Commun. Math. Phys. 96, 15–57 (1984)

Communications in Mathematical Physics

Computation Theory of Cellular Automata

Stephen Wolfram*

The Institute for Advanced Study, Princeton, NJ 08540, USA

Abstract. Self-organizing behaviour in cellular automata is discussed as a computational process. Formal language theory is used to extend dynamical systems theory descriptions of cellular automata. The sets of configurations generated after a finite number of time steps of cellular automaton evolution are shown to form regular languages. Many examples are given. The sizes of the minimal grammars for these languages provide measures of the complexities of the sets. This complexity is usually found to be non-decreasing with time. The limit sets generated by some classes of cellular automata correspond to regular languages. For other classes of cellular automata they appear to correspond to more complicated languages. Many properties of these sets are then formally non-computable. It is suggested that such undecidability is common in these and other dynamical systems.

1. Introduction

Systems that follow the second law of thermodynamics evolve with time to maximal entropy and complete disorder, destroying any order initially present. Cellular automata are examples of mathematical systems which may instead exhibit "self-organizing" behaviour[1]. Even starting from complete disorder, their irreversible evolution can spontaneously generate ordered structure. One coarse indication of such self-organization is a decrease of entropy with time. This paper discusses an approach to a more complete mathematical characterization of self-organizing processes in cellular automata, and possible quantitative measures of the "complexity" generated by them. The evolution of cellular automata is viewed as a computation which processes information specified as the initial state. The structure of the output from such information processing is then described using

* Work supported in part by the U.S. Office of Naval Research under contract number N 00014-80-C-0657

1 An introduction to cellular automata in this context, together with many references is given in [1]. Further results are given in [2, 3], and are surveyed in [4, 5]

the mathematical theory of formal languages (e.g. [6–8]). Detailed results and examples for simpler cases are presented, and some general conjectures are outlined. Computation and formal language theory may in general be expected to play a role in the theory of non-equilibrium and self-organizing systems analogous to the role of information theory in conventional statistical mechanics.

A one dimensional cellular automaton consists of a line of sites, with each site taking on a finite set of possible values, updated in discrete time steps according to a deterministic rule involving a local neighbourhood of sites around it. The value of site i at time step t is denoted $a_i^{(t)}$ and is a symbol chosen from the alphabet

$$S=\{0, 1, \ldots, k-1\}\,. \tag{1.1}$$

The possible sequences of these symbols form the set Σ of cellular automaton configurations $A^{(t)}$. Most of this paper concerns the evolution of infinite sequences $\Sigma=S^Z$; finite sequences $\Sigma=S^N$ flanked by quiescent sites (with say value 0) may also be considered. At each time step each site value is updated according to the values of a neighbourhood of $2r+1$ sites around it by a local rule

$$\phi: S^{2r+1}\to S \tag{1.2}$$

of the form[2]

$$a_i^{(t)}=\phi[a_{i-r}^{(t-1)}, a_{i-r+1}^{(t-1)}, \ldots, a_{i+r}^{(t-1)}]\,. \tag{1.3}$$

This local rule leads to a global mapping

$$\Phi: \Sigma\to\Sigma \tag{1.4}$$

on complete cellular automaton configurations. Then in general

$$\Omega^{(t+1)}=\Phi\Omega^{(t)}\subseteq\Omega^{(t)}, \tag{1.5}$$

where

$$\Omega^{(t)}=\Phi^t\Sigma \tag{1.6}$$

is the set (ensemble) of configurations generated after t iterated applications of Φ (t time steps).

Formal languages consist of sets of words formed from strings of symbols in a finite alphabet S according to definite grammatical rules. Sets of cellular automaton configurations may thus be considered as formal languages, with each word in the language representing a cellular automaton configuration. Such infinite sets of configurations are then completely specified by finite sets of grammatical rules. (This descriptive use of formal grammars may be contrasted with the use of their transformation rules to define the dynamical evolution of developmental or L systems (e.g. [9]).)

Figure 1.1 gives typical examples of the evolution of cellular automata from disordered initial states according to various rules ϕ. Structure of varying complexity is seen to be formed. Four basic classes of behaviour are found in

