Simulations between alternating CA, alternating TM and circuit families[*]

Frank Reischle and Thomas Worsch
Fakultät für Informatik, Universität Karlsruhe

Abstract

Variants of cellular automata consisting of alternating instead of deterministic finite automata are investigated, so-called uniform alternating CA (ACA) and two types of nonuniform ACA. The former two have been considered by Matamala (1997). It is shown that the nonuniform ACA are time equivalent. The main contributions are fast simulations of ACA by uniform circuit families and vice versa. It is shown that nonuniform ACA are time equivalent to circuit families with unbounded fan-in, and that uniform ACA are time equivalent to circuit families with constant fan-in. Hence uniform ACA and alternating TM are time equivalent, too, solving a problem left open by Matamala. The results also give some evidence that a linear time simulation of nonuniform ACA by ATM is "unlikely" to exist.

1 Introduction

The standard model of deterministic cellular automata (CA) has been generalized in several directions, e.g. nondeterministic CA or stochastic CA. Recently they have also been extended using the concept of alternation [1]. First results have been obtained by Krithivasan and Mahajan [2]. They are considering a rather restricted model, one would call a special type of uniform ACA (UACA), using the terminology of Matamala [3].

Besides uniform ACA he considers so-called weak ACA (WACA) and one variant of nonuniform ACA (denoted $\exists\forall$–ACA in this paper). In [3] it is shown that UACA and WACA are time equivalent (i.e. they can simulate each other with only constant slowdown), that these models can simulate

[*]This is technical report 9/98 of the Department of Informatics, University of Karlsruhe. It is also available at `http://liinwww.ira.uka.de/~worsch/papers/`.

alternating Turing machines (ATM) with constant slowdown, and that these models can be simulated by $\exists\forall$–ACA with constant slowdown.

The rest of the paper is organized as follows: In Section 2 some basic notation is introduced as well as some concepts which are common to all alternating devices. In Section 3 two variants of nonuniform ACA ($\exists\forall$–ACA and $\forall\exists$–ACA) are considered. We prove that these models can simulate each other with only constant slowdown. Section 4 is devoted to uniform ACA (UACA), weak ACA (WACA) and alternating TM (ATM). We show that they are *all* time equivalent, solving a problem left open in [3]. In Section 5 relations between uniform resp. nonuniform ACA and circuit families with bounded fan-in resp. unbounded fan-in are investigated. Therefore the question whether nonuniform ACA are time equivalent to ATM is related to the corresponding problem for circuit families with (un-)bounded fan-in, giving some indication that nonuniform ACA may be somewhat more powerful than ATM.

2 Basic notions

In this paper we are only dealing with one-dimensional cellular automata (CA) with von Neumann neighborhood N of radius 1. Thus a deterministic CA is specified by a *set of states* Q and a *local transition function*[1] $\delta : Q^N \to Q$ specifying for each *local configuration* $l : N \to Q$ the new state $\delta(l)$ of the central cell. A *(global) configuration* is a mapping $c : \mathbf{Z} \to Q$. The state of cell i in c is denoted c_i and the local configuration "observed" by cell i in its neighborhood is denoted $c_{i+N} : N \to Q : n \mapsto c_{i+n}$. For a configuration c denote by c^N the mapping $c^N : \mathbf{Z} \to Q^N : i \mapsto c_{i+N}$. For a subset $T \subseteq \mathbf{Z}$ we write $c \vdash_{T,\delta} c'$ iff $i \in T \Longrightarrow c'_i = \delta(c^N_i)$ (where we have written c^N_i instead of $c^N(i)$). The relation $c \vdash_{\mathbf{Z},\delta} c'$ is abbreviated to $c \vdash_\delta c'$ and describes one step of a deterministic CA according to the local rule δ.

Of course the usual definition of deterministic CA is captured by the above seemingly unnecessarily complicated formalism. The reason for introducing it is, that it will turn out to be useful for the definition of alternating CA.

For all types of alternating CA it is assumed that the set of states $Q = Q_\exists \cup Q_\forall$ is partitioned into *existential states* $q \in Q_\exists$ and *universal states* $q \in Q_\forall$. The *local transition function* is now of the form $\nu : Q^N \to 2^Q$, specifying a *subset* $\nu(l)$ of possible new states for a cell observing l in its neighborhood. A configuration is defined as for deterministic CA.

[1] B^A denotes the set of all mappings from A to B.

For the recognition of formal languages an *input alphabet* $A \subset Q$ and a set $F \subset Q$ of *accepting final states* has to be specified, as well as a *quiescent state* $\Box$. For the quiescent local configuration $l_\Box : n \mapsto \Box$ the local function has to satisfy $\nu(l_\Box) = \{\Box\}$. In the *initial configuration* c_w for an input $w = x_1 \cdots x_n \in A^+$ of length $n \geq 1$ cell i holds input symbol x_i (for $1 \leq i \leq n$) and all other cells are in state $\Box$. A configuration c is *accepting* iff $c_1 \in F$.

The recognition of formal languages by alternating devices is usually defined using the notion of a *computation tree.* Its nodes are configurations and the root is always an initial configuration. In general there are several computation trees with the same root. A computation tree is accepting iff all of its leaves are accepting configurations. A word w is accepted if there is an accepting computation tree with c_w as root.

A word w is accepted in time t if there is an accepting computation tree of height $\leq t$ with c_w as root. For ACA we say that w is accepted in space s if every configuration in this tree contains at most s cells in a non-quiescent state.

Two machines (be it CA, TM, ...) are called *equivalent* iff they recognize the same language. They are called *time equivalent* iff they are equivalent and their time[2] complexities only differ by a constant factor.

The main differences between the various types of alternating CA lie in the definitions of which trees are legal computation trees. This topic is treated separately in the following sections.

3 Nonuniform alternating CA

First we define two types of *nonuniform cellular automata*, denoted as $\exists\forall$–ACA and $\forall\exists$–ACA. While $\exists\forall$–ACA are considered in [3] the $\forall\exists$–ACA are equally natural to look at.

