

RAIRO

INFORMATIQUE THÉORIQUE

W. BUCHER

K. CULIK II

On real time and linear time cellular automata

RAIRO – Informatique théorique, tome 18, n° 4 (1984), p. 307-325.

http://www.numdam.org/item?id=ITA_1984__18_4_307_0

© AFCET, 1984, tous droits réservés.

L'accès aux archives de la revue « RAIRO – Informatique théorique » implique l'accord avec les conditions générales d'utilisation (<http://www.numdam.org/legal.php>). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques

<http://www.numdam.org/>

ON REAL TIME AND LINEAR TIME CELLULAR AUTOMATA (*)

by W. BUCHER ⁽¹⁾ and K. CULIK II ⁽²⁾

Communicated by J. BERSTEL

Abstract. — *The recognition power of one-way and two-way cellular automata under various time restrictions is compared. For example, it is shown that an arbitrary linear time is equivalent to $2n$ -time in this sense. Generalized cellular automata (GCA) are introduced and it is shown that real time GCA are equivalent to $2n$ -time cellular automata. Various restricted classes of GCA are shown to be equivalent to GCA.*

Résumé. — *Nous considérons les classes des langages acceptés par des automates cellulaires « one-way » et « two-way » sous diverses restrictions de temps. Par exemple, nous démontrons que la classe des langages reconnus en temps linéaire quelconque est la même que la classe reconnue en temps $2n$. On introduit des automates cellulaires généralisés (GCA) et on démontre que les GCA reconnaissant en temps réel sont équivalents aux automates cellulaires travaillant en temps $2n$. Nous démontrons que les GCA soumis à diverses restrictions restent équivalents aux GCA généraux.*

1. INTRODUCTION

Motivated by the advent of *VLSI* technology there has been increasing interest in devices for parallel computation, in particular systolic arrays, cf. [10]. The simplest, and presently practically probably the most important one, is a 1-dimensional array of uniformly interconnected identical processors working in synchronous manner. Various abstract models of this type which differ in the way the inputs are read and the processors are interconnected have been studied already for long time. We will study the computational power of the real- and linear time cellular automata ([1, 8, 3, 11, 12]), of the linear time one-way cellular automata ([1, 6, 8, 12]), and investigate some of their closure properties.

(*) Received in May 1983, revised in October 1983.

This work has been done during the second author's visit at the Institute für Informationsverarbeitung Graz, Austria.

⁽¹⁾ Institute für Informationsverarbeitung, Technical University of Graz, A-8010 Graz, Austria.

⁽²⁾ Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada.

The paper is organized as follows:

Section 2 introduces the basic concepts of cellular automata (*CA*), one-way cellular automata (*OCA*), and trellis automata. Two lemmas are proved in this section, the first one relating different time complexity classes, the second one allowing to restart computations on cellular automata. In the next section we show that the class of linear time *OCA* languages and the classes of real time and linear time *CA* languages are closed under length-multiplying morphisms and under inverse morphisms. The same technique is also used to show that for $c_1, c_2 > 1$, c_1n -time *OCA* (*CA*) and c_2n -time *OCA* (*CA*) are equivalent. This also implies that for each $c > 1$ the class of cn -time *OCA* (*CA*) languages is equal to the class of linear time *OCA* (*CA*) languages. In the last section we introduce the generalized cellular automata (*GCA*). A *GCA* differs from a *CA* in the way the accepting configurations are specified. A cellular automaton accepts a string in time $T(n)$ if the rightmost symbol computed at this time (from the input string) is an accepting one. A *GCA* accepts an input string if the string of symbols computed at time $T(n)$ belongs to a given *CA* language. We show that the real time *GCA* are equivalent to the $2n$ -time *CA* automata and therefore also to the linear time *CA* automata. This new class of cellular automata is of interest because of the robustness of its definition, we show that various alternative types of cellular automata (accepting in the middle, accepting anywhere, accepting by specific regular patterns) are all equivalent to *GCA*. Actually, all these classes can be formally described as special classes (normal forms) of *GCA*. In the closing remarks we discuss a candidate for showing that the real-time *CA* languages are not closed under reversal — an open problem [11]. Proving this would imply the proper inclusion of the real time *CA* languages in the real time *GCA* languages (linear time *CA* languages) which are closed under reversal.

We then summarize what is known about the closure under reversal of various language classes recognized by a linear array of processors, and show a diagram summarizing their inclusion relations shown in [11, 1, 4] and here.

2. DEFINITIONS AND BASIC LEMMAS

Throughout this paper we will assume that # is a special symbol not contained in any of the alphabets considered.

Three types of devices are studied in this paper: cellular automata, one-way cellular automata and trellis automata. They have already been studied

before by several authors, e. g. [11, 2, 3, 6, 12]. In [1] common formalism was developed for these objects which allows to define them conveniently in terms of "local" functions. We will follow their notation, the essence being contained in the following definitions. \mathbb{N} (\mathbb{R} , resp.) denotes the set of natural (real, resp.) numbers.

DEFINITION 2.1: A *cellular automaton* (CA for short) is a system $C = (\Sigma, A, A_0, M)$, where $A_0 \subseteq A$, $\Sigma \subseteq A$ are alphabets and M is a function $M: (A \cup \{ \# \})^3 \rightarrow A \cup \{ \# \}$ satisfying

$$M(a, b, c) = \# \quad \text{if and only if} \quad b = \#.$$

M induces a length preserving $m: A^+ \rightarrow A^+$ in the following way:

$$m(a) = M(\#, a, \#)$$

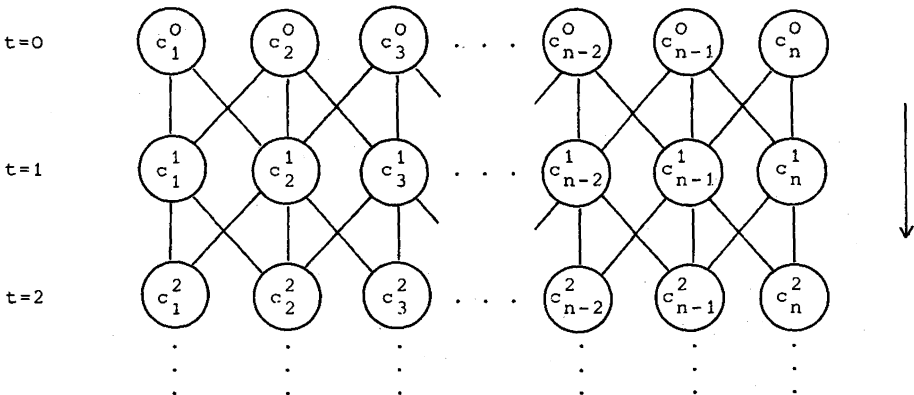
$$m(a_1 a_2 \dots a_n) = M(\#, a_1, a_2) \dots M(a_{n-1}, a_n, \#) \text{ for } n \geq 2.$$

Let $f: \mathbb{N} \rightarrow \mathbb{N}$ be a function and L a subset of Σ^+ . We say that L is accepted by C in time $f(n)$ if and only if

$$L = \{ w \in \Sigma^+ \mid m^{f(|w|)}(w) \in A^* A_0 \}.$$

We say that $f: \mathbb{N} \rightarrow \mathbb{N}$ is *CA-constructible*, if there is a CA $C = (\{ a \}, A, A_0, M)$ such that $m^k(a^n) \in A^* A_0$ if and only if $k = f(n)$. The class of all languages accepted by CA's in time $f(n)$ will be denoted by $CA(f(n))$.

