Kuwait J. Sci. & Eng. **24**(2), 1997

Description of words by cellular automata

ANTON ČERNÝ

Department of Mathematics and Computer Sciences, Kuwait University, PO Box 5969
Safat 13060, Kuwait
cerny@math-1.sci.kuniv.edu.kw

ABSTRACT

We propose one- and two-dimensional cellular automata as a generative device aimed to describe (one-way) infinite one- and two-dimensional words. Each infinite word can be described by a single infinite computation generating its growing prefixes. The finitary one-dimensional version of the model describes precisely the class of context-sensitive languages.

INTRODUCTION

The classical theory of formal languages deals with finite sequences of symbols—words, strings which are linear (one-dimensional) in their character. In the present paper we will deal with *array words* (finite 2-dimensional words), *sequences* (infinite linear words), and trellises (infinite two-dimensional words) as well. Investigations of infinite words, being of importance e.g. for better understanding the regularities in long sequences of symbols (Lothaire 1983), started long before the "computing age" (Thue 1906, 1912). Trellises can be viewed as an infinite generalization of array words or, dually, as a two-dimensional generalization of sequences. They have been used for example, for description of placement of a heterogeneous network of processors (Culik II *et al.* 1984) proposed for studying parallel algorithms for VLSI implementation, and for description of the computational process of linear cellular automata (Choffrut & Culik II 1984, Ibarra *et al.* 1985b). Several properties of trellises with structural regularities have been further studied in Korec 1985, Černý & Gruska (1986a, 1986b).

Recognizing devices can be easily applied to 2-dimensional objects (Inoue & Takanami 1988). On the other hand, the standard rewriting tools for description of languages cannot always be directly generalized to the 2-dimensional case. No straightforward two-dimensional extension of the essential concept of a rewriting grammar seems to be at hand. Rewriting a symbol (or a subword) inside a 2-dimensional word by even a linear word would "push" some parts of the word apart, thus destroying the current context of symbols in the other dimension. Several mechanisms for controlling the rewriting have been designed (Siromoney *et al.* 1984, Subramanian *et al.* 1985). Rewriting just the border of a sentential form

(Lucanu 1987) does not seem to be satisfactory, since internal parts of words cannot be influenced.

The present work has been devoted to designing a rewriting tool based on the concept of cellular automata. Two-dimensional cellular automata can match the shape of array words and allow rewriting inside the word. On the other hand, cellular automata with a unidirectional information flow used as a deterministic rewriting tool are shown to be capable of stepwise generation of a reasonably large class of infinite words.

Cellular automata were introduced as a formal model featuring some crucial aspects of behavior of simple biological systems. Therefore the computational process of cellular automata in one or more dimensions became the subject of investigations (von Neumann 1966, Blum & Hewitt 1967, Cole 1969, Wolfram 1984b, 1985). From a formal languages point of view, cellular automata are more traditionally used as word recognizers (e.g. Ibarra *et al.* 1985a). In such a case, a word provided as input in some standard way (for the particular model), is accepted if the automaton enters a required accepting configuration. Cellular automata have been considered as word generators, as well (Wolfram 1984a). A word is considered to be generated by a cellular automaton if it is contained (in a required form) in some configuration entered by the automaton during its computation starting from a usually fixed initial configuration. In our approach, in accordance with the usual concept of a rewriting grammar, one computation will result in generating just one word. Our model will be defined for both the 1-dimensional and 2-dimensional case. We will investigate the 1-dimensional case only, and whenever the results can be easily extended, to two dimensions.

The model—generating cellular automaton—consists of cells organized in a sequence or a 2-dimensional rectangular grid. Each cell, responsible for rewriting one symbol of the resulting word, behaves as a finite-state device communicating with its neighbors. The communication is limited to the left-to-right (plus bottom-up in the 2-dimensional case) direction. This restriction enables a simulation of a generating cellular automaton by observing just the cells corresponding to the generated word. We will consider array words to be rectangular words, though our model—after some modification of its definition—could be suitable to generate words of irregular shape as well. The nondeterministic version of the model is shown to be strong enough to describe a reasonably wide class of finitary languages and context-sensitive languages. The infinite words described by the deterministic version include tag sequences (Cobham 1972), generalized Pascal triangles (Korec 1985), and modular trellises (Gruska 1984, Černý & Gruska 1986a).

BASIC NOTIONS AND NOTATIONS

Throughout this paper the symbol ρ will represent the dimension of the case under consideration. Though the results valid for the 2-dimensional case can mostly be generalized to higher dimensions, we will restrict ourselves to the cases $\rho = 1, 2$. The expressions ω^1, ω^2, $*^1$, $*^2$, used in common formulations for $\rho = 1, 2$, will be considered to be identical with ω, $\omega\omega$, $*$, $**$, respectively.

The set $\{0, 1, 2, \ldots\}$ of all natural numbers will be denoted as Nat. For $p, q \in \text{Nat}$ we denote $[p] = \{i \in \text{Nat} \mid i < p\} = \{0, 1, \ldots, p-1\}$, and $[p, q] = [p] \times [q] = \{(i, j) \in \text{Nat}^2 \mid i < p, j < q\}$. Hence $[0] = [0, 0] = \varnothing$. For $u \in \text{Nat}^\rho$ we will use the

notation $|u|_s$, $|u|_p$ with the following meaning. For $\rho = 1$ and $u = i \in \mathrm{Nat}$, $|u|_s = |u|_p = i$. For $\rho = 2$ and $u = (i, j) \in \mathrm{Nat}^2$, $|u|_s = i + j$, $|u|_p = ij$.

An alphabet is a nonempty finite set of symbols. A finite ρ-dimensional word (of size u, or a u-word for some $u \in \mathrm{Nat}$) over an alphabet A is a mapping $w\colon [u] \to A$. An infinite word (of size ω^ρ, or a ω^ρ-word) over the alphabet A is a mapping $w\colon \mathrm{Nat}^\rho \to A$. The empty word—the only word of size 0 identical to the only word of size (0, 0)—will be denoted by λ. The size (called the length in the case $\rho = 1$). of w is denoted by $|w|$. Instead of $w(x)$, we will often denote the symbol of w at position $x \in \mathrm{Nat}^\rho$ as w_x. For $\rho = 1$ we write in the usual way $w = w_0\ w_1\ \ldots$; using this convention, the concatenation of the words x, y (x finite, y finite or infinite) is defined as the word xy. For $\rho = 2$ we depict w as a rectangle (Fig. 1). Finite two-dimensional words, infinite one- and two-dimensional words are called array words, sequences, and trellises, respectively. A finite ρ-dimensional word w' is a prefix of a (finite or infinite) ρ-dimensional word w if w' as a mapping is a restriction of the mapping w.

For $\rho = 1$, the parent of a position $i \in \mathrm{Nat}$ is the position $i - 1$ (we will consider the virtual parent -1 of the position 0 as well). For $\rho = 2$, the parents of the position $(i, j) \in \mathrm{Nat}^2$ are the positions $(i - 1, j)$ and $(i, j - 1)$ (the virtual parents of (0, 0) are $(-1, 0)$ and $(0, -1)$), and i and j are coordinates of the position (i, j). The n-th column (the n-th row] consists of all positions (i, j) with $i = n$. [$j = n$], the n-th left [right] diagonal consists of all positions (i, j) such that $i + j = n$ [$i - j = n$]. A position is called the son of its parent(s).

$$\begin{array}{cccc} \cdots & \cdots & \cdots & \cdots \\ w_{0,2} & w_{1,2} & w_{2,2} & \cdots \\ w_{0,1} & w_{1,1} & w_{2,1} & \cdots \\ w_{0,0} & w_{1,0} & w_{2,0} & \cdots \end{array}$$

Fig. 1.