For a configuration c let $\exists(c) = \{i \in \mathbf{Z} \mid c_i \in Q_\exists\}$ and $\forall(c) = \{i \in \mathbf{Z} \mid c_i \in Q_\forall\}$. Let $\bar{c}$ and $\bar{c}'$ be two mappings $\mathbf{Z} \to Q \cup Q^N$ and $T \subset \mathbf{Z}$ a subset of cells. We write $\bar{c} \vdash_{T,\nu} \bar{c}'$ iff $i \notin T \Longrightarrow \bar{c}_i = \bar{c}'_i$ and $i \in T \Longrightarrow \bar{c}_i \in Q^N \wedge \bar{c}'_i \in \nu(\bar{c}_i)$. Sometimes T is called the subset of *active* cells.

Using subsets $\bar{C}$, $\bar{C}'$, existential and universal "substeps" are now defined as follows: $\bar{c} \vDash^\forall_T \bar{C}'$ iff $\bar{C}' = \{\bar{c}' \mid \bar{c} \vdash_{T,\nu} \bar{c}'\}$ and $\bar{C} \vDash^\exists_T \bar{C}'$ iff $|\bar{C}| = |\bar{C}'|$ and for each $\bar{c} \in \bar{C}$ there is a $\bar{c}' \in \bar{C}'$ such that $\bar{c} \vdash_{T,\nu} \bar{c}'$.

The one-step relation for $\exists\forall$–ACA is now defined as $c \vDash^{\exists\forall} C'$ iff there is

[2]For ACA and TM this means the number of steps, for circuit families their depth.

a $\bar{c}$, such that[3] $c^N \models^{\exists}_{\exists(c)} \bar{c} \models^{\forall}_{\forall(c)} C'$. For $\forall\exists$–ACA we use $c \models^{\forall\exists} C$ iff there is a set $\bar{C}$, such that $c^N \models^{\forall}_{\forall(c)} \bar{C} \models^{\exists}_{\exists(c)} C'$. One should note that for both substeps the subset of cells which are active are defined via the original configuration c.

A tree of configurations is a computation tree for an $\exists\forall$–ACA iff for each non-leaf node c the set $\mathrm{Succ}(c)$ of its successors in the tree satisfies $c \models^{\exists\forall} \mathrm{Succ}(c)$. Analogously for $\forall\exists$–ACA all non-leaves c have to satisfy $c \models^{\forall\exists} \mathrm{Succ}(c)$.

Lemma 3.1 *For each $\exists\forall$–ACA there is a time equivalent one such that all configurations occuring in any computation tree are such that either all non-quiescent states are existential or they are all universal. The same holds for $\forall\exists$–ACA.*

Proof. The proof is only given for $\exists\forall$–ACA with von Neumann neighborhood $N = \{-1, 0, 1\}$. (The proofs for other neighborhoods and for $\forall\exists$–ACA can be done analogously). Let Q denote the set of states of an $\exists\forall$–ACA M. We construct another $\exists\forall$–ACA M' with state set Q', the same input alphabet A and the same quiescent state $\square$, which uses one full step (called an existential step) to simulate an existential substep and a subsequent full step (called a universal step) to simulate the subsequent universal substep of a step of M.

Without loss of generality assume that a cell will never enter the quiescent state if it is not already in it.

The simulation of an existential substep (resp. a universal substep) is indicated by a special flag (E resp. U) which is stored additionally in each non-quiescent cell of M'. The state set of M' is $Q' = A \cup \{\square\} \cup (\{\mathrm{E}, \mathrm{U}\} \times Q) \cup (\{\mathrm{U}\} \times Q^3)$.

For $x, y, z \in Q$ the states $(\mathrm{U}, y), (\mathrm{U}, xyz) \in Q'$ are universal. States $(\mathrm{E}, y) \in Q'$ are chosen to be existential as well as all states in A, because the first step of M' will be an existential one. By convention $\square$ is existential, too (although it doesn't matter). The set of accepting final states for M' is $F' = F \cup (\{\mathrm{E}\} \times F)$, where F is the set of accepting final states for M.

It remains to define the transition function ν' of M' (in terms of ν of M). This will be done in such a way that for an arbitrary configuration of M' which occurs during a computation starting with some initial configuration c_w the flags of all non-quiescent cells are either all E or they are all U; hence in each such configuration either all non-quiescent states are existential or they are all universal.

[3] We are identifying x and $\{x\}$ here.

For $x, y, z \in Q$ and $x', y', z' \in Q'$, using $\nu'(x', y', z')$ as a shorthand notation[4] define:

$$\begin{aligned}
\nu'((\mathrm{E},x),(\mathrm{E},y),(\mathrm{E},z)) &= \begin{cases} \{(\mathrm{U},\hat{y}) \mid \hat{y} \in \nu(x,y,z)\} & \text{if } y \in Q_\exists \\ \{(\mathrm{U},xyz)\} & \text{if } y \in Q_\forall \end{cases} \\
\nu'(x',(\mathrm{U},y),z') &= \{(\mathrm{E},y)\} \\
\nu'(x',(\mathrm{U},xyz),z') &= \{(\mathrm{E},\hat{y}) \mid \hat{y} \in \nu(x,y,z)\}
\end{aligned}$$

Furthermore a cell in state $x \in A$ acts as if it were in state (E, x). A cell in state (E, x) (or acting as such because it is in state $x \in A$) treats a neighbor in state $y \in A \cup \{\square\}$ as if that were in state (E, y). Analogously a cell in state (U, x) treats a neighbor in state $\square$ as if that were in state (U, y).

For a cell in state $\square$ we choose ν' as follows: If it sees a neighbor with flag E it essentially acts like a cell in state $(\mathrm{E}, \square)$ applying the above defined rule. But there is one exception. If according to ν one of the new states is again $\square$, then ν' does not prescribe $(\mathrm{U}, \square)$ as the corresponding new state, but simply $\square$. This ensured that the space complexity of M' is the same as that of M. If a cell in state $\square$ sees a neighbor with flag U it does nothing. i.e. remains in the quiescent state.

For all other local configurations ν' can be defined arbitrarily because they will never occur in configurations belonging to simulations of M.