It is often convenient to represent the functioning of a cellular automaton on a time-space diagram (cf. [7, 11]), each row corresponding to the configuration of an n -cell array at a certain time $t \geq 0$. At time $t=0$, the i -th cell contains the i -th letter of the input word w . As time goes by the different values are computed according to the transition function M . We denote by c_i^t the value of cell i at time $t \geq 0$, i. e. $c_i^t = M(c_{i-1}^{t-1}, c_i^{t-1}, c_{i+1}^{t-1})$.



In one-way cellular automaton information is not allowed to flow both to the left and to the right, but (usually) only to the left. Formally we obtain

DEFINITION 2.2: A one-way cellular automaton (OCA for short) is a system $O = (\Sigma, A, A_0, M)$ where Σ , A_0 and A are as in Definition 2.1 and $M : (A \cup \{ \# \})^2 \rightarrow A \cup \{ \# \}$ is a transition function satisfying $M(a, b) = \#$ if and only if $a = b = \#$. As before we extend M to $m : A^+ \rightarrow A^+$ by putting

$$m(a) = M(a, \#) \text{ and}$$

$$m(a_1 \dots a_n) = M(a_1, a_2)M(a_2, a_3) \dots M(a_n, \#) \text{ if } n \geq 2.$$

Given a function $f : \mathbb{N} \rightarrow \mathbb{N}$ and a subset $L \subseteq \Sigma^+$ we say L is accepted by the OCA in time $f(n)$ if and only if

$$L = \{ w \in \Sigma^+ \mid m^{f(|w|)}(w) \in A_0 A^* \}.$$

The notions OCA-constructible function and OCA($f(n)$) are defined as for CA and a time-space diagram may be used similar to the one in figure 1, but now the arrows from cells c_i^t , c_{i-1}^{t+1} have to be omitted. Accepting cell is the leftmost one.

It was shown in [1] that OCA automata are equivalent in accepting power to the (homogenous) trellis automata introduced in [2].

DEFINITION 2.3: A trellis automaton (TA for short) is a system $T = (\Sigma, A, A_0, M)$, where Σ , A_0 and A are as before and $M : (A \cup \{ \# \})^2 \rightarrow A \cup \{ \# \}$ is a transition function satisfying $M(a, b) = \#$ if and only if $a = b = \#$.

Given any word $w = a_1 \dots a_n \in (A \cup \{ \# \})^+$, $n \geq 2$, we define

$$m(a_1 \dots a_n) = M(a_1, a_2) \dots M(a_{n-1}, a_n).$$

A subset $L \subseteq \Sigma^+$ is said to be accepted by T in time $f(n)$, if and only if

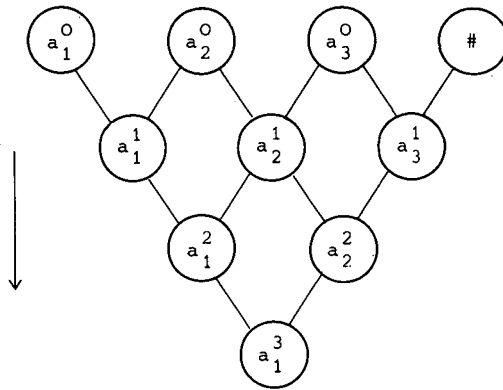
$$L = \{ w \in \Sigma^+ \mid m^{f(|w|)}(w \#^{f(|w|) - |w| + 1}) \in A_0 \}.$$

As in the preceding definitions, TA($f(n)$) is the class of all languages accepted in time $f(n)$ by trellis automata and the notion of TA-constructability carries over as well.

The recognition of word $w \in L$ can be viewed as a computation on an actual trellis as follows.

The word $w \#^{f(|w|) - |w| + 1}$ is fed at the top level of the trellis. Each node computes a value depending on the inputs it receives from the two upper nodes directly connected to it and sends it to the two neighbours just below. The computation can be imagined as flowing down level by level from the top to the bottom whose value decides whether the word is accepted or not.

In our definitions languages have to be accepted "exactly" in time $f(n)$. However, sometimes in complexity theory, acceptance "within time $f(n)$ "



is considered. In fact, for the devices and time functions considered in this paper these forms of acceptance are equivalent. Let for $X \in \{ CA, OCA, TA \}$, $X(\leq f(n))$ denote the class of languages accepted by a device X within time $f(n)$. Then the following holds.

PROPOSITION 2.1: *Let $X \in \{ CA, OCA, TA \}$ and $T_1, T_2 : \mathbb{N} \rightarrow \mathbb{N}$ be X -constructible and assume that for all $n \in \mathbb{N}$ $T_1(n) \leq T_2(n)$.*

Then $X(\leq T_1(n)) \subseteq X(\leq T_2(n))$ and $X(T_1(n)) \subseteq X(\leq T_2(n))$. (In other words, $X(\leq T_1(n)) = X(T_1(n))$ and $X(T_1(n)) \subseteq X(T_2(n))$.)

Proof. — We give a proof merely for CA , the other proofs are similar and left to the reader. Let $L \in CA(\leq T_1(n))$, i.e. there is a $CAC = (\Sigma, A, A_0, M)$ such that $x \in L \cap \Sigma^+$ if and only if there is $k \leq f(|x|)$ such that $m^k(x) \in A^*A_0$. Now, because of the CA -constructibility of $T_1(n)$ and $T_2(n)$, there are CA 's $C_i = (\{a\}, A_i, A_{i,0}, M_i)$ such that $m_i^k(a^n) \in A_i^*A_{i,0}$ if and only if $k = T_i(n)$. Define a CA C' as follows.

$$C' = (\Sigma, A', A'_0, M')$$

$$A' = A \cup (A \cup \{Y, N\}) \times (A_1 \cup \{Y, N\}) \times A_2$$

$$A'_0 = \{Y\} \times \{Y\} \times A_{2,0}$$

$$M'(a_1, a_2, a_3) = (M(a_1, a_2, a_3), M_1(b_1, b_2, b_3), M_2(b_1, b_2, b_3)) \text{ where}$$

(initialization) $b_i = \#$ if $a_i = \#$ and $b_i = a$ if $a_i \in A$.

