I F I G

RESEARCH

REPORT

INSTITUT FÜR INFORMATIK

A TIME HIERARCHY FOR BOUNDED ONE-WAY CELLULAR AUTOMATA

Andreas Klein Martin Kutrib

IFIG RESEARCH REPORT 0101

JANUARY 2001

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 0101, January 2001

A Time Hierarchy for Bounded One-Way Cellular Automata

Andreas Klein Martin Kutrib[1]

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

Abstract. Space-bounded one-way cellular language acceptors (OCA) are investigated. The only inclusion known to be strict in their time hierarchy from real-time to exponential-time is between real-time and linear-time! We show the surprising result that there exists an infinite hierarchy of properly included OCA-language families in that range. A generalization of a method of Terrier is shown that provides a tool for proving that languages are not acceptable by OCAs with small time bounds. By such a language and a translation result the hierarchies are established.

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

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

1 Introduction

Linear arrays of interacting finite automata are models for massively parallel language acceptors. Their advantages are simplicity and uniformity. It has turned out that a large array of not very powerful processing elements operating in parallel can be programmed to be very powerful.

One type of system is of particular interest: the cellular automata whose homogeneously interconnected deterministic finite automata (the cells) work synchronously at discrete time steps obeying one common transition function. Here we are interested in a very simple type of cellular automata. The arrays are real-space bounded, i.e., the number of cells is bounded by the number of input symbols, and each cell is connected to its immediate neighbor to the right only. Due to the resulting information flow from right to left such devices are called one-way cellular automata (OCA). If the cells are connected to their both immediate neighbors the information flow becomes two-way and the device is a (two-way) cellular automaton (CA).

Although parallel language recognition by (O)CAs has been studied for more than a quarter of a century some important questions are still open. In particular, only little is known of proper inclusions in the time hierarchy. Most of the early languages known not to be real-time but linear-time OCA-languages are due to the fact that every unary real-time OCA-language is regular [4]. In [9] and [10] a method has been shown that allows proofs of non-acceptance for non-unary languages in real-time OCAs. Utilizing these ideas the non-closure of real-time OCA-languages under concatenation could be shown.

Since for separating the complexity classes in question there are no other general algebraic methods available, specific languages as potential candidates are of particular interest. In [5] several positive results have been presented. Surprisingly, so far there was only one inclusion known to be strict in the time hierarchy from real-time to exponential-time. It is the inclusion between real-time and linear-time languages. In [2] the existence of a non-real-time OCA-language that is acceptable in $n + \log(n)$-time has been proved yielding a lower upper bound for the strict inclusion. Another valuable tool for exploring the OCA time hierarchy is the possible linear speed-up [7] from $n + r(n)$ to $n + \varepsilon \cdot r(n)$ for $\varepsilon > 0$.

The main contribution of the present paper is to show that there exists an infinite time hierarchy of properly included language families. These families are located in the range between real-time and linear-time. The surprising result covers the lower part of the time hierarchy in detail.

The paper is organized as follows: In Section 2 we define the basic notions and the model in question. Since for almost all infinite hierarchies in complexity theory the constructibility of the bounding functions is indispensable, in Section 3 we present our notion of constructibility in OCAs and prove that it covers a wider range of functions than the usual approach. Section 4 is devoted to a generalization of the method in [10] to time complexities beyond real-time. This key tool is utilized to obtain a certain language not acceptable with a given time bound. Finally, in Section 5 the corresponding proper inclusion is extended to an infinite time hierarchy by translation arguments.

2 Basic notions

We denote 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 λ and the reversal of a word w by w^R. For the length of w we write $|w|$. 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 $f^{[i]}$, $i \in \mathbb{N}$. If f is increasing then its inverse is defined according to $f^{-1}(n) = \min\{m \in \mathbb{N} \mid f(m) \geq n\}$. As usual we define the set of functions that grow strictly less than f by $o(f) = \{g : \mathbb{N}_0 \to \mathbb{N} \mid \lim_{n\to\infty} \frac{g(n)}{f(n)} = 0\}$. In terms of orders of magnitude f is an upper bound of the set $O(f) = \{g : \mathbb{N}_0 \to \mathbb{N} \mid \exists\, n_0, c \in \mathbb{N} : \forall\, n \geq n_0 : g(n) \leq c \cdot f(n)\}$. Conversely, f is a lower bound of the set $\Omega(f) = \{g : \mathbb{N}_0 \to \mathbb{N} \mid f \in O(g)\}$.