In an existential step cells storing a universal state of M only collect the states of their neighbors and alter their flag, resulting in a state (U, xyz). Cells in a state (U, y) "did their work" during the previous existential step and do nothing except changing their flag during a universal step. In the other cases the cells of M' work as they would in C and alter their flag.

Now that M' is defined, we are going to prove that it works as it should. Denote by $\mathrm{E} \otimes C$ the set of configurations of M' which is obtained from the set C of configurations of M by adding the E flag in every non-quiescent cell in every configuration, analogously for $U \otimes C$, and in the cases of "c instead of C".

Suppose M makes one $\exists\forall$-step $c^N \vDash^{\exists}_{\exists(c)} \bar{c} \vDash^{\forall}_{\forall(c)} \hat{C}$. First we will show, that for M' holds: $\mathrm{E} \otimes c \vDash^{\exists\forall} \mathrm{U} \otimes \bar{c} \vDash^{\exists\forall} \mathrm{E} \otimes \hat{C}$.

For the simulation of the existential substep because of the definition of ν' and since $\exists(\mathrm{E} \otimes c) = \mathbf{Z}$ for M' one gets $(\mathrm{E} \otimes c)^N \vDash^{\exists}_{\exists(\mathrm{E}\otimes c)} \mathrm{U} \otimes \bar{c}$. Since $\forall(\mathrm{E} \otimes c) = \emptyset$ furthermore $\mathrm{U} \otimes \bar{c} \vDash^{\forall}_{\forall(\mathrm{E}\otimes c)} \mathrm{U} \otimes \bar{c}$ holds. That is, M' simulates the existential substep of M by the full step $\mathrm{E} \otimes c \vDash^{\exists\forall} \mathrm{U} \otimes \bar{c}$.

[4] for $\nu'(l)$ where l satisfies $l(-1) = x'$, $l(0) = y'$ and $l(1) = z'$

The considerations for the simulations of the universal substep $\bar{c} \models^{\forall}_{\forall(c)} \hat{C}$ of M are slightly more complicated. It has been defined above that quiescent cells observing neighbors in states (U, y) stay in state $\Box$. Hence during the existential substep of M' starting with configuration $\mathrm{U} \otimes \bar{c}$ "nothing interesting happens". This means that there is only one $\bar{c}' : \mathbf{Z} \to Q \cup Q^3$ with $(\mathrm{U} \otimes \bar{c})^N \models^{\exists}_{\exists(\mathrm{U}\otimes\bar{c})} \bar{c}'$, and it satisfies $\bar{c}'_i = \Box$ if $\bar{c}_i = \Box$ and $\bar{c}'_i = (\mathrm{U} \otimes \bar{c})^N_i$ otherwise. The latter happens for the cells in $\forall(\mathrm{U} \otimes \bar{c})$ which are exactly the non-quiescent cells of $\mathrm{U} \otimes \bar{c}$.

It remains to be shown that the $\tilde{c}$ satisfying $\bar{c}' \models^{\forall}_{\forall(\mathrm{U}\otimes\bar{c})} \tilde{c}$ are exactly the $\mathrm{E} \otimes \hat{c}$ for $\hat{c} \in \hat{C}$. It is clear that the flag of each non-quiescent state has to be E. To see that $\bar{c}' \vdash_{\forall(\mathrm{U}\otimes\bar{c})} \mathrm{E} \otimes \hat{c}$ holds, consider for an arbitrary non-quiescent cell i the cases whether it was in an existential or a universal state in configuration c separately:

1. $c_i \in Q_{\exists}$**, i.e.** $(\mathrm{U} \otimes \bar{c})_i = (\mathrm{U}, \bar{c}_i)$**:** According to the definition of ν' only the flag is toggled in cell i, and hence $(\mathrm{E} \otimes \hat{c})_i = (\mathrm{E}, \bar{c}_i)$ for any $\hat{c} \in \hat{C}$.

2. $c_i \in Q_{\forall}$**, i.e.** $(\mathrm{U} \otimes \bar{c})_i = (\mathrm{U}, c_{i-1}c_ic_{i+1})$**:** In M $\hat{c}_i \in \nu'(c_{i-1}c_ic_{i+1})$ holds, and this is exactly what is enforced by ν' for M'.

On the other hand it should be clear that because of the definition of ν' whenever $\bar{c}' \vdash_{\forall(\mathrm{U}\otimes\bar{c})} \mathrm{E} \otimes \hat{c}$, then $\hat{c} \in \hat{C}$.

Thus for M' holds $\mathrm{U} \otimes \bar{c} \models^{\exists\forall} \mathrm{E} \otimes \hat{C}$ and therefore every M-step $c \models^{\exists\forall} \hat{C}$ can be simulated by 2 M'-steps

$$\mathrm{E} \otimes c \models^{\exists\forall} \mathrm{U} \otimes \bar{c} \models^{\exists\forall} \mathrm{E} \otimes \hat{C} \; .$$

Conversely, for 2 steps of M' starting in a configuration of the form $\mathrm{E} \otimes c$, there has to be a $\bar{c}$ such that

$$\mathrm{E} \otimes c \models^{\exists\forall} \mathrm{U} \otimes \bar{c} \models^{\exists\forall} \mathrm{E} \otimes \hat{C} \; .$$

It is then not difficult to see that this implies that for M one has

$$c^N \models^{\exists}_{\exists(c)} \bar{c} \models^{\forall}_{\forall(c)} \hat{C} \quad \text{i.e.} \quad c \models^{\exists\forall} \hat{C} \; .$$

Hence, given an input word, for every accepting computation tree for M of height h there is one for M' of height $2h$ and vice versa. ∎

Essentially the same argument as above leads to the following result:

Theorem 3.2 *For all time bounds t and space bounds s holds:*

$$\exists\forall\text{–}\textsc{Aca–Time–Space}(\mathrm{O}(t), s) = \forall\exists\text{–}\textsc{Aca–Time–Space}(\mathrm{O}(t), s) \; .$$

4 Uniform ACA versus bounded fan-in circuits

Uniform ACA have been defined in [3] as follows. A deterministic transition function $\delta : Q^N \to Q$ is said to be *compatible* with a nondeterministic transition function $\nu : Q^N \to 2^Q$, written as $\delta \in \nu$ for short, iff for all $l \in Q^N$: $\delta(l) \in \nu(l)$. A configuration c of a UACA is called *existential* (resp. *universal*) iff c_1 is existential (resp. universal). A tree of configurations is a computation tree for a UACA iff each existential configuration c has exactly one successor c' such that $c \vdash_\delta c'$ for some $\delta \in \nu$ and for each universal configuration c the set of successors is $\mathrm{Succ}(c) = \{c' \mid \text{there is a } \delta \in \nu : c \vdash_\delta c'\}$. Hence the main difference to nonuniform ACA is that if in a configuration c the same local configuration occurs several times, in a UACA all cells observing it will enter the *same* new state, while in a nonuniform ACA they don't have to.