2 The notation used here differs slightly from that of [2]. In particular, F in [2] is denoted here as ϕ

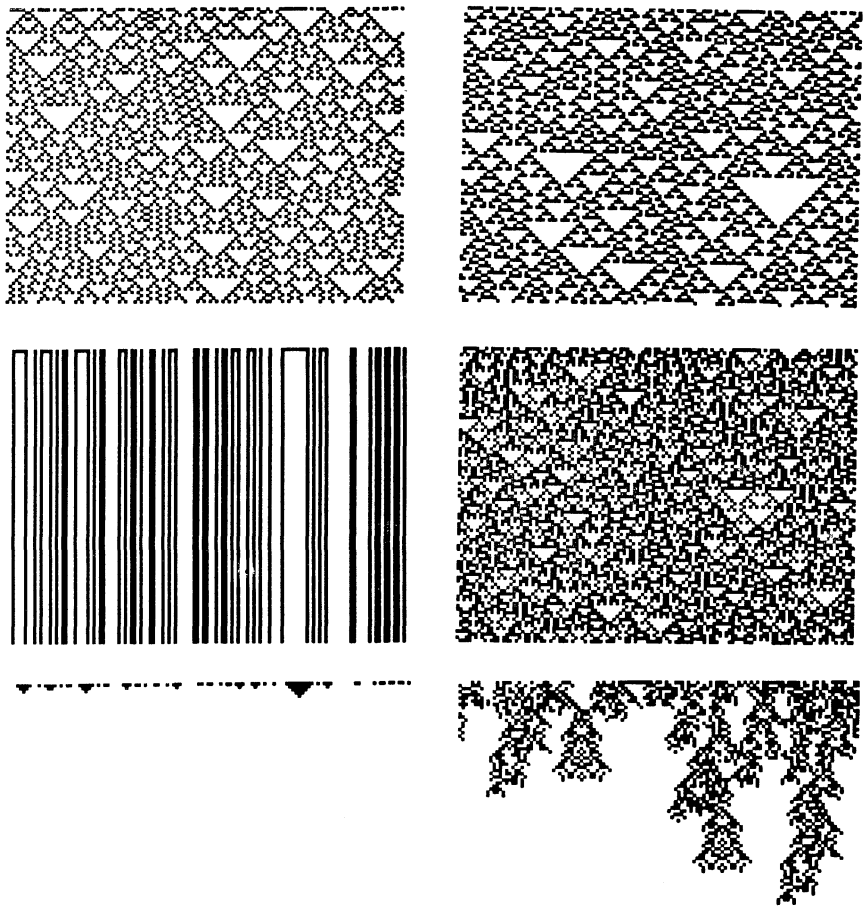


Fig. 1.1. Evolution of cellular automata with various typical local rules ϕ. The initial state is disordered; successive lines show configurations obtained at successive time steps. Four qualitative classes of behaviour are seen. (The first five rules shown have $k=2$ and $r=1$, and rule numbers 18, 22, 76, 90 and 128, respectively [1]. The last rule has $k=2$, $r=2$, and totalistic code number 20 [2])

these and other cellular automata [2]. In order of increasing apparent complexity, qualitative characterizations of these classes are as follows:

1. Tends to a spatially homogeneous state.
2. Yields a sequence of simple stable or periodic structures.
3. Exhibits chaotic aperiodic behaviour.
4. Yields complicated localized structures, some propagating.

Approaches based on dynamical systems theory (e.g. [10, 11]) suggest some quantitative characterizations of these classes: the first three are analogous to the limit points, limits cycles and chaotic ("strange") attractors found in continuous dynamical systems. The fourth class exhibits more complex behaviour, and, as discussed below, is conjectured [2] to be capable of universal computation (e.g. [6,

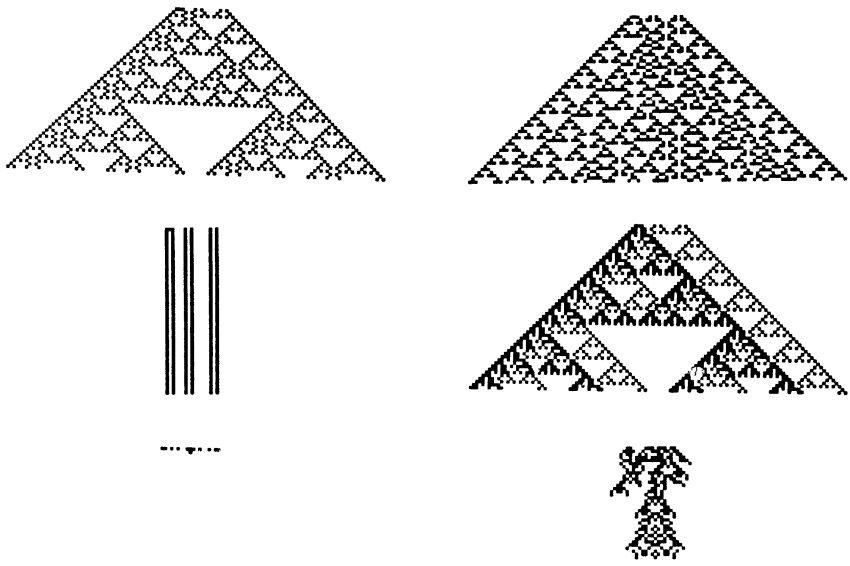


Fig. 1.2. Evolution of cellular automata with various typical local rules from finite initial states. (The rules shown are the same as in Fig. 1)

7, 8]). The formal language theory approach discussed in this paper provides more precise and complete characterizations of the classes and their complexity.

The four classes of cellular automata generate distinctive patterns by evolution from finite initial configurations, as illustrated in Fig. 1.2:

1. Pattern disappears with time.
2. Pattern evolves to a fixed finite size.
3. Pattern grows indefinitely at a fixed rate.
4. Pattern grows and contracts with time.

The classes are also distinguished by the effects of small changes in initial configurations:

1. No change in final state.
2. Changes only in a region of finite size.
3. Changes over a region of ever-increasing size.
4. Irregular changes.

"Information" associated with the initial state thus propagates only a finite distance in classes 1 and 2, but may propagate an infinite distance in classes 3 and 4. In class 3, it typically propagates at a fixed positive speed.

The grammar of a formal language gives rules for generating or recognizing the words in the language. An idealized computer (such as a Turing machine) may be constructed to implement these rules. Such a computer may be taken to consist of a "central processing unit" with a fixed finite number of internal states, together with a "memory" or "tape." Four types of formal language are conventionally identified, roughly characterized by the size of the memory in computers that implement them (e.g. [7]):

0. Unrestricted languages[3]: indefinitely large memory.

1. Context-sensitive languages: memory proportional to input word length.

2. Context-free languages: memory arranged in a stack, with a fixed number of elements available at a given time.

3. Regular languages: no memory.

These four types of languages (essentially) form a hierarchy, with type 0 the most general. Only type 0 languages require full universal computers; the other three types of language are associated with progressively simpler types of computer (linear-bounded automata, pushdown automata, and finite automata, respectively).