Let A be an alphabet and let $u \in \mathrm{Nat}^\rho$. The sets of all ρ-dimensional finite words, u-words, infinite words over A will be denoted by $A^{*\rho}$, A^u, and $A^{\omega\rho}$ respectively. If there is no danger of confusion, we will identify a singleton set with its only element, e.g. for a symbol a, $a^{\omega\omega} = \{a\}^{\omega\omega}$. A ρ-dimensional finitary language [language of infinite words] over A is any subset of $A^{*\rho}$ [of $A^{\omega\rho}$]. We will not consider mixed languages consisting of both finite and infinite words. If $\rho = 1$, the concatenation of a finitary language L with a language M of finite or of infinite words is defined as $LM = \{xy \mid x \in L, y \in M\}$.

Let Σ, Γ be two alphabets. A morphism from Σ^* to Γ^* is a mapping $h\colon \Sigma^* \to \Gamma^*$ such that for $x, y \in \Sigma^*$ $h(xy) = h(x)h(y)$. The morphism h is completely described by its values on Σ. If $h(\Sigma) \subseteq \Sigma^p$ for some $p \geqslant 1$, h is called p-uniform. One possible generalization of uniform morphisms can be given as follows. For a mapping $h\colon \Sigma^m \to \Gamma^p$, $m, p \geqslant 1$, a (m, p)-substitution $h\colon \Sigma^* \to \Gamma^*$ is a partial mapping defined for words over Σ of length being a multiple of m. $h(w)$ is obtained from such a word w by dividing w into consecutive subwords of length m and replacing each such subword z by $h(z)$. Clearly, $h(xy) = h(x)h(y)$ whenever h is applicable to the words x and y. h is called size increasing if $p > m$. In the 2-dimensional case we will use the analogical notion of $((m, n), (p, q))$-substitution (size increasing if $p > m$ and $q > n$), which replaces (m, n)-words by (p, q)-words. For $m = n = 1$ this substitution will be

called (p, q)-uniform morphism despite the fact that we do not have a 2-dimensional analogue to the general notion of morphism. (m, p)-substitutions and $((m, n), (p, q))$-substitutions will be called block substitutions. We will apply morphisms and block substitutions in an obvious way to infinite words as well, without providing a special definition.

GENERATING CELLULAR AUTOMATON

The generating cellular automaton which we are going to describe here resembles more to an (extended) L-system than a phrase grammar. All letters of the sentential form are being rewritten in parallel. Each letter is replaced by a letter again. Thus the only way for a word to grow is at the end by, at most, one letter in each step. We distinguish nonterminal and terminal symbols, as is usual in grammars.

Informally, a linear generating cellular automaton (1-gca) is a one-way infinite sequence (working tape) of cells. The cells are numbered starting from 0. Each cell contains one symbol (nonterminal or terminal) of the working alphabet of the *gca*. The symbol will be called the state of the cell. A computation of a *gca* consists of synchronized steps. In each step each cell will change its state depending on its current state and the state of its parent (left neighbor). The generating cellular automaton is defined as homogeneous, i.e. the transition function describing the state changes is for all cells the same. No terminal state (symbol) can be changed any more. Neither can be the "dead" nonterminal state \$. If some cell enters the state \$, the whole computation is unsuccessful, i.e. it does not result in generating any terminal word. Initially, all cells are in the blank (quiescent) state B which can be changed just if the parent is not in the state B, and cannot be changed back to B any more. The (virtual) parent of the leftmost cell is by default in the state \$. The generated word (finite or infinite) consists of terminal symbols only.

A 2-dimensional generating cellular automaton (*2-gca*) will consist of an infinite rectangular grid of cells called a trellis structure. We use a similar terminology for trellis structures as for trellises. A 2-*gca* computes in a similar way as a *gca* except that the next state of each cell depends of its current state and the current states of both of its parents. The details can be seen from the following common definition.

Definition 1. A nondeterministic ρ-dimensional generating cellular automaton (ρ-gca) is a triple $A = (N, T, \delta)$, where N and T are two disjoint alphabets, of nonterminal and terminal symbols, respectively, N containing two special symbols B (blank) and \$ (dead). $\delta\colon V^\rho \times V \to 2^V$, where $V = N \cup T$, is the transition mapping, such that for all $X \in V^\rho$, and $a \in T$, $\delta(X, a) = \{a\}$, $\delta(X, \$) = \{\$\}$, $\delta(B^\rho, B) = \{B\}$, and if $Y \neq B$ then $B \notin \delta(X, Y)$.

A ρ-*gca* will be called deterministic (ρ-*dgca* in short), if for all $X \in V^\rho$, and $Y \in V$ $\mathrm{card}(\delta(X, Y)) = 1$. In this case we consider δ to be a function $\delta\colon V^{\rho+1} \to V$. A configuration of A is any element of the set $C_A = \{c \in V^{\omega\rho} \mid$ just finitely many values of c are different from $B\}$. A computational step of A is the binary relation $\Rightarrow_A$ on C_A defined for $x, y \in C_A$ as $x \Rightarrow_A y$ iff $y(i) \in \delta(N_r(x, s), x(s))$ for all $s \in \mathrm{Nat}^\rho$, where the neighborhood $N_r(x, s)$ at position s in x is defined as $N_1(x, s) = x(i-1)$ if $s = i \in \mathrm{Nat}$, and $N_2(x, s) = (x(i, j-1), x(i-1, j))$ if $s = (i, j) \in \mathrm{Nat}^2$. By default, for $\rho = 1$ $x(-1) = \$$, and for $\rho = 2$ $x(i, -1) = x(-1, j) = \$$ for any $i, j \in \mathrm{Nat}$. A compu-

tation of A is a sequence $C = (c_0, c_1, \ldots)$ of configurations from C_A, such that $c_0 = B^{\omega\rho}$, and for $i \geqslant 0$ $c_i \Rightarrow_A c_{i+1}$.

In the following, consider any number $t \in \mathrm{Nat}$ and function f: Nat $\to$ Nat. We will say that a nonempty word $w \in T^u$, $u \in \mathrm{Nat}^\rho$, is generated by A (in time t) if there exists a computation $C = (c_0, c_1, \ldots)$ of A, such that for some $k < t$, w is a prefix of c_k and $c_k(s) = \mathrm{B}$ for $s \notin [u]$. The empty word is generated by A (in time 0), if $B \in \delta(\$^\rho, B)$. The language of all words generated by A will be denoted by $L(A)$. The language $L(A)$ is generated by A in time f if each nonempty word in $L(A)$ of size $u \in \mathrm{Nat}^\rho$ is generated by A in time $f(|u|_s)$.