For completeness we mention the so-called weak ACA (WACA). They are "essentially" CA with deterministic cells with the exception of one cell, say cell 1, which is an alternating one. WACA are time equivalent to UACA [3].

One of the main contributions of this paper are results concerning the relations between alternating CA and uniform circuit families (UCIR). In this section UACA will be shown to be time equivalent to UCIR with bounded fan-in gates. As a corollary one also gets the time equivalence of UACA and ATM, solving a problem left open in [3]. In order to have the same set of input symbols for all models, we restrict ourselves to the input alphabet $A = \{0, 1\}$ below.

A circuit family with bounded fan-in consists of a circuit C_n with n inputs and one output for each integer $n \geq 1$, such that each C_n consists of $\neg$-gates (having one entry[5]) and $\wedge$- and $\vee$-gates with two entries. It is sometimes convenient to assume that there are "gates" producing the constants 0 (resp. 1) as output. Such devices obviously can be built from the $\neg$-, $\wedge$- and $\vee$-gates using one input bit. Furthermore we assume that the n input bits are provided at the exits of "input gates" (having no entries) and that the result of the circuit is available at the exit of an "output gate" (having one entry and doing nothing).

The size of a circuit is the total number of entries of all gates. For circuits with bounded fan-in this differs from the number of gates only by a constant factor (which we will ignore throughout the paper) and has the advantage that it actually is the definition used for circuits with unbounded

[5]Sometimes we don't use the word input to avoid confusions with the inputs of the circuit, and analogously for "exit" instead of output of a gate.

fan-in gates (see next section). The depth of a circuit is the length of the longest path from an input to the output. These notions are generalized to circuit families in the obvious way.

An important topic in the definition of UCIR is a concept also called *uniformity* (which has nothing to do with the uniformity condition for ACA). Different versions are used in the literature; see [5] for an overview. In this paper we will use the following (which is also applicable to UCIR with unbounded fan-in): To the gates of each circuit C_n numbers v of length $\mathrm{O}(\log \mathrm{size}(C_n))$ have to be assigned such there is a deterministic TM which on input of (n, v, i) can compute the type of gate v, its number e of entries and for $i \leq e$ the number v' of the gate providing the input for the i-th entry of v, using space complexity $\mathrm{O}(\log \mathrm{size}(C_n))$. Note that from this also follows the possibility to compute the number v_o of the output gate of a circuit using the same space complexity (by simply checking all numbers v one after the other until the one with the correct type is found).

Lemma 4.1 *For functions $t \geq n$ and $s \geq n$ which can be computed in space $\mathrm{O}(s)$:*

$$\text{UACA–TIME–SPACE}(\mathrm{O}(t), \mathrm{O}(s)) \subseteq \text{UCIR–DEPTH–SIZE}(\mathrm{O}(t), 2^{\mathrm{O}(s)})$$

Proof. Let C be an UACA with input alphabet $A = \{0, 1\}$, state set Q and transition function ν. For each input size n a circuit C_n will be constructed which accepts an input word w of length n, if and only if C accepts w.

The circuit consists of an "upper" and a "lower" part (the flow of information in the circuit is from the top to the bottom). The upper part of the circuit is similar to the construction for the proof of [5, Theorem 3]. It checks for all $w \in A^n$ whether C accepts w in time $t(n)$ or not: For all $1 \leq i \leq t(n)$ and all configurations c of C occupying space $s(n)$ or less – there are $2^{\mathrm{O}(s(n))}$ such configurations – the circuit contains a gate labeled (i, c). Imagine the gates being arranged in $t(n)$ levels with the first index *de*creasing from the top to the bottom. Furthermore there is a zero level with 2^n gates labeled $(0, c_w)$ (where the c_w are the initial configurations of C.

The type of a gate is $\wedge$ (resp. $\vee$) if the configuration is universal (resp. existential). The entries of gate (i, c) are connected to the exits of all gates $(i + 1, c')$ such that c' is a successor configuration of c. In general there are more than two successors of a configuration c, but for the UACA there are not more than $K := |Q|^{(|Q|^{|N|})}$, i.e. a constant number. So the "gates" described above can be implemented as binary trees of height $\leq \lceil ld(K) \rceil$ consisting

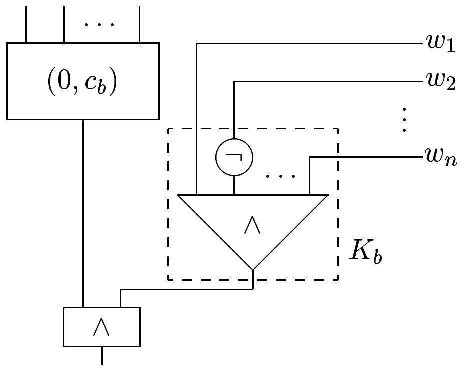

Figure 1: For each $b \in \{0,1\}^n$ there is a gate $(0, c_b)$ and a comparator circuit K_b whose i-th entry is negated $\Leftrightarrow b_i = 0$.

of ordinary gates with in-degree 2. Thus the overall depth of the circuit is increased only by a constant factor[6]. As the circuit is constructed from lower levels to higher levels a gate labeled with an accepting configuration is replaced by a constant 1 and gates labeled with a configuration using more than $s(n)$ space and gates on a hypothetic level $t(n) + 1$ are replaced by a constant 0. Now the following can be shown by induction (cf. [5]): the output of a gate $(0, c_w)$ is 1 iff there is an accepting tree for C on w of height $\leq t(n)$.