A one-way resp. two-way cellular array is a linear array of identical deterministic finite state machines, sometimes called cells, which are connected to their nearest neighbor to the right resp. to their both nearest neighbors. The array is bounded by cells in a distinguished so-called boundary state. For convenience we identify the cells by positive integers. The state transition depends on the current state of each cell and the current state(s) of its neighbor(s). The transition function is applied to all cells synchronously at discrete time steps. Formally:

Definition 1 *A* one-way cellular automaton (OCA) *is a system* $\langle S, \delta, \#, A, F\rangle$, *where*

1. S *is the finite, nonempty set of* cell states,
2. $\# \notin S$ *is the* boundary state,
3. $A \subseteq S$ *is the nonempty set of* input symbols,
4. $F \subseteq S$ *is the set of* accepting (or final) states, *and*
5. $\delta : (S \cup \{\#\})^2 \to S$ *is the* local transition function.

If the flow of information is extended to two-way the resulting device is a *(two-way) cellular array* (CA) and the local transition function maps from $(S\cup\{\#\})^3$ to S.

A *configuration* of a cellular automaton at some time $t \geq 0$ is a description of its global state, which is actually a mapping $c_t : [1, \ldots, n] \to S$ for $n \in \mathbb{N}$.

The configuration at time 0 is defined by the initial sequence of states. For a given input $w = a_1 \cdots a_n \in A^+$ we set $c_{0,w}(i) = a_i$ for $1 \leq i \leq n$. 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 for $t \geq 0$ be a configuration, then its successor configuration is as follows:

$$\begin{aligned} c_{t+1} = \Delta(c_t) \iff & \\ & c_{t+1}(1) = \delta\big(\#, c_t(1), c_t(2)\big) \\ & c_{t+1}(i) = \delta\big(c_t(i-1), c_t(i), c_t(i+1)\big), i \in \{2, \dots, n-1\} \\ & c_{t+1}(n) = \delta\big(c_t(n-1), c_t(n), \#\big) \end{aligned}$$

for CAs and correspondingly for OCAs. Thus, Δ is induced by δ.

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

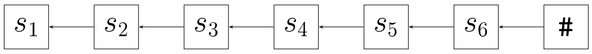

Figure 1: A one-way cellular automaton.

An input w is accepted by an (O)CA if at some time i during its course of computation the leftmost cell enters an accepting state.

Definition 2 *Let $\mathcal{M} = \langle S, \delta, \#, A, F\rangle$ be an* (O)CA.

1. *An input $w \in A^+$ is* accepted *by $\mathcal{M}$ if there exists a time step $i \in \mathbb{N}$ such that $c_i(1) \in F$ holds for the configuration $c_i = \Delta^{[i]}(c_{0,w})$.*
2. *$L(\mathcal{M}) = \{w \in A^+ \mid w$ is accepted by $\mathcal{M}\}$ is the* language accepted *by $\mathcal{M}$.*
3. *Let $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$, be a mapping. If all $w \in L(\mathcal{M})$ can be accepted with at most $t(|w|)$ time steps, then L is said to be of time complexity t.*

The family of all languages that are acceptable by some OCA (CA) with time complexity t is denoted by $\mathscr{L}_t(\mathrm{OCA})$ ($\mathscr{L}_t(\mathrm{CA})$). If t equals the identity function $id(n) = n$, acceptance is said to be in *real-time*, and if t is equal to $k \cdot id$ for an arbitrary rational number $k > 1$, then acceptance is carried out in *linear-time*. Correspondingly, we write $\mathscr{L}_{rt}((\mathrm{O})\mathrm{CA})$ and $\mathscr{L}_{lt}((\mathrm{O})\mathrm{CA})$.

In this article we prove:

Theorem 3 *Let $r_1, r_2 : \mathbb{N} \to \mathbb{N}$ be two functions such that $r_2 \cdot \log^2(r_2) \in o(r_1)$ and r_1^{-1} is constructible, then*

$$\mathscr{L}_{n+r_2(n)}(\mathrm{OCA}) \subset \mathscr{L}_{n+r_1(n)}(\mathrm{OCA})$$

Example 4 Let $0 \leq p < q \leq 1$ be two rational numbers. Clearly, $n^p \cdot \log^2(n^p)$ is of order $o(n^q)$. In the next section the constructibility of the inverse of n^q will be established. Thus, an application of Theorem 3 yields the strict inclusion

$$\mathscr{L}_{n+n^p}(\mathrm{OCA}) \subset \mathscr{L}_{n+n^q}(\mathrm{OCA})$$

3 Constructible Functions

For the proof of Theorem 3 it will be necessary to control the lengths of words with respect to some internal substructures. The following notion of constructibility expresses the idea that the length of a word relative to the length of a subword should be computable.

