I F I G
RESEARCH
REPORT

INSTITUT FÜR INFORMATIK

REAL-TIME LANGUAGE RECOGNITION BY ALTERNATING CELLULAR AUTOMATA

Thomas Buchholz Andreas Klein
Martin Kutrib

IFIG RESEARCH REPORT 9904
MARCH 1999

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

JUSTUS-LIEBIG-
UNIVERSITÄT
GIESSEN

IFIG Research Report

IFIG Research Report 9904, March 1999

Real-Time Language Recognition by Alternating Cellular Automata

Thomas Buchholz[1] Andreas Klein

Martin Kutrib[2]

Institute of Informatics, University of Giessen

Arndtstr. 2, D-35392 Giessen, Germany

Abstract. The capabilities of alternating cellular automata (ACA) to accept formal languages are investigated. Several notions of alternation in cellular automata have been proposed. Here we study so-called nonuniform ACAs. Our considerations center on space bounded real-time computations. In particular, we prove that there is no difference in acceptance power regardless whether one-way or two-way communication lines are provided. Moreover, the relations between real-time ACAs and deterministic (CA) and nondeterministic (NCA) cellular automata are investigated. It is proved that even the real-time ACAs gain exponential speed-up against nondeterministic NCAs. Comparing ACAs with deterministic CAs it is shown that real-time ACAs are strictly more powerful than real-time CAs.

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

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

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

1 Introduction

Linear arrays of finite automata can be regarded as models for massively parallel computers. Mainly they differ in how the automata are interconnected and in how the input is supplied. Here we are investigating arrays with two very simple interconnection patterns. Each node is connected to its both immediate neighbors or to its right immediate neighbor only. Correspondingly they are said to have two-way or one-way communication lines. The input mode is parallel. At initial time each automaton fetches an input symbol. Such arrays are commonly called cellular automata.

Although deterministic, nondeterministic and alternating finite automata have the same computing capability there appear to be essential differences when they are used to construct deterministic (CA), nondeterministic (NCA) and alternating (ACA) cellular automata. (We use the denotation OCA, NOCA and AOCA to indicate one-way communicationlines.) For example, it is a famous open problem whether or not CAs and OCAs have the same computing power ($\mathscr{L}(\text{OCA}) =?\ \mathscr{L}(\text{CA})$) [11] but the problem is solved for nondeterministic arrays ($\mathscr{L}(\text{NOCA}) = \mathscr{L}(\text{NCA})$) [3]. It is known that the real-time OCA languages are properly contained in the linear-time OCA languages ($\mathscr{L}_{rt}(\text{OCA}) \subset \mathscr{L}_{lt}(\text{OCA})$) [2, 13, 5]. But on the other hand $\mathscr{L}_{rt}(\text{NOCA}) = \mathscr{L}_{lt}(\text{NOCA})$ has been shown in [1]. Since $\mathscr{L}_{lt}(\text{NOCA}) = \mathscr{L}_{rt}(\text{NCA})$ (which follows from the closure of $\mathscr{L}_{rt}(\text{NOCA})$ under reversal [1] and $\mathscr{L}_{lt}(\text{NOCA}) = \mathscr{L}_{rt}^{R}(\text{NCA})$) we have the identity $\mathscr{L}_{rt}(\text{NOCA}) = \mathscr{L}_{rt}(\text{NCA})$. For deterministic arrays it holds $\mathscr{L}_{rt}(\text{OCA}) \subset \mathscr{L}_{rt}(\text{CA})$ [11].

Here we consider arrays built by alternating finite automata. In [7] from the point of view of time-varying cellular automata first results concerning a restricted variant of ACAs are shown. In a second work on alternating cellular automata [8] three models are distinguished. In *nonuniform* ACAs each cell computes its next state independently according to the local transition function. In *uniform* ACAs at every time step one deterministic local transition is nondeterministically chosen from a finite set of such functions and is applied to all the cells. The last notion defines the *weak* ACAs where only the leftmost cell of the array is an alternating automaton; all the others are nondeterministic. In [8] it is shown that nonuniform ACAs are the most powerful of the devices and that linear-time weak and uniform ACAs coincide. Some other results deal with simulations between alternating Turing machines and ACAs. This topic is also the main contribution of [10] where the simulation results of [8] are extended and some others are shown.

Our main interest are nonuniform ACAs under real-time restriction. The basic notions are defined in the next section. Section 3 is devoted to the question whether or not two-way ACAs are more powerful than one-way AOCAs. We prove the answer to be ‘no’. Especially, the equivalence between ACAs and AOCAs is shown for all time complexities. A second result in Section 3 is the important technical lemma which states that a specific subclass of ACAs can be sped up by a constant factor as long as the time complexity does not fall below real-time. For such devices, especially, the equivalence of real-time and linear-time follows. In Section 4 the relations between real-time ACAs and deterministic and nondeterministic cellular automata are investigated. It is proved that even the real-time ACAs gain exponential speed-up against nondeterministic NCAs. Comparing ACAs with deterministic CAs it is shown that real-time ACAs are strictly more powerful than real-time CAs. The latter result becomes important in so far as it is not known whether one of the following inclusions is strict:

$$\mathscr{L}_{rt}(\mathrm{CA}) \subseteq \mathscr{L}_{lt}(\mathrm{CA}) \subseteq \mathscr{L}(\mathrm{OCA}) \subseteq \mathscr{L}(\mathrm{CA}) \subseteq \mathscr{L}(\mathrm{NCA})$$

2 Basic notions

We denote the rational numbers by $\mathbb{Q}$, the integers by $\mathbb{Z}$, the positive integers $\{1, 2, \dots\}$ by $\mathbb{N}$, the set $\mathbb{N} \cup \{0\}$ by $\mathbb{N}_0$ and the powerset of a set S by 2^S. The empty word is denoted by ε and the reversal of a word w by w^R. For the length of w we write $|w|$.

