Sublinearly Space Bounded Iterative Arrays

Andreas Malcher; Carlo Mereghetti; Beatrice Palano

Nr. 1/07

Frankfurter Informatik-Berichte

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

ISSN 1616–9107

[illegible]

Sublinearly Space Bounded Iterative Arrays

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

Carlo Mereghetti
Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano
via Comelico 39/41, 20135 Milano, Italy
E-Mail: mereghetti@dsi.unimi.it

Beatrice Palano
Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano
via Comelico 39/41, 20135 Milano, Italy
E-Mail: palano@dsi.unimi.it

Abstract. Iterative arrays (IAs) are a parallel computational model with a sequential processing of the input. They are one-dimensional arrays of interacting identical deterministic finite automata. In this note, realtime-IAs with sublinear space bounds are used to accept formal languages. The existence of a proper hierarchy of space complexity classes between logarithmic and linear space bounds is proved. Furthermore, an optimal space lower bound for non-regular language recognition is shown.

Key words: Iterative arrays, cellular automata, space bounded computations, decidability questions, formal languages, theory of computation

1 Introduction

Iterative arrays (IAs, for short) are computational devices consisting of an array of identical deterministic finite automata — called cells — which themselves are homogeneously interconnected with their both neighbors. An IA reads the input sequentially via a distinguished *communication cell.* The

*Corresponding author.

state of each cell is changed at discrete time steps by applying its transition function synchronously. Cole [2] was the first who studied formal language aspects of IAs. A survey on such aspects may be found in [4]. Some very recent results concern communication-restricted IAs [5, 12] and reversible IAs [6].

The *space* used by IAs considers, as a function of the input length, the number of cells activated along computations. In the general model, as many cells as the input is long may be used. Here, we consider realtime-IAs which are allowed to use only a *sublinear amount of space*.

As a main result, we exhibit an infinite proper hierarchy of classes of languages accepted between logarithmic and linear space bounds. For sublogarithmic space bounds, we prove that only regular languages can be accepted. Finally, some decidability questions on space bounded realtime-IAs are studied.

2 Definitions

We assume that the reader is familiar with the common notions of formal language and recursion theory (see, e.g., [3]).

Let Σ^* denote the set of all words over a finite alphabet Σ and $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$, with ε the empty word. The reversal of a word w is denoted by w^R. For the length of w, we write $|w|$. Set inclusion and strict set inclusion is denoted by $\subseteq$ and $\subset$, respectively. Let REG denote the family of regular languages. In this paper we do not distinguish whether a language L contains the empty string ε or not, i.e., we identify L with $L \setminus \{\varepsilon\}$. With log we denote the logarithm to the base 2.

Let S be a class of recursively enumerable languages. With a slight abuse of terminology, we say that a language L has the property S if L belongs to S. Given a Turing machine M, we denote by $T(M)$ the language accepted by M, and by $\langle M\rangle$ a suitable encoding of M. We set $L_S = \{\langle M\rangle \mid T(M) \in S\}$. If L_S is recursive (resp. recursively enumerable) we say the property S is decidable (resp. semidecidable).

Details and results on iterative arrays may be found, e.g., in [4]. An IA is depicted in the following figure.

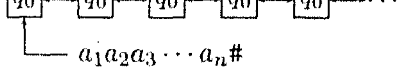

Figure 1: An iterative array.

It consists of a linear array of identical deterministic finite state automata called cells. At the beginning, all cells are in a designated quiescent state q_0. Each cell is connected with its left and right neighbor, except clearly the leftmost cell having only a right connection. The leftmost cell is the *communication cell,* which processes one input symbol at each time step. The local transition function is applied to each cell at the same time step. When the whole input is read, the end-of-input symbol # is processed.