Definition 5 *A function $f : \mathbb{N} \to \mathbb{N}$ is* constructible *if there exists an λ-free homomorphism h and a language $L \in \mathscr{L}_{rt}(\mathrm{OCA})$ such that*

$$h(L) = \{a^{f(n)-n}b^n \mid n \in \mathbb{N}\}$$

Since constructible functions describe the length of the whole word dependent on the length of a subword it is obvious that each constructible function must be greater than or equal to the identity. At a first glance this notion of constructibility might look somehow unusual or restrictive. But λ-free homomorphisms are very powerful so the family of (in this sense) constructible functions is very rich, and is, in fact, a generalization of the usual notion. The remainder of this section is devoted to clarify the presented notion and its power.

The next lemma states that we can restrict our considerations to length preserving homomorphisms. The advantage is that for length preserving homomorphisms each word in L is known to be of length $f(m)$ for some $m \in \mathbb{N}$.

Lemma 6 *Let $f : \mathbb{N} \to \mathbb{N}$ be a constructible function. Then there exists a length preserving λ-free homomorphism h and a language $L \in \mathscr{L}_{rt}(\mathrm{OCA})$ such that*

$$h(L) = \{a^{f(n)-n}b^n \mid n \in \mathbb{N}\}$$

The proof follows immediately from a proof in [1] where the closure of $\mathscr{L}_{rt}(\mathrm{OCA})$ under λ-free homomorphisms is characterized by OCAs with limited nondeterminism.

Given an increasing constructible function $f : \mathbb{N} \to \mathbb{N}$ and a language $L_a \subseteq A^+$ acceptable by some OCA with time complexity $n + r(n)$, where $r : \mathbb{N} \to \mathbb{N}$, we now define a language that plays an important role in the sequel. Let the language $L_f \subseteq B^+$ be a witness for the constructibility of f, i.e., $L_f \in \mathscr{L}_{rt}(\mathrm{OCA})$ and $h(L_f) = \{a^{f(n)-n}b^n \mid n \in \mathbb{N}\}$ for a length preserving λ-free homomorphism h. The language $L_1(L_a, L_f) \subseteq (A \cup \{\sqcup\}) \times B)^+$ is constructed as follows

1. The second component of each word w in $L_1(L_a, L_f)$ is a word of L_f that implies that w is of length $f(m)$ for some $m \in \mathbb{N}$.
2. The first component of w contains exactly $f(m) - m$ blank symbols and m non-blank symbols.
3. The non-blank symbols in the first component of w form a word in L_a.

The following proposition is used in later sections. Besides, it is an example that demonstrates how to use constructible functions. In Lemma 13 we will prove that the shown bound for the time complexity of L_1 is minimal.

Proposition 7 *The language $L_1(L_a, L_f)$ is acceptable by some* OCA *with time complexity $n + r(f^{-1}(n))$.*

Proof. We construct an OCA $\mathcal{A}$ with three registers that accepts L_1 obeying the time complexity $n + r(f^{-1}(n))$.

In its first register $\mathcal{A}$ verifies that the second component of each word in L_1 is a word of L_f. By definition of L_f this can be done in real-time.

In its second register $\mathcal{A}$ checks that the first component of L_1 contains exactly $f(m) - m$ blank symbols. Because it can be verified that the second component of L_1 belongs to L_f, we know that the first $f(m) - m$ symbols of the second component are mapped to a's and the last m symbols of the second component are mapped to b's. The task is to check that the number of a's in the second component is equal to the number of blank symbols in the first component. Therefore, $\mathcal{A}$ shifts the blank symbols from right to left. Each symbol a in the second component consumes one blank symbol. A signal that goes from the right to the left with full speed can check that no blank symbol has reached the leftmost cell and that each letter a has consumed one blank symbol, i.e., that the number of a's is equal to the number of blank symbols. The test can be done in real-time.