An alternating cellular automaton is a linear array of identical alternating finite automata, sometimes called cells, where each of them is connected to its both nearest neighbors (one to the left and one to the right). For our convenience we identify the cells by positive integers. The state transition of the cells depends on the actual state of the cell itself and the actual states of its both neighbors. The finite automata work synchronously at discrete time steps. Their states are partitioned into existential and universal ones. What makes a, so far, nondeterministic computation to an alternating computation is the mode of acceptance, which will be defined with respect to the partitioning. More formally:

Definition 1 *An* alternating cellular automaton *(*ACA*) is a system* $(S, \delta, \#, A, F)$ *where*

a) *S is the finite, nonempty set of* states *which is partitioned into* existential *(S_e) and* universal *(S_u) states: $S = S_e \cup S_u$,*
b) *$\# \notin S$ is the* boundary state,
c) *$A \subseteq S$ is the nonempty set of* input symbols,
d) *$F \subseteq S$ is the set of* accepting states,
e) *δ is the finite, nonempty set of* local transition functions *which map from $(S \cup \{\#\})^3$ to S.*

Let $\mathcal{M} = (S, \delta, \#, A, F)$ be an ACA. A *configuration* of $\mathcal{M}$ at some time $t \geq 0$ is a description of its global state, which is actually a mapping $c_t : [1, \dots, n] \to S$ for $n \in \mathbb{N}$. The configuration at time 0 is defined by the initial sequence of states. For a given input word $w = w_1 \cdots w_n \in A^+$ we set $c_{0,w}(i) := w_i$, $1 \leq i \leq n$. Subsequent configurations are chosen according to the global transition Δ:

Let $n \in \mathbb{N}$ be a positive integer and c resp. c' be two configurations defined by $s_1, \dots, s_n \in S$ resp. $s'_1, \dots, s'_n \in S$.

$$c' \in \Delta(c) \iff \exists\, \delta_1, \dots, \delta_n \in \delta : \quad s'_1 = \delta_1(\#, s_1, s_2), s'_2 = \delta_2(s_1, s_2, s_3), \dots, s'_n = \delta_n(s_{n-1}, s_n, \#)$$

Thus, Δ is induced by δ. Observe, that one can equivalently define ACAs by requiring just one unique nondeterministic local transition that maps from $\left(S \cup \{\#\}\right)^3$ to $\left(2^S \setminus \emptyset\right)$. But with an eye towards later constructions we are requiring a finite, nonempty set of deterministic local transitions from which each cell nondeterministically chooses one at every time step. Obviously, both definitions yield equivalent devices.

The evolution of $\mathcal{M}$ is represented by its computation tree.

The *computation tree* $T_{\mathcal{M},w}$ of $\mathcal{M}$ under input $w \in A^+$ is a tree whose nodes are labeled by configurations. The root of $T_{\mathcal{M},w}$ is labeled by $c_{0,w}$. The children of a node labeled by a configuration c are the nodes labeled by the possible successor configuartions of c. Thus, the node c has exactly $|\Delta(c)|$ children.

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

If the flow of information is restricted to one-way, the resulting device is an *alternating one-way cellular automaton* (AOCA). I.e. the next state of each cell depends on the actual state of the cell itself and the state of its immediate neighbor to the right. Thus, we have information flow from right to left. Accordingly acceptance is indicated by the leftmost cell of the array: A *configuration c is accepting* iff $c(1) \in F$.

In order to define *accepting computations* on input words we need the notion of accepting subtrees.

Definition 2 *Let $\mathcal{M} = (S, \delta, \#, A, F)$ be an* ACA *or an* AOCA *and $T_{\mathcal{M},w}$ be its computation tree for an input word $w \in A^n$, $n \in \mathbb{N}$. A finite subtree T' of $T_{\mathcal{M},w}$ is said to be an* accepting subtree *iff it fulfills the following conditions:*

a) The root of T' is the root of $T_{\mathcal{M},w}$.

b) If $c' \in \Delta(c)$ is a child of a non-leaf node labeled by a configuration c then the set of all its children is $\left\{c'' \in \Delta(c) \mid c''(i) = c'(i) \text{ for all } 1 \le i \le n \text{ such that } c(i) \in S_e\right\}$.

c) The leafs of T' are labeled by accepting configurations.

From the computational point of view an accepting subtree is built by letting all the cells in existential states do their nondeterministic guesses and, subsequently, spawning all possible distinct offspring configurations with respect to the cells in universal states.

Conversely, one could build the subtree by spawning all possible distinct offspring configurations with respect to the cells in universal states at first, and letting cells in existential states do their guesses in each offspring configuration independently. Fortunately, it has been shown [10] that both methods lead to time complexities which differ at most by a constant factor. Moreover, the proofs given in the following can easily be adapted to that mode of acceptance such that both methods are equivalent in the framework in question.

Definition 3 *Let $\mathcal{M} = (S, \delta, \#, A, F)$ be an* ACA *or an* AOCA*.*

a) *A word $w \in A^+$ is* accepted *by $\mathcal{M}$ if there exists an accepting subtree of $T_{\mathcal{M},w}$.*

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

c) *Let $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$, be a mapping. If for all $w \in L(\mathcal{M})$ there exists an accepting subtree of $T_{\mathcal{M},w}$ the height of which is less than $t(|w|)$, then L is said to be of* time complexity *t.*

An ACA (AOCA) $\mathcal{M}$ is *nondeterministic* if the state set consists of existential states only. An accepting subtree is now a list of configurations which corresponds to a possible computation path of $\mathcal{M}$. Nondeterministic cellular automata are denoted by NCA resp. NOCA.

An ACA (AOCA) is *deterministic* if the set δ of local transition functions is a singleton. In these cases the course of computation is unique for a given input word w and, thus, the whole computation tree is a list of configurations. Deterministic cellular automata are denoted by CA resp. OCA.

The family of all languages which can be accepted by a device POLY with time complexity t is denoted by $\mathscr{L}_t(\text{POLY})$. If t equals the *identity function* $id(n) := n$ acceptance is said to be in real-time and we write $\mathscr{L}_{rt}(\text{POLY})$. The *linear-time* languages $\mathscr{L}_{lt}(\text{POLY})$ are defined according to

$$\mathscr{L}_{lt}(\text{POLY}) := \bigcup_{k \in \mathbb{Q},\, k \geq 1} \mathscr{L}_{k \cdot id}(\text{POLY})$$

3 Equivalence of one-way and two-way information flow and linear speed-up

This section is devoted to the relationship between ACAs and AOCAs and the speed-up of a restricted version that becomes important in subseqent proofs. The main results are that for arbitrary time complexities there is no difference in acceptance power between one-way and two-way information flow and the possibility to speed up so-called uniformly universal ACAs and AOCAs by a constant factor as long as they do not fall below real-time. Especially by the latter result we can show the results in the next sections even for real-time language families.

Thus, in $\mathcal{M}'$ either all cells are in existential or all cells are in universal states and, moreover, this changes from time step to time step. Since 0 is even the computation starts with existential states.

Without loss of generality we may assume that once a cell becomes accepting it remains in accepting states permanently. Such a behavior is simply implemented by setting a flag in an additional register that will never be unset. It remembers the cell that it has been in an accepting state before.