Informally, C' works like C on the first track, like C_1 on the second and like C_2 on the third, and if the rightmost cell contains an accepting symbol in its first track at a time $k \leq T_1(|x|)$, Y is kept in the first track till time $T_1(|x|)$, from time $T_1(|x|)$ till $T_2(|x|)$ this symbol Y is then kept in the second track;

if Y is not contained in the first track at time $T_1(|x|)$, then N is propagated in the second track. We assume that Y, N are not in $A \cup A_1 \cup A_2$.

Formally we put

$$M'((a_1, a_2, a_3), (b_1, b_2, b_3), (c_1, c_2, c_3)) = (M(a_1, b_1, c_1), M_1(a_2, b_2, c_2), M_2(a_3, b_3, c_3)) \\ \text{if } (c_1, c_2, c_3) \neq (\#, \#, \#)$$

$$M'((a_1, a_2, a_3), (b_1, b_2, b_3), (\#, \#, \#)) = (M(a_1, b_1, \#), M_1(a_2, b_2, \#), M_2(a_3, b_3, \#)) \\ \text{if } M(a_1, b_1, \#) \notin A_0$$

$$M'((a_1, a_2, a_3), (b_1, b_2, b_3), (\#, \#, \#)) = (Y, Z, M_2(a_3, b_3, \#)) \text{ if } M(a_1, b_1, \#) \in A_0.$$

where $Z = Y$ if $M_1(a_2, b_2, \#) \in A_{1,0}$ and $Z = M_1(a_2, b_2, \#)$ otherwise.

$$M'((a_1, a_2, a_3), (b_1, b_2, b_3), (\#, \#, \#)) = (M(a_1, b_1, \#), N, M_2(a_3, b_3, \#)) \\ \text{if } M(a_1, b_1, \#) \notin A_0, b_1 \neq Y, \text{ and } M_1(a_2, b_2, \#) \in A_{1,0}.$$

$$M'((a_1, a_2, a_3), (Y, b_2, b_3), (\#, \#, \#)) = (Y, Z, M_2(a_3, b_3, \#)) \text{ where} \\ Z = Y \text{ if } b_2 = Y \text{ or } M_1(a_2, b_2, \#) \in A_{1,0}, \\ Z = N \text{ if } b_2 = N, \\ Z = M_1(a_2, b_2, \#) \text{ otherwise.}$$

Symbols Y and N may be treated arbitrarily in cells left to the rightmost one. This shows that $X(\leq T_1(n)) \subseteq X(T_2(n))$.

The proof that $X(T_1(n)) \subseteq X(\leq T_2(n))$ is quite similar: one only has to produce a special accepting symbol Y exactly at time $T_1(n)$ and at no other time. This guarantees that no other words are accepted within time $T_2(n)$. Since the construction is similar to the one in the first step, we omit the details. \square

The following relationships among the various complexity classes have been established in [1] and [11].

LEMMA 2.1:

$$(i) \quad CA(T(n)) = CA\left(\frac{T(n)}{k} + n\right) \text{ for } k \in \mathbb{N}, [11]$$

$$(ii) \quad OCA(T(n)) = TA(T(n)), [1]$$

$$(iii) \quad CA(n) = OCA(2n), [1]$$

$$(iv) \quad X(T(n)) = X(T(n) + 1), \text{ for } X \in \{CA, OCA, TA\}, [1].$$

Here $T(n)$ is an arbitrary function from the natural numbers to the natural numbers satisfying $T(n) \geq n - 1$. Note that this restriction is necessary for TA and meaningful for CA and OCA , since otherwise the output would be independent of the first input symbol. Of special importance for our investigations are the functions $T(n) = n$ giving rise to the classes of real-time X -languages and $T(n) = \lfloor cn \rfloor$ giving rise to classes of linear time X -languages.

It was shown in [6] that if $L \in OCA(n)$ then $L\mathcal{L}^* \in OCA(n)$. Since $OCA(n)$ is

closed under reversal [1], we immediately obtain the following lemma needed in the following sections.

LEMMA 2.2: *OCA(n) is closed under reversal and if Σ_1, Σ_2 are arbitrary alphabets and $L \in OCA(n)$, then $\Sigma_1^* L \Sigma_2^* \in OCA(n)$.*

In the proofs of section 4 we need to be able to restart computation on a CA after $|x|$ time steps, where x is an arbitrary input. The difficulty is that each cell at time $|x|$ has to know that a new computation starts at that time. Fortunately, this difficulty of synchronizing the cells may be overcome by a suitable modification of an optimal solution to the firing squad problem (see [13]).

LEMMA 2.3 (Synchronization Lemma): *Let Σ be an alphabet and ϕ a symbol not in Σ . There exists a CA $C = (\Sigma, A, A_0, M)$ such that for all $x \in \Sigma^+$ the following holds:*

- (i) $m^{|x|}(x) = \phi^{|x|}$
- (ii) if $0 \leq k < |x|$ then $m^k(x) \notin A^* \phi A^*$.

Proof: We modify an optimal solution of the firing squad problem: Given letters a, b, c there exists a CA $C_1 = (\{a, b\}, A_1, A_{1,0}, M_1)$ such that

- (i) $m_1^{2n}(ba^n) = c^{n+1}$, (ii) if $0 \leq k < 2n$, then $m_1^k(ba^n) \notin A^* c A^*$, and (iii) $M_1(a, a, a) = M_1(a, a, \#) = a$. (See [13, 11]).

The idea of the proof now is to divide the input string in two equally-sized words and apply the above mentioned firing squad algorithm C_1 to the left subword and a reversed version to the right subword.

Formally, let $C_2 = (\{a, b\}, A_1, A_{1,0}, M_2)$ be the CA defined by $M_2(a_1, a_2, a_3) = M_1(a_3, a_2, a_1)$. Then

- (i) $m_2^{2n}(a^n b) = c^{n+1}$, (ii) if $0 \leq k < 2n$, then $m_2^k(a^n b) \notin A^* c A^*$, and (iii) $M_2(a, a, a) = M_2(\#, a, a) = a$.

Let $C = (\Sigma, A, A_0, M)$ be the following CA. Since we do not accept words, we do not specify A_0 .

$A = \Sigma \cup \{c', \phi, a\} \cup A_1 \times (\{l, \bar{l}, r, \bar{r}\} \cup A_1)$, where $c', \phi, l, \bar{l}, r, \bar{r}$ are not in $A_1 \cup \Sigma$.

Initialization:

- $M(a_1, a_2, a_3) = a$
- $M(\#, a_1, a_2) = (b, l)$
- $M(a_1, a_2, \#) = (b, r)$, if $a_1, a_2, a_3 \in \Sigma$.