The grammatical rules for a formal language may be specified as "productions" which define transformations or rewriting rules for strings of symbols. In addition to the set S of "terminal" symbols s_i which appear directly in the words of the language, one introduces a set U of intermediate "non-terminal" symbols u_i. To generate words in the language, one begins with a particular non-terminal "start" symbol, then uses applicable productions in turn eventually to obtain strings containing only terminal symbols. The different types of languages involve productions of different kinds:

0. Arbitrary productions.

1. Productions $\alpha_1 \to \alpha_2$ for which $|\alpha_2| \geqq |\alpha_1|$, where α_i is an arbitrary string of terminal and non-terminal symbols, and $|\alpha_i|$ is its length.

2. Productions of the form $u_i \to \alpha_j$ only (with a fixed bound on $|\alpha_j|$).

3. Productions of the form $u_i \to s_j u_k$ or $u_i \to s_j$ only.

Words in languages are recognized (or "parsed") by finding sequences of inverse productions that transform the words back to the start symbol.

The grammars for regular (type 3) languages may be specified by the finite state transition graphs for finite automata that recognize them. Each arc in such a graph carries a symbol s_i from the alphabet S. The nodes in the graph are labelled by non-terminal symbols, and connected according to the production rules of the grammar. Words in the language correspond to paths through the state transition graph. The (set) entropy of the language, defined as the exponential rate of increase in the number of words with length (see Sect. 3), is then given by the logarithm of the largest eigenvalue of the adjacency matrix for the state transition graph. This eigenvalue is always an algebraic integer.

The set of all possible sequences of zeroes and ones forms a trivial regular language, corresponding to a finite automaton with the state transition graph of Fig. 1.3a. Exclusion of all sequences with pairs of adjacent ones (so that any 1 must be followed by a 0) yields the regular language of Fig. 1.3b. The set of sequences in which, say, an even number of isolated ones appear between every 0110 block, again forms a regular language, now specified by the graph of Fig. 1.3c.

Regular expressions provide a convenient notation for regular languages. For example, ((0*)(1*))* represents all possible sequences of zeroes and ones, corresponding to Fig. 1.3a. Here α* denotes an arbitrary number of repetitions of the

3 Also known as general, phrase-structure, and semi-Thue languages

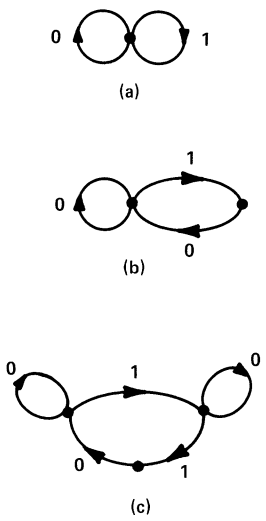


Fig. 1.3a–c. State transition graphs for deterministic finite automata (DFA) corresponding to some regular languages: **a** the set of all possible sequences of zeroes and ones; **b** sequences in which 11 never occurs; **c** sequences in which an even number of isolated 1's appear between each 0110 block. Words in the languages correspond to sequences of symbols on arcs in paths through the DFA state transition graphs. The three DFA shown have successively larger numbers of states Ξ, and the sets of symbol sequences they represent may be considered to have successively larger "regular language complexities"

string α. With this notation, (0*(10)*)* represents Fig. 1.3b, and (0(0*)1(0*)1)* represents Fig. 1.3c.

Many regular grammars may in general yield the same regular language. However, it is always possible to find the simplest grammar for a given regular language (Myhill-Nerode theorem (e.g. [7])), whose corresponding finite automaton has the minimal number of states (nodes). This minimal number of states provides a measure of the "complexity" Ξ of the regular language. The regular languages of Fig. 1.3a–c are thus deemed to have progressively greater regular language complexities.

Section 2 shows that the sets of configurations $\Omega^{(t)}$ generated by any finite number of steps in the evolution of a cellular automaton form a regular language. For some cellular automata, the complexities of the regular languages obtained tend to a fixed limit after a few time steps, yielding a large time limiting set of configurations corresponding to a regular language. In general, it appears that the limit sets for all cellular automata that exhibit only class 1 or 2 behaviour are given by regular languages. For most class 3 and 4 cellular automata, however, the regular language complexities $\Xi^{(t)}$ of the sets $\Omega^{(t)}$ increase rapidly with time, presumably leading to non-regular language limit sets.

Set of symbol sequences analogous to sets of cellular automaton configurations are obtained from the "symbolic dynamics" of continuous dynamical systems, in which the values of real number parameters are divided into discrete bins, labelled by symbols (e.g. [10, 11]). The simplest symbol sequences obtained in

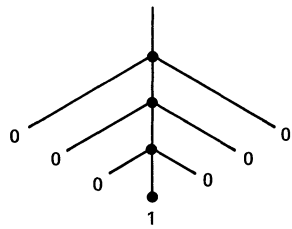


Fig. 1.4. The derivation tree for a word in the context-free language consisting of sequences of the form 0^n10^n

this way are "full shifts," corresponding to trivial regular languages Σ containing all possible sequences of symbols. More complicated systems yield finite complement languages, or "subshifts of finite type," in which a finite set of fixed blocks of symbols is excluded. "Sofic" systems, equivalent to general regular languages, have also been studied [12]. There is nevertheless evidence that, just as in cellular automata, regular languages are inadequate to describe the complete symbolic dynamics of even quite simple continuous dynamical systems.

Context-free (type 2) languages are generalizations of regular languages. Words in context-free languages may be viewed as sequences of terminal nodes (leaves) in trees constructed according to context-free grammatical rules. Each non-terminal symbol in the context-free grammar is taken to correspond to a type of tree node. The production rules for the non-terminal symbol then specify its possible descendents in the tree. For each word in the language, there corresponds such a "derivation" tree, rooted at the start symbol. (In most context-free languages, there are "ambiguous" words, obtained from multiple distinct derivation trees.) The syntax for most practical computer languages is supposed to be context-free. Each grammatical production rule corresponds to a subexpression with a particular structure (such as $u \bigcirc v$); the subexpressions may be arbitrarily nested [as in $((a \bigcirc (b \bigcirc c)) \bigcirc d) \bigcirc e$)], corresponding to arbitrary derivation trees.