The next result states that one-way information flow in alternating cellular automata is as powerful as two-way information flow. This, on one hand, gives us a normalization since for proofs and constructions it is often useful to reduce the technical challenge to one-way transitions and, on the other hand, indicates the power of alternations since it is well known that deterministic one-way languages form a proper subset of the deterministic two-way languages: $\mathscr{L}_{rt}(\mathrm{OCA}) \subset \mathscr{L}_{rt}(\mathrm{CA})$ [11].

Theorem 4 *Let $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$, be a mapping. Then*

$$\mathscr{L}_t(\mathrm{AOCA}) = \mathscr{L}_t(\mathrm{ACA})$$

Proof. For structural reasons it suffices to show $\mathscr{L}_t(\mathrm{ACA}) \subseteq \mathscr{L}_t(\mathrm{AOCA})$.

The idea for the simulation of an ACA by an AOCA without any loss of time is as follows: A cell of the AOCA 'knows' the actual states of itself and of its neighbor to the right. Additionally, it guesses the state of its neighbor to the left nondeterministically and simulates the two-way transition of the ACA. In order to verify whether or not the guesses are correct each cell stores its guessed state and its old state in additional registers. After performing a simulation step the verification can simply be done by comparing the old state of a cell with the guessed state of its neighbor to the right. Thus, the verification is done by the neighbor to the left of a cell, respectively.

Obviously, the guesses of the leftmost cell are not verified. But we can restrict the local transition as follows: If the initial state of a cell is an existential one all the guessed left neighbor states that are not the border state are marked by a '-' during the first time step. If the initial state of a cell is an universal one all the guessed left neighbor states that are not the border state are marked by a '+'. The effect of these marks is that the cells with a '-' will never and the cells with a '+' will always accept. Thus, if the cell is not the leftmost cell this behavior does not affect the overall computation result. But if the cell is the leftmost cell only the correct guesses are relevant during the remaining computation.

Moreover, al left border state is guessed by a cell if and only if that cell has guessed a left border state at the first time step, and the not marked cells are only allowed to switch to accepting states if they have guessed a left border state. Therefore, to guess a left border state at every time step is the only way for the leftmost cell to become accepting. But exactly in these cases it has simulated the correct behavior of the leftmost cell of the two-way ACA.

Up to now we kept quiet about a crucial point. Whereas the verification itself is a deterministic task which can be performed by cells in existential as well as in universal states, responding to the result of the verification needs further mechanisms.

We distinguish two cases: If the old state of a cell is an existential one and the verification by the left neighboring cell fails then the latter sends an error signal to the left that prevents the not marked cells passed through from accepting. Therefore, in an accepting subtree there are only nodes labeled by configurations in which existential cells have guessed right and, hence, have simulated the two-way transition correctly. If the verification succeeds no further reaction is necessary.

In the second case the old state of a cell is an universal one. If the verification by the left neighboring cell fails it sends an error signal to the left that enforces all not marked cells passed through to switch into an accepting state. Again, if the verification succeeds no further reaction is necessary.

What is the effect of these mechanisms: In an accepting subtree in all configurations with a common predecessor cells that have been existential in the predecessor are in the same states, respectively. Due to the first case these cells have simulated the two-way transition correctly. Since all siblings (spawned by universal states) have to lead to subtrees with accepting leafs but acceptance according to the two-way ACA depends on the configurations with correct guesses only all configurations with wrong guesses are forced to accept to achieve the desired behavior.

Altogether it follows that the AOCA can simulate the ACA without any loss of time. □

Corollary 5 $\mathscr{L}_{rt}(\text{AOCA}) = \mathscr{L}_{rt}(\text{ACA})$

As we have shown extending the information flow from one-way to two-way does not lead to more powerful devices. The next lemma states that increasing the computation time by a constant factor does not either if we restrict the computations to the uniformly universal mode. A corresponding result does not hold for deterministic cellular automata. Instead, $\mathscr{L}_{rt}(\text{OCA}) \subset \mathscr{L}_{lt}(\text{OCA})$ has been shown [2, 13, 5]. The relationship is a famous open problem for deterministic two-way devices (e.g. [4, 12]).

Uniform ACAs have been introduced in [7, 8]. The main difference between uniform ACAs and (nonuniform) ACAs is the induction of the global transition. Whereas in an ACA at every time step each cell chooses independently one local transition from the set in an uniform ACA at every time step one local transition is choosen globally and applied to all the cells:

Let $\mathcal{M} = (S, \delta, \#, A, F)$ be an uniform ACA, $n \in \mathbb{N}$ be a positive integer and c resp. c' be two configurations defined by $s_1, \dots, s_n \in S$ resp. $s'_1, \dots, s'_n \in S$.

$$\begin{aligned} c' \in \Delta(c) \iff & \exists\, \delta_u \in \delta : \\ & s'_1 = \delta_u(\#, s_1, s_2), s'_2 = \delta_u(s_1, s_2, s_3), \dots, s'_n = \delta_u(s_{n-1}, s_n, \#) \end{aligned}$$

Thus, in a computation tree of an uniform ACA each node has at most $|\delta|$ successors. Now a whole configuration is labeled universal (existential) if the leftmost cell is in an universal (existential) state. An accepting subtree is a finite subtree of the computation tree that includes all (one) of the successors of a universal (existential) node. As usual all leafs have to be labeled with accepting configurations.

Now we are considering a computation mode that is nonuniform for existential and uniform for universal states. It is called *uniformly universal* mode and the corresponding devices are denoted by UUACA and UUAOCA. Besides their technical importance such devices are interesting for their own.

For our purposes it is sufficient to consider UUAOCAs $\mathcal{M} = (S, \delta, \#, A, F)$ which are *alternation normalized* as follows:

$$\begin{aligned} &A \subseteq S_e \text{ and } \forall\, \delta_i \in \delta : \\ &\qquad \big(\forall\, s_1, s_2 \in S_e \cup \{\#\} : \delta_i(s_1, s_2) \in S_u \text{ and} \\ &\qquad\ \ \forall\, s_1, s_2 \in S_u \cup \{\#\} : \delta_i(s_1, s_2) \in S_e\big) \end{aligned}$$

Thus, at every even time step all the cells are in existential and at every odd time step all the cells are in universal states.

Lemma 6 *Let $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$, be a mapping and $k \in \mathbb{Q}$, $k \geq 1$, be a constant. For every alternation normalized* UUAOCA $\mathcal{M}$ *that accepts a language $L(\mathcal{M})$ with time complexity $k \cdot t$ there exists an alternation normalized* UUAOCA $\mathcal{M}'$ *with time complexity t such that $L(\mathcal{M}) = L(\mathcal{M}')$ and vice versa.*

Proof. Let $\mathcal{M} = (S, \delta, \#, A, F)$. W.l.o.g. we can assume that the cells in $\mathcal{M}$ once they have entered an accepting state are remaining in accepting states.