The "lower part" of C_n uses the input bits $w_1, \ldots, w_n$ to select the proper gate $(0, c_w)$ whose output becomes the output of the whole circuit (see Figure 1): To this end there are 2^n "comparators" K_b each comparing $w = w_1 \cdots w_n$ to one of the constant bit patterns $b = b_1 \cdots b_n \in A^n$ producing a 1 as output iff $w = b$. The outputs of K_b and the gate labeled $(0, c_b)$ are fed to an $\wedge$-gate. The outputs of these 2^n $\wedge$-gates are fed to a tree of $\vee$-gates (with height $\mathrm{ld}(2^n) = n$). The root of this tree is the output of the circuit and the following holds: The output of the constructed circuit is 1 iff C accepts w in time $\leq t(n)$.

The depth of the circuit is at most $\lceil ld(K) \rceil \cdot (t(n)+1) + \lceil ld(n) \rceil + 1 + n \in \mathrm{O}(t(n))$. The size of the circuit is dominated by the size of its upper part which is $2^{\mathrm{O}(s(n))} \cdot (t(n)+1)$. Since $t(n) \leq 2^{\mathrm{O}(S(n))}$, the overall size is $2^{\mathrm{O}(s(n))}$.

In order to prove that this circuit family is uniform, we describe a deterministic TM M' that constructs the circuit C_n (in the sense stated above)

[6]that only depends on the ACA C and not on the length of the input.

using space $\mathrm{O}(\log \operatorname{size}(C_n)) = \mathrm{O}(s(n))$. Hence M' is allowed to store a constant number of gate labels (i, c) on its tape. The interesting point is how M' determines the labels $(i+1, c')$ of the gates which provide an input for a given gate (i, c). Suppose that $\{\delta \mid \delta \in \nu\} = \{\delta_1, \ldots, \delta_{\tilde{K}}\}$. We may assume that each gate in the "upper" part has exactly $\tilde{K}$ entries (although it may happen that $\delta_i \neq \delta_j$, but $c \vdash_{\delta_i} c'$ and $c \vdash_{\delta_j} c'$ lead to the same successor configuration); our construction still works if a gate gets the same input from the same gate on different entries. M' can determine the label $(i+1, c')$ of the gate connected to its j-th entry by increasing the first index and applying δ_j to the cells of c in the bounds given by s. By looking at c_1 M' can determine the type of a gate (i, c) and if it has to replaced by a constant 1. Due to the assumptions made to t and s, M' can also determine when a gate must be replaced by a constant 0. ■

Note that the assumption $t(n) \geq n$ is no real restriction because a UACA needs at least n steps until every input symbol might have had an influence on the state of the origin cell.

Since it is known [5], that for t and s which can be linearly approximated[7]:

$$\text{UCIR–DEPTH–SIZE}(\mathrm{O}(t), 2^{\mathrm{O}(s)}) \subseteq \text{ATM–TIME–SPACE}(\mathrm{O}(t), \mathrm{O}(s))$$

together with [3]

$$\text{ATM–TIME–SPACE}(\mathrm{O}(t), \mathrm{O}(s)) \subseteq \text{UACA–TIME–SPACE}(\mathrm{O}(t), \mathrm{O}(s)) ,$$

one also gets the opposite inclusion as in Lemma 4.1 and therefore:

Theorem 4.2 *For $t(n) \geq n$ which can be linearly approximated holds:*

$$\text{UACA–TIME}(\mathrm{O}(t)) = \text{ATM–TIME}(\mathrm{O}(t)) = \text{UCIR–DEPTH}(\mathrm{O}(t)) .$$

5 Nonuniform ACA and circuits

In the previous section uniform ACA have been shown to be time equivalent to circuit families with bounded fan-in. In the following an analogous relation between nonuniform ACA and circuit families with unbounded fan-in gates will be established. ACA will also be compared to circuits with bounded fan-in gates and ATM.

[7]I.e. there is a $\hat{t}$, $t \leq \hat{t} \in \mathrm{O}(t)$, (resp. $\hat{s}$) which is time constructible.

In a circuit with unbounded fan-in (UbUcir) $\wedge$- and $\vee$-gates are allowed with an arbitrary number of entries. The uniformity condition used is exactly the same as for circuits with bounded fan-in above.

The proof in section 4 cannot be applied unchanged to nonuniform Aca because unlike for Uaca the number of successors of a configuration of a nonuniform Aca ($\exists\forall$–Aca or $\forall\exists$–Aca) cannot be bounded by a constant[8]. This leads to the idea to compare nonuniform Aca with uniform circuit families with unbounded fan-in gates.

Lemma 5.1 *For functions $t \geq n$ and $s \geq n$ which can be computed in space* $\mathrm{O}(s)$*:*

$$\exists\forall\text{–Aca–Time–Space}(\mathrm{O}(t), \mathrm{O}(s)) \subseteq \text{UbUcir–Depth–Size}(\mathrm{O}(t), 2^{\mathrm{O}(s)})$$

Proof. Recall the proof of Lemma 4.1. Here, a similar construction is used, but there are the following changes: A gate labeled (i, c) in the upper part of the circuit now is replaced by a tree of height 2 of gates with unbounded fan-in (see Figure 2). The root of the tree is a $\vee$ gate whose entries are connected to intermediate $\wedge$ nodes which correspond to the $\bar{c}$ such that $c^N \models^{\exists}_{\exists(c)} \bar{c}$. The entries of these intermediate gates in turn are connected to the exits of the gates $(i+1, \hat{c})$, for all $\hat{c} \in \hat{C}$, where $\bar{c} \models^{\forall}_{\forall(c)} \hat{C}$.