Regular languages correspond to context-free languages whose derivation trees consist only of a "trunk" sprouting a sequence of leaves, one at a time. An example of a context-free language not represented by any regular grammar is the sequence of strings of the form 0^n10^n for any n. (Here, as elsewhere, α^n represents n-fold repetition of the string α.) A derivation tree for a word in this language is shown in Fig. 1.4. In general, the productions of any context-free language may in fact be arranged so that all derivation trees are binary (Chomsky normal form)[4].

At each point in the generation of a word in a regular language, the next symbol depends only on the current finite automaton state, and not on any previous history. (Regular language words may thus be considered as Markov chains.) To generate words in a context-free language, however, one must maintain a "stack" (last-in first-out memory), which at each point represents the part of the derivation tree above the symbol (tree leaf) just generated. In this way, words in context-free languages may exhibit certain long-range correlations, as illustrated in Fig. 1.4. (In

47 Compare many implementations of the LISP programming language. Also, compare with models of multiparticle production cascade processes (e.g. [13])

practical computer languages, these long-range correlations are typically manifest in the pairing of parentheses separated by many subexpressions.)

The production rules of a context-free grammar specify transformations for individual non-terminal symbols, independent of the "context" in which they appear. Context-sensitive grammars represent a generalization in which transformations for a particular symbol may depend on the strings of symbols that precede or follow it (its "context"). However, the transformation or production rule for any string α_1 is required to yield a longer (or equal length) string α_2. The set of all strings of the form $0^n1^n0^n$ for any n forms a context-sensitive language, not represented by any context-free or simpler language. The words in a context-sensitive language may be viewed as formed from sequences of terminal nodes in a directed graph. The graph is a derivation tree rooted at the start symbol, but with connections representing context sensitivities added. The requirement $|\alpha_2| \geqq |\alpha_1|$ implies that there are progressively more nodes at each stage: the length of a word in context-sensitive language thus gives an upper bound on the number of nodes that occur at any stage in its derivation. A machine that recognizes words in a context-sensitive language by enumerating all applicable derivation graphs need therefore only have a memory as large as the words to be recognized.

Unrestricted (type 0) languages are associated with universal computers. A system is considered capable of "universal computation" if, with some particular input, it can simulate the behaviour of any other computational system[5]. A universal computer may thus be "programmed" to implement any finite algorithm. A universal Turing machine has an infinite memory, and a central processing unit with a particular "instruction set." (The "simplest" known universal Turing machine has seven internal states, and a memory arranged as a line of sites, each having four possible values, and with one site accessible to the central processing unit at each time step (e.g. [8]).) Several quite different systems capable of universal computation have also been found. Among these are string manipulation systems which directly apply the production rules of type 0 languages; machines with one, infinite precision, arithmetic register; logic circuits analogous to those of practical digital electronic computers; and mathematical systems such as λ-calculus (general recursive functions). Some cellular automata have also been proved capable of universal computation. For example, a one-dimensional cellular automaton with $k=18$ and $r=1$ is equivalent to the simplest known universal Turing machine (e.g. [14]). (A two-dimensional cellular automata, the "Game of Life", with $k=2$ and a nine site neighbourhood, has also been proved computationally universal (e.g. [15]).) It is conjectured that all cellular automata in the fourth class indicated above are in fact capable of universal computation [2].

There are many problems which can be stated in finite terms, but which are "undecidable" in a finite time, even for a universal computer[6]. An example is the "halting problem": to determine whether a particular computer will "halt" in a finite time, given particular input. The only way to predict the behaviour of some

5 Although there are some mathematically-defined operations which they cannot perform (as discussed below), it seems likely that the usual class of "universal computers" can simulate the behaviour of any physically-realizable system

6 This is a form of Godel's theorem, in which the processes of mathematical proof are formalized in the operation of a computer

system **S** is to execute some procedure in a universal computer; but if, for example, **S** is itself a universal computer, then the procedure must reduce to a direct simulation, and can run no more than a finite amount faster than the evolution of **S** itself. The infinite time behaviour of **S** cannot therefore be determined in general in a finite time. For a cellular automaton, an analogue of the halting problem is to determine whether a particular finite initial configuration will ultimately evolve to the null configuration.

Any problem which depends on the results of infinite information processing may potentially be undecidable. However, when the information processing is sufficiently simple, there may be a finite "short-cut" procedure to determine the solution. For example, the information processing corresponding to the evolution of cellular automata with only class 1 or 2 behaviour appears to be sufficiently simple that their infinite time behaviour may be found by finite computation. Many problems concerning the infinite time behaviour of class 3 and 4 cellular automata may, however, be undecidable. For example, the entropies of the invariant sets for class 3 and 4 cellular automata may in general be non-computable numbers. This would be the case if the languages corresponding to these limit sets were of type 0 or 1.

It seems likely, in fact, that the consequences of infinite evolution in many dynamical systems may not described in finite mathematical terms, so that many questions concerning their limiting behaviour are formally undecidable. Many features of the behaviour of such systems may be determined effectively only by explicit simulation: no general predictions are possible.

Even for results that can in principle be obtained by finite computation there is a wide variation in the magnitude of time (or memory resources) required. Several classes of finite computations may be distinguished (e.g. [7]).

The first class (denoted P) consists of problems that can be solved by a deterministic procedure in a time given by some polynomial function of the size of their input. For example, finding the successor of a length n sequence in a cellular automaton takes (at most) a time linear in n, and is therefore a problem in the class P. Since most universal computers can simulate any other computer in a polynomial time, the times required on different computers usually differ at most by a polynomial transformation, and the set of problems in class P is defined almost independent of computer.