Now in a first part we are going to construct an UUAOCA $\mathcal{M}'$ that simulates $\mathcal{M}$ with double speed. Basically, the idea is to simulate a sequence of four steps of $\mathcal{M}$ by two steps in $\mathcal{M}'$.

From time $4 \cdot t$, $t \in \mathbb{N}_0$, an accepting computation of $\mathcal{M}$ runs through configurations such that $\exists\, c_{4 \cdot t+1} \in \Delta(c_{4 \cdot t})$ such that $\forall\, c_{4 \cdot t+2} \in \Delta(c_{4 \cdot t+1})$ such that $\exists\, c_{4 \cdot t+3} \in \Delta(c_{4 \cdot t+2})$ such that $\forall\, c_{4 \cdot (t+1)} \in \Delta(c_{4 \cdot t+3})$.

Note, that due to the normalization the configurations of only universal states have at most $|\delta|$ successors, respectively (i.e. $|\Delta(c_{4 \cdot t+1})| \leq |\delta|$ and $|\Delta(c_{4 \cdot t+3})| \leq |\delta|$).

The expression is replaced by the equivalent expression $\exists\, c_{4 \cdot t+1} \in \Delta(c_{4 \cdot t})$ such that $\exists\, c_{4 \cdot t+3}(c_{4 \cdot t+2})$ such that $\forall\, c_{4 \cdot t+2} \in \Delta(c_{4 \cdot t+1})$ such that $\forall\, c_{4 \cdot (t+1)} \in \Delta\big(c_{4 \cdot t+3}(c_{4 \cdot t+2})\big)$ where $c_{4 \cdot t+3}(c_{4 \cdot t+2})$ denotes a mapping which gives a configuration $c_{4 \cdot t+3}$ depending on $c_{4 \cdot t+2}$. (Again, there exists only a finite number of possible configurations $c_{4 \cdot t+2}$.)

For a moment let us assume that in $\mathcal{M}$ every cell is connected to its three immediate neighbors to the right such that the local transition depends on the actual states of four cells. Under this assumption both successive existential as well as both successive universal steps can be performed in one step respectively. Obviously, the resulting UUAOCA $\mathcal{M}' = (S', \delta', \#, A, F')$ is again alternation normalized. Formally it is constructed as follows. Let $\delta = (\delta_1, \ldots, \delta_d)$ and therefore $|\delta| = d \in \mathbb{N}$.

$S' := S \cup (S \times \delta^d)$, $\quad S'_e := S$, $\quad S'_u := S \times \delta^d$, $\quad F' := F \cup (F \times \delta^d)$,

$\delta' := \{\delta^e_{i,j_1,\ldots,j_d} \mid 1 \leq i, j_1, \ldots, j_d \leq d\} \cup \{\delta^u_{i,j} \mid 1 \leq i, j \leq d\}$

$\forall\, p_1 \in S,\ p_2, p_3, p_4 \in S \cup \{\#\}$:

$$\delta^e_{i,j_1,\ldots,j_d}\big(p_1, p_2, p_3, p_4\big) := (\delta_i(p_1, p_2), \delta_{j_1}, \ldots, \delta_{j_d})$$

$\forall\,(p_1, f_{1_1}, \ldots, f_{1_d}) \in S \times \delta^d$, $(p_m, f_{m_1}, \ldots, f_{m_d}) \in (S \times \delta^d) \cup \{\#\}, 2 \leq m \leq 4$:

$$\begin{aligned}
&\delta^u_{i,j}\big((p_1, f_{1_1}, \ldots, f_{1_d}), (p_2, f_{2_1}, \ldots, f_{2_d}), (p_3, f_{3_1}, \ldots, f_{3_d}), (p_4, f_{4_1}, \ldots, f_{4_d})\big) := \\
&\qquad\qquad\qquad \delta_j(r_1, r_2) \\
&\qquad\qquad \text{where } r_1 = f_{1_i}(q_1, q_2), \quad r_2 = f_{2_i}(q_2, q_3) \\
&\qquad\qquad \text{where } q_1 = \delta_i(p_1, p_2), \quad q_2 = \delta_i(p_2, p_3), \quad q_3 = \delta_i(p_3, p_4)
\end{aligned}$$

The construction of the local transitions of $\mathcal{M}'$ for existential states simulates a corresponding local transition of $\mathcal{M}$ and additionally guesses d local transitions of $\mathcal{M}$. By these d transitions the mapping is realized that gives a configuration $c_{4\cdot t+3}$ depending on $c_{4\cdot t+2}$. Each cell uses one of the local transitions in each of the d possible configurations $c_{4\cdot t+2}$ which yields an independently guessed successor, respectively.

Let $w \in A^n$ be a word belonging to $L(\mathcal{M})$. In order to prove the correctness of the construction we show that for an accepting subtree T of the computation tree $T_{\mathcal{M},w}$ of $\mathcal{M}$ there exists an accepting subtree T' of the computation tree $T_{\mathcal{M}',w}$ of $\mathcal{M}'$ such that the nodes on level $2 \cdot t \in \mathbb{N}$ of T' are exactly the nodes on level $4 \cdot t$ of T and vice versa. Due to the normalization of $\mathcal{M}$ and $\mathcal{M}'$ all configurations on levels in question consist of existential states only.

The assertion holds true for $t = 0$ since the initial configurations $c_{0,w}$ of $\mathcal{M}$ and $c'_{0,w}$ of $\mathcal{M}'$ (and thus the roots of T and T') are identical.

Let $c_{4\cdot t}$ be a node of T on level $4 \cdot t$. By inductive hypothesis exists a node $c'_{2\cdot t}$ of T' on level $2 \cdot t$ where $c_{4\cdot t} = c'_{2\cdot t}$ and vice versa.

"$\Longrightarrow$" Let $c_{4\cdot t+1}$ be the unique successor node of $c_{4\cdot t}$ in T ($c_{4\cdot t}$ consists of existential states only). The successors of $c_{4\cdot t+1}$ (generated by the applications of $\delta_1, \ldots, \delta_d$) are denoted by $c^{(1)}_{4\cdot t+2}, \ldots, c^{(d)}_{4\cdot t+2}$ ($c_{4\cdot t+1}$ consists of universal states only) and their unique successors by $c^{(l)}_{4\cdot t+3}$, $1 \leq l \leq d$. The next step yields the successors $c^{(l,1)}_{4\cdot(t+1)}, \ldots, c^{(l,d)}_{4\cdot(t+1)}$ of $c^{(l)}_{4\cdot t+3}$ (generated by the applications of $\delta_1, \ldots, \delta_d$) where $1 \leq l \leq d$.