Denote for $n \in \mathrm{Nat}$ $u_1(n) = n$ and $u_2(n) = (n, n)$. We will say that an infinite word $w \in T^{\omega\rho}$ is generated by A (in time f), if for each $n \in \mathrm{Nat}$ the prefix of w of size $u_\rho(n)$ is a prefix of some word generated by A (in time $f(\rho n)$). An infinite word $w \in T^{\omega\rho}$ is directly generated by A (in time f) if there exists an infinite computation $(C = (c_0, c_1, \ldots)$ of A such that for each $n \in \mathrm{Nat}$ the prefix of w of size u_ρ is a prefix of some c_k (where $k \leqslant f(pn)$). The language of all infinite words from $T^{\omega\rho}$ [directly] generated by A will be denoted by $L^{\omega\rho}(A)$ $[L_d^{\omega\rho}(A)]$. The language $L^{\omega\rho}(A)$ $[L_d^{\omega\rho}(A)]$ is [directly] generated by A in time f: Nat $\to$ Nat, if each of its elements is [directly] generated by A in time f. In the case of deterministic *gca* the language $L_d^{\omega\rho}(A)$ contains at most one infinite word. We will denote such a word by $D(A)$ for both $\rho = 1, 2$. Since $L^{\omega\rho}(A) = \varnothing$ in this case, we will say that $D(A)$ is generated by A (omitting "directly").

The class of all ρ-dimensional [deterministic] generating cellular automata will be denoted by ρ-$[D]GCA$. For the class of languages of a particular type generated by the particular class of automata we will combine the notation introduced above, e.g. both L_d^ω(1-$DGCA$) and D(1-$DGCA$) will denote the class of all sequences directly generated by deterministic 1-*gca*.

Let us just note that the languages of infinite words generated and directly generated by the same automaton need not be the same, as illustrated by the following example.

Example 1. Let A be a gca defined as $A = (\{B, \$, X, Y\}, \{a\}, \delta)$, where $\delta(\$, B) = \{X\}$, $\delta(\$, X) = \{Y, a\}$, $\delta(\$, Y) = \{Y, a\}$, $\delta(Y, B) = \{X\}$, $\delta(Y, Y) = \{Y\}$, $\delta(X, B) = \{B\}$, $\delta(a, Y) = \{a\}$, $\delta(a, X) = \{a\}$, $\delta(a, B) = \{B\}$. The complication resulting in the generated word a^5 is depicted in Fig. 2:

A computation of A is either infinite, and does not produce any terminal at position 0, or it is finite (i.e. only a finite number of cells become nonblank—see Fig. 2). Clearly, $L(A) = a^*$, $L^\omega(A) = a^\omega$, $L_d^\omega(A) = \varnothing$. Hence $L^\omega(A) \neq L_d^\omega(A)$.

The next example, being a slight modification of Example 1, provides a language of sequences directly generated by a 1-*gca*.

Example 2. We consider a *gca* B similar to that from Example 1, just B contains one more terminal symbol b and the rule $\delta(a, B) = \{B\}$ is replaced by the rules $\delta(a, B) = \{b\}$, $\delta(b, B) = \{b\}$. It is clear that $L_d^\omega(B) = a^*b^\omega$. Assume the same language is being generated by some gca B'. Then each word from a^* is a prefix of some finite

0. *BBBBBB* ...	3. *YXBBBB* ...	6. *aYYBBB* ...	9. *aaaaXB* ...
1. *XBBBBB* ...	4. *YYBBBB* ...	7. *aaYXBB* ...	10. *aaaaaB* ...
2. *YBBBBB*...	5. *YYXBBB* ...	8. *aaaYBB* ...	11. *aaaaaB* ...

Fig. 2.

word generated by B', hence the word a^ω is generated by B', but does not belong to $L_d^\omega(B)$, which is a contradiction.

GENERATIVE POWER OF THE MODEL

To get an idea about the generative power of our model, we will compare it to the recognizing and generating power of the Turing machine. We will consider a single-tape Turing machine with (if $\rho = 1$) a one-way infinite tape or (if $\rho = 2$) a quarter-plane tape.

When simulating a ρ-*gca* by a ρ-dimensional Turing machine (a ρ-TM for short), each step of the ρ-*gca* can be simulated by visiting all nonblank cells by the head of the ρ-TM. Since a ρ-*gca* working in time $f(n)$ can contain as much as $O(f^\rho(n))$ nonblank cells, a ρ-TM can simulate a ρ-*gca* in time $O(f^{\rho+1}(n))$. On the other hand, a nondeterministic ρ-*gca* is capable of simulation of a ρ-TM. We will give the idea of the simulation for the one-dimensional case, but the technique can be easily extended to two dimensions. Simulation of the left-to-right move of the Turing-machine head is straightforward. To simulate a right-to-left move, each cell can at any time (nondeterministically) decide that the head has moved to its position. In the following step the son cell has to check whether the decision was correct. Hence if a cell contains the head which is to be moved left, this cell has to still remember the situation in the next step to be able to approve the correct decision of the parent. A cell, having discovered an error, enters the dead state \$ to "veto" the further computation; in such a case no terminal word is generated. If the ρ-TM works in time $f(n)$, the simulating ρ-*gca* will work in time $O(f(n))$.

When finite words are generated, the ρ-gca is restricted to the minimal working space (corresponding to the size of the resulting terminal word) and therefore 1-*gca* are equivalent to linear bounded automata. We can summarize our basic observations as follows:

Proposition 1. ρ-*gca* and ρ-dimensional Turing machines can be mutually simulated in polynomial time. The class $L(1\text{-}GCA)$ is equal to the class of all extended (i.e. with λ possibly added) context-sensitive languages.

We will now reconsider the simulation of the models will respect to the description of languages of infinite words. Since the standard versions of Turing machines are designed to work with finite words only, we take the liberty of considering a Turing model stepwise (directly) generating an infinite terminal word on its tape in a similar way, as it happens on the gca tape. (We hope no more details need be provided). The above mutual simulation then works in the case of a direct description of languages of infinite words as well. As a consequence we find for example that each generated language of infinite words can be directly generated. Indeed, consider a 1-*gca* generating some language of sequences and a Turing machine stepwise simulating all its computations. Such a Turing machine can be modified to directly generate an infinite sequence by nondeterministically prolonging its output whenever it discovers that the current string of output is a proper prefix of the finite word currently generated in the simulation of the 1-*gca*. The Turing machine can then be simulated by another 1-*gca* directly generating the same language. A similar construction can be performed for 2 dimensions. Taking into consideration Example 2 we have the following lemma.

Lemma 1. $L^{\omega\rho}(\rho\text{-}GCA)$ is a proper subclass of $L_d^{\omega\rho}(GCA)$.

In the case of finitary languages the deterministic versions of our model are not of interest, since they are able to generate just singleton languages. The situation is different in the case of languages of infinite words, when description of a singleton language (i.e. of a single infinite word) may be of great practical importance. Therefore we have to consider the simulation of deterministic generating models as well. The straightforward simulation of a ρ-*gca* by a ρ-TM leads to the following lemma:

Lemma 2. To each ρ-*gca* generating an infinite word in time f(n) there exists an equivalent ρ-dimensional deterministic Turing machine generating the same word in time $O(f^{\rho+1}(n))$.

The technique from the beginning of this section cannot be used to prove the converse of the previous lemma, since it is entirely based on nondeterminism. In fact, there is a difference in the generative power of deterministic *gca* and Turing machines, and the converse of Lemma 2 is not true.