In order to verify that the non-blank symbols in the first component form a word of L_a the automaton $\mathcal{A}$ simulates the OCA that accepts L_a. But for every blank-symbol $\mathcal{A}$ has to be delayed for one time step, until it receives the necessary information for the next simulation step. Therefore, $\mathcal{A}$ needs $m + r(m) + (f(m) - m)$ steps for the simulation ($m + r(m)$ time steps for the simulation itself and $f(m) - m$ time steps delaying time). Substituting $m = f^{-1}(n)$ completes the proof. □

Now we prove that the family of constructible functions is very rich. In particular, all Fischer-constructible functions are constructible in the sense of Definition 5. A function f is said to be *Fischer-constructible* if there exists an unbounded two-way CA such that the initially leftmost cell enters a final state at time $i \in \mathbb{N}$ if and only if $i = f(m)$ for some $m \in \mathbb{N}$. Thus, the Fischer-constructibility is an important notion that meets the intuition of constructible functions. For a detailed study of these functions see [8] where also the name has been introduced according to the author of [6].

For example, n^k for $k \in \mathbb{N}$, 2^n, $n!$, and p_n, where p_n is the nth prime number, are Fischer-constructible. Moreover, the class is closed under several operations.

Lemma 8 *If a function $f : \mathbb{N} \to \mathbb{N}$ is Fischer-constructible, then it is constructible in the sense of Definition 5.*

Proof. In [3] it is shown that for every language $L' \in \mathscr{L}_{lt}(\mathrm{CA})$ there exists a language $L \in \mathscr{L}_{rt}(\mathrm{OCA})$ and an λ-free homomorphism h such that $h(L) = L'$.

Therefore, it is sufficient to prove that for Fischer-constructible functions f the languages $\{a^{f(n)-n}b^n \mid n \in \mathbb{N}\}$ are linear-time CA-languages:

The initially leftmost cell of an appropriate CA starts the construction of f, i.e., it enters a final state exactly at the time steps $f(1)$, $f(2), \dots$ In addition, the rightmost cell sends initially a signal to the left that runs with full speed. The CA accepts a word of the form a^+b^+ if and only if this signal arrives at the leftmost cell at a time step $f(m)$ for some $m \in \mathbb{N}$ and the number of b's is equal to m. The details of the easy CA construction are straightforward. □

Without proof we mention that the class of constructible functions is closed under several operations such as addition, multiplication or composition.

4 Equivalence Classes

In order to prove lower bounds for the time complexity we generalize a lemma that gives a necessary condition for a language to be real-time acceptable by an OCA. At first we need the following definition:

Definition 9 *Let L be a language and X and Y be two sets of words. Two words w and w' are equivalent with respect to L, X and Y (in short (L, X, Y)-equivalent) if and only if $xwy \in L \iff xw'y \in L$ for all $x \in X$ and $y \in Y$.*

Let $L_d \subset \{0, 1, (,), |\}^+$ be a language whose words are of the form

$$x\,(\,x_1 \,|\, y_1\,) \cdots (\,x_n \,|\, y_n\,)\,y$$

where $x, x_i, y, y_i \in \{0,1\}^*$ for $1 \le i \le n$, and $(\,x \,|\, y\,) = (\,x_i \,|\, y_i\,)$ for at least one $i \in \{1, \dots, n\}$.

The language L_d can be thought as a dictionary. The task for the OCA is to check whether the pair $(\,x \,|\, y\,)$ appears in the dictionary or not.

Proposition 10 *Let $X = Y = \{0,1\}^*$. Two words $w = (\,x_1 \,|\, y_1\,) \cdots (\,x_n \,|\, y_n\,)$ and $w' = (\,x'_1 \,|\, y'_1\,) \cdots (\,x'_m \,|\, y'_m\,)$ are equivalent with respect to L_d, X and Y if and only if $\{(\,x_1 \,|\, y_1\,), \dots, (\,x_n \,|\, y_n\,)\} = \{(\,x'_1 \,|\, y'_1\,), \dots, (\,x'_m \,|\, y'_m\,)\}$.*

Proof. First assume that the two sets are equal. Let $x \in X$ and $y \in Y$, then $xwy \in L_d$ implies $(\,x \,|\, y\,) = (\,x_i \,|\, y_i\,)$ for some i. Since the two sets are equal we have $(\,x \,|\, y\,) = (\,x'_j \,|\, y'_j\,)$ for some j. Therefore, $xwy \in L_d$ implies $xw'y \in L_d$ and vice versa, i.e., w and w' are (L_d, X, Y)-equivalent.

Now assume the two sets are not equal. Without loss of generality we can assume that there exist $x \in X$ and $y \in Y$ with $(\,x \,|\, y\,) = (\,x_i \,|\, y_i\,)$ for some i, but $(\,x \,|\, y\,) \neq (\,x'_j \,|\, y'_j\,)$ for all $j = 1, \dots, m$. Then $xwy \in L_d$ but $xw'y \notin L_d$ and, thus, w and w' are not (L_d, X, Y)-equivalent. □

Now we are prepared to formulate the lemma we are going to use in order to prove lower bounds for the time complexities. For the special case $L \in \mathscr{L}_{rt}(\mathrm{OCA})$ the lemma has been shown in [10].