By the construction and the hypothesis there exists the successor configuration $c'_{2\cdot t+1}$ of $c'_{2\cdot t}$ on level $2\cdot t+1$ of T' such that $c'_{2\cdot t+1}(x) = \big(c_{4\cdot t+1}(x), f_{x_1}, \ldots, f_{x_d}\big)$ where $f_{x_l}\big(c^{(l)}_{4\cdot t+2}(x), c^{(l)}_{4\cdot t+2}(x+1)\big) = c^{(l)}_{4\cdot t+3}(x)$, $1 \leq l \leq d$, $1 \leq x \leq n$.

The successors of $c'_{2\cdot t+1}$ are obtained by applying all of the d^2 mappings $\delta^u_{i,j}$, $1 \leq i, j \leq d$, to all the cells.

We obtain the configuration $c^{(i,j)}_{4\cdot(t+1)}$, $1 \leq i, j \leq d$, by applying the local transition $\delta^u_{i,j}$ to $c'_{2\cdot t+1}$: With the denotations of the construction we have

$$\begin{aligned}
&q_1 = \delta_i\big(c_{4\cdot t+1}(x), c_{4\cdot t+1}(x+1)\big) = c^{(i)}_{4\cdot t+2}(x), \\
&q_2 = \delta_i\big(c_{4\cdot t+1}(x+1), c_{4\cdot t+1}(x+2)\big) = c^{(i)}_{4\cdot t+2}(x+1), \\
&q_3 = \delta_i\big(c_{4\cdot t+1}(x+2), c_{4\cdot t+1}(x+3)\big) = c^{(i)}_{4\cdot t+2}(x+2) \text{ and} \\
&r_1 = f_{x_i}\big(c^{(i)}_{4\cdot t+2}(x), c^{(i)}_{4\cdot t+2}(x+1)\big) = c^{(i)}_{4\cdot t+3}(x), \\
&r_2 = f_{x+1_i}\big(c^{(i)}_{4\cdot t+2}(x+1), c^{(i)}_{4\cdot t+2}(x+2)\big) = c^{(i)}_{4\cdot t+3}(x+1) \text{ and, hence,} \\
&c'^{(i,j)}_{2\cdot(t+1)}(x) = \delta_j\big(c^{(i)}_{4\cdot t+3}(x), c^{(i)}_{4\cdot t+3}(x+1)\big) = c^{(i,j)}_{4\cdot(t+1)}(x).
\end{aligned}$$

"$\Longleftarrow$" Let $c'_{2\cdot t+1} = (p_1, f_{1_1}, \ldots, f_{1_d}) \cdots (p_n, f_{n_1}, \ldots, f_{n_d})$ be the unique successor node of $c'_{2\cdot t}$ in T'.

The successors of $c'_{2\cdot t+1}$ (generated by the applications of $\delta^u_{i,j}$, $1 \leq i, j \leq d$) are denoted by $c'^{(i,j)}_{2\cdot(t+1)}$.

By the construction and the hypothesis there exist $\delta_x \in \delta$ such that $p_x = \delta_x\big(c_{4\cdot t}(x), c_{4\cdot t}(x+1)\big) = c_{4\cdot t+1}(x)$, $1 \leq x \leq n$. We obtain the configuration $c'^{(i,j)}_{2\cdot(t+1)}$, $1 \leq i, j \leq d$, by applying the local transition δ_i to $c_{4\cdot t+1}$, applying the local transition f_{x_i} to the cell x in the successor of $c_{4\cdot t+1}$ and, finally, applying the local transition δ_j:

$$
\begin{aligned}
&c^{(i)}_{4\cdot t+2}(x) = \delta_i\big(c_{4\cdot t+1}(x), c_{4\cdot t+1}(x+1)\big) = \delta_i(p_x, p_{x+1}) = q_x,\ 1 \leq x \leq n.\\
&c^{(i)}_{4\cdot t+3}(x) = f_{x_i}\big(c_{4\cdot t+2}(x), c_{4\cdot t+2}(x+1)\big) = f_{x_i}(q_x, q_{x+1}) = r_x,\ 1 \leq x \leq n.\\
&c^{(i,j)}_{4\cdot(t+1)}(x) = \delta_j\big(c_{4\cdot t+3}(x), c_{4\cdot t+3}(x+1)\big) = \delta_j(r_x, r_{x+1}) =\\
&\quad \delta^u_{i,j}\big((p_x, f_{x_1}, \ldots, f_{x_d}), \ldots (p_{x+3}, f_{x+3_1}, \ldots, f_{x+3_d})\big) = c'^{(i,j)}_{2\cdot(t+1)}(x),\ 1 \leq x \leq n.
\end{aligned}
$$

From "$\Longrightarrow$" and "$\Longleftarrow$" follows the correctness of the construction. Therefore $L(\mathcal{M}) = L(\mathcal{M}')$ and the time complexity of $\mathcal{M}'$ is half times the time complexity of $\mathcal{M}$.

Now we have to show the result for automata in which the cells are connected to one neighbor only. The idea is to compress the input such that each cell of $\mathcal{M}'$ simulates three adjacent cells of $\mathcal{M}$ in different registers. Due to the normalization all simulated cells are either universal or existential and there appear no conflicts.

The compression is done in the first existential time step. The compressed input is guessed by the leftmost $\lceil \frac{n}{3} \rceil$ cells. In [1] it has been shown how the verification of the guess can be done deterministically in real-time.

Since $\mathcal{M}'$ is again an alternation normalized UUAOCA the whole method can be iterated i times until $2^i \geq k$ to obtain the required speed-up. □

The construction of Lemma 6 bounds the possible speed-up by the time needed for the verification of the guessed compression. Therefore, computations cannot be faster than real-time.

Corollary 7 *Let $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$, be a mapping and $k \in \mathbb{Q}$, $k \geq 1$, be a constant. For every alternation normalized* UUACA *$\mathcal{M}$ that accepts a language $L(\mathcal{M})$ with time complexity $k \cdot t$ there exists an alternation normalized* UUACA *$\mathcal{M}'$ with time complexity t such that $L(\mathcal{M}) = L(\mathcal{M}')$ and vice versa.*

Proof. The construction of Theorem 4 does not affect the status of the cells (i.e. whether they are existential or universal). Therefore, for a given alternation normalized UUACA there exists an equivalent alternation normalized UUAOCA with the same time complexity. The UUAOCA can be sped up and the resulting automaton, trivially, can be transformed into an alternation normalized UUACA again. □

4 Comparisons with (non)deterministic cellular automata

It is a famous open problem whether or not the inclusion $\mathscr{L}_{rt}(\text{CA}) \subseteq ?\ \mathscr{L}_{lt}(\text{CA})$ is a proper one. Moreover, the seemingly easier problem $\mathscr{L}_{rt}(\text{CA}) \subseteq ?\ \mathscr{L}(\text{CA})$ is open, too. The same holds for nondeterministic cellular automata: It is not known whether or not the inclusion $\mathscr{L}_{rt}(\text{NCA}) \subseteq ?\ \mathscr{L}(\text{NCA})$ is strict.