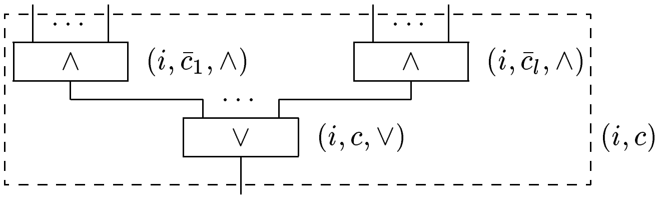

Figure 2: The "gate" (i, c) is implemented by one $\vee$-gate and an $\wedge$-gate for each $\bar{c}_j$ with $c^N \models^{\exists}_{\exists(c)} \bar{c}_j$.

Gates labeled with accepting configurations are replaced by a constant 1 and gates labeled by a configuration using more than space $s(n)$ or gates on level $t(n)+1$ are replaced by 0. It can be shown by induction on the level i that a gate (i, c) produces a 1 as output if and only if c is the root of an

[8]It depends on the number of cells in a configuration that are in a universal state observing a local configuration which allows at least two new states, i.e. $|\mathrm{Succ}(c)|$ depends on the space complexity and in general cannot be bounded by a constant.

accepting computation tree of height $\leq t(n) - i$ of the $\exists\forall$–ACA. Therefore the output of a gate labeled $(0, c_w)$ is 1 iff the $\exists\forall$–ACA accepts w.

The comparators selecting the output of the proper gate $(0, c_w)$ can be implemented using a single $\wedge$-gate with fan-in $2n$, making use of input symbols and their negations. Likewise the outputs of these $\wedge$-gates are fed to a single $\vee$-gate with fan-in 2^n whose exit is the output of the circuit.

The depth of the circuit C_n with n inputs is $2(t(n)+1)+2 = \mathrm{O}(t(n))$. The circuit contains up to $2^{\mathrm{O}(s(n))} \cdot (t(n)+1) = 2^{\mathrm{O}(s(n))}$ $\vee$-gates and $2^{\mathrm{O}(s(n))} \cdot 2^{\mathrm{O}(s(n))} \cdot (t(n)+1) = 2^{\mathrm{O}(s(n))}$ $\wedge$-gates. That is, the maximum fan-in of one gate is bounded by $2^{\mathrm{O}(s(n))}$ and therefore the size of the circuit is bounded by $2^{\mathrm{O}(s(n))} \cdot 2^{\mathrm{O}(s(n))} = 2^{\mathrm{O}(s(n))}$.

In order to show the uniformity of the circuit family, i.e. that the circuit C_n can be constructed by a TM M' using space $\mathrm{O}(s(n))$, the following numbering of gates will be used: For each (i, c) the corresponding $\vee$-gate is labeled $(i, c, \vee)$ and the intermediate $\wedge$-gates are labeled $(i, \bar{c}_j, \wedge)$. Suppose the gate that is connected to the j-th entry of gate $(i, c, \vee)$ has to be determined. For each existential state c_k in the configuration c, M' stores the set $\nu(c_{k+N})$ of possible successor states on its tape. This requires at most $|Q| \cdot s(n)$ additional storage. Now M' picks from each of these sets an appropiate successor state to build a configuration $\bar{c}_j$ and thus to obtain the label $(i, \bar{c}_j, \wedge)$ of the gate in question. Using a similar technique, the gate $(i+1, c', \vee)$ that is connected to the j-th entry of an $\wedge$-gate can be determined. ∎

If in the above construction gates with large fan-in are replaced by trees of gates with fan-in 2 one gets the following result.

Lemma 5.2 *For functions t and s satisfying the requirements of Lemma 5.1 holds:*

$$\begin{aligned}
\exists\forall\text{–ACA–TIME–SPACE}(\mathrm{O}(t), \mathrm{O}(s)) &\subseteq \text{UCIR–DEPTH–SIZE}(\mathrm{O}(st), 2^{\mathrm{O}(s)}) \\
\exists\forall\text{–ACA–TIME}(\mathrm{O}(t)) &\subseteq \text{UCIR–DEPTH}(\mathrm{O}(t^2))
\end{aligned}$$

Proof. First, the same construction as above is used. Then, each gate with a fan-in $f > 2$ is replaced by a tree of height $\lceil ld(f) \rceil$ of gates with fan-in 2. The maximum fan-in cannot exceed the number of gates in the UBUCIR-circuit[9] and the constructed circuits contain $2^{\mathrm{O}(s(n))} \cdot (t(n)+1)$ gates, where $2^{\mathrm{O}(s(n))}$ is the number of possible configurations occupying space $\mathrm{O}(s(n))$).

[9]Note, that its size is $\geq$ its depth which in turn is $\geq$ the time complexity of the ACA which is $\geq n$.

Since $t \leq 2^{\mathrm{O}(s(n))}$ (see [1] for a similar result for Atm), the number of gates and thus the maximum fan-in is bounded by $2^{\mathrm{O}(s(n))}$. Therefore the depth of the Ucir-circuit is bounded by $s(n)t(n)$ which is $\mathrm{O}(t^2(n))$. ∎

From Lemma 5.2 and the known relations between Ucir and Atm [5, Theorem 3] also follows the fact stated in Lemma 5.3 below which is Theorem 1 in [3].

In our opinion the proof of the latter needs some modification. First we describe a nonuniform Aca as a counterexample for which the proof fails and then the corrected proof of a direct simulation of nonuniform Aca by Atm is shown.

Consider an $\exists\forall$–Aca C being in the following configuration (with 7 nonquiescent cells)

$\square$	z	x	z	u	z	x	z	$\square$

where $\square$, z and x are existential states and u is a universal state. Suppose the transition function of C is defined by $(\_, z, \_) \mapsto \{\bar{z}\}$, $(z, x, z) \mapsto \{x_1, x_2\}$, $(z, u, z) \mapsto \{u_1, u_2\}$, $(\_, \square, \_) \mapsto \{\square\}$, … (other values are not of interest for our example).

We will now consider one possible configuration tree of the simulating Atm M (as described in [3, Theorem 1]) and have a look at the part of this tree (depicted in Figure 3) that simulates the step of C starting in the configuration shown above.