Lemma 11 *Let $r : \mathbb{N} \to \mathbb{N}$ be an increasing function, $L \in \mathscr{L}_{n+r(n)}(\mathrm{OCA})$ and X and Y be two sets of words. Let s be the minimal number of states needed by an OCA to accept L in $n + r(n)$ time steps.*

If all words in X are of length m_1 and all words in Y are of length m_2, then the number N of (L, X, Y)-equivalence classes of the words at most of length $n - m_1 - m_2$ is bounded by

$$N \leq s^{m_1|X|} s^{(m_2+r(n))|Y|}$$

Proof. Let $\mathcal{A}$ be an OCA with s states that accepts L in $n + r(n)$ time steps. Let S be the state set of $\mathcal{A}$.

We consider the computation of $\mathcal{A}$ on the word xwy for some $x \in X$ and $y \in Y$. After $|w|$ time steps the interesting part of the configuration of $\mathcal{A}$ can be described by $f_w(x) f'_w(y)$ where (cf. Figure 2)

1. $f_w(x) \in S^*$ and $f'_w(y) \in S^*$.

2. $|f_w(x)| = |x|$ and $|f'_w(y)| = y + r(n)$. During the remaining $|xwy| + r(|xwy|) - |w| \leq |x| + |y| + r(n)$ time steps the result of the computation of $\mathcal{A}$ depends only on the states of the $|x| + |y| + r(n)$ leftmost cells.

3. $f'_w(y)$ depends only on w and y since no information can move from left to right.

4. $f_w(x)$ depends only on w and x since during $|w|$ time steps only the leftmost $|x| + |w|$ cells can influence the states of the leftmost $|x|$ cells.

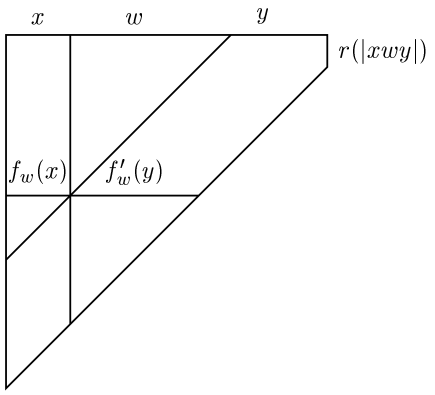

Figure 2: OCA computation in the proof of Lemma 11.

If $f_w(x) = f_{w'}(x)$ and $f'_w(y) = f'_{w'}(y)$ for all $x \in X$ and $y \in Y$, then w and w' are equivalent with respect to L, X and Y. Thus, if w and w' are not equivalent, then $f_w \neq f_{w'}$ or $f'_w \neq f'_{w'}$.

Now we count the number of functions f_w and f'_w. Since f_w maps X into the set S^{m_1} which contains s^{m_1} elements, the number of different functions f_w is bounded by $(s^{m_1})^{|X|}$. Analogously, it follows that the number of different functions f'_w is bounded by $(s^{m_2+r(n)})^{|Y|}$.

Since each upper bound on the number of pairs (f_w, f'_w) is also an upper bound on the number of (L, X, Y)-equivalence classes the lemma follows. $\square$

Now we apply the lemma to the language L_d.

Proposition 12 *Let $r : \mathbb{N} \to \mathbb{N}$ be a function. If $r(n)\log^2(r(n)) \in o(n)$, then L_d is not acceptable by an* OCA *with time complexity $n + r(n)$ but L_d belongs to $\mathscr{L}_{lt}$*(OCA).

Proof. For fixed $m_1 \in \mathbb{N}$ and $m_2 \in \mathbb{N}$ we investigate all words of the form $(\, x_1 \,|\, y_1 \,) \cdots (\, x_k \,|\, y_k \,)$ with $x_i \in \{0,1\}^{m_1}$ and $y_i \in \{0,1\}^{m_2}$ for all $i \in \{1, \dots, k\}$ and $(\, x_i \,|\, y_i \,) \neq (\, x_j \,|\, y_j \,)$ for $i \neq j$. We call this words of type (m_1, m_2).

Since there are at most $2^{m_1+m_2}$ different pairs $(\, x_i, x_j \,)$ the length of words of type (m_1, m_2) is at most $2^{m_1+m_2} \cdot (m_1 + m_2 + 3)$.