In this paper, we are interested in *realtime-IAs.* Formally, a (deterministic) realtime-IA A is defined as

$$A = (Q, q_0, \#, \Sigma, \delta, \delta_0, F),$$

where $Q \neq \emptyset$ is the finite set of cell states, $q_0 \in Q$ is the quiescent (initial) state, $\# \notin \Sigma$ is the end-of-input symbol, Σ is the input alphabet, and $F \subseteq Q$ is the set of accepting cell states. A computational step is locally defined by the transition function $\delta : Q^3 \to Q$ for non-communication cells and $\delta_0 : Q^2 \times (\Sigma \cup \{\#\}) \to Q$ for the communication cell. The function δ satisfies: *(i)* $\delta(q_0, q_0, q_0) = q_0$, and *(ii)* $\delta(a, b, c) = q_0 \Rightarrow b = q_0$. Roughly speaking, condition *(i)* means that a cell may become *active* (i.e., assume a state in $Q \setminus \{q_0\}$) only if its left neighbor is already active, while condition *(ii)* states that, once active, a cell cannot enter q_0 any more. The function δ_0 satisfies condition *(ii)*, *mutatis mutandis.*

Let $a_1 \cdots a_n\#$ be the input word for A. A configuration of A at some time $t \geq 0$ is a pair (w_t, c_t), where $w_t \in \Sigma^*\# \cup \{\varepsilon\}$ is the non-read input sequence and $c_t : \mathbb{N} \to Q$ maps the single cells to their current states. For instance, $c_t(0)$ is the state of the communication cell at time t. The initial configuration (w_0, c_0) is defined by $w_0 = a_1 \cdots a_n\#$ and $c_0(i) = q_0$, for $i \in \mathbb{N}$. For $0 < t \leq n+1$, the global transition function Δ of A is induced by δ and δ_0 as $(w_t, c_t) = \Delta(w_{t-1}, c_{t-1})$, where:

$$c_t(0) = \delta_0(c_{t-1}(0), c_{t-1}(1), \sigma),$$

with

$$\sigma = \begin{cases} a_t & \text{if } t \leq n \\ \# & \text{if } t = n+1, \end{cases}$$

while, for $i \geq 1$

$$c_t(i) = \delta(c_{t-1}(i-1), c_{t-1}(i), c_{t-1}(i+1)),$$

and $\sigma w_t = w_{t-1}$.

The computation of A on input $a_1 \cdots a_n\#$ is the sequence $(w_0, c_0), \ldots, (w_{n+1}, c_{n+1})$ where for any $0 < t \leq n+1$, $(w_t, c_t) = \Delta(w_{t-1}, c_{t-1})$. The word $a_1 \cdots a_n \in \Sigma^*$ is *accepted* by A if and only if $c_{n+1}(0) \in F$.

It should be noted that the acceptance of a word is sometimes defined slightly different, i.e., a word is accepted if and only if there is a time step $0 \leq t \leq n+1$ at which the communication cell enters an accepting state. It is easy to see that for realtime-IAs both acceptance modes are equivalent.

The language accepted by A is the set $T(A) \subseteq \Sigma^*$ of all words accepted by A.

We denote by $\mathcal{L}_{rt}(\mathrm{IA})$ the class of languages accepted by realtime-IAs. It holds that $\mathrm{REG} \subset \mathcal{L}_{rt}(\mathrm{IA})$. Inclusion is trivial, yet it is well known that some (but not all) context-free and context-sensitive languages are accepted by realtime-IAs (see, e.g., [4] and the next section).

3 Space Bounded Iterative Arrays

In the general model, along their computations, IAs may use as many cells as the input length. It is natural to investigate a sublinear cells usage. In analogy with the Turing machine model, we call space the amount of cells used by an IA. Formally, the space used in the computation $(x\#, c_0), \ldots, (\varepsilon, c_{|x|+1})$ of a realtime-IA A on the word $x \in \Sigma^*$ is defined as

$$S(x) = \max\{i \in \mathbb{N} \mid c_{|x|+1}(i) \neq q_0\}.$$

The *strong space complexity* of A is the function $S : \mathbb{N} \to \mathbb{N}$ defined as

$$S(n) = \max\{S(x) \mid x \in \Sigma^*, |x| = n\}.$$

Hence, $S(n)$ counts the maximum number of cells activated during the computations on words of length n. It is easy to see that, for any realtime-IA, $S(n) \leq n+1$. In this paper, we focus on *sublinearly* strongly space bounded realtime-IAs, i.e., with $S(n) = o(n)$. We denote by $\mathcal{L}_{rt}(S(n)\text{-IA})$ the class of languages accepted by $S(n)$ strongly space bounded realtime-IAs.