Since $\mathscr{L}(\text{NCA})$ coincides with the context-sensitive languages and $\mathscr{L}(\text{CA})$ with the deterministic context-sensitive languages the properness of the inclusion $\mathscr{L}_{rt}(\text{CA}) \subseteq ?\ \mathscr{L}(\text{NCA})$ is also open due to the open problems mentioned and the famous lba-problem (i.e. in our terms $\mathscr{L}(\text{CA}) =?\ \mathscr{L}(\text{NCA})$). The open problem $\mathscr{L}_{rt}(\text{CA}) \subseteq ?\ \mathscr{L}_{rt}(\text{NCA})$ stresses the dilemma. Altogether, the following inclusions follow for structural reasons but it is not known whether one of them is strict.

$$\mathscr{L}_{rt}(\text{CA}) \subseteq \mathscr{L}(\text{OCA}) \subseteq \mathscr{L}(\text{CA}) \subseteq \mathscr{L}(\text{NCA}) \text{ and}$$
$$\mathscr{L}_{rt}(\text{CA}) \subseteq \mathscr{L}_{rt}(\text{NCA}) \subseteq \mathscr{L}(\text{NCA}).$$

In the present section we compare real-time ACAs to deterministic and nondeterministic cellular automata. The next result shows that adding alternations to nondeterministic computations yields enormous speed-ups.

Theorem 8 $\mathscr{L}(\text{NCA}) \subseteq \mathscr{L}_{rt}(\text{ACA})$

Proof. Let $\mathcal{M} = (S, \delta, \#, A, F)$ be an NCA. Since the number of cells is bounded by the length of the input n the number of configurations is bounded by $|S|^n$. Therefore, we can assume that $\mathcal{M}$ has the exponential time complexity $|S|^n$.

The following construction is done for $|S| = 2$. A generalization is straightforward. The first step is to define an alternation normalized UUACA $\mathcal{M}' = (S', \delta', \#, A, F')$ which needs $3 \cdot n$ time steps to simulate 2^n steps of the ACA $\mathcal{M}$. For the main part of $\mathcal{M}'$ (a deterministic track is added later) we set

$S' := A \cup S^2 \cup S^3, \quad S'_e := A \cup S^2, \quad S'_u := S^3,$

$\delta' := \{\delta^e_{i,j,k} \mid 1 \le i,j \le |S|, 1 \le k \le |F|\} \cup \{\delta^e_i \mid 1 \le i \le |S|\} \cup \{\delta^u_1, \delta^u_2\}.$

Let n denote the length of the input, $\delta = \{\delta_1, \dots, \delta_d\}$, $S = \{s_1, \dots, s_r\}$ and $F = \{f_1, \dots, f_v\}$.

In its first time step $\mathcal{M}'$ saves its input configuration, guesses an accepting configuration of $\mathcal{M}$ and another configuration of $\mathcal{M}$:

$\forall\, p_1, p_2 \in A, p_3 \in A \cup \{\#\}$:

$$\begin{aligned} \delta^e_{i,j,k}(p_1, p_2, p_3) &:= (p_2, s_i, s_j) \\ \delta^e_{i,j,k}(\#, p_2, p_3) &:= (p_2, s_i, f_k), \quad 1 \le i,j \le |S|, \quad 1 \le k \le |F| \end{aligned}$$

The idea is to guess successively an accepting computation path $c_0, \dots, c_{2^n}$ of $\mathcal{M}$. The configuration on the second track should be $c_{2^{n-1}}$ and the accepting configuration on the third track should be c_{2^n}.

From now on at every universal step for each configuration two offsprings are spawned. One gets the configurations on the first and second track and the other the configurations on the second and third track:

$\forall\, (p_{1,1}, p_{1,2}, p_{1,3}), (p_{3,1}, p_{3,2}, p_{3,3}) \in S^3 \cup \{\#\}, (p_{2,1}, p_{2,2}, p_{2,3}) \in S^3$:

$$\begin{aligned} \delta_1^u\big((p_{1,1}, p_{1,2}, p_{1,3}), (p_{2,1}, p_{2,2}, p_{2,3}), (p_{3,1}, p_{3,2}, p_{3,3})\big) &:= (p_{2,1}, p_{2,2}) \\ \delta_2^u\big((p_{1,1}, p_{1,2}, p_{1,3}), (p_{2,1}, p_{2,2}, p_{2,3}), (p_{3,1}, p_{3,2}, p_{3,3})\big) &:= (p_{2,2}, p_{2,3}) \end{aligned}$$

Thus, c'_1 represents the configurations c_0, $c_{2^{n-1}}$ and c_{2^n}. Its both successors represent the configuration pairs $(c_0, c_{2^{n-1}})$ and $(c_{2^{n-1}}, c_{2^n})$.

In every further existential step the configuration between the represented configurations is guessed:

$\forall\, (p_{1,1}, p_{1,2}), (p_{3,1}, p_{3,2}) \in S^2 \cup \{\#\}, (p_{2,1}, p_{2,2}) \in S^2$:

$$\delta_i^e\big((p_{1,1}, p_{1,2}), (p_{2,1}, p_{2,2}), (p_{3,1}, p_{3,2})\big) := (p_{2,1}, s_i, p_{2,2}), \quad 1 \leq i \leq |S|$$

Thus, the two possible configurations of $\mathcal{M}'$ at time step 3 are representing the configuration triples $(c_0, c_{2^{n-2}}, c_{2 \cdot 2^{n-2}})$ and $(c_{2 \cdot 2^{n-2}}, c_{3 \cdot 2^{n-2}}, c_{4 \cdot 2^{n-2}})$. One time step later we have the four pairs $(c_0, c_{2^{n-2}})$, $(c_{2^{n-2}}, c_{2 \cdot 2^{n-2}})$, $(c_{2 \cdot 2^{n-2}}, c_{3 \cdot 2^{n-2}})$ and $(c_{3 \cdot 2^{n-2}}, c_{4 \cdot 2^{n-2}})$.

