsers

10/1995 Rußmann, Arnd:
Dynamic LL(k) Parsing

11/1995 Leyendecker, Thomas ; Oehler, Peter ; Waldschmidt, Klaus:
Spezifikation hybrider Systeme

12/1995 Cerone, Antonio ; Maggiolo-Schettini, Andrea:
Time-based Expressivity of Times Petri Nets

1/1996 Schütz, Marko ; Schmidt-Schauß, Manfred:
A Constructive Calculus Using Abstract Reduction for Context Analysis (nicht erschienen)

2/1996 Schmidt-Schauß, Manfred:
CPE: A Calculus for Proving Equivalence of Expressions in a Nonstrict Functional Language

1/1997 Kemp, Rainer:
On the Expected Number of Nodes at Level k in 0-balanced Trees

2/1997 Nebel, Markus:
New Results on the Stack Ramification of Binary Trees

3/1997 Nebel, Markus:
On the Average Complexity of the Membership Problem for a Generalized Dyck Language

4/1997 Liebehenschel, Jens:
Ranking and Unranking of Lexicographically Ordered Words: An Average-Case Analysis

5/1997 Kappes, Martin:
On the Generative Capacity of Bracketed Contextual Grammars

1/1998 Arlt, B. ; Brause, R.:
The Principal Independent Components of Images. *Elektronisch publiziert unter URL http://www.informatik.uni-frankfurt.de/fbreports/fbreport1-98.ps.gz*

2/1998 Miltrup, Matthias ; Schnitger, Georg:
Large Deviation Results for Quadratic Forms

3/1998 Miltrup, Matthias ; Schnitger, Georg:
Neural Networks and Efficient Associative Memory

4/1998 Kappes, Martin:
Multi-Bracketed Contextual Grammars

5/1998 Liebehenschel, Jens:
Lexicographical Generation of a Generalized Dyck Language

6/1998 Kemp, Rainer:
On the Joint Distribution of the Nodes in Uniform Multidimensional Binary Trees

7/1998 Liebehenschel, Jens:
Ranking and Unranking of a Generalized Dyck Language

8/1998 Grimm, Christoph ; Waldschmidt, Klaus:
Hybride Datenflußgraphen

9/1998 Kappes, Martin:
Multi-Bracketed Contextual Rewriting Grammars

1/1999 Kemp, Rainer:
On Leftist Simply Generated Trees

2/1999 Kemp, Rainer:
A One-to-one Correspondence Between a Class of Leftist Trees and Binary Trees

3/1999 Kappes, Martin:
Combining Contextual Grammars and Tree Adjoining Grammars

4/1999 Kappes, Martin:
Descriptional Complexity of Deterministic Finite Automata with Multiple Initial States

5/1999 Nebel, Markus E.:
New Knowledge on AVL-Trees

6/1999 Manfred Schmidt-Schauß, Marko Schütz (editors):
13^{th} International Workshop on Unification

7/1999 Brause, R.; Langsdorf, T.; Hepp, M.:
Credit Card Fraud Detection by Adaptive Neural Data Mining. *Elektronisch publiziert unter URL http://www.informatik.uni-frankfurt.de/fbreports/fbreport7-99.ps.gz*

8/1999 Kappes, Martin:
External Multi-Bracketed Contextual Grammars

9/1999 Priese, Claus P.:
A Flexible Type-Extensible Object-Relational DataBase Wrapper-Architecture

10/1999 Liebehenschel, Jens:
The Connection between Lexicographical Generation and Ranking

11/1999 Brause, R.; Arlt, B.; Tratar, E.:
A Scale-Invariant Object Recognition System for Content-based Queries in Image Databases. *Elektronisch publiziert unter URL http://www.informatik.uni-frankfurt.de/fbreports/fbreport11-99.ps.gz*

12/1999 Kappes, M.; Klemm, R. P.; Kintala, C. M. R.:
Determining Component-based Software System Reliability is Inherently Impossible

13/1999 Kappes, Martin:
Multi-Bracketed Contextual Rewriting Grammars With Obligatory Rewriting

14/1999 Kemp, Rainer:
On the Expected Number of Leftist Nodes in Simply Generated Trees

1/2000 Kemp, Rainer:
On the Average Shape of Dynamically Growing Trees

2/2000 Arlt, B.; Brause, R.; Tratar, E.:
MASCOT: A Mechanism for Attention-based Scale-invariant Object Recognition in Images. *Elektronisch publiziert unter URL http://www.cs.uni-frankfurt.de/fbreports/fbreport2-00.pdf*

3/2000 Heuschen, Frank; Waldschmidt, Klaus:
Bewertung analoger und digitaler Schaltungen der Signalverarbeitung

4/2000 Hamker, Fred H.; Paetz, Jürgen; Thöne, Sven; Brause, Rüdiger; Hanisch, Ernst:
Erkennung kritischer Zustände von Patienten mit der Diagnose „Septischer Schock" mit einem RBF-Netz. *Elektronisch publiziert unter URL http://www.cs.uni-frankfurt.de/fbreports/fbreport04-00.pdf*

1/2001 Nebel, Markus E.:
A Unified Approach to the Analysis of Horton-Strahler Parameters of Binary Tree Structures

2/2001 Nebel, Markus E.:
Combinatorial Properties of RNA Secondary Structures

3/2001 Nebel, Markus E.:
Investigation of the Bernoulli-Model for RNA Secondary Structures

4/2001 Malcher, Andreas:
Descriptional Complexity of Cellular Automata and Decidability Questions

1/2002 Paetz, Jürgen:
Durchschnittsbasierte Generalisierungsregeln; Teil I: Grundlagen

2/2002 Paetz, Jürgen; Brause, Rüdiger:
Durchschnittsbasierte Generalisierungsregeln Teil II: Analyse von Daten septischer Schock-Patienten

3/2002 Nießner, Frank:
Decomposition of Deterministic ω-regular Liveness Properties and Reduction of Corresponding Automata

4/2002 Kim, Pok-Son:
Das $\mathcal{RSV}$-Problem ist $\mathcal{NP}$-vollständig

5/2002 Nebel, Markus E.:
On a Statistical Filter for RNA Secondary Structures

6/2002 Malcher, Andreas:
Minimizing Finite Automata is Computationally Hard

1/2003 Malcher, Andreas:
On One-Way Cellular Automata with a Fixed Number of Cells

2/2003 Malcher, Andreas:
On Two-Way Communication in Cellular Automata with a Fixed Number of Cells

3/2003 Malcher, Andreas:
On the Descriptional Complexity of Iterative Arrays