Let us start by investigating space requirements for recognizing certain languages. To simplify our constructions, we use a typical programming tool for IAs called *tracks*. Informally, each cell $c(i)$ of an IA A is divided into *subcells* $c(i,1), \ldots, c(i,k)$, for a constant k. The jth track of A is the sequence $c(0,j), c(1,j), \ldots$. Different tasks can be simultaneously carried on in the tracks. One may easily notice that the use of this trick does not affect our considerations on space.

Lemma 1 *The language $\{a^m b^m c^m \mid m \geq 1\}$ can be accepted by a realtime-IA in* $\log(n)$ *strong space.*

Proof: Realtime-IAs can implement binary counters by storing values in their cells. The information to be communicated among cells are carry-overs and the position of the most significant bit of the counter. In [5], a detailed construction for communication restricted[1] realtime-IAs is given which works also for general realtime-IAs. It may also be observed that the number of cells needed to count the natural number n is $\lfloor \log(n) + 1 \rfloor$. By grouping the first two cells into the communication cell, we obtain $\log(n)$ as space upper bound.

Now, consider an IA which has three tracks. In the first two tracks, binary counters are implemented according to the above construction. While reading a's, both counters are incremented by one for each a. When reading b's, the counter in the first track is decremented by one for each b. Finally, when reading c's, the counter in the second track is decremented by one for each c. The correct format of the input, i.e., $a^+b^+c^+$, can be checked in the last track by the communication cell. If both counters have been decremented to zero at that moment when the end-of-input symbol is read, then the input is accepted, otherwise it is rejected. This realtime-IA is easily seen to be $\log(n)$ strongly space bounded. □

Lemma 2 *The language $\{a^{m^k} \mid m \geq 1\}$, for fixed $k \geq 2$, can be accepted by a realtime-IA in $\sqrt[k]{n}$ strong space.*

Proof: We consider the construction of signals of ratio m^k presented in [8]. This construction works in real time and needs a cellular device with two-way communication. Therefore, it can be implemented by a realtime-IA A. On the word a^n, the communication cell of A enters an accepting state whenever a designated state is obtained which identifies time steps of the form p^k, for $1 \leq p \leq \lfloor \sqrt[k]{n} \rfloor$. Moreover, it can be observed that at most $\lfloor \sqrt[k]{n} + 1 \rfloor$ cells are used. Again, by grouping the first two cells into the communication cell, we obtain a realtime-IA in $\sqrt[k]{n}$ strong space accepting $\{a^{m^k} \mid m \geq 1\}$. □

Lemma 3 *The language $P = \{wcw^R \mid w \in \{a,b\}^*\}$ can be accepted by a realtime-IA in linear strong space.*

[1]Roughly speaking, in the communication restricted model [5, 12], cells can exchange only "one bit of information."

Proof: In [1], a construction is presented which shows how realtime-IAs can simulate a stack. The correct format of the input, i.e., $\{a, b\}^*c\{a, b\}^*$ can be checked in the communication cell via an additional track. Thus, we may assume that the input has the form wcx with $w, x \in \{a, b\}^*$. Now, w is read and each symbol is pushed into the cells in a stack-like manner. Having read the separating symbol c, in every subsequent time step one symbol is popped from the cells and matched against the input x. Finally, when the input is read and all symbols have been matched correctly, the word is accepted, otherwise it is rejected. It can be observed from the construction in [1] that at most $|w|$ cells are used for an input wcx, with $w, x \in \{a, b\}^*$. Hence, a linear amount of space is sufficient. □

We will see in Lemma 6, that a linear amount of space is also necessary for recognizing the language P.

4 Proper Space Hierarchy

Let us now start to build a proper hierarchy for sublogarithmically space bounded realtime-IAs. The first level is stated in the following

Lemma 4 REG $\subset \mathcal{L}_{rt}(\log(n)\text{-IA})$.

Proof: Lemma 1 gives an example of a non-regular language accepted by a realtime-IA in $\log(n)$ strong space. □

To investigate higher levels of the hierarchy, we introduce some terminology. Given a configuration (w_t, c_t) of a realtime-IA, we call *array state* either the sequence $c_t(0), c_t(1), \ldots, c_t(k)$ of the states of all active cells at time t (i.e., with $k \in \mathbb{N}$ satisfying $c_t(k) \neq q_0$ and $c_t(k+1) = q_0$), or the empty sequence representing $c_t(i) = q_0$, for $i \in \mathbb{N}$.