The point is that the simulation by M can produce successor configurations which differ in a cell that was in an existential state (like the 3rd cell from the left in the configurations in the bottom row of Figure 3). This however is impossible for a $\exists\forall$–Aca by definition. Therefore one could construct a $\exists\forall$–Aca C such that the "simulating" Atm M recognizes a language $L(M) \neq L(C)$.

The corrected simulation of nonuniform Aca by Atm is shown next. The same basic idea is used, but the main difference is that the updates of cells in an existential state and cells in an universal state are done in separate phases of the simulation.

Lemma 5.3 *For all time bounds t and space bounds s holds:*

$$\exists\forall\text{–Aca–Time–Space}(t, s) \quad \subseteq \quad \text{Atm–Time–Space}(\mathrm{O}(t^2), \mathrm{O}(s))$$

Proof. Let C be a $\exists\forall$–Aca with state set $Q = Q_\exists \cup Q_\forall$, input alphabet A, von Neumann neighborhood of radius 1 and the transition function ν. In the

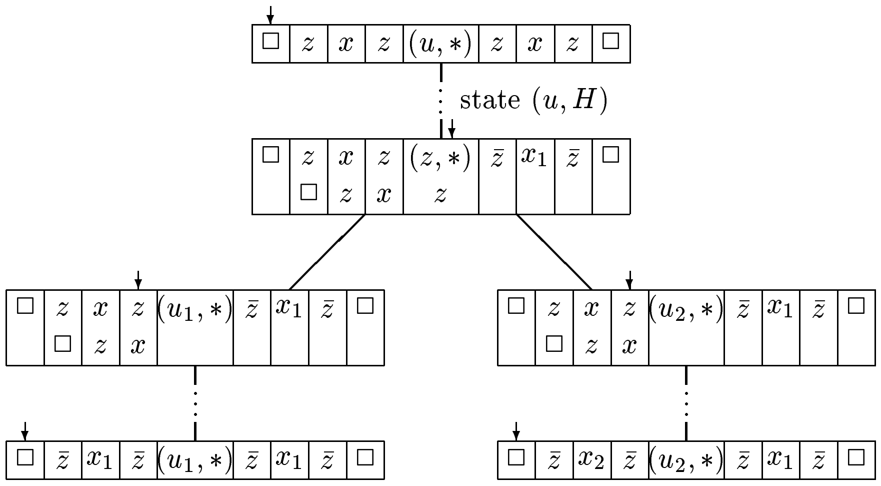

Figure 3: Simulation of one step of C by M. The arrows point at the current head positions. The states of M are not shown, except for the configuration where the universal branching occurs.

following an ATM M with one work tape will be described that simulates one step of C within $\mathrm{O}(s(n))$ steps. For this purpose states from Q are stored as symbols on the tape of M.

The simulation of one step is done in 3 phases. In each phase M moves its head across the tape segment with all squares storing non-quiescent states. In the first phase M copies the contents of each field in this segment to the right neighboring square. In this phase the ATM works exclusively in a deterministic fashion. In the second phase M moves its head in the opposite direction over the interesting tape segment. Whenever M reads a symbol that represents an existential state the ATM enters an existential state and carries out the transition according to ν. All other symbols are copied into the left neighbor square. In the third phase M carries out the transitions for all universal states with itself being in an universal state. When a symbol that represents an accepting state of C is written by M in the second or third phase into the square with index 1 (this square therefore has to be marked with a special symbol at the beginning of the computation), M enters an accepting state. The simulation of the following step of C works analogously with all directions of head movements reversed.

Hence, the squares of M's tape must store up to 3 states from Q and a marker in square 1. The state set of M must contain Q, a flag that shows in which phase it is at the moment, a flag that shows the current direction of head movement and a special accepting state.

The space complexity of M is $s+2$ and the time complexity is $t \cdot \mathrm{O}(s) = \mathrm{O}(t^2)$. ∎

We are now going to prove a kind of reverse result compared to Lemma 5.1, thus establishing a close relation between nonuniform ACA and circuit families with unbounded fan-in (satisfying certain conditions).

Lemma 5.4 *Let $t \geq n$ and $s \geq n$ be functions which can be computed in space* $\mathrm{O}(s)$ *and which satisfy $t \in \Omega(s)$. Then it holds:*

$$\textsc{UbUcir–Depth–Size}(\mathrm{O}(t), 2^{\mathrm{O}(s)}) \subseteq \exists\forall\textsc{–Aca–Time–Space}(\mathrm{O}(t), \mathrm{O}(s))$$

Proof. Let C_n be the circuit with n inputs, having depth $\mathrm{O}(t)$ and size $2^{\mathrm{O}(s)}$. We have to construct an $\exists\forall$–ACA accepting the same $w \in A^n$ as C_n, satisfying the time and space bounds $\mathrm{O}(t)$ and $\mathrm{O}(s)$. The construction is modeled on the one in the proof for the analogous simulation of UCIR by ATM [5, Theorem 4]. It essentially relies on a procedure **value** with two integer parameters v and i. Its task is to accept, if the i-th entry of gate v will get a 1 as input. In order to "simulate" a circuit for an input w, **value** is called with the number v_o of the output gate of the circuit (see the fourth paragraph of the introductory remarks of Section 4) and 1 (as the number of its only entry).

Before considering a detailed implementation of **value** for ACA consider the following "high level" description sequentially executing two steps **A)** and **B)** for given input parameters v and i:

A) universally do in parallel:

```
    1) $v' \leftarrow$ guess existentially the number of a gate
    2) $\tau' \leftarrow$ guess existentially the type of $v'$
```

```
B) universally do in parallel:
    1) check that during guess in A1) above all participating cells guessed
       that they should do something existentially
    2) do sequentially:
        21) check that the output of $v'$ is the $i$-th input of $v$
        22) check that $v'$ is of type $\tau'$
    3) depending on the type $\tau'$ do one of the following:
        31) for input gates: check that the corresponding input bit is 1
        32) if $\tau' = \vee$: sequentially do
            a) $i' \leftarrow$ guess existentially a number
            b) universally do in parallel
                i) check that value$(v', i')$ is 1
                ii) check that during guess in B32a) above all cells
                    guessed that they should do something existentially
        33) if $\tau' = \wedge$: (analogously to $\tau' = \vee$)
            a) $i' \leftarrow$ guess universally a number (i.e. check all numbers)
            b) universally do in parallel
                i) check that value$(v', i')$ is 1
                ii) check that during guess in B33a) above all cells
                    guessed that they should do something universally
```

The crucial point of the construction is to make sure that an ACA needs only a constant number of steps between the start of **value** and the recursive calls to **value** happening inside. For this to be true, some numbers (v', i') have to be guessed in constant time, although they consist of $k = \log \text{size}(C_n)$ bits. Furthermore the passing of parameters to the recursion must be done quickly.