Concluding inductively it is easy to see that at time $1 \leq t \leq n$ there exist 2^t configurations of $\mathcal{M}'$ representing the pairs $(c_0, c_{2^{n-t}})$, $(c_{2^{n-t}}, c_{2 \cdot 2^{n-t}})$, $(c_{2 \cdot 2^{n-t}}, c_{3 \cdot 2^{n-t}}), \ldots, (c_{(2^t - 1) \cdot 2^{n-t}}, c_{2^t \cdot 2^{n-t}})$.

For $t = n$ we obtain (c_0, c_1), $(c_1, c_2), \ldots, (c_{2^n - 1}, c_{2^n})$. Now $\mathcal{M}'$ can locally check whether the second element of a pair is a valid successor of the first elements of the cell and its neighbors according to the local transitions of $\mathcal{M}$. If the check succeeds $\mathcal{M}'$ has guessed an accepting computation path of $\mathcal{M}$ and accepts the input.

In order to perform the check each cell of $\mathcal{M}'$ has to be aware of the time step $2 \cdot n$. For this purpose a deterministic FSSP algorithm [9, 14] is started on an additional track which synchronizes the cells at time $2 \cdot n$. Altogether the result of the check is available at time step $2 \cdot n + 1$ and needs another $n - 1$ time steps to get into the leftmost cell. We conclude that $\mathcal{M}'$ has time complexity $3 \cdot n$. By Lemma 6 the alternation normalized UUACA $\mathcal{M}'$ can be sped up to real-time.

It remains to show that $\mathscr{L}_{rt}(\text{UUACA}) \subseteq \mathscr{L}_{rt}(\text{ACA})$. The proof is a straightforward adaption of the proof that (nonuniform) ACAs are at least as powerful as uniform ACAs [8]. $\square$

Corollary 9 $\mathscr{L}(\text{NCA}) \subseteq \mathscr{L}_{rt}(\text{AOCA})$

Extending the previously mentioned chains of inclusions by the last result we obtain

$$\begin{aligned} &\mathscr{L}_{rt}(\text{CA}) \subseteq \mathscr{L}(\text{OCA}) \subseteq \mathscr{L}(\text{CA}) \subseteq \mathscr{L}(\text{NCA}) \subseteq \mathscr{L}_{rt}(\text{ACA}) \text{ and} \\ &\mathscr{L}_{rt}(\text{CA}) \subseteq \mathscr{L}_{rt}(\text{NCA}) \subseteq \mathscr{L}(\text{NCA}) \subseteq \mathscr{L}_{rt}(\text{ACA}). \end{aligned}$$

The next result shows that in both chains one of the inclusions *is* a proper one. It states $\mathscr{L}_{rt}(\text{CA}) \subset \mathscr{L}_{rt}(\text{ACA})$. We prove the inclusion by the use of a specific kind of deterministic cellular spaces as connecting pieces. A *deterministic cellular space* (CS) works like a deterministic cellular automaton. The difference is the unbounded number of cells. In cellular spaces there exists a so-called *quiescent state* q_0 such that the local transition satisfies $\delta(q_0, q_0, q_0) = q_0$. At time 0 all the cells from $\mathbb{Z}$ except the cells $1, \ldots, n$ which get the input are in the quiescent state. Obviously, at every time step the number of nonquiescent cells increases at most by 2.

In [6] an infinite hierarchy of language families has been shown: If $r \in \mathbb{Q}$, $r \geq 1$, and $\varepsilon \in \mathbb{Q}$, $\varepsilon > 0$, then $\mathscr{L}_{n^r}(\text{CS}) \subset \mathscr{L}_{n^{r+\varepsilon}}(\text{CS})$.

Especially, for $r = 1$ and $\varepsilon = 1$ it holds $\mathscr{L}_{rt}(\text{CS}) \subset \mathscr{L}_{n^2}(\text{CS})$.

Cellular spaces which are bounded to the left (and unbounded to the right) are equivalent to the original model since both halflines can be simulated on different tracks in parallel. Moreover, one obtains again an equivalent model if the number of cells is bounded by the time complexity. Let $w = w_1 \cdots w_n$ be an input word and $s : \mathbb{N} \to \mathbb{N}$, $s(n) \geq n$, be a mapping. The family of languages acceptable by deterministic cellular automata with initial configuration

$$c_{0,w} : [1, \ldots, s(|w|)] \to S, \quad c_{0,w} = \begin{cases} w_i & \text{if } 1 \leq i \leq |w| \\ q_0 & \text{if } |w| + 1 \leq i \leq s(|w|) \end{cases}$$

is denoted by $\mathscr{L}_{t,s}(\text{CA})$. It follows immediately $\mathscr{L}_t(\text{CS}) = \mathscr{L}_{t,t}(\text{CA})$ (here we assume always $\delta(q_0, q_0, \#) = q_0$).

Theorem 10 $\mathscr{L}_{rt}(\text{CA}) \subset \mathscr{L}_{rt}(\text{ACA})$

Proof. From $\mathscr{L}_t(\text{CS}) = \mathscr{L}_{t,t}(\text{CA})$ for $t = id$ we obtain $\mathscr{L}_{rt}(\text{CS}) = \mathscr{L}_{n,n}(\text{CA})$. The latter family is based on simultaneously n-time bounded and n-space bounded cellular automata, i.e. real-time (classical) cellular automata. Thus, $\mathscr{L}_{rt}(\text{CS}) = \mathscr{L}_{n,n}(\text{CA}) = \mathscr{L}_{rt}(\text{CA})$. By the result in [6] for $r = 1$ and $\varepsilon = 1$ it follows $\mathscr{L}_{rt}(\text{CS}) \subset \mathscr{L}_{n^2}(\text{CS})$ which is equivalent to $\mathscr{L}_{rt}(\text{CA}) \subset \mathscr{L}_{n^2}(\text{CS}) = \mathscr{L}_{n^2,n^2}(\text{CA})$. Now in order to prove the theorem we have to show $\mathscr{L}_{n^2,n^2}(\text{CA}) \subseteq \mathscr{L}_{rt}(\text{ACA})$.

The following construction results in an alternation normalized UUACA $\mathcal{M}'$ which simulates a simultaneously n^2-time bounded and n^2-space bounded CA $\mathcal{M}$. Note that a deterministic computation task is per se alternation normalized and meets the conditions of the uniformly universal computation mode.

Let $c_0, \ldots, c_{n^2}$ denote the configurations of an accepting computation path of $\mathcal{M}$ on input $w = w_1 \cdots w_n$. $\mathcal{M}'$ gets the input $c'_0(i) = c_0(i) = w_i$, $1 \leq i \leq n$, and 'knows' $c_0(i)$, $n + 1 \leq i \leq n^2$, to be the quiescent state q_0.