Lemma 3. There exists an infinite word over a 2-symbol terminal alphabet, which can be directly generated by a ρ-dimensional deterministic Turing machine, but cannot be directly generated by a ρ-*dgca*.

Proof. (sketch for $\rho = 1$). It is decidable which terminal symbol (if any) will be generated by a 1-*dgca* at a given position since the configuration of the first i cells will be repeated after at most $|V|^i$ steps (V being the working alphabet). A diagonalization argument can be applied. All 1-*gca*'s can be ordered to a sequence and a Turing machine constructed, generating a sequence which differs at the i-th position from the sequence generated by the i-th 1-*gca*.

NONDETERMINISTIC GENERATING CELLULAR AUTOMATA

In this section we will observe some elementary properties of *gca*'s. In the case of finite words, the working space of the gca is limited to the size of the generated word. There are just finitely many different symbol patterns possible in this space. A repetition of a pattern in a computation can be avoided. Choosing for m the size of the working alphabet, we obtain a trivial upper bound on the computation time.

Proposition 2. For each ρ-*gca* A there exists a constant m such that each word of size $u \in \mathrm{Nat}^\rho$ is generated by A in time $m^{|u|_\rho}$.

A speed-up theorem can be easily proved for the *gca*'s:

Theorem 1. To each ρ-*gca* A [directly] generating a language in time $f(n)$, and $k \in \mathrm{Nat}$, there exists an equivalent ρ-*gca* A', deterministic if A is deterministic, [directly] generating the same language in time $\lceil f(n)/k \rceil + n$.

Proof. The proof is based on a simulation of k steps of each cell in one. In A' each nonblank nonterminal symbol will be a k-tuple of symbols of A corresponding to the sequence of states of the current cell during k consecutive steps. If one of the

symbols in the k-tuple is terminal, then the current cell will turn to this terminal state in the very next step. The speed-up is applicable only after waking up the cells.

The classes $L(\rho\text{-}GCA)$, $L^{\omega\rho}(\rho\text{-}GCA)$, $L_d^{\omega\rho}(\rho\text{-}GCA)$ will not be subjects of our investigations, since they can be described by Turing machines as well. The proofs of properties of these classes based on ρ-*gca*'s can be of interest just from the point of view of programming techniques used for ρ-*gca*'s. As an example we provide the following theorem.

Theorem 2. The class $L_d^{\omega}(\rho\text{-}GCA)$ is closed under mapping by a λ-free (uniform for $\rho = 2$) morphism.

Proof. Consider a λ-free morphism $h\colon \Sigma^* \to \Gamma^*$, and a 1-*gca* A with the terminal alphabet Σ. The 1-*gca* A' generating $h(L_d^{\omega}(A))$ will work as follows. Each cell of A' when waked up divides itself into two floors. The first floor of the cells simulates the computation of A. The second floor of each cell is a buffer capable of storing up to $k+1$ symbols of Γ, where k is the maximum length of a word in $h(\Sigma)$. The second floor in a cell becomes active when the first floor enters a state $a \in \Sigma$. In this case the buffer is filled by $h(a)$, leaving the rightmost position free. Hence the second floor will stepwise build the homomorphic image of the sequence generated by A. This image will be propagated to the right, leaving always the leftmost symbol as the resulting terminal symbol in the corresponding cell. The propagating will be done by synchronized shifting of the contents of the second floors to the right. In a shift step a cell will shift the contents of its buffer by one position to the right (the rightmost symbol is erased—sent to the son cell) while the rightmost symbol from the parent's buffer (in the previous state) is accepted as the new leftmost symbol (no symbol is accepted if the parent is in a state from Γ). The shift synchronization requires that whenever a cell sends a symbol, the son cell will accept it. Therefore two additional flags are contained in each cell. The first is set when the cell has just sent a symbol, and the second when the cell has just accepted a symbol. If the second flag of a cell does not match with the first flag of its parent, the cell enters the dead state, and vetoes the computation. The decision of each cell, whether to perform a shift, is done nondeterministically. A successful shift is accomplished only if all cells, starting from the first being in a nonterminal state and up to one having a free rightmost position in its buffer, decide to shift in the same moment, otherwise the computation does not generate any sequence.

Let $\rho = 2$. By applying the above technique to each column, we can observe that the class (2-GCA) is closed under mapping by $(1, q)$-morphisms. Because of symmetry, it is closed under mapping by $(p, 1)$-morphisms. Since each (p, q)-morphism can be written as a composition of a $(1, q)$-morphism with a $(p, 1)$-morphism, the class (2-GCA) is closed under mapping by (p, q)-morphisms.

Theorem 3. The class $L_d^{\omega\rho}(\rho\text{-}GCA)$ is closed under the inverse of a uniform morphism.

Proof. If the morphism is u-uniform, $u \in \mathrm{Nat}^{\rho}$, it is enough to simulate $|u|_p$ cells in one, and nondeterministically replace the resulting terminal u-word in each cell by one of the symbols from its inverse image in the morphism.

Corollary 1. The class $L_d^{\omega^\rho}(\rho\text{-}GCA)$ is closed under mapping by block substitutions.

DETERMINISTIC GENERATING CELLULAR AUTOMATA

This section will be devoted to studying the properties of deterministic cellular automata as a descriptive tool of infinite sequences and infinite trellises. Despite the weaker generating power than Turing machines (Lemma 3), the generated class of infinite words is rich enough to include several classes investigated in the literature. First we consider the complexity of the generating of infinite words by *dgca.*

Lemma 4. For each *dgca* A generating an infinite word there exists a constant m such that the cell at the position $u \in \mathrm{Nat}^\rho$ becomes terminal after at most $m(|u|_s + 1)$ steps.

Proof. Let $m = |V|$ where V is the size of the working alphabet of A. A cell becomes terminal in at most m steps after her parent(s). The proof can be performed by induction on p.

The following theorem is an immediate consequence of Lemma 4.

Theorem 4. For each *dgca* A there exists a constant m such that the word $D(A)$ is generated in time mn.

The previous result can be combined with the assertion of Theorem 1, where k can be chosen to be arbitrarily large. We obtain:

Theorem 5. Let $\varepsilon > 0$. For each ρ-*dgca* A there exists ρ-*dgca* A' generating the word $D(A)$ in time $[(1 + \varepsilon)n]$.

The least time to generate a sequence (trellis) is n. Since in time $n +$ const just ultimately periodic sequences (in the 2-dimensional case only so-called regular trellises) can be generated, while the generative power of the *dgca* is greater, (as we will show further on in this section), there are at least two different classes induced by the time complexity of generation by dgca. We leave open the question whether there is a finer hierarchy between time n and $[(1 + \varepsilon)n]$.

There is a trade off, in Theorem 5, between the generating speed and the size of the dgca. Following the proof technique of Theorem 1, each symbol in A' is a k-tuple of symbols of the working alphabet V of A. We have to choose for k the cardinality m of V. The working alphabet of A' then contains more than m^m symbols.

Next we will investigate some closure properties of *dgca.* We will use the notion of a finite state transducer (*fst*) being an important concept both in the theory of finite (Rovan 1986) or infinite (Latteaux 1990) words. It is a generalization of the notion of a morphism, hence for the 2-dimensional case we are able to introduce a uniform fst only. Informally, a fst applied on a word (finite or infinite) will at each position of the word enter some internal state and replace (uniformly in the 2-dimensional case)

the current symbol by a finite word. The state and the replacement word depend on the current symbol and the *fst* state(s) at the parent(s) position(s).