This is achieved by using k subsequent cells, each of which guesses one bit. But there is a further complication. Sometimes a number has to be guessed existentially, e.g. in **B32a)**, but in **B33a)** numbers have to be guessed universally. The information which is the case is present at some cell, but it cannot be provided to all k guessing cells in constant time. The solution is as follows: Guessing is done in two steps. In the first one, each of the k cells guesses whether the number guessing has to be done existentially or universally, sets a flag indicating its choice and enters an existential or universal state accordingly. In the second step each cell guesses a bit. Afterwards, see **B32bii)** (resp. **B33bii)**), it is verified that all cells have correctly guessed that an existential (resp. universal) guess was needed. This can be done in k steps by sending τ' to the guessing cells.

This way the next recursion level is always called after a constant number of steps, and the time for verification only contributes a *summand* of $\log \text{size}(C_n) \in \mathrm{O}(s)$ to the overall execution time. All other considerations are analogous to the proof in [5]. ■

As a corollary one obtains:

Theorem 5.5 *For functions t and s satisfying the requirements of Lemmata 5.1 and 5.4 holds:*

$$\exists\forall\text{–}\textsc{Aca}\text{–}\textsc{Time}\text{–}\textsc{Space}(\mathrm{O}(t),\mathrm{O}(s)) = \textsc{UbUcir}\text{–}\textsc{Depth}\text{–}\textsc{Size}(\mathrm{O}(t),2^{\mathrm{O}(s)}) \ .$$

It is not known whether UbUcir of depth $\mathrm{O}(d)$ can be simulated by Ucir of depth $\mathrm{O}(d)$. The general expectation seems to be that this is not possible. Therefore Lemma 5.4 together with Theorem 4.2 can be taken as an indication, that a linear-time simulation of nonuniform Aca by Atm is “unlikely” to exist.

Furthermore one should note that in the comparisons of UbUcir and Atm (instead of Aca) the depth of the circuits usually corresponds to number of alternations (and *not* to the time of the Atm; see [4, Theorem 7.3.7]):

Theorem 5.6 *If the s and t are $\geq \log$ and constructible in time $\mathrm{O}(s)$:*

$$\textsc{UbUcir}\text{–}\textsc{Depth}\text{–}\textsc{Size}(\mathrm{O}(t),2^{\mathrm{O}(s)}) = \textsc{Atm}\text{–}\textsc{Alter}\text{–}\textsc{Space}(\mathrm{O}(t),\mathrm{O}(s)) \ .$$

To put it the other way round, because of Theorem 5.5 alternation depth of Atm is linearly related to the time of Aca:

Corollary 5.7 *If the functions s and t satisfy the prerequisites of Theorems 5.5 and 5.6:*

$$\exists\forall\text{–}\textsc{Aca}\text{–}\textsc{Time}\text{–}\textsc{Space}(\mathrm{O}(t),\mathrm{O}(s)) = \textsc{Atm}\text{–}\textsc{Alter}\text{–}\textsc{Space}(\mathrm{O}(t),\mathrm{O}(s)) \ .$$

6 Conclusions

The results obtained above together with those proved in [3] lead to the relatively simple situation depicted in Figure 4. An arrow from A to B with a label $\mathrm{O}(f(t))$ indicates that each machine of type A with time complexity t can be simulated by a machine of type B with time complexity $\mathrm{O}(f(t))$. Thick lines indicate results obtained in this paper.

$$\begin{tikzcd}
 & \forall\exists\text{–}\textsc{Aca} \arrow[r, leftrightarrow, "\mathrm{O}(t)"] & \exists\forall\text{–}\textsc{Aca} \arrow[r, leftrightarrow, "\mathrm{O}(t)"] \arrow[d, "\mathrm{O}(t^2)"] \arrow[dr, "\mathrm{O}(t^2)"] & \textsc{UbUcir} \\
\textsc{Waca} \arrow[r, leftrightarrow, "\mathrm{O}(t)"'] & \textsc{Uaca} \arrow[ur, "\mathrm{O}(t)"] \arrow[r, leftrightarrow, "\mathrm{O}(t)"'] & \textsc{Ucir} \arrow[r, leftrightarrow, "\mathrm{O}(t)"'] & \textsc{Atm}
\end{tikzcd}$$

Figure 4: Simulations between different ACA, UCIR and ATM

Thus, as long as some standard constructibility requirements are fulfilled, uniform ACA correspond to circuit families with bounded fan-in and nonuniform ACA correspond to circuit families with unbounded fan-in.
CA

are usually defined in such a way that it takes n steps before each input symbol could possibly have an influence on the outcome of the computation. Therefore the above results only hold for time complexities $t \geq n$. On the other hand for uniform circuit families there is usually some interest in sublinear, e.g. polylogarithmic depths. It is therefore natural to look at modifications of (A)CA which allow sublinear computation times and to extend the results presented in this paper to a wider range of resource bounds. This will be done in a forthcoming paper.

References

[1] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[2] Kamala Krithivasan and Meena Mahajan. Nondeterministic, probabilistic and alternating computations on cellular array models. *Theoretical Computer Science*, 143(1):23–49, May 1995.

[3] Martín Matamala. Alternation on cellular automata. *Theoretical Computer Science*, 180(1-2):229–241, 1997.

[4] Karl Rüdiger Reischuk. *Einfürung in die Komplexitätstheorie.* B.G. Teubner, Stuttgart, 1990.

[5] Walter L. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22(3):365–383, 1981.