The key idea for $\mathcal{M}'$ is to guess the states $c_{n^2-n}(1), \ldots, c_{n^2-n}(n)$ existentially during the first time step and subsequently to spawn two offspring computations universally. One of them is the deterministic task to simulate $\mathcal{M}$ on input $c_{n^2-n}(1), \ldots, c_{n^2-n}(n)$ for n time steps in order to check whether $\mathcal{M}$ would accept. The second offspring has to verify whether the guess has been correct (i.e.

$\mathcal{M}$ produces a corresponding configuration at time step $n^2 - n$). Therefore, at the third time step it guesses the states $c_{n^2-2\cdot n}(1), \dots, c_{n^2-2\cdot n}(2 \cdot n)$ two times on three tracks: On one track in the compressed form (i.e. every cell contains two states), on another track the states $c_{n^2-2\cdot n}(1), \dots, c_{n^2-2\cdot n}(n)$ and on a third track the states $c_{n^2-2\cdot n}(n+1), \dots, c_{n^2-2\cdot n}(2 \cdot n)$. (Whether or not the guess yields two times the same sequence can deterministically be checked. Details are omitted here.) At the next time step $\mathcal{M}'$ universally spawns three offsprings: One of them is the deterministic task to simulate $\mathcal{M}$ on $c_{n^2-2\cdot n}(1), \dots, c_{n^2-2\cdot n}(2 \cdot n)$ for n time steps to check whether $\mathcal{M}$ would compute the previously guessed states $c_{n^2-n}(1), \dots, c_{n^2-n}(n)$ and, thus, to verify the previous guess. The second and third offsprings have to verify whether the new guesses are correct. The second offspring guesses $c_{n^2-3\cdot n}(1), \dots, c_{n^2-3\cdot n}(2 \cdot n)$ and iterates the described procedure. The third task has to guess the states $c_{n^2-3\cdot n}(1), \dots, c_{n^2-3\cdot n}(3 \cdot n)$ two times at four tracks: In the compressed form (i.e. three states per cell) and $c_{n^2-3\cdot n}(1), \dots, c_{n^2-3\cdot n}(n)$ and $c_{n^2-3\cdot n}(n+1), \dots, c_{n^2-3\cdot n}(2 \cdot n)$ and $c_{n^2-3\cdot n}(2n+1), \dots, c_{n^2-3\cdot n}(3 \cdot n)$ on separate tracks. (Again, whether or not the guess yields two times the same sequence can deterministically be checked.) Now a corresponding procedure is iterated. After the guessing one offspring simulates $\mathcal{M}$ for n time steps on $c_{n^2-3\cdot n}(1), \dots, c_{n^2-3\cdot n}(3 \cdot n)$ in order to verify whether it computes the states $c_{n^2-2\cdot n}(n+1), \dots, c_{n^2-2\cdot n}(2 \cdot n)$, and another three offsprings are verifying the guesses $c_{n^2-3\cdot n}(1), \dots, c_{n^2-3\cdot n}(n)$ and $c_{n^2-3\cdot n}(n+1), \dots, c_{n^2-3\cdot n}(2 \cdot n)$ and $c_{n^2-3\cdot n}(2n+1), \dots, c_{n^2-3\cdot n}(3 \cdot n)$.

Concluding inductively at time $2 \cdot i$ there exist offspring computations for the verification of $c_{n^2-i\cdot n}(j \cdot n + 1), \dots, c_{n^2-i\cdot n}((j+1) \cdot n)$ where $2 \leq i \leq n$ and $0 \leq j \leq i - 1$.

For $i = n$ the sequences $c_0(j \cdot n + 1), \dots, c_0((j+1) \cdot n)$ have to be verified. This can be done by checking whether the states match the initial input. For this reason the cells have to be aware of the time step $2 \cdot n$ what can be achieved by providing a deterministic FSSP algorithm on an additional track as has been done in the previous proof. Moreover, the computations have to know to which initial input symbols their sequences have to be compared. These that verify the sequences $c_{n^2-i\cdot n}(1), \dots, c_{n^2-i\cdot n}(n)$ behave slightly different. They are spawning three (instead of four) offsprings at every universal step. Since exactly the sequence $c_0(1), \dots, c_0(n)$ has to be compared to the input $w_1 \cdots w_n$ whereas all other sequences simply have to be compared to $q_0, \dots, q_0$, the verification can be done.

The FSSP fires at time $2 \cdot n$. Afterwards the (partially) simulation of $\mathcal{M}$ needs another n time steps. To collect the result of that simulation and to get it into the leftmost cell needs at most n further time steps. Altogether, $\mathcal{M}'$ has the time complexity $4 \cdot n$. Following the last steps of the proof of Theorem 8 the alternation normalized UUACA $\mathcal{M}'$ can be sped up to real-time and one can conclude $L(\mathcal{M}') \in \mathscr{L}_{rt}(\text{ACA})$. $\square$

Another interpretation of the last theorem is the possibility to save time and space simultaneously when adding alternation to a deterministic computation.

Moreover, now we know that at least one of the following inclusions is strict:

$$\mathscr{L}_{rt}(\mathrm{CA}) \subseteq \mathscr{L}_{rt}(\mathrm{NCA}) \subseteq \mathscr{L}_{rt}(\mathrm{ACA})$$

Acknowledgements

The authors are grateful to Thomas Worsch for valuable comments on the proof of Theorem 8.

References

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

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

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

[4] Ibarra, O. H. and Jiang, T. *Relating the power of cellular arrays to their closure properties*. Theoret. Comput. Sci. 57 (1988), 225–238.

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

[6] Ibarra, O. H., Kim, S. M., and Moran, S. *Sequential machine characterizations of trellis and cellular automata and applications*. SIAM J. Comput. 14 (1985), 426–447.

[7] Krithivasan, K. and Mahajan, M. *Nondeterministic, probabilistic and alternating computations on cellular array models*. Theoret. Comput. Sci. 143 (1995), 23–49.

[8] Matamala, M. *Alternation on cellular automata*. Theoret. Comput. Sci. 180 (1997), 229–241.

[9] Mazoyer, J. *A six-state minimal time solution to the firing squad synchronization problem*. Theoret. Comput. Sci. 50 (1987), 183–238.

[10] Reischle, F. and Worsch, Th. *Simulations between alternating CA, alternating TM and circuit families*. Technical Report 9/98, Fakultät für Informatik, Universität Karlsruhe, 1998.

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

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

[13] Umeo, H., Morita, K., and Sugata, K. *Deterministic one-way simulation of two-way real-time cellular automata and its related problems*. Inform. Process. Lett. 14 (1982), 158–161.

[14] Waksman, A. *An optimum solution to the firing squad synchronization problem*. Inform. Control 9 (1966), 66–78.