Nondeterministic polynomial time problems (NP) form a second class. Solutions to such problems may not necessarily be obtained in a polynomial time by a systematic procedure, but the correctness of a candidate solution, once guessed, can be tested in a polynomial time. Clearly $P \subseteq NP$, and there is considerable circumstantial evidence that $P \neq NP$. The problem of finding a pre-image for a length n sequence under cellular automaton evolution is in the class NP.

The problem classes P and NP are characterized by the times required for computations. One may also consider the class of problems PSPACE that require memory space given by a polynomial function of the size of the input, but may take an arbitrary time. There is again circumstantial evidence that $P \subset PSPACE$.

Just as there exist universal computers which, when given particular input, can simulate any other computer, so, analogously, there exist "NP-complete" (or

"PSPACE-complete") problems which, with particular input, correspond to any NP (or PSPACE) problem of a particular size (e.g. [6, 7]). Many NP and PSPACE complete problems are known. An example of an NP-complete problem is "satisfiability": finding truth values for n variables which make a particular Boolean expression true. If $P \neq \mathrm{NP}$ then there is essentially no faster method to solve this problem than to try the 2^n possible sets of values. (It appears that any method must at least require a time larger than any polynomial in n.) As discussed in Sect. 6, it is likely that the problem of finding pre-images for sequences in certain cellular automata, or of determining whether particular sequences are ever generated, is NP-complete. This would imply that no simple description exists even for some finite time properties of cellular automata: results may be found essentially only by explicit simulation of all possibilities.

2. Construction of Finite Time Sets

This section describes the construction of the set of configurations $\Omega^{(t)}$ generated after a finite number of time steps t of cellular automaton evolution, starting from the set $\Omega^{(0)} = \Sigma$ of all possible configurations. It is shown that $\Omega^{(t)}$ may be represented as a regular language (cf. [2, 16]), and an explicit construction of the minimal grammar for this language is given. Section 3 describes some properties of such grammars, and Sect. 4 discusses their form for a variety of cellular automata.

To describe the construction we begin with a simple example. The procedure followed may be generalized directly.

Consider the construction of the set $\Omega^{(1)}$ generated by one time step in the evolution of the $k=2$, $r=1$ cellular automaton with a local rule ϕ given by ("rule number 76" [1])

$$111\to 0,\quad 110\to 1,\quad 101\to 0,\quad 100\to 0,\quad 011\to 1,\quad 010\to 1,\quad 001\to 0,\quad 000\to 0. \tag{2.1}$$

The value $a_i^{(1)}$ of a site at position i in a configuration $A^{(1)} = \Phi A^{(0)} \in \Omega^{(1)}$ depends on a neighbourhood of three sites $\{a_{i-1}^{(0)}, a_i^{(0)}, a_{i+1}^{(0)}\}$ in the preceding configuration $A^{(0)} = \Omega^{(0)}$. The adjacent site $a_{i+1}^{(1)}$ depends on the overlapping neighbourhood $\{a_i^{(0)}, a_{i+1}^{(0)}, a_{i+2}^{(0)}\}$. The dependence of $a_{i+1}^{(1)}$ on $a_i^{(0)}$ associated with this two-site overlap in neighbourhoods may be represented by the graph g of Fig. 2.1 (analogous to a de Bruijn graph [17]). The nodes in the graph represent the overlaps $\{a_i^{(0)}, a_{i+1}^{(0)}\}$. These nodes are joined by directed arcs corresponding to three-site neighbourhoods. The local cellular automaton rule ϕ of Eq. (2.1) defines a transformation for each three-site neighbourhood, and thus associates a symbol with each arc of g. Each possible path through g corresponds to a particular initial configuration $A^{(0)}$. The successor $A^{(1)}$ of each initial configuration is given by the sequence of symbols associated with the arcs on the path. The sequences of symbols obtained by following all possible paths through g thus correspond to all possible configurations $A^{(1)}$ obtained after one time step in the evolution of the cellular automaton (2.1). The complete set $\Omega^{(1)}$ may thus be represented by the graph g. It is clear that not all possible sequences of 0's and 1's can appear in the configurations of $\Omega^{(1)}$. For example, no path in g can include the sequence 111, and thus no configuration in $\Omega^{(1)}$ can contain a block of sites 111.

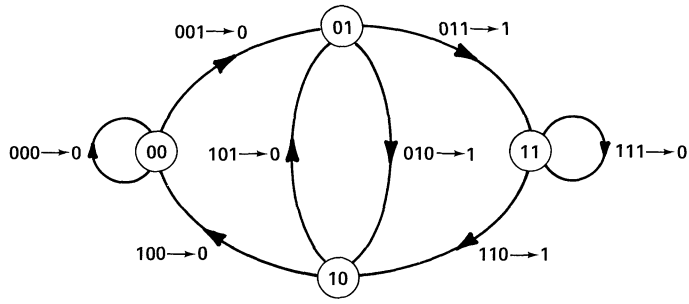


Fig. 2.1. The state transition graph g for a non-deterministic finite automaton (NDFA) that generates configurations obtained after one time step in the evolution of the $k=2$, $r=1$ cellular automaton with rule number 76 [Eq. (2.1)]. Possible sequences of site values are represented by possible paths through the graph. The nodes in the graph are labelled by pairs of initial site values; the arcs then correspond to triples of initial site values. Each such triple is mapped under rule number 76 to a particular site value. The graph with arcs labelled by these site values corresponds to all possible configurations obtained after one time step. Note that the basic graph is the same for all $k=2$, $r=1$ cellular automata; only the images of the initial site value triples change from one rule to another