The signal $l(r)$ propagates to the right (left) at unit speed till it reaches the middle while in the first track the firing squad computation takes place. Note that this simulation only works because (iii) holds for C_1 and C_2 .

$$M(a, a, a) = a$$

$$M((a_1, l), (a_2, l), (a_3, l)) = M((a_1, l), (a_2, l), a_3) = M((a_1, l), a_2, a_3) \\ = (M_1(a_1, a_2, a_3), l) \text{ for } a_1, a_2, a_3 \in A_1$$

$$M((a_1, r), (a_2, r), (a_3, r)) = M(a_1, (a_2, r), (a_3, r)) = M(a_1, a_2, (a_3, r)) \\ = (M_2(a_1, a_2, a_3), r) \text{ for } a_1, a_2, a_3 \in A_1$$

$$M(\#, (a_1, l), (a_2, l)) = M(\#, (a_1, l), a_2) = (M_1(\#, a_1, a_2), l)$$

$$M((a_1, r), (a_2, r), \#) = M(a_1, (a_2, r), \#) = (M_2(a_1, a_2, \#), r).$$

According to whether n is odd or even, one of the following situations will occur when the signals reach the middle:

$$M((a_1, l), a, (a_2, r)) = (M_1(a_1, a, \#), M_2(\#, a, a_2))$$

or two signals l and r are in the second tracks of two adjacent cells:

$$M((a_1, l), (a_2, l), (a_3, r)) = (M_1(a_1, a_2, \#), l)$$

$$M((a_1, l), (a_2, r), (a_3, r)) = (M_2(\#, a_2, a_3), r).$$

For odd n now barred symbols \bar{l} and \bar{r} are propagated to the sides at unit speed.

$$M((a_1, l), (a_2, l), (a_3, a_4)) = (M_1(a_1, a_2, a_3), \bar{l})$$

$$M((a_1, a_2), (a_3, r), (a_4, r)) = (M_2(a_2, a_3, a_4), \bar{r})$$

$$M((a_1, \overset{(-)}{l}), (a_2, a_3), (a_4, \overset{(-)}{r})) = (M_1(a_1, a_2, \#), M_2(\#, a_3, a_4))$$

$$M((a_1, \overset{(-)}{l}), (a_2, \overset{(-)}{l}), (a_3, \bar{l})) = (M_1(a_1, a_2, a_3), \bar{l})$$

$$M((a_1, \bar{r}), (a_2, \overset{(-)}{r}), (a_3, \overset{(-)}{r})) = (M_2(a_1, a_2, a_3), \bar{r}).$$

In the above transitions it is assumed that none of the arguments has first component c .

Finally,

$$M(\#, (c, l), (c, l)) = M((c, l), (c, l), (c, l)) = \dots \\ = M((c, r), (c, r), \#) = c'$$

$$M(\#, c', c') = M(c', c', c') = M(c', c', \#) = \phi$$

in the even case and

$$M(\#, (c, l), (c, \bar{l})) = M((c, \overset{(-)}{l}), (c, \bar{l}), (c, \bar{l})) = M((c, \bar{l}), (c, \bar{l}), (c, c)) = \dots \\ = M((c, \bar{r}), (c, r), \#) = \phi$$

in the odd case simulate the firing. Note that in the even case a waiting step has to be inserted.

Some special cases may occur for small n , but we leave the appropriate definitions to the reader, and also the straightforward verification that (i) and (ii) hold for C . \square

3. CLOSURE PROPERTIES OF LINEAR TIME CA AND OCA

Various closure properties for cellular and one-way cellular automata have been studied in the literature. Using a two-track technique, it is readily seen that for arbitrary functions $f(n)$ the classes $CA(f(n))$ and $OCA(f(n))$ are Boolean algebras (see [2, 11, 6]). It was shown in [1] that $OCA(n)$ is closed under reversal and in [11] that the class of linear time CA languages, that is the class of languages L such that there is $c, c \in \mathbb{R}$, with the property that $L \in CA(\lceil cn \rceil)$, is also closed under reversal. In view of Prop. 2.1 (i) and the easily established fact that $CA(f(n)) \subseteq CA(kf(n))$ for arbitrary $k \in \mathbb{N}$ and $f(n)$, this is equivalent to saying that $CA(\lceil cn \rceil)$ is closed under reversal for $c \in \mathbb{R}, c > 1$. However, for the class $CA(n)$ of real-time CA languages the reversal problem is open so far, and it is also open for classes of OCA languages different from $OCA(n)$.

In this section we give two closure results for linear time CA and OCA languages, namely, we prove that the classes $OCA(\lceil cn \rceil)$ and $CA(\lceil cn \rceil)$ are closed under inverse morphisms and injective length-multiplying morphisms, where a morphism ϕ is called length-multiplying if the images $\phi(a)$ of all letters have the same length.

The main tool for proving these facts are the following two lemmas:

LEMMA 3.1 [2]: *$OCA(n)$ is closed under inverse morphisms and injective length-multiplying morphisms.*

LEMMA 3.2: *Let $L \subseteq \Sigma^+, b$ a letter not in Σ and $c \geq 1$ a real number such that $f(n) = \lceil cn \rceil$ is constructible for the respective device. Then*

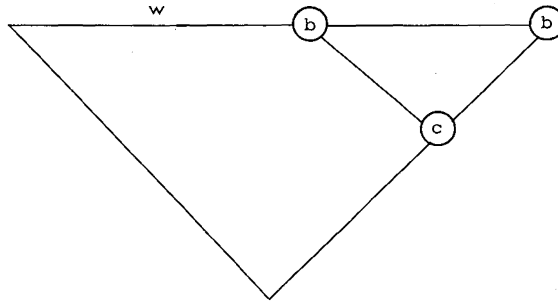
- (i) $L \in OCA(f(n))$ if and only if $\{wb^{f(|w|)-|w|} \mid w \in L\} \in OCA(n)$.
- (ii) $L \in CA(f(n))$ if and only if $\{b^{f(|w|)-|w|}wb^{f(|w|)} \mid w \in L\} \in OCA(n)$.

Proof: The technique for simulating CA's and OCA's by trellis automata, ([1]), actually shows the only if-direction. We only have to treat the b 's as blanks $\#$. This works without further restrictions for CA's since then in the simulating trellis automaton the actual processing and accepting is done within the space corresponding to w . (Note that treating b as $\#$ has the consequence that outside this space no other letter than b may occur in the time-space diagram; and b is not accepting symbol). However, if we simulate our OCA by a trellis automaton, some words wb^k might be accepted where $k \neq f(|w|) - |w|$. Therefore we have to intersect the language accepted by the simulating trellis automaton with $\{wb^{f(|w|)-|w|} \mid w \in \Sigma^+\}$, which is in

$OCA(n)$ since it is a linear language. We leave the formal constructions to the reader.

To prove the converse we have to settle the problem that within the b blocks some computation might take place which is forbidden for $\#$'s.

(i) Assume $L_1 = \{wb^{f(|w|-|w|)} \mid w \in L\}$ is in $OCA(n)$, and consequently in $TA(n) = TA(n-1)$. Let $T = (\Sigma \cup \{b\}, A, A_0, M)$ be a trellis automaton accepting L_1 in time $n-1$ and consider the actual trellis corresponding to $wb^{f(|w|-|w|)}$.