Definition 2. A ρ-dimensional finite-state transducer (ρ-*fst*) is a quintuple $F = (K, \Sigma, \Gamma, \delta, q_I)$, where Σ and Γ are two alphabets, of input and output symbols, respectively, K is a finite set of states containing the initial state q_I. δ is the transition mapping defined for $\rho = 1$ as[2] $\delta: K \times \Sigma \to 2^{K \times \Gamma^*}$ where for all $q \in K$ and $X \in \Sigma$, $\delta(q, X)$ is finite, and for $\rho = 2$: $\delta: K^2 \times \Sigma \to 2^{K \times \Gamma^{m,n}}$ where $m, n \geqslant 1$ are fixed. F is called (m, n)-uniform and denoted as 2-*fst*(m, n) in this case. (For our purpose it is enough to define a single-letter-input transducer. More general definitions have been considered in literature.)

F is called deterministic (ρ-*dfst*), if for all $q \in K^\rho$, $X \in \Sigma$ $\mathrm{card}(\delta(q, X)) = 1$. In this case we consider δ to be a function $\delta: K^\rho \times \Sigma \to K \times \Gamma^*$. F is called nonerasing if for all $q, q' \in K$, $X \in \Sigma$ $(q', \lambda) \notin (q, X)$. For $\rho = 1$ F is called uniform if for some $p \in \mathrm{Nat}$ $\delta(K \times \Sigma) \subset 2^{K \times \Gamma^p}$.

Let us denote $\Delta = \cup_{s \in K^\rho,\, X \in \Sigma} \delta(q, X)$ (thus Δ is a finite set of pairs—we will use Δ as an alphabet). For a word $w \in \Sigma^{*\rho} \cup \Sigma^{\omega\rho}$ let $\sigma_F(w) \in \Delta^{|w|}$ be the language consisting of all words σ_w such that for $u \in [|w|]$ $[u \in \mathrm{Nat}^\rho$ for an infinite $w]$ $\sigma_w(u) = (q_u, z_u)$, satisfying $\sigma_w(u) \in \delta(s_\rho(u), w)$ where for $\rho = 1$ and $u = 1 \in \mathrm{Nat}$ $s_1(u) = q_{i-1}$, and for $\rho = 2$ and $u = (i, j) \in \mathrm{Nat}^2$ $s_2(u) = (q_{i, j-1}, q_{i-1, j})$. By default, for $\rho = 1$ $q_{-1} = q_I$, and for $\rho = 2$ $q_{0,-1} = q_{-1,0} = q_I$. The (finite-state) transduction by F is the mapping τ_F: $\Sigma^{*\rho} \cup \Sigma^{\omega\rho} \to \Gamma^{*\rho} \cup \Gamma^{\omega\rho}$ defined for $w \in \Sigma^{*\rho} \cup \Sigma^{\omega\rho}$ as $\tau_F(w) = \pi_F(\sigma_F(w))$, where π_F: $\Delta^{*\rho} \cup \Delta^{\omega\rho} \to \Gamma^{*\rho} \cup \Gamma^{\omega\rho}$ is the projection replacing each symbol $(s, z) \in \Delta$ by z.

To characterize the result of application of finite-state transductions to the class $L_d^{\omega\rho}(\rho\text{-}DGCA)$, we have to distinguish the cases $\rho = 1, 2$, since just a uniform transduction can be applied in the 2-dimensional case.

Lemma 5. For each 1-*dgca* A and a nonerasing 1-[*d*]*fst* F there is a 1-[d]gca A' such that $L_d^\omega(A') = \tau_F(L_d^\omega(A))$.

Proof. For F we will use the notation from Definition 2. If F is deterministic, each cell of the 1-*dgca* A' will again, in the same way as in the proof of Theorem 2 divide itself to two floors. A cell will in addition apply the F's transition function to compute the state of F for its position and initially fill its buffer as soon as it has both reached a terminal state and observed the state of F in the parent's cell. The deterministic synchronization of the shift steps will be done in the following way. The speed of shifting must be slow enough not to reach a cell which has not created its buffer yet. A cell in A enters a terminal state at most m steps after its parent, where m is the size of the working alphabet of A. Therefore a shift will be performed in every m-th step only. To achieve this delay, each cell of A' will be counting cyclically from 1 to m.

Now let F be nondeterministic. The nondeterministic synchronization method from the proof of Theorem 2 can be applied to construct the nondeterministic ρ-*gca* A'.

We will illustrate the technique of the proof of Lemma 5, in the following example.

Example 3. Assume the simplest possible 1-*dgca* A generating a^ω, and the 1-*dfst* $F = \{\{q_I, q\}, \{a\}, \{x, y\}, \delta, q_I\}$ where $\delta(q_I, a) = (q, xy)$, $\delta(q, a) = q_I, x)$. The working alphabet of A consists of a, B and \$ only, we will take $m = 2$, since \$ will never appear in a cell. Since $|xy| = 2$, $|x| = 1$, the length of the buffer will be 3. The contents of a cell which is divided into floors will be depicted as a quadruple [terminal symbol from A, state of F, buffer, counter]. The generating computation for a^ω starts as $BBBBBB \ldots$, $aBBBBB \ldots$, $aaBBBB \ldots$, $aaaBBB \ldots$, $aaaaBB \ldots$ $\ldots$. A few initial steps of the computation of the *gca* A' are depicted in Fig. 3.

Lemma 6. For each 2-*dgca* A and a 2-[d]*fst*(m, n) F there is a 2-[d]*gca* A' such that $L_d^{\omega\omega}(A') = \tau_F(L_d^{\omega\omega}(A))$.

Proof. We will use for F the notation from Definition 2. First we construct a [d]*gca* A'' such that $L_d^{\omega\omega}(A'') = \sigma_F(L_d^{\omega\omega}(A))$. A computation of A'' consists of a simulation of A followed by stepwise application of the transition function of F. We have to show the existence of a *gca* A' generating the language $\tau_F(L_d^{\omega\omega}(A)) = \pi_F(\sigma_F(L_d^{\omega\omega}(A))) = \pi_F(L_d^{\omega\omega}(A''))$. If F is nondeterministic, then A'' is as well. Since π_F is a (m, n)-uniform morphism, we can apply Theorem 2. If F is deterministic, then A'' is as well. Since π_F is a composition of a (m, l)-morphism with a (l, n)-morphism, it is enough to observe that the method of the proof of Lemma 5 can be applied in parallel either in each row or in each column of 2-*dgca*, and therefore the class $L_d^{\omega\omega}(2\text{-}DGCA)$ is closed under mapping by both (m, l)-morphisms and (l, n)-morphisms.

For our next consideration we introduce the notion of a canonical uniform morphism. For an alphabet Σ and $u \in \mathrm{Nat}^\rho$ we will use the finite set $\Sigma_u = \{\langle w\rangle \mid w \in \Sigma^u\}$ as an alphabet. The u-uniform morphism $h\colon \Sigma_u^{*\rho} \to \Sigma^{*\rho}$ defined for $w \in \Sigma^u$ by $h(\langle w\rangle) = w$ will be called the canonical u-morphism over Σ. Clearly, h is one-to-one.