As has been shown in Proposition 10 two words are equivalent iff the sets of subwords are equal. Thus, there are $2^{2^{m_1+m_2}}$ words of type (m_1, m_2) which belong to different equivalence classes with respect to L_d, $X = \{0,1\}^{m_1}$ and $Y = \{0,1\}^{m_2}$. (For each subset of $X \times Y$ there exists one equivalence class.)

Assume L_d belongs to $\mathscr{L}_{n+r(n)}$(OCA), then an accepting OCA must be able to distinguish all these equivalence classes. By Lemma 11 there must exist a number of states s such that

$$2^{2^{m_1+m_2}} \leq s^{m_1 2^{m_1}} s^{(m_2+r(n))2^{m_2}}$$

for $n = 2^{m_1+m_2} \cdot (m_1 + m_2 + 3) + m_1 + m_2$.

In order to obtain a contradiction let $m_1 = c \cdot 2^{m_2}$ for some arbitrary rational number c. Approximating the order of n we obtain

$$n \in O(2^{m_1+m_2} \cdot (m_1 + m_2)) = O(2^{m_1+m_2} \cdot m_1) = O(2^{m_1} 2^{m_2} \cdot m_1) = O(2^{m_1} m_1^2)$$

Since $r(n)\log^2(r(n)) \in o(n)$ it holds $r(2^{m_1} m_1^2)\log^2(r(2^{m_1} m_1^2)) \in o(2^{m_1} m_1^2)$. Observe $2^{m_1}\log^2(2^{m_1}) = 2^{m_1} m_1^2$ and, therefore, $r(2^{m_1} m_1^2)$ and, hence, $r(n)$ must be of order $o(2^{m_1})$. It follows

$$\begin{aligned} s^{m_1 2^{m_1}} s^{(m_2+r(n))2^{m_2}} &= s^{m_1 2^{m_1}} s^{(m_2+o(2^{m_1}))2^{m_2}} = s^{m_1 2^{m_1}} s^{o(2^{m_1})2^{m_2}} \\ &= s^{m_1 2^{m_1} + o(2^{m_1} 2^{m_2})} = s^{c 2^{m_2} 2^{m_1} + o(2^{m_1} 2^{m_2})} = s^{c 2^{m_2+m_1} + o(2^{m_1+m_2})} \end{aligned}$$

Since c has been arbitrarily chosen we can let it go to 0 and obtain

$$= s^{o(2^{m_1+m_2})} = o(2^{2^{m_1+m_2}})$$

This is a contradiction, thus, L is not acceptable by an OCA in $n + r(n)$ time.

To see that L is acceptable in linear-time, we construct an appropriate OCA. Starting with an input word of the form $x\,(\, x_1 \,|\, y_1 \,) \cdots (\, x_m \,|\, y_m \,)\, y$ the OCA shifts the subword y with full speed to the left. During the first n time steps the OCA marks all pairs $(\, x_i \,|\, y_i \,)$ with $y_i = y$. Each marked pair starts moving to the left with half speed. Each time a pair $(\, x_i \,|\, y \,)$ reaches the left hand side the OCA checks whether $x_i = x$. The pairs of the form $(\, x_i \,|\, y\,)$ reach the leftmost cell sequentially because y moves with full speed but the pairs of form $(\, x_i \,|\, y\,)$ with half speed only. This guarantees that the OCA has sufficient time to check whether $x = x_i$. Figure 3 illustrates the computation. The basic task for the OCA is to check whether $y = y_i$. This is equivalent to the acceptance of the real-time OCA-language $\{w \bullet w \mid w \in \{0,1\}^+\}$. □

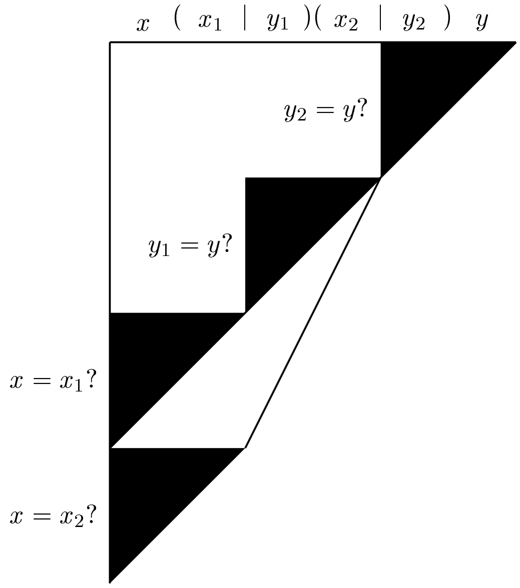

Figure 3: Linear-time acceptance of L_d. The black triangles mark the areas where a check of the form $y_i = y$ takes place. It is easy to see that the black triangels are disjoint, i.e., the checks can be done one after the other with a finite number of states.