The processing in the time-space triangle defined by the three nodes labelled b, b and c is independent of w . Moreover, the node labels on the line connecting the left node b and the node c form an ultimately periodic sequence independent of w and the node labels on each level in the triangle are the same. We now define a trellis automaton $T_1 = (\Sigma, A_1, A_{1,0}, M_1)$ in the following way. We may assume that $b \notin M(A \times A)$.

$$A_1 = A \cup A \times A$$

$$A_{1,0} = A_0$$

$$M_1(a_1, a_2) = M(a_1, a_2) \quad \text{if } a_1, a_2 \neq b$$

$$M_1(a, \#) = (M(a, b), M(b, b)) \quad \text{if } a \neq b$$

$$M_1(a_1, (a_2, a_3)) = M(a_1, a_2)$$

$$M_1((a_1, a_2), \#) = (M(a_1, a_2), M(a_2, a_2))$$

$M_1(x, y, z) = b_0$ otherwise, where b_0 is a sink symbol. This proves that $L \in OCA(f(n))$.

(ii) Let $L_1 = \{b^{f(|w|-|w|)}wb^{f(|w|)} \mid w \in L\}$ be in $OCA(n) = TA(n-1)$. The proof that L is in $CA(f(n))$ is essentially an application of the folding technique developed in [5], see also [1]. However to avoid too complicated constructions we first reduce the language L_1 . The case $c = 1$ has because of $CA(n) = OCA(2n)$ already been settled by (i). Assume first that $c \geq 2$ and let $k \geq 2$ be the least integer such that $f(n) \leq kn$. Then $L_2 = \{b^{(k-1)|w|}wb^{2(k-1)|w|} \mid w \in L\}$ is an $OCA(n)$ language, since it is the intersection of the three $OCA(n)$ languages

$b^*L_1b^*, \{b^{(k-1)|w|} | w \in \Sigma^+\} b^*$, and $b^* \{wb^{2(k-1)|w|} | w \in \Sigma^+\}$. Define a morphism $\varphi : (\Sigma \cup \{b\})^* \rightarrow (\Sigma \cup \{b\})^*$ by $\varphi(a) = a$ for $a \in \Sigma$ and $\varphi(b) = b^{k-1}$. Then $\varphi^{-1}(L_2) = \{b^{|w|}wb^{2|w|} | w \in L\}$ is an $OCA(n)$ language. Because of $CA(2n) \subseteq CA(\lceil cn \rceil)$ for $c \geq 2$, we will be done if we can show that L is in $CA(f(n))$ for $c \leq 2$.

Let $T = (\Sigma \cup \{b\}, A, A_0, M)$ be a trellis automaton accepting L_1 in time $n - 1$. We fold the trellis three times and then double the speed to obtain a CA accepting L in time $f(n)$. Note that the construction is the same for each $c \leq 2$, since appropriate timing is provided by the function $f(n)$. Informally, the folding is done in the following way:

$$b^n a_1 \dots a_n b^{2n} \quad \blacktriangleright \quad \begin{array}{c} \underline{b \quad \dots \quad b} \\ | \\ \underline{a_1 \quad \dots \quad a_n} \\ | \\ b \quad \dots \quad b \\ | \\ \underline{b \quad \dots \quad b} \end{array}$$

Formally we put $C = (\Sigma, A_1, A_{1,0}, M_1)$ where

$$A_1 = \Sigma \cup A \times A \times A \times A$$

$$A_{1,0} = A \times A_0 \times A \times A.$$

Let for $x, y, z \in A$ $g(x, y, z) = M(M(x, y), M(y, z))$.

$$M_1(a_1, a_2, a_3) = (g(b, b, b), g(a_1, a_2, a_3), g(b, b, b), g(b, b, b))$$

$$M_1(\#, a_1, a_2) = (g(b, b, a_1), g(b, a_1, a_2), g(b, b, b), g(b, b, b))$$

$$M_1(a_1, a_2, \#) = (g(\#, b, b), g(a_1, a_2, b), g(a_2, b, b), g(b, b, \#))$$

$$M_1(b_1, c_1, d_1, e_1), (b_2, c_2, d_2, e_2), (b_3, c_3, d_3, e_3))$$

$$= (g(b_3, b_2, b_1), g(c_1, c_2, c_3), g(d_3, d_2, d_1), g(e_1, e_2, e_3))$$

$$M_1(\#, (b_1, c_1, d_1, e_1), (b_2, c_2, d_2, e_2))$$

$$= (g(b_2, b_1, c_1), g(b_1, c_1, c_2), g(d_2, d_1, e_1), g(d_1, e_1, e_2))$$

$$M_1((b_1, c_1, d_1, e_1), (b_2, c_2, d_2, e_2), \#)$$

$$= (g(\#, b_2, b_1), g(c_1, c_2, d_2), g(c_2, d_2, d_1), g(e_1, e_2, \#)),$$

where $a_i \in \Sigma$ and $b_i, c_i, d_i, e_i \in A$.

It is now straightforward to verify that C accepts L in time $f(n)$. □

THEOREM 3.1: *Let c be a real number, $c > 1$, and assume that $f(n) = \lceil cn \rceil$ is constructible for the respective device. Then*

- (i) $OCA(f(n)) = OCA(2n)$
- (ii) $CA(f(n)) = CA(2n)$.

Proof: (ii) is already known, however it may be reproved along the same

lines as (i) by use of Lemma 3.2 and the inverse morphism theorem for $OCA(n)$. We omit the details.

(i) The proof of Lemma 3.2 has already shown that $OCA(\lfloor cn \rfloor) = OCA(2n)$ for $c \geq 2$. Clearly $\left\lceil \frac{k+1}{k} n \right\rceil$ is OCA -constructible for an arbitrary natural number k , consequently because of Prop. 2.1 it is sufficient to show that $OCA(2n) \subseteq OCA\left(\left\lceil \frac{k+1}{k} n \right\rceil\right)$. Let $L \in OCA(2n)$ and let for $1 \leq m \leq k$ L_m be the language $\{wb^{|w|} \mid w \in L, |w| \equiv m(k)\}$. Then $L_m \in OCA(n)$. Since

$$\{wb^{|w|+k-m} \mid w \in \Sigma^+\}$$

is a linear language,

$$L_m b^{k-m} = L_m b^* \cap \{wb^{|w|+k-m} \mid w \in \Sigma^+\} \in OCA(n).$$

Define a morphism φ by $\varphi(a) = a$ for $a \in \text{alph}(L)$ and $\varphi(b) = b^k$. Then

$$\begin{aligned} \varphi^{-1}(L_m b^{k-m}) &= \{wb^{\frac{|w|+k-m}{k}} \mid w \in L, |w| \equiv m(k)\} \\ &= \{wb^{\lceil \frac{1}{k}|w| \rceil} \mid w \in L, |w| \equiv m(k)\} \in OCA(n). \end{aligned}$$

Consequently, $L' = \{wb^{\lceil \frac{1}{k}|w| \rceil} \mid w \in L\}$ is in $OCA(n)$ and this means $L \in OCA\left(\left\lceil \frac{k+1}{k} n \right\rceil\right)$. \square

THEOREM 3.2: *Let $X \in \{CA, OCA\}$ and let $c \geq 1$ be a real number such that $f(n) = \lfloor cn \rfloor$ is X -constructible. Then $X(\lfloor cn \rfloor)$ is closed under injective length-multiplying morphisms.*

Proof: The proof is an application of the preceding lemmas and is the same for CA and OCA . Therefore we restrict to CA .