Lemma 7. For each ρ-*dgca* A, $u \in \mathrm{Nat}^\rho$, and for the canonical u-uniform morphism h over the terminal alphabet of A there exists a ρ-*dgca* A' such that $D(A') = h^{-1}(D(A))$.

B	*B*	B	*B*	*B*	...
[*a*, *q*, *xy*-, 0]	*B*	*B*	*B*	*B*	...
[*a*, *q*, *xy*-, 1]	[*a*, q_I, *x*--, 0]	*B*	*B*	*B*	...
[*a*, *q*, *xy*-, 2]	[*a*, q_I, *x*--, 0]	[*a*, *q*, *xy*-, 0]	*B*	*B*	...
[*a*, *q*, -*xy*, 1]	[*a*, q_I, *x*--, 1]	[*a*, *q*, *xy*-, 0]	[*a*, q_I, *x*--, 0]	*B*	...
[*a*, *q*, -*xy*, 2]	[*a*, q_I, *x*--, 2]	[*a*, *q*, *xy*-, 0]	[*a*, q_I, *x*--, 0]	[*a*, *q*, *xy*-, 0]	...
x	[*a*, q_I, *yx*-, 1]	[*a*, *q*, *xy*-, 1]	[*a*, q_I, *x*--, 0]	[*a*, *q*, *xy*-, 0]	...
x	[*a*, q_I, *yx*-, 2]	[*a*, *q*, *xy*-, 2]	[*a*, q_I, *x*--, 0]	[*a*, *q*, *xy*-, 0]	...
x	[*a*, q_I, -*yx*, 1]	[*a*, *q*, *xy*-, 1]	[*a*, q_I, *x*--, 1]	[*a*, *q*, *xy*-, 0]	...
x	[*a*, q_I, -*yx*, 2]	[*a*, *q*, *xy*-, 2]	[*a*, q_I, *x*--, 2]	[*a*, *q*, *xy*-, 0]	...
x	*y*	[*a*, *q*, *xxy*, 1]	[*a*, q_I, *x*--, 1]	[*a*, *q*, *xy*-, 1]	...
x	*y*	[*a*, *q*, *xxy*, 2]	[*a*, q_I, *x*--, 2]	[*a*, *q*, *xy*-, 2]	...
x	*y*	[*a*, *q*, -*xx*, 1]	[*a*, q_I, *yx*-, 1]	[*a*, *q*, xy-, 1]	...
x	*y*	[*a*, *q*, -*xx*, 2]	[*a*, q_I, *yx*-, 2]	[*a*, *q*, *xy*-, 2]	
x	*y*	*x*	[*a*, q_I, *xyx*, 1]	[*a*, *q*, *xy*-, 1]	...
x	*y*	*x*	[*a*, q_I, *xyx*, 2]	[*a*, *q*, *xy*-, 2]	...
x	*y*	*x*	[*a*, q_I, -*xy*, 1]	[*a*, *q*, *xxy*, 1]	...
x	*y*	*x*	[*a*, q_I, -*xy*, 2]	[*a*, *q*, *xxy*, 2]	...
x	*y*	*x*	*x*	[*a*, *q*, *yxx*, 1]	...

Fig. 3. Initial computation steps of *gca* A'; each line corresponds to one computational step.

Proof. A' simulates $|u|_p$ cells of A in one. If the resulting u-tuple is a terminal word w, then the cell enters the terminal state $\langle w \rangle$.

Since each (uniform) morphism is a special case of a finite-state transduction, and each substitution is a composition of an inverse of a canonical morphism and a uniform morphism, we have the following properties of the class of words generated by ρ-*dgca*.

Theorem 6. The class $D(\rho\text{-}DGCA)$ is closed under [uniform for $\rho = 2$] deterministic nonerasing finite-state transductions and under block substitutions.

The next theorem shows that the class of infinite words generated by ρ-*dgca* has some properties which can be required from a reasonable descriptive device.

Theorem 7. The class $D(1\text{-}DGCA)$ is closed under the operations of adding, removing or changing finitely many symbols.

The class $D(2\text{-}DGCA)$ is closed under the operations of removing finitely many rows and columns, adding finitely many rows and columns being sequences from $D(1\text{-}DGCA)$, and changing finitely many symbols.

Proof. Consider a 1-*dgca* A. Any finite change in $D(A)$ can be performed by repeatedly adding and/or removing one symbol at the beginning of the sequence. This can be achieved by simulating the i-th cell of A by the $(i+1)$-th cell (when adding a symbol) or $(i-1)$-th cell (when removing a symbol) of a new 1-*dgca*, with an appropriate behavior of the leftmost cell. Now let A be a 2-*dgca*. It is easy to observe that each row and each column of $D(A)$ is from $D(1\text{-}DGCA)$. Again, changing finitely many rows, columns, or symbols of the trellis can be accomplished by repeatedly adding and/or removing the 0-th row or the 0-th column. The method from the 1-dimensional case can be applied again, however, when adding a new sequence, the underlying 1-dgca generating the new row or column must be encoded, being used only by the border cells of the new 2-*dgca*.

The next operation we consider is the shuffling of sequences. We provide just an informal definition. A regular (m, n)-shuffle of two sequences s_1 and s_2 $(m, n \geqslant 1)$ is a sequence consisting of the first m symbols of s_1 followed by the first n symbols of s_2, then the next m symbols of s_1, the next n symbols of s_2, etc. There is no apparent way of "2-dimensional shuffling", but trellises can be shuffled rowwise [columnwise]. A regular (m, n)-row-shuffle [column-shuffle] of two trellises is defined as the shuffle of the corresponding sequences of rows [columns].

Theorem 8. The class $D(1\text{-}DGCA)$ is closed under regular shuffles.

The class $D(2\text{-}DGCA)$ is closed under regular row and column shuffles.

Proof. To generate the regular (m, n)-shuffle of the sequences s_1, s_2, let us first consider a 1-*dgca* A generating in parallel the inverse image of s_1 in the canonical m-uniform morphism and the inverse image of s_2 in the canonical n-uniform morphism. Applying on $D(A)$ the uniform morphism which maps each symbol-pair $(\langle w_1 \rangle, \langle w_2 \rangle)$, where $|w_1| = m$, $|w_2| = n$, to the word $w_1 w_2$, will result in the required

shuffle of s_1 and s_2. The assertion follows from Theorem 6. The proof technique can be directly extended to the 2-dimensional case.

A *2-gca* is in fact a generalization of the definition of regular trellises (Culik *et al.* 1984) or generalized Pascal triangles (Korec 1985). The one-dimensional equivalent of a regular trellis is a periodic sequence. An infinite word is strictly regular if it is directly generated by *ρ-dgca* without nonterminal symbols (except *B* and \$); it is called regular if it is an image of a strictly regular word in some uniform morphism. The class $D(\rho\text{-}DGCA)$ clearly contains all strictly regular words, and, since it is closed under uniform morphisms, all regular ρ-dimensional infinite words. Another class—modular trellises—has been described in (Černý & Gruska 1986a, 1986b), based on the term tag sequence (Cobham 1972). A tag sequence can be roughly described as an infinite fixed point of a morphism. We will use the term modular sequence instead. Modular words can be described as follows (uniformity is required for $\rho = 2$). Let Σ be an alphabet. An infinite word $w \in \Sigma^{\omega\rho}$ is strictly modular if it is a fixed point of some morphism $h\colon \Sigma^{*\rho} \to \Sigma^{*\rho}$ and no finite prefix of w is a fixed point of h. If h is u-uniform for some $u \in (\mathrm{Nat} - \{0, 1\})^{\rho}$, w is called strictly u-uniformly modular. *A* word from $\Gamma^{*\rho}$ is [u-uniformly] modular if it is an image of a strictly [u-uniformly] modular word in a 1-uniform morphism. It is not straightforward to see that each [uniformly] modular word can be directly generated by a *ρ-dgca*. We will prove a more general assertion.