The graph g of Fig. 2.1 may be considered as the state transition graph for a finite automaton which generates the formal language $\Omega^{(1)}$. Each node of g corresponds to a state of the finite automaton, and each arc to a transition in the finite automaton, or equivalently to a production rule in the grammar represented by the finite automaton. The set $\Omega^{(1)}$ thus forms a regular language. Labelling the states in g as u_0, u_1, u_2, u_3, the productions in the grammar for this language are:

$$\begin{aligned} &u_0 \to 0u_0\,, \quad u_0 \to 0u_1\,, \quad u_1 \to 1u_2\,, \quad u_1 \to 1u_3\,, \quad u_2 \to 0u_0\,, \\ &\qquad u_2 \to 0u_1\,, \quad u_3 \to 0u_3\,, \quad u_3 \to 1u_2\,. \end{aligned} \tag{2.2}$$

This finite set of rules provides a complete specification of the infinite set $\Omega^{(1)}$.

Each path through g corresponds uniquely to a particular initial configuration $A^{(0)}$. But several different paths may yield the same successor configuration $A^{(1)}$. Each such path corresponds to a distinct inverse image of $A^{(1)}$ under Φ. Enumeration of paths in g shows, for example, that there are 5 distinct inverse images for the sequence 00 under the cellular automaton mapping (2.1), 5 also for 01 and 10, and 1 for 11.

The finite automaton g of Fig. 2.1 is not the only possible one that generates the language $\Omega^{(1)}$. An alternative finite automaton $\bar{g}$ is shown in Fig. 2.2, and may be considered "simpler" than g since it has fewer states. $\bar{g}$ is obtained from g by combining the 00 and 10 nodes, which are equivalent in that only paths carrying the same symbol sequences pass through these nodes. The complete set of symbol sequences generated by the possible paths through $\bar{g}$ is identical to that generated by possible paths through g.

The finite automata g and $\bar{g}$ are non-deterministic in the sense that multiple arcs carrying the same symbol emanate from some nodes, so that several distinct paths may generate the same word in the formal language. It is convenient for many purposes to find deterministic finite automata (DFA) equivalent to the non-

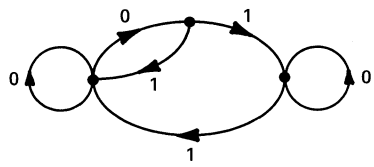


Fig. 2.2. The state transition graph $\bar{g}$ for an alternative NDFA that generates the language $\Omega^{(1)}$ obtained after one time step of evolution according to rule 76. This NDFA is obtained by combining two equivalent states in the NDFA g of Fig. 2.1

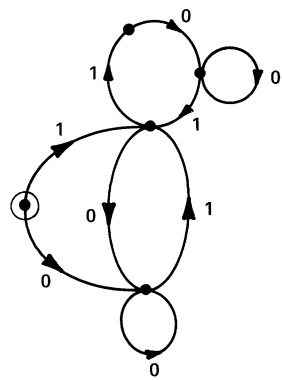


Fig. 2.3. The state transition graph G for a deterministic finite automaton (DFA) obtained from the non-deterministic finite automaton of Fig. 2.1 by the subset construction [and represented by the productions of Eq. (2.3)]. Here and in other DFA graphs, the start node ψ_S is shown encircled. Words in the regular language $\Omega^{(1)}$ correspond to paths through G, starting at ψ_S

deterministic finite automata (NDFA) g and $\bar{g}$. Such DFA may always be found by the standard "subset construction" (e.g. [6, 7]).

Consider for example the construction of a DFA G equivalent to the NDFA g of Fig. 2.1. Let ψ be the set of all possible subsets of the set of nodes $\{u_i\}$ (the power set of $\{u_i\}$). There are $2^4 = 16$ elements ψ_i of ψ; each potentially corresponds to a state in G. The construction of G begins from the "start node" $\psi_S = \{u_0, u_1, u_2, u_3\}$. This node is joined by a 0 arc to the node $\{u_0, u_1, u_3\}$ corresponding to the set of NDFA states reached by a 0 arc according to (2.2) from any of the u_i in ψ_S. An analogous procedure is applied for each arc at each node in G. The resulting graph is shown in Fig. 2.3, and may be represented by the productions

$$\begin{aligned}
\psi_S = \{u_0, u_1, u_2, u_3\} &\to 0\{u_0, u_1, u_3\}, \quad \{u_0, u_1, u_2, u_3\} \to 1\{u_2, u_3\}, \\
\{u_0, u_1, u_3\} &\to 0\{u_0, u_1, u_3\}, \quad \{u_0, u_1, u_3\} \to 1\{u_2, u_3\}, \\
\{u_2, u_3\} &\to 0\{u_0, u_1, u_3\}, \quad \{u_2, u_3\} \to 1\{u_2\}, \\
\{u_2\} &\to 0\{u_0, u_1\}, \quad \{u_2\} \to 1\{\}, \\
\{u_0, u_1\} &\to 0\{u_0, u_1\}, \quad \{u_0, u_1\} \to 1\{u_2, u_3\}.
\end{aligned} \tag{2.3}$$

Notice that only 5 of the 16 possible ψ_i are reached by transitions from ψ_S. The production in Eq. (2.3) yielding the null set $\{\}$ (often denoted ε) signifies the absence of an arc carrying the symbol 1 emanating from the $\{u_2\}$ node.