5 Time Hierarchies

The last section is devoted to the proof of the main result which has been stated in Theorem 3. The next step towards the proof is a translation lemma that allows to extend a single proper inclusion to a time hierarchy.

Lemma 13 *Let $t_1, t_2 : \mathbb{N} \to \mathbb{N}$ be two functions and let L_a be a language that belongs to $\mathscr{L}_{n+t_1(n)}(\mathrm{OCA})$ but is not acceptable by any OCA within $n+o(t_2(n))$ time as can be shown by applying Lemma 11. Further let $f : \mathbb{N} \to \mathbb{N}$ be a constructible function and $r_1, r_2 : \mathbb{N} \to \mathbb{N}$ be two functions such that $r_1(f(n)) \in \Omega(t_1(n))$ and $r_2(f(n)) \in o(t_2(n))$. Then*

$$\mathscr{L}_{n+r_2(n)}(\mathrm{OCA}) \subset \mathscr{L}_{n+r_1(n)}(\mathrm{OCA})$$

Proof. For $f(n) \in O(n)$ we have $r_2(O(n)) \in o(t_2(n))$ what implies $r_2(n) \in o(t_2(n))$ and, thus, $L_a \notin \mathscr{L}_{n+r_2(n)}(\mathrm{OCA})$. Conversely, $r_1(O(n)) \in \Omega(t_1(n))$

and, therefore, $r_1(n) \in \Omega(t_1(n))$. It follows $L_a \in \mathscr{L}_{n+r_1(n)}(\mathrm{OCA})$ and, hence, the assertion.

In order to prove the lemma for $n \in o(f(n))$ let L_f be a language that proves the constructibility of f in Lemma 6. At first we show that we can always find such an L_f whose words are of the form $a^n w b^n$ with $|w| = f(n) - 2n$.

By $w/3$ we denote the word w compressed by the factor 3, i.e., one symbol of $w/3$ is interpreted as three symbols of w.

Now define $\bar{L}_f$ such that $a^{f(n)-n-\frac{1}{3}f(n)}(w/3)b^n \in \bar{L}_f$ iff $w \in L_f$. Clearly, the words of $\bar{L}_f$ are of the desired form since $n \in o(f(n))$. Moreover, there exists a trivial λ-free, length preserving homomorphism that maps $\bar{L}_f$ to $\{a^{f(n)-n}b^n \mid n \in \mathbb{N}\}$. Also, $\bar{L}_f$ belongs to $\mathscr{L}_{rt}(\mathrm{OCA})$ since an OCA can verify in real-time that an input w

- belongs to L_f,
- the length of the word $a^{f(n)-n-\frac{1}{3}f(n)}(w/3)b^n \in \bar{L}_f$ is equal to the length of w, and
- that n is equal to the number of b's in $h(w)$ where h denotes the homomorphism that maps L_f to $\{a^{f(n)-n}b^n \mid n \in \mathbb{N}\}$.

Thus, from now on we may assume w.l.o.g. that the words of L_f are of the form $a^n w b^n$ with $|w| = f(n) - 2n$. From Proposition 7 follows that the language $L_1(L_a, L_f)$ belongs to $\mathscr{L}_{n+t_1(f^{-1}(n))}(\mathrm{OCA})$. By the assumption on $r_1(f(n))$ we obtain $r_1(n) = r_1(f(f^{-1}(n))) \in \Omega(t_1(f^{-1}(n)))$ and, therefore, $L_1(L_a, L_f) \in \mathscr{L}_{n+r_1(n)}(\mathrm{OCA})$.

It remains to show that $L_1(L_a, L_f) \notin \mathscr{L}_{n+r_2(n)}(\mathrm{OCA})$.

Since $r_2(f(n)) \in o(t_2(n))$ and L_a is not acceptable within $n + o(t_2(n))$ time by any OCA, the language L_a is not acceptable within $n + r_2(f(n))$ time by any OCA, either. Due to the assumption, by Lemma 11 for every $s \in \mathbb{N}$ there must exist sets X and Y and an $n \in \mathbb{N}$ such that all words in X are of length m_1, all words in Y are of length m_2, and the number of (L_a, X, Y)-equivalence classes of the words at most of length $n - m_1 - m_2$ is not bounded by $s^{m_1|X|}s^{(m_2+r_2(f(n)))|Y|}$.