A deterministic finite-state transduction is called prolongable in symbol a, if it is induced by a nonerasing *ρ-dfst* F such that (we are using the notation from Definition 2) $\Sigma = \Gamma$, $a \in \Sigma$, $\delta(q_I, a) = (q, z)$, where a is a prefix of z and $z \neq a$. It is easy to see that the powers $(\tau_F)^n(a)$ form a strictly growing chain, i.e. each one is a proper prefix of the next. The limit of this chain is an infinite word, being a fixed point of τ_F. This infinite word can be in the linear case stepwise constructed as follows. A partial result of the construction can be described as a triple (α, q, β), consisting of a state q of F, and two words α, β over Σ, such that $\alpha\beta$ is a prefix of the constructed sequence. α is the processed part, and β the part ready for processing. The initial triple is (λ, q_I, a). If $\delta(q, b) = (q', z)$, then the triple (x, q, by) will be changed to (xb, q', yz). The dfst is nonerasing and (except for the initial triple) always $\tau_F(\alpha) = \alpha\beta$; therefore the process is infinite, generating the required sequence. A similar construction can be performed in the 2-dimensional case for a uniform 2-dfst.

Theorem 9. The class $D(\rho\text{-}DGCA)$ contains all fixed points of prolongable deterministic (uniform for $\rho = 2$) finite-state transductions. In particular, it contains all fixed points of (uniform for $\rho = 2$) prolongable morphisms.

Proof. Let $\rho = 1$. The above construction of the fixed point can be used by a 1-*dgca*. The resulting sequence will be stepwise created and passed in the reversed form to the right. The front symbol reaching a blank cell will be the resulting terminal state for this cell. Having found out its terminal value and obtained the current state of F from its parent, each cell will compute the next state q of F and the next portion z of the sequence. The portion z, when created, will be reversed and kept in the cell until it can be added to the rear end of the propagating sequence. Shifts of the sequence may be performed in each step, and no special synchronization is necessary. By default, the state q_I and front-end symbol a are passed to the first cell.

Let $\rho = 2$. The construction from the linear case can be used here twice. The (m, n)-uniform transduction is first applied as an n-uniform transduction for the rows, while columns of symbols of height m are considered to be "terminal multisymbols". A blank cell will behave as follows. When the front of the sequence of multisymbols coming along the row arrives, the cell will take the first multisymbol, reverse it, and keep it to be added to the end of the sequence coming along the column. When the front of the sequence of symbols coming along the column arrives, the cell will take the first symbol—this will be the resulting symbol for the cell—and wait to receive the states of F from both its parents. Then it will create its (m, n)-portion, reverse the order of columns, and keep it to be added to the end of the sequence of multisymbols going along the row. Since the transduction is uniform, the process is fully synchronized.

As a consequence, we obtain that all (uniformly for $\rho = 2$) modular words can be directly generated *ρ-dgca*. Since the classes or regular and modular words are mutually incomparable (Černý & Gruska 1986a), we have:

Theorem 10. The class $D(\rho\text{-}DGCA)$ contains the class of all regular and the class of all (uniformly for $\rho = 2$) modular sequences as proper subclasses.

The next theorem indicates that the class $D(\rho\text{-}DGCA)$ is rich enough, since very basic problems are undecidable.

Theorem 11. It is undecidable for a given *ρ-dgca*, whether a given symbol will appear in any of its cells or whether $D(A)$ is empty.

Proof. It is enough to prove the assertions for 1-*dgca*. We may code the computation of a 1-*dgca* A by a trellis t over its working alphabet by taking for $t(i, j)$ the contents of the cell j of A after the $(i + j)$-th step. The value $t(i, j)$ (of the cell j of A in time $i + j$) is fully determined by $t(i, j - 1)$ and $t(l - 1, j)$. Hence t is a strictly regular trellis (or a generalized Pascal triangle). According to Korec (1985), generalized Pascal triangles can simulate computations of deterministic Turing machines in (some parts of) the left diagonals. A 1-dgca can in parallel simulate a Turing machine with an empty input (in the above sense) and generate the sequence a^ω. The generating process will be interrupted iff the Turing machine halts. The undecidability of the symbol occurrence and of the halting problem for Turing machines implies the assertion of the theorem.

Given a position $u \in \mathrm{Nat}^\rho$ in a strictly uniformly modular infinite word w, the value $w(u)$ can be computed by a finite-state automaton (*fsa*) induced by the iterated morphism (Cobham 1972, Černý & Gruska 1986a). For $\rho = 1$, and a strictly modular sequence w generated by iteration of p-uniform morphism h on an alphabet Σ prolongable in $a \in \Sigma$, the *fsa* has Σ as the set of states, $[p]$ as the input alphabet, a as the initial state, and the transition function will map a state $b \in \Sigma$ and a symbol $j \in [p]$ to the j-th symbol of $h(b)$. When applied to the p-ary notation of a position i, the resulting state is $w(i)$. A similar effect can be achieved in the 2-dimensional case by coding a position (i, j) in the alphabet $[p, q]$.

A sequence is p-uniformly modular iff it is a fixed point of some (m, r)-substitution, where the ratio r/m is a positive power of p, and a 2-dimensional analogue is

also true (Černý & Gruska 1986a). Hence each fixed point of a block substitution can be positionally computed by a *fsa* and belongs to $D(\rho\text{-}DGCA)$. We will show that this remains true even when (for $\rho = 1$) $r > m$ and n is not a multiple of m. First we need a rather technical lemma.

Lemma 8. Let Σ be an alphabet, $1 \leqslant m < r$, and let σ be a (m, r)-substitution on Σ having a fixed point $w \in \Sigma^\omega$. For each $t \in$ Nat there is a sequence of words $x_0, x_1, \ldots, x_k$ of length mr and corresponding sequences of integers $j_0, j_1, \ldots, j_k = 0$; $t_0 = t$, $t_1, \ldots, t_k = 0$; such that for each $0 \leqslant i \leqslant k$

(1) x_i is a subword of w starting at position t_i and for $i \neq k$ x_i is a subword of $\sigma(x_{i+1})$ starting at position j_i

(2) $j_i = t_i \bmod r$ and for $i \neq 0$ $t_{i+1} = m\,[t_i/r]$, i.e. $t_i = t_{i+1}(r/m) + j_i$

Proof. The value $w(t)$ at a given position $t > 0$ is determined by the subword x_1 of w of length $|x_1| = mr$, starting at position $t_1 = m\,[t/r]$. $w(t)$ is contained in the image $\sigma(x_1)$ (of length r^2). The distance $j_0 = t - t_1(r/m)$ between $w(t)$ and the first symbol of $\sigma(x_1)$ is less than r, therefore the subword x_0 of w of length $|x_0| = mr$, starting at position $t_0 = t$ is contained in $\sigma(x_1)$ and completely determined by x_1 and j_0. In a similar way, x_1 is fully determined by the subword x_2 of w of length mr starting at the position $t_2 = m\,[t_1/r]$, and $j_1 = t_1 - t_2(r/m) < r$. By iterating this process we will finish with the word x_k being the initial subword of w of length mr.