Lemma 5 *For* $k \geq 2$, $\mathcal{L}_{rt}(\sqrt[k]{n}\text{-IA}) \subset \mathcal{L}_{rt}(\text{IA})$.

Proof: By Lemma 3, the language $P = \{wcw^R \mid w \in \{a, b\}^*\}$ can be accepted by a realtime-IA. Now, assume by contradiction that P is accepted by a realtime-IA A with state set Q in $\sqrt[k]{n}$ strong space. Consider A having the word wcx, with $|w| = |x|$, as input. It is clear that, upon reading the symbol c, A must have at least $2^{|w|}$ different array states. Otherwise, A would wrongly accept a word wcx such that $x \neq w^R$. On the other hand, the maximum number of different array states which can be obtained on an input of length $2|w|+1$ is bounded by $|Q|^{\sqrt[k]{2|w|+1}}$. Then, we obtain the inequality

$$|Q|^{\sqrt[k]{2|w|+1}} \geq 2^{|w|},$$

which is equivalent to

$$(2|w|+1)(\log(|Q|))^k \geq |w|^k.$$

This is a contradiction for $|w|$ being sufficiently large. $\square$

More generally, one may show

Lemma 6 *For any* $S(n) = o(n)$, $\mathcal{L}_{rt}(S(n)\text{-IA}) \subset \mathcal{L}_{rt}(\text{IA})$.

Proof: By the same technique used to prove Lemma 5, we get $P \notin \mathcal{L}_{rt}(S(n)\text{-IA})$, for $S(n) = o(n)$. $\square$

Lemma 7 *For* $k \geq 2$, $\mathcal{L}_{rt}(\sqrt[k+1]{n}\text{-IA}) \subset \mathcal{L}_{rt}(\sqrt[k]{n}\text{-IA})$.

Proof: We consider the language

$$P' = \{c^{(|w|+1)^k - 2|w| - 1} wcw^R \mid w \in \{a,b\}^*\},$$

and first show that P' can be accepted by a realtime-IA A in $\sqrt[k]{n}$ strong space. A has four tracks. In the first track, the communication cell checks whether the input has the correct format, i.e., $c^*\{a,b\}^*c\{a,b\}^*$. In the second track, we implement the recognition of the language $\{c^{m^k} \mid m \geq 1\}$ according to the construction of Lemma 2. Notice that words in P' have length $(|w|+1)^k$. Any time a number of initial c's in the form p^k, for $p \geq 1$, is counted, the communication cell of the second track enters an accepting state and a special symbol is pushed into the third track which is organized as a stack. We notice that, for words in P', the stack height is $|w|$ at the end of initial c's processing (we will show this fact later). Then, when the first symbol from $\{a,b\}$ is read: in the fourth track, we start the recognition of the language $P = \{wcw^R \mid w \in \{a,b\}^*\}$ as in Lemma 3; additionally, in the third track, one symbol per each a or b read is popped from the stack up to the next c. At the end of the parsing of a word in P': we enter a certain state in the first track; an accepting state is entered both in the second and fourth track, and the stack in the third track is empty up to that symbol which has just arisen from the second track. Thus, we can accept upon reading the end-of-input symbol. In all other cases, one or more tracks reveal a failure and the word is rejected.

It remains to be shown that, having processed the initial c's of words in P', the height of the stack implemented in the third track is $|w|$. It's enough to see that $(|w|+1)^k - 2|w| - 1 \geq |w|^k$. In fact, for $k \geq 2$:

$$(|w|+1)^k - 2|w| - 1 =$$
$$\sum_{i=0}^{k-2} \binom{k}{i} |w|^{k-i} + (k-2)|w| \geq |w|^k.$$