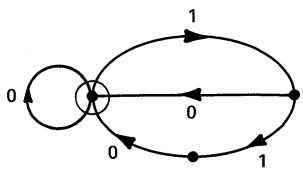


Fig. 2.4. The state transition graph $\bar{G}$ for the minimal DFA that generates the regular language $\Omega^{(1)}$ obtained after one time step of evolution according to cellular automaton rule 76. The graph is obtained by combining equivalent nodes in the DFA G of Fig. 2.3. It has the smallest possible number of nodes

The DFA G of Fig. 2.3 provides an alternative complete description of the language $\Omega^{(1)}$ represented by the NDFA g and $\bar{g}$ of Figs. 2.1 and 2.2. Possible sequences of symbols in words of $\Omega^{(1)}$ correspond to possible paths through G, starting at ψ_S. Consider the procedure for recognizing whether a sequence α can occur in $\Omega^{(1)}$. If α can occur, then it must correspond to a path through the NDFA g, starting at some node. The set of possible paths through g is represented by a single path through the DFA G. The start state ψ_S in G corresponds to the set of all possible states in g. As each symbol in the sequence α is scanned, the DFA G makes a transition to a state representing the set of states that g could reach at that point. The sequence α can thus occur in a word of $\Omega^{(1)}$ if and only if it corresponds to a path in G. The deterministic nature of G ensures that this path is unique.

Complete cellular automaton configurations consist of infinite sequences of symbols, and correspond to infinite paths in the DFA graph G. The possible words in $\Omega^{(1)}$ may thus be generated by following all possible paths through G.

Just as for the NDFA g, some of the states in the DFA G are equivalent, and may be combined. Two states are equivalent if and only if transitions from them with all possible symbols (here 0 or 1) lead to equivalent states. An equivalent DFA $\bar{G}$ shown in Fig. 2.4 may thus be obtained by representing each equivalence class of states in G by a single state. It may be shown that this DFA is the minimal one that recognizes the language $\Omega^{(1)}$ [18, 6, 7]. It is unique (up to state relabellings), and has fewer states than any equivalent DFA. Such a procedure yields the minimal form for any DFA; the analogous procedure for NDFA does not, however, necessarily yield a minimal form.

In most cases, the minimal DFA that generates all (two-way) infinite words of a regular language is the same as the minimal DFA constructed above that recognizes all finite (or one-way infinite) sequences of symbols in words of the language. In some cases, such as that of Fig. 2.5 (the set $\Omega^{(1)}$ for rule number 18), however, the latter DFA may contain additional "transient" subgraphs rooted at ψ_S, feeding into the main graph. The set of infinite paths through these transient subgraphs is typically a subset of the set of infinite paths in the main graph.

The minimal DFA $\bar{G}$ of Fig. 2.4 provides a simple description of the regular language $\Omega^{(1)}$. Regular expressions, mentioned in Sect. 1, provide a convenient notation for this and other regular languages. In terms of regular expressions,

$$\Omega^{(1)} = ((0^*)1(0 \vee 10)), \tag{2.10}$$

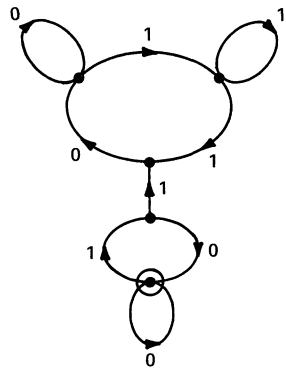


Fig. 2.5. The state transition graph $\bar{G}$ for the minimal DFA corresponding to the regular language $\Omega^{(1)}$ obtained after one time step in the evolution of cellular automaton rule 18. This graph contains a "transient" subgraph rooted at the start state, feeding into the main graph. All symbol sequences occurring at any point in a word $\Omega^{(1)}$ may be recognized as corresponding to paths through $\bar{G}$ beginning at the start state. Complete words in $\Omega^{(1)}$ may nevertheless be generated as possible infinite paths in $\bar{G}$, with the transient subgraph removed

where infinite repetition to form each infinite word is understood. Here α^* represents an arbitrary number (possibly zero) of repetitions of the string α, and $\alpha_1 \vee \alpha_2$ stands for α_1 or α_2.

The example discussed so far generalizes immediately to show that the set $\Omega^{(t)}$ of configurations generated by t time steps of evolution according to any cellular automaton rule forms a regular language. Constructions analogous to those described above give grammars for these languages. The number of states in the initial NDFA g is in general k^{2rt}. (Two examples are shown in Fig. 2.6; graphs for successively larger values of rt may be obtained by a recursive construction [17].) The size of the DFA G obtained from g by the subset construction may be as large as $2^{k^{2rt}}-1$, but is usually much smaller. (Note that the "reject" state $\{\}$ is not counted in the size of the grammar.)

As an example, consider the language $\Omega^{(2)}$ generated by two time steps in the evolution of the cellular automaton (2.1). The original NDFA g which corresponds to this language has 16 states, and the DFA G obtained from it by the subset construction has nine states. Nevertheless, the resulting minimal DFA $\bar{G}$ has just three states, and is in fact identical to that found for $\Omega^{(1)}$ as shown in Fig. 2.4. Since $\bar{G}$ gives a complete (finite) specification of the languages $\Omega^{(t)}$, this implies that

$$\Omega^{(1)} = \Omega^{(2)} = \Phi\Omega^{(1)} \tag{2.11}$$

in this case. $\Omega^{(1)}$ is thus the limit set for the evolution of the cellular automaton of Eq. (2.1).

3. Properties of Finite Time Sets

This section discusses some properties of the regular language sets $\Omega^{(t)}$ generated by a finite number of steps of cellular automaton evolution, and constructed by the procedure of Sect. 2.