Theorem 12. The class $D(\rho\text{-}DGCA)$ contains all fixed points of size increasing ρ-dimensional block substitutions.

Proof. Let $\rho = 1$. Let σ be a (m, r)-substitution on an alphabet Σ having a fixed point $w \in \Sigma^\omega$, $1 \leqslant m < r$. Consider a position $t \geqslant 1$ in w. According to (2) of Lemma 8, the sequence $j_0, j_1, \ldots, j_k$ is fully determined by t and forms a kind of notation for the number t expressed in base r/m. Indeed, $t = j_k(r/m)^k + j_{k-1}(r/m)^{k-1} + \cdots + j_0$, for example, 17 can be expressed in base 3/2 as 21012. This notation is independent of the leading zeroes and arithmetic is performed in the usual way, but each carry has to be multiplied by m. Based on Lemma 6.5 we can construct a finite automaton F for determining the value $w(t)$. The states of the automaton are the words over the terminal alphabet of A of length mr. The input alphabet is $[r]$. The initial state is the prefix of w of length mr. The transition maps a state s and input $j \in [r]$ to the state being the subword of length mr of $\sigma(s)$ starting at position j. For the input sequence being the (r/m)-notation of t, F will reach a state with the initial letter $w(t)$, as seen from (2) of Lemma 8.

We will give an idea how to construct a 1-*dgca* A such that $w = D(A)$. A cell of A will simulate F to computer $w(t)$ for its own position t. The cell will receive (digit-by-digit) from its parent the (r/m)-notation $t_{r/m}$ of t. If $t_{r/m}$ is provided in reversed order, then the cell can easily increase it by 1 and send it on to its son. To be able to use the reversed notation, we have to rebuild F to follow the reversed input. This is a technical exercise on finite automata using some modification of standard techniques of reversing automata and constructing deterministic ones. The proof can be directly adapted for the case $\rho = 2$.

ACKNOWLEDGEMENTS

This work was supported by grant No. SM058 of Kuwait University, and partially by grant No. MSV SRI/1477/94 of the Slovak ministry of Education and Science.

REFERENCES

Blum, M. & Hewitt, C. 1967. Automata on a two-dimensional tape. Proceedings of the IEEE Symposium on Switching and Automata theory: 135–160.

Černý, A. & Gruska, J. 1986a. Modular trellises. In: **Rosenberg, I & Salomaa, A. (Eds.).** The book of *L*, pp. 45–61, Springer, Berlin.

Černý, A. & Gruska, J. 1986b. Modular real-time trellis automata. Fundamenta Informaticae **IX:** 253–282.

Choffrut, C. & Culik II, K. 1984. On real-time cellular automata and trellis automata. Acta Informatica **21:** 393–407.

Cobham, A. 1972. Uniform tag sequences. Mathematical Systems Theory **6:** 164–192.

Cole, S.N. 1969. Real-time computation by n-dimensional iterative arrays of finite-state machines. IEEE Transactions on Computers C-18 (4).

Culik II, K., Gruska, J. & Salomaa, A. 1984. Systolic trellis automata. International Journal of Computer Mathematics **15:** 195–212.

Gruska, J. 1984. Systolic automata—power, characterizations, nonhomogeneity. Proceedings of Mathematical Foundations of Computer Science '84, Prague, Springer, LNCS 176, Berlin.

Ibarra, O.H., Palis, M.A. & Kim, S.M. 1985a. Fast parallel language recognition by cellular automata, Theoretical Computer Science **41 (2–3):** 231–246.

Ibarra, O.H., Kim, S.M. & Moran, S. 1985b. Sequential machine characterizations of trellis and cellular automata and applications. SIAM Journal of Computation **14 (2):** 426–447.

Inoue, K. & Takanami, I. 1988. A survey of two-dimensional automata theory. Proceedings of the International Meeting of Young Computer Scientists '88, Smolenice, Slovakia: 21–35.

Korec, I. 1985. Generalized Pascal triangles. Acta Mathematica of Universitas Comenianae **XLVI–XLVII:** 93–130.

Korec, I. 1990. Pascal triangles modulo n and modular trellises. Computers and Artificial Intelligence **9 (2):** 105–113.

Latteaux, M., & Timmerman, E. 1990. Rational ω-transductions. Proceedings Mathematical Foundations of Computer Science '90, B. Bystrica, Springer LNCS 452, Berlin.

Lothaire, H. 1983. Combinatorics on words. Addison-Wesley, Reading, Mass., 238 pp.

Lucanu, D. 1987. Several properties of array languages. Information Sciences **43:** 185–203.

Neumann, J. von 1966. Theory of self-reproducing automata. University of Illinois Press, Urbana.

Rovan, B. 1986. Complexity classes of *g*-systems are AFL. Acta Mathematica Universitas Comenianae **XLVII–XLIV:** 283–297.

Siromoney, R., Subramanian, K.G. & Dare, R. 1984. Infinite arrays and controlled deterministic table OL array systems. Theoretical Computer Science **33:** 3–11.

Subramanian, K.G., Siromoney, R. & Siromoney, G. 1985. A note on an extension of matrix grammars generating two-dimensional languages, Information Sciences **35:** 223–233.

Thue, A. 1906. Über unendliche Zeichenreihen. Videnskabs-Selskabets Skifter, Mathematik, Kristiania **7:** 1–22.

Thue, A. 1912. Über die gegenseitige Lage gleichen Teile gewisser Zeichenreihen. Videnskapsselskapets Skifter, Mathematik Kristiania **1:** 1–67.

Wolfram, S. 1984a. Computation theory of cellular automata. Communications in Mathematical Physics **96 (1):** 15–57.

Wolfram, S. 1984b. Universality and complexity in cellular automata. Physica D—Nonlinear Phenomena **10 (1–2):** 1–35.

Wolfram, S. 1985. Two-dimensional cellular automata. Journal of Statistical Physics **38 (5–6):** 901–946.

(Accepted 23 December 1996)

توصيف الكلمات باستخدام الأوتوماتا الخلويـة

أنطون شيرني
قسم الرياضيات وعلم الحاسوب،
كلية العلوم بجامعة الكويت،
ص . ب 5969، 13060 الصفاة، الكويت.

خلاصة

يعرض هذا البحث استخدام كل من الأوتوماتا أحادية الأبعاد وكذلك الأوتوماتا الخلويـة ثنائية الأبعاد كوسيلة انتاجية لتوصيف الكلمات اللانهائية ذات الأبعاد الأحاديـة والثنائيـة، حيث أن كل كلمة لا نهائية يمكن توصيفها باستخدام حساب لا نهائي يولد بداياتها المتزايدة. اضافة الى ذلك يوضح البحث أن الصورة المحدودة أحادية البعد من النظام يمكنها توصيف اللغات ذات الحساسية للمضمون.