Define
$$X' = \{(x_1, a) \cdots (x_{m_1}, a) \mid x = x_1 \dots x_{m_1} \in X\}$$
and
$$Y' = \{(y_1, b) \cdots (y_{m_2}, b) \mid y = y_1 \dots y_{m_2} \in Y\}$$
and for every word $v = v_1 \cdots v_{n-m_1-m_2}$ a word v' by
$$v' = (v_1, w_1) \cdots (v_{n-m_1-m_2}, w_{n-m_1-m_2})(\sqcup, w_{n-m_1-m_2+1}) \cdots (\sqcup, w_{f(n)-m_1-m_2})$$
where $a^{m_1} w_1 \cdots w_{f(n)-m_1-m_2} b^{m_2}$ is a word of L_f. (Remember that each word in L_f starts with n symbols a and ends with n symbols b and $m_1 + m_2 \leq n$.)

For $x \in X$ and $y \in Y$ let x' and y' denote the corresponding words in X' and Y'.

By construction $xvy \in L_a$ iff $x'v'y' \in L_1$. (The word $x'v'y'$ belongs to L_1 if the second component of $x'v'y'$ is a word in L_f, which is always true, and the first component of $x'v'y'$ is a word in L_a concatenated with some blank symbols, i.e., $xvy \in L_a$.) Thus, the (L_a, X, Y)-equivalence classes have corresponding (L_1, X', Y')-equivalence classes and the number of (L_1, X', Y')-equivalence classes under the words whose length is at most $f(n) - m_1 - m_2$ is not bounded by $s^{m_1|X|}s^{(m_2+r_2(f(n)))|Y|}$.

Applying Lemma 11 with L_1, X', Y' and $f(n)$ in place of L_a, X, Y and n yields that $L_1 \notin \mathscr{L}_{n+r_2(n)}(\text{OCA})$. This completes the proof. □

Finally the main Theorem 3 is just a combination of the preceding lemmas:

Theorem 3. Let $r_1, r_2 : \mathbb{N} \to \mathbb{N}$ be two functions such that $r_2 \cdot \log^2(r_2) \in o(r_1)$ and r_1^{-1} is constructible, then

$$\mathscr{L}_{n+r_2(n)}(\text{OCA}) \subset \mathscr{L}_{n+r_1(n)}(\text{OCA})$$

Proof. Proposition 12 shows that the previously defined language L_d is acceptable in linear-time but is not acceptable in $n+r(n)$ time if $r(n)\log^2(r(n)) \in o(n)$. Now set $t_1(n) = n$ and t_2 such that $t_2(n)\log^2(t_2(n)) = n$.

Inserting yields $r(n)\log^2(r(n)) \in o(t_2(n)\log^2(t_2(n)))$. We conclude $r(n) \in o(t_2(n))$ and, thus, L_d is not acceptable in $n + o(t_2(n))$ time. In order to apply Lemma 13 we consider the constructible function $f = r_1^{-1}$.

Clearly, $r_1(f(n)) = n \in \Omega(n) = \Omega(t_1(n))$.

Since $r_2(n)\log^2(r_2(n)) \in o(r_1(n))$ we have

$$r_2(f(n))\log^2(r_2(f(n))) \in o(r_1(f(n))) = o(n) = o(t_2(n)\log^2(t_2(n)))$$

We conclude $r_2(f(n)) \in o(t_2(n))$.

Now all conditions of Lemma 13 are satisfied and an application proves the assertion $\mathscr{L}_{n+r_2(n)}(\text{OCA}) \subset \mathscr{L}_{n+r_1(n)}(\text{OCA})$. □

References

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

[2] Buchholz, Th., Klein, A., and Kutrib, M. *On tally languages and generalized interacting automata*. Developments in Language Theory IV. Foundations, Applications, and Perspectives, 2000, pp. 316–325.

[3] Buchholz, Th., Klein, A., and Kutrib, M. *On interacting automata with limited nondeterminism*. Fund. Inform. (2001), to appear.

[4] Choffrut, C. and Čulik II, K. *On real-time cellular automata and trellis automata*. Acta Inf. 21 (1984), 393–407.

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

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

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

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

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

[10] Terrier, V. *Language not recognizable in real time by one-way cellular automata*. Theoret. Comput. Sci. 156 (1996), 281–287.