Let $L \in CA(2n)$ and $\varphi : \text{alph}(L)^* \rightarrow \Delta^*$ be an injective length-multiplying morphism such that $|\varphi(a)| = k$ for $a \in \text{alph}(L)$. Let b be a letter not in $\text{alph}(L) \cup \Delta$ and extend φ to $(\text{alph}(L) \cup \{b\})^*$ by $\varphi(b) = b^k$. Because of Lemma 3.2 $\{b^{|w|}wb^{2|w|} \mid w \in L\}$ is in $OCA(n)$ and consequently

$$\{b^{k|w|}\varphi(w)b^{2k|w|} \mid w \in L\} = \{b^{|\varphi(w)|}\varphi(w)b^{2|\varphi(w)|} \mid w \in L\}$$

is in $OCA(n)$. But this is equivalent to $\varphi(L) \in CA(2n)$. \square

THEOREM 3.3: *Let $X \in \{CA, OCA\}$ and let $c \geq 1$ be a constant such that $f(n) = \lfloor cn \rfloor$ is X -constructible. Then $X(\lfloor cn \rfloor)$ is closed under inverse morphisms.*

Proof: Because of the preceding discussion and the fact that $OCA(2n) = CA(n)$,

it remains to show that $CA(2n)$ and $OCA(2n)$ are closed under inverse morphisms.

Let $L \in OCA(2n)$ and $\varphi : \Delta^* \rightarrow \text{alph}(L)^*$ be a morphism. Let $k \geq 1$ be a natural number such that for all $a \in \Delta$ $|\varphi(a)| \leq k$. Then for all $w \in L$

$$|\varphi^{-1}(w)| \geq \frac{|w|}{k}.$$

Let b be a letter not in $\Delta \cup \text{alph}(L)$ and extend φ by $\varphi(b) = b$. Since $L \in OCA(2n)$, $\{wb^{|w|} \mid w \in L\} \in OCA(n)$, and consequently $\{\varphi^{-1}(w)b^{|w|} \mid w \in L\} \in OCA(n)$. The number of b 's in $\varphi^{-1}(w)b^{|w|}$ is not greater than $k|\varphi^{-1}(w)|$, and therefore $\{\varphi^{-1}(w)b^{k|\varphi^{-1}(w)|} \mid w \in L\} = \{\varphi^{-1}(w)b^{|w|} \mid w \in L\} b^* \cap \{wb^{k|w|} \mid w \in \Delta^+\}$ is in $OCA(n)$. This is equivalent to saying $\varphi^{-1}(L) \in OCA((k+1)n) = OCA(2n)$.

If L is a $CA(2n)$ language, using lemma 3.2 we can conclude in the same way that $\{b^{|w|}\varphi^{-1}(w)b^{2|w|} \mid w \in L\}$ is an $OCA(n)$ language and therefore $b^* \{b^{|w|}\varphi^{-1}(w)b^{2|w|} \mid w \in L\} b^*$ is an $OCA(n)$ language. Since

$$\{b^{k|w|}w \mid w \in \Delta^+\} b^+ \cap b^+ \{wb^{(k+1)|w|} \mid w \in \Delta^+\} = \{b^{k|w|}wb^{(k+1)|w|} \mid w \in \Delta^+\}$$

is an $OCA(n)$ language, $\{b^{k|\varphi^{-1}(w)|}\varphi^{-1}(w)b^{(k+1)|\varphi^{-1}(w)|} \mid w \in L\}$ is also an $OCA(n)$ language. Therefore $\varphi^{-1}(L) \in CA((k+1)n) = CA(2n)$. \square

4. GENERALIZED CELLULAR AUTOMATA

As mentioned in section 2, it is an open problem whether the class of real time cellular automata languages is closed under reversal. From an automata point of view, the reversal of $CA(n)$ is precisely the class of languages accepted in real time by a CA where the accepting cell is the leftmost one. Formally, a reversed real time CA is a system $C = (\Sigma, A, A_0, M)$, where Σ, A, A_0, M are as for a CA , but $L \subseteq \Sigma^+$ is accepted by C in real time if and only if $L = \{w \in \Sigma^+ \mid m^{|w|}(w) \in A_0 A^*\}$. In fact, real-time CA and reversed real-time CA process data in the same way and they merely differ by the contents of the array at time n . This leads in a natural way to the following

DEFINITION 4.1: A *generalized cellular automaton* (GCA for short) is a system $C = (\Sigma, A, K, M)$, where Σ, A and M are as in Definition 2.1 and $K \subseteq A^+$ is a real-time CA language.

A language $L \subseteq \Sigma^+$ is said to be accepted by the generalized cellular automaton in time $f(n)$ if and only if

$$L = \{w \in \Sigma^+ \mid m^{f(|w|)}(w) \in K\},$$

where m is the extension of M as for cellular automata. $GCA(f(n))$ denotes the class of languages accepted by generalized cellular automata in time $f(n)$.

For the rest of this paper we are only interested in $GCA(n)$, so when saying L is accepted by a generalized cellular automaton we tacitly assume that $L \in GCA(n)$.

An obvious consequence of the definition of GCA is that the real-time CA languages and the reversed real-time CA languages are GCA languages. However, a stronger result holds.

LEMMA 4.1: $CA(2n) \subseteq GCA(n)$.

Proof: Let $L \in CA(2n)$. By the results of section 3, for a letter $b \notin \text{alph}(L)$ the language $L' = \{ b^{|w|} w b^{2|w|} \mid w \in L \}$ is an $OCA(n)$ language. Both

$$L_1 = b \{ b^{|w|} w b^{2|w|} \mid w \in L, |w| \equiv 1(2) \} b^2$$

and

$$L_2 = \{ b^{|w|} w b^{2|w|} \mid w \in L, |w| \equiv 0(2) \}$$

are $OCA(n)$ languages, consequently $L_3 = L_1 \cup L_2$ is an $OCA(n)$ language.