Since A must accept in a strong acceptance mode, we have to make sure that A uses at most $\sqrt[k]{|z|}$ cells on input z, disregarding acceptance or rejection. To this aim, we modify A such that the input is rejected whenever the third or fourth track wants to use a cell which has not been already active in the second track. Let us show that this new algorithm is correct and takes place in $\sqrt[k]{|z|}$ strong space. To this aim, it is enough to show that: *(i)* the second track never uses more than $\sqrt[k]{|z|}$ cells, and that *(ii)* $z \in P'$ implies that, along the computation on z, the third and fourth track never exceed second track's space usage. The first property is ensured by Lemma 2. Concerning the second property, we notice that the third and fourth tracks essentially contain stacks of height $|w|$. Yet, we can concentrate only on the third track, since the fourth track grows only after the initial c's have been read. Let t be a time step during the processing of the initial c's, i.e., $1 \leq t \leq (|w|+1)^k - 2|w| - 1$. At this time, at least $\lfloor \sqrt[k]{t} \rfloor$ cells are active in the second track, while p cells are active in the third track, with p satisfying $p^k \leq t < (p+1)^k$. Clearly, we get $\lfloor \sqrt[k]{t} \rfloor \geq p$. Hence, we conclude that P' can be accepted by a realtime-IA in $\sqrt[k]{n}$ strong space.

Next, we show that P' cannot be accepted by a realtime-IA in $\sqrt[k+1]{n}$ strong space. By contradiction, assume that P' is accepted by such an IA A' with state set Q. As in the proof of Lemma 5, on input a word $c^{(|w|+1)^k - 2|w| - 1} wcx$, with $|w| = |x|$, A' must be able to represent at least $2^{|w|}$ different array states upon reading the symbol c between w and x. The maximum number of different array states of A' on an input of length $(|w|+1)^k$ is bounded by $|Q|^{\sqrt[k+1]{(|w|+1)^k}}$. Then, we must require $|Q|^{\sqrt[k+1]{(|w|+1)^k}} \geq 2^{|w|}$ or, equivalently,

$$(|w|+1)^k (\log(|Q|))^{k+1} \geq |w|^{k+1},$$

which can also be written as

$$\sum_{i=0}^{k} \binom{k}{i} |w|^{k-i} (\log(|Q|))^{k+1} \geq |w|^{k+1}.$$

Dividing by $|w|^{k+1} > 0$, we get

$$\sum_{i=0}^{k} \binom{k}{i} |w|^{-i-1} (\log(|Q|))^{k+1} \geq 1.$$

This is clearly a contradiction since, for any $0 \leq i \leq k$, we have that $|w|^{-i-1}$ converges to 0 as $|w|$ grows. □

With similar considerations we obtain

Corollary: For $k \geq 2$, $\mathcal{L}_{rt}(\log(n)\text{-IA}) \subset \mathcal{L}_{rt}(\sqrt[k]{n}\text{-IA})$.

In conclusion, we are able to exhibit the claimed infinite space hierarchy:

Theorem 1 *For $k \geq 2$,*

- $\text{REG} \subset \mathcal{L}_{rt}(\log(n)\text{-IA})$,
- $\mathcal{L}_{rt}(\log(n)\text{-IA}) \subset \mathcal{L}_{rt}(\sqrt[k+1]{n}\text{-IA})$,
- $\mathcal{L}_{rt}(\sqrt[k+1]{n}\text{-IA}) \subset \mathcal{L}_{rt}(\sqrt[k]{n}\text{-IA})$,
- $\mathcal{L}_{rt}(\sqrt[k]{n}\text{-IA}) \subset \mathcal{L}_{rt}(\text{IA})$.

5 Some Decidability Questions

Some constructions used in the previous section are useful to tackle decidability questions on space bounded realtime-IAs. Let us begin with the following

Lemma 8 *Let A be a realtime-IA. A realtime-IA A' working in* $\log(n)$ *strong space can be effectively constructed such that $T(A') = \{c^{2^{|w|}-1} w c^{|w|} \mid w \in T(A)\}$, with c being a new alphabet symbol.*

Proof: We construct an IA A' with two tracks. The first track implements a binary counter which is increased while reading the initial c's. After reading $2^{|w|} - 1$ c's, $|w|$ cells are activated to store the binary number $1^{|w|}$. When reading the first non-c: *(i)* in the second track, the computation of A on word w is simulated; *(ii)* in the first track, a signal from left to right is sent checking whether all cells contain a 1. If this signal reaches the cell which contains the most significant bit, another signal from right to left is started. If this new signal has arrived at the communication cell upon reading the end-of-input symbol, and w has been accepted on the second track, then A'

accepts the input. Otherwise, the input is rejected. It can be observed that on any accepting or non-accepting computation, at most as many cells as the length of the binary counter are used. Thus, A' is a realtime-IA working in $\log(n)$ strong space. □

It is shown in [7] that almost all decidability questions for realtime-IAs are undecidable and not semidecidable. The same results can be extended to the restricted model of realtime-IAs working in $\log(n)$ strong space.

Theorem 2 *Emptiness, finiteness, infiniteness, universality, equivalence, inclusion, regularity, and context-freedom are not semidecidable for realtime-IAs working in* $\log(n)$ *strong space.*

Proof: Assume, by contradiction, that emptiness is semidecidable for realtime-IAs $\log(n)$ strongly space bounded. Let A be a realtime-IA and construct a realtime-IA A' $\log(n)$ strongly space bounded as in Lemma 8. Clearly, $T(A)$ is empty if and only if $T(A')$ is empty. This would show that emptiness is semidecidable for realtime-IAs, which contradicts [7]. With the same argument, we can show the non-semidecidability of finiteness and infiniteness. By using the track programming tool, it is not hard to prove that realtime-IAs working in $\log(n)$ strong space are closed under *intersection, union and complementation.* So, universality is also not semidecidable. Now, consider two realtime-IAs A and B working in $\log(n)$ strong space. Since $T(A) \subseteq T(B)$ if and only if $T(A) \cap T(B)^c = \emptyset$, we obtain the non-semidecidability of inclusion and, hence, equivalence. Finally, consider the language $T(A') \cdot \{a^m b^m c^m \mid m \geq 1\}$, with $T(A')$ defined as in Lemma 8 and some new alphabet symbols a, b, c. According to Lemma 1 and Lemma 8, such a language is easily seen to be accepted by a realtime-IA A'' in $\log(n)$ strong space. Since $T(A'')$ is regular (context-free) if and only if $T(A')$ is empty, we obtain the non-semidecidability of regularity (context-freedom) for $\log(n)$ strongly space bounded realtime-IAs. □

6 Lower Bounds for Recognizing Non-Regular Languages

According to Theorem 2, restricting to a logarithmic cell usage still leads to non-semidecidable questions. So, as a further restriction, we could consider realtime-IAs which are allowed to use only a sublogarithmic number of cells. Notice that an analogous investigation has been carried on even for space bounded Turing machines (see, e.g., [9, 10] for a survey on the sublogarithmic space world).

The next theorem shows that sublogarithmic space bounds reduce the computational capacity of realtime-IAs to the regular languages.

Theorem 3 *Let A be a realtime-IA $S(n)$ strongly space bounded. Then, either $S(n) \geq c \cdot \log(n)$, for some constant $c > 0$ and infinitely many n, or A accepts a regular language and the space used by A is bounded by a constant.*

Proof: We adapt the "cut-and-glue" technique in [11] to space bounded realtime-IAs. Let $L = T(A)$ and let $L[k]$ be the set of words in L requiring exactly k cells. We notice that $L[k] \neq \emptyset$ for infinitely many k. Otherwise, the space used by A would be bounded by a constant and L would be regular, as one may easily verify. Fix a k such that $L[k] \neq \emptyset$, and let $w = a_1 a_2 \cdots a_n$ be one of the shortest words in $L[k]$. Consider now the computation $(w_0, c_0), \ldots, (w_{n+1}, c_{n+1})$ of A on the input word $w\#$. Since $w \in L[k]$, the array state c_{n+1} consists of exactly k cells. Now, suppose there exist $0 \leq i < j \leq n+1$ satisfying $c_i = c_j$. Then, the sequence $(w_0, c_0), \ldots, (w_i, c_i), (w_{j+1}, c_{j+1}), \ldots, (w_{n+1}, c_{n+1})$ would be an accepting computation for the input word $a_1 \cdots a_i a_{j+1} \cdots a_n\#$, or for $a_1 \cdots a_i\#$ whenever $j = n$. Moreover, we observe that such a computation uses exactly k cells, witnessing membership of $a_1 \cdots a_i a_{j+1} \cdots a_n$ (or $a_1 \cdots a_i$) in $L[k]$. This clearly contradicts the fact that n is the length of the shortest words in $L[k]$. Thus, there must exist at least $n+2$ pairwise different array states consisting of k cells, and this for infinitely many k. Hence, we get $n + 2 \leq |Q|^k$, where Q is the state set of A. Taking logarithms, we obtain $k \geq c \cdot \log(n)$, for some positive constant c. This shows the claim. □