Let c be a letter not in $\{b\} \cup \text{alph}(L)$ and define a morphism $\varphi : (\{b, c\} \cup \text{alph}(L))^* \rightarrow (\{b\} \cup \text{alph}(L))^*$ by $\varphi(b) = b^2$, $\varphi(c) = b^4$, $\varphi(a) = a$

for $a \in \text{alph}(L)$. Then $\varphi^{-1}(L_3) \cap b^*(\text{alph}(L))^*c^* = \{ b^{\lceil \frac{|w|}{2} \rceil} w c^{\lceil \frac{|w|}{2} \rceil} \mid w \in L \}$ is an

$OCA(n)$ language. If $T = (\text{alph}(L), A, A_0, M)$ is a trellis automaton accepting $\varphi^{-1}(L_3)$ in time $n - 1$ we similarly to the construction in the proof of Lemma 3.2

fold the trellis using three tracks, the first corresponding to the b -block and the third to the c -block, and double the speed. The resulting $GCA(\text{alph}(L), A_1, K, M_1)$ accepts L in real time, where

$$K = \bigcup_{n \geq 0} A_1^{\lceil \frac{n}{2} \rceil - 1} A_{1,0} A_1^{\lceil \frac{n}{2} \rceil}, \quad A_{1,0} = A \times A_0 \times A.$$

The details are straightforward and left to the reader. □

The converse of Lemma 4.1 also holds.

LEMMA 4.2: $GCA(n) \subseteq CA(2n)$.

Proof: The proof is a direct consequence of the Synchronization Lemma and the definition of generalized cellular automaton.

Let $L \in GCA(n)$ be accepted by the generalized cellular automaton $C_1 = (\Sigma, A, K, M)$, let C_2 be a cellular automaton accepting K in real time, and let C_3 be a cellular automaton as in the Synchronization Lemma. A cellular automaton C_4 accepting L in $2n$ -time works as follows. Given $w \in \Sigma^+$,

C_4 on a first track works like C_1 and on a second track like C_3 . At time $|w|$ the second tracks of all the cells of the array contain ϕ . Now C_4 processes the content of the first track like C_2 and leaves unchanged the second track. If A_0 is the set of accepting symbols of C_2 , $A_0 \times \{\phi\}$ is the set of accepting symbols of C_4 . \square

Combining Lemmas 4.1 and 4.2 we obtain

THEOREM 4.1: $GCA(n) = CA(2n)$.

An immediate consequence of the results of section 3 and the above theorem is

THEOREM 4.2: $GCA(n)$ is a Boolean algebra closed under reversal, injective length-multiplying morphisms and inverse morphisms.

The proofs of Lemmas 4.1 and 4.2 show that in the definition of GCA we could have allowed that K be an arbitrary linear time CA language instead of real time. Theorem 4.1 then still would be valid.

The simulation technique in Lemma 4.2 also works if we consider $GCA(kn)$, where k is an arbitrary natural number, since the Synchronization Lemma enables us to start a new computation at time kn .

THEOREM 4.3: For arbitrary natural numbers k , $GCA(kn) = GCA(n)$.

Sometimes for practical applications the definition of GCA might be too general, in the sense that a more tractable content of the cellular array at time n would be more convenient. An example are the real time cellular automata, where K is required to be of the form A^*A_0 . However, we strongly believe that this special form of accepting language is not sufficient in general, in other words, that $CA(n)$ does not equal $GCA(n)$.

The rest of this section is devoted to showing that for three special types of CA languages K the thus restricted GCA 's are equivalent in accepting power to arbitrary GCA 's. One type actually was already considered in the proof of Lemma 4.1, where we constructed a GCA accepting in the middle.

DEFINITION 4.2:

(i) A $GCA(\Sigma, A, K, M)$ "accepts by regular sets" (is a $rGCA$ for short), if K is a regular subset of A^* .

(ii) A $GCA(\Sigma, A, K, M)$ "accepts in the middle" (is a $mGCA$ for short), if $K = \bigcup_{n \geq 1} A^{\lfloor \frac{n}{2} \rfloor - 1} A_0 A^{\lfloor \frac{n}{2} \rfloor}$, where A_0 is a subset of A .

(iii) A $GCA(\Sigma, A, K, M)$ "accepts anywhere" (is an $aGCA$ for short), if $K = A^*A_0A^*$, where A_0 is a subset of A .

For time functions $f(n)$ the classes $xGCA(f(n))$ are defined in the obvious way.

THEOREM 4.4:

$$rGCA(n) = mGCA(n) = aGCA(n) = GCA(n).$$

Proof: That $mGCA(n)$ is equal to $GCA(n)$ has already been shown in Lemma 4.1. Since $aGCA(n)$ by definition is contained in $rGCA(n)$, there remains to be shown that $GCA(n)$ is a subset of $aGCA(n)$, or equivalently, that $mGCA(n)$ is a subset of $aGCA(n)$. This is accomplished by sending a half-speed signal from the leftmost cell to the middle.

Let L be a language accepted by the $mGCA(n)$ $C = (\Sigma, A, K, M)$, where K is specified by the subset A_0 of A . Define an $aGCA(n)$ $C_1 = (\Sigma, A_1, K_1, M_1)$ in the following way.

$$A_1 = A \cup A \times \{A, 2\}, \text{ where we assume that } 1, 2 \text{ are not in } A$$

$$K_1 = A^\dagger(A_0 \times \{1, 2\})A^\dagger$$

$$M_1(a, b, c) = M(a, b, c)$$

$$M_1(\#, a, b) = (M(\#, a, b), 1) \quad \text{for } a, b \in \Sigma$$

$$M_1((a, 1), b, c) = M_1(a, b, c)$$

$$M_1(\#, (a, 1), b) = (M(\#, a, b), 2)$$

$$M_1((a, 2), b, c) = (M(a, b, c), 1)$$

$$M_1(x, y, z) = M(x', y', z') \text{ otherwise, where } x' \text{ equals } x \text{ for } x \in A \cup \{\#\} \text{ and } x' \text{ is the first component of } x \text{ otherwise.}$$

If not specified, symbols are assumed to be in A . We also assume without loss of generality that symbols of Σ are not generated by C at any time $t > 0$.

At time n the i -th cell contains an element of $A \times \{1, 2\}$ if and only if $i = \left\lceil \frac{n}{2} \right\rceil$, and consequently w is accepted by C_1 if and only if it is accepted by C . \square

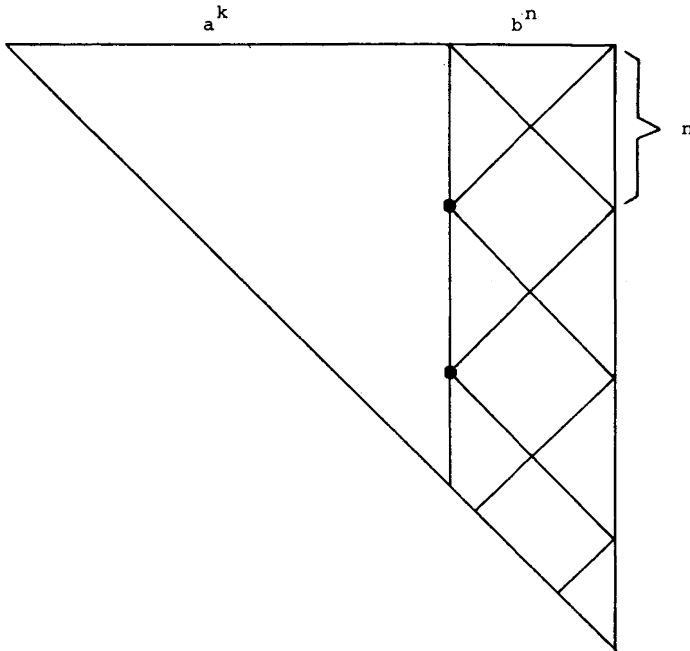
It is clear that by modifying the morphism φ in the proof of Lemma 4.1 we could also construct GCA 's accepting at the $[cn]$ -th cell for an arbitrary constant c , $0 < c < 1$, without losing accepting power. For various other restrictions on K this is also true. However, we do not discuss this here.

5. CONCLUSION

In section 3 we have shown a speed-up theorem for linear time OCA languages. The essential tool in the proof of this result was the transformation of a given language L to a real time OCA language L' (Lemma 3.2). In general such a transformation is not guaranteed. However, if we assume that $T(n)$ is OCA -constructible and $\{a^n b^{T(n)-n} \mid n \in \mathbb{N}\}$ is in $OCA(n)$, then it

can be shown in a similar way as in section 3 that for an arbitrary natural number k $OCA(T(n)) = OCA(k(T(n) - n) + n) = OCA\left(\left\lceil \frac{T(n) - n}{k} \right\rceil + n\right)$. This suggests the further investigation of real-time OCA languages corresponding to binary integer relations, namely sets of the form $\{a^k b^n \mid (k, n) \in R\}$ for some relation $R \subseteq \mathbb{N} \times \mathbb{N}$.

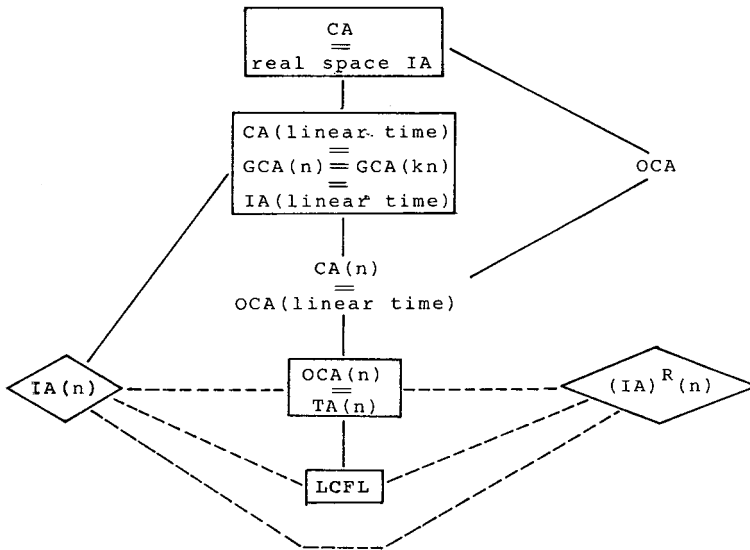
The work reported in this paper was initiated by the problem whether $CA(n)$ is closed under reversal, [11]. This problem remains open, we conjecture that it is not closed. A candidate for a language in $CA(n)$ but not in the reversal of $CA(n)$ is $L = \{a^k b^n \mid n, k \in \mathbb{N}, n \text{ divides } k\}$. L is in $CA(n)$, since we can bounce signals at unit speed in the b -block and create special symbols in the cell corresponding to the leftmost b at time $rn, r \in \mathbb{N}$, and test whether the signal coming from the leftmost a meets such a special symbol.



On the other hand, we conjecture that $\tilde{L} = \{a^k b^n \mid k, n \in \mathbb{N}, k \text{ divides } n\}$ is not in $CA(n)$. Note that a proof that $CA(n)$ is not closed under reversal would also give a proof for the (generally accepted) conjecture that linear time CA 's are more powerful than real-time CA 's, $CA(n) \not\subseteq CA(2n)$.

The following diagram summarizes what is known about the reversal problem

for various classes and also gives the set theoretic inclusion for the respective classes. Here \boxed{X} means that class X is closed under reversal, $\square X$ that X is not closed under reversal, and X that the problem is open. For the definition of iterative arrays (IA for short) we refer the reader to [4]. No time-specification for a class means arbitrary time, straight lines mean set theoretic inclusion and dotted lines incomparability. $(IA)^R(n)$ is the reversal of $IA(n)$ and $LCFL$ the class of linear context-free languages.



REFERENCES

1. C. CHOFFRUT, K. CULIK II, *On real-time cellular automata and trellis automata*, Research Report F 114, Institute für Informationsverarbeitung, Technical University of Graz, 1983.
2. K. CULIK II, J. GRUSKA & A. SALOMMAA, *Systolic trellis automata (for VLSI)*, Research Report CS-81-34, Dept. of Comp. Sci., University of Waterloo, 1981.
3. K. CULIK II, J. GRUSKA & A. SALOMMAA, *Systolic trellis automata: Stability, Decidability and Complexity*, Res. Rep. CS-82-04, Dept. of Comp. Sci., University of Waterloo, 1982.
4. S. N. COLE, *Real-time computation by n-dimensional iterative arrays of finite-state machines*, I.E.E.E. Trans. on Comp., Vol. 18, 1969, pp. 349-365.
5. K. CULIK II, J. PACHL, *Folding and Unrolling Systolic Arrays*, ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, Ottawa, August 1982.

6. C. R. DYER, *One Way Bounded Cellular Automata*, Inform. and Control, Vol. 44, 1980, pp. 261-281.
7. P. C. FISCHER, *Generation of primes by a one-dimensional real-time iterative array*, J. Assoc. Comput. Mach., Vol. 12, 1965, pp. 388-394.
8. F. C. HENNIE, *Iterative Arrays of Logical Circuits*, MIT Press, Cambridge Mass., 1961.
9. S. P. KOSARAJU, *On some open problems in the theory of cellular automata*, I.E.E.E. Trans. Computers, Vol. C-23, 1974, pp. 561-565.
10. H. T. KUNG, *Why Systolic Architecture?* Computer Magazine, January 1982.
11. A. R. SMITH III, *Real-time language recognition by one-dimensional cellular automata*, J. Comput. System Sci., Vol. 6, 1972, pp. 233-253.
12. H. UMEO, K. MORITA, K. SUGATA, *Deterministic one-way simulation of two-way real-time cellular automata and its related problems*, Information Proc. Letters, Vol. 14, 1982, pp. 158-161.
13. A. WAKSMAN, *An optimum solution to the firing squad synchronization problem*, Inform. and Control, Vol. 9, 1966, pp. 66-78.