We conclude by observing that the logarithmic space lower bound for non-regular language acceptance in Theorem 3 is *optimal.* It is enough, in fact, to consider Lemma 1 where a non-regular language is shown to be accepted by a realtime-IA in $\log(n)$ strong space.

References

[1] Buchholz, T., Kutrib, M.: Some relations between massively parallel arrays. Parallel Comput. **23** (1997) 1643–1662.

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

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

[4] Kutrib, M.: Automata arrays and context-free languages. In: Where Mathematics, Computer Science and Biology Meet, Kluwer Academic Publishers (2001) 139–148.

[5] Kutrib, M., Malcher, A.: Fast iterative arrays with restricted intercell communication: constructions and decidability. In: Královič, R., Urzyczyn, P. (Eds.): Mathematical Foundations of Computer Science (MFCS 2006), LNCS 4162, Springer-Verlag (2006) 634–645.

[6] Kutrib, M., Malcher, A.: Real-time reversible iterative arrays. In: Csuhaj-Varjú, E., Ésik, Z. (Eds.): Fundamentals of Computation Theory (FCT 2007), LNCS 4639, Springer-Verlag (2007) 376–387.

[7] Malcher, A.: On the descriptional complexity of iterative arrays. IEICE Trans. Inf. Sci. **E87-D** (2004) 721–725.

[8] Mazoyer, J., Terrier, V.: Signals in one-dimensional cellular automata. Theoret. Comp. Sci. **217** (1999) 53–80.

[9] Mereghetti, C.: The descriptional power of sublogarithmic resource bounded Turing machines. In: Geffert, V., Pighizzini, G. (Eds.): Descriptional Complexity of Formal Systems (DCFS 2007), Univerzity P. J. Šafárik, Košice, Slovakia (2007) 12-26.

[10] Szepietowski, A.: Turing machines with sublogarithmic space. LNCS 843, Springer-Verlag, Berlin, 1994.

[11] Stearns, R.E., Hartmanis, J., Lewis II, P.M.: Hierarchies of memory limited computations. In: IEEE Conf. Record. on Switching Circuit Theory and Logical Design (1965) 179–190.

[12] Umeo, H., Kamikawa, N.: Real-time generation of primes by a 1-bit communication cellular automaton. Fundam. Inform. **58** (2003) 421–435.

Interne Berichte am Institut für 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):
13th 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

4/2003 Kemp, Rainer:
On the Expected Number of Leftist Nodes in Dynamically Growing Trees

5/2003 Nebel, Markus E.:
Identifying Good Predictions of RNA Secondary Structure

1/2004 Meise, Christian:
Zwischenbericht zum Projekt BeCom

2/2004 Malcher, Andreas:
On Non-Recursive Trade-Offs Between Finite-Turn Pushdown Automata

3/2004 Kappes, Martin; Nießner, Frank:
Reliability and Conciseness of ω-Language Representations by Finite ω-Automata

1/2005 Paetz, Jürgen:
Anwendungs- und Optimierungspotenzial eines Neuro-Fuzzy-Systems im Wirkstoffdesign

2/2005 Metzler, Dirk:
Robust E-Values for Gapped Local Alignments

3/2005 Weinard, Maik:
Deciding the FIFO Stability of Networks in Polynomial Time

4/2005 Abawi, Daniel F.:
Analyse und Bewertung von Erstellungssystemen für Augmented Reality-Anwendungen

5/2005 Abawi, Daniel F.:
Die Verdeckungsdarstellungsproblematik bei Augmented Reality-Anwendungen

1/2006 Metzler, Dirk; Nebel, Markus E.:
Predicting RNA Secondary Structures with Pseudoknots by MCMC Sampling

1/2007 Malcher, Andreas; Mereghetti, Carlo; Palano, Beatrice:
Sublinearly Space Bounded Iterative Arrays