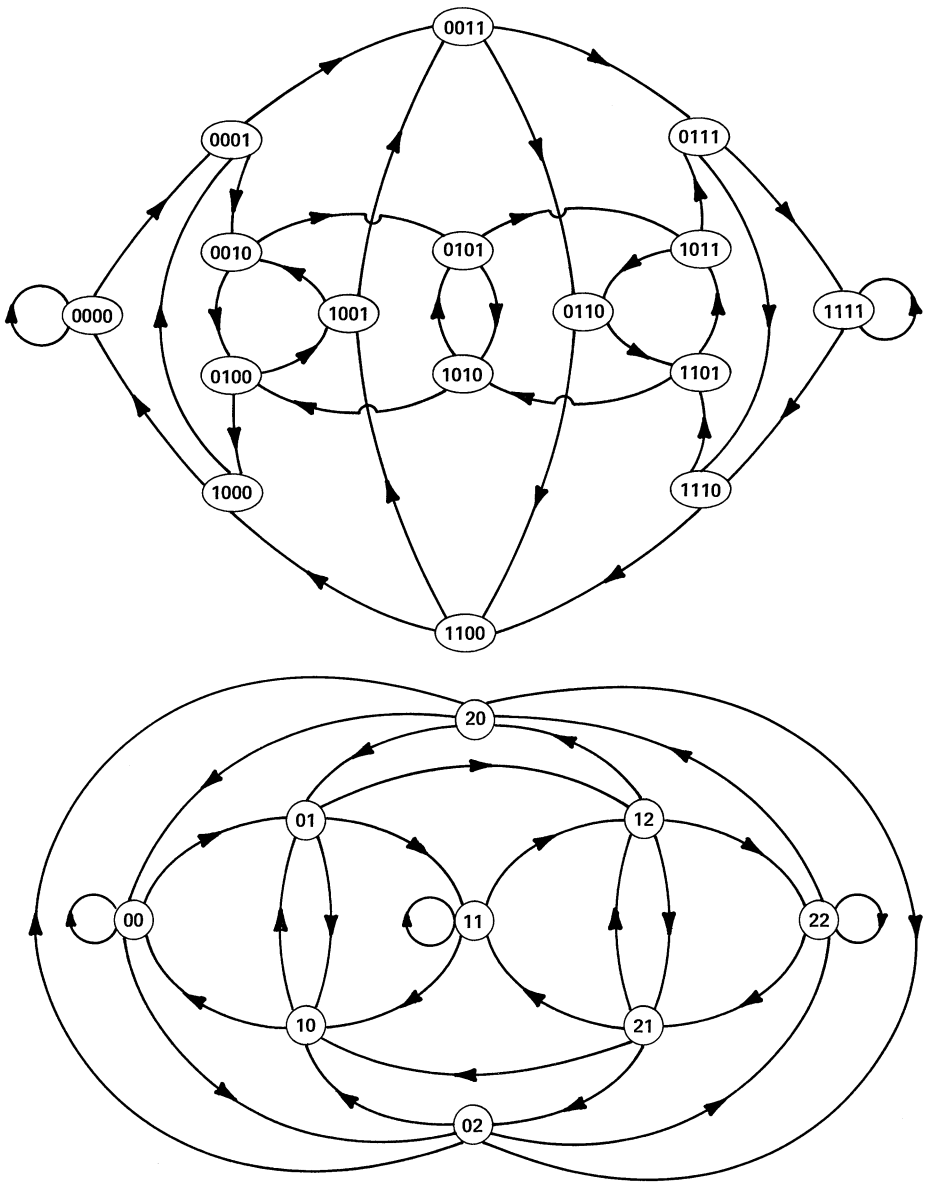


Fig. 2.6. Non-deterministic finite automaton graphs (de Bruijn graphs) analogous to Fig. 2.1 for the cases $k=2$, $r=2$, and $k=3$, $r=1$

We consider as a sample set the 32 "legal" cellular automaton rules with $k=2$ and $r=1$. A rule ϕ is considered legal if it is symmetric, and maps the null configuration (with all site values 0) to itself. Each of the 256 possible $k=2$, $r=1$ cellular automaton rules is conveniently labelled by a "rule number," defined as the decimal equivalent of the sequence of binary digits $\phi[1, 1, 1]$, $\phi[1, 1, 0], \ldots,$ $\phi[0, 0, 0]$ (analogous to Eq. (2.1)) [1].

Table 1. Numbers of nodes $\Xi^{(t)}$ (and arcs) in minimal deterministic finite automata (DFA) representing regular languages corresponding to sets of configurations $\Omega^{(t)}$ generated after t time steps in the evolution of legal $k=2$, $r=1$ cellular automata. Each configuration corresponds to a path through the DFA state transition graph. The construction of Sect. 2 yields the DFA with the minimal number of nodes (states) $\Xi^{(t)}$ that generates a given regular language $\Omega^{(t)}$. This DFA may be considered to give the shortest specification of $\Omega^{(t)}$ viewed as a regular language. Its size $\Xi^{(t)}$ measures the "complexity" of $\Omega^{(t)}$. The initial ($t=0$) set of configurations include all possible sequences of zeroes and ones, and correspond to a trivial regular language. Cellular automata with only class 1 or 2 behaviour yield regular languages whose complexities become constant, or increase as polynomials in t. Cellular automata capable of class 3 or 4 behaviour usually lead to rapidly-increasing complexities. Bounds on these complexities are given when their exact calculation exceeded available computational resources. Some of the results in this table were obtained using the methods of [52] and [53]

Rule	$\Xi^{(0)}$	$\Xi^{(1)}$	$\Xi^{(2)}$	$\Xi^{(3)}$	$\Xi^{(4)}$
0	1 (2)	1 (1)	1 (1)	1 (1)	1 (1)
4	1 (2)	2 (3)	2 (3)	2 (3)	2 (3)
18	1 (2)	5 (9)	47 (91)	143 (270)	$\gtrsim$ 20000
22	1 (2)	15 (29)	280 (551)	4506 (8963)	$\gtrsim$ 20000
32	1 (2)	2 (3)	5 (7)	7 (9)	9 (11)
36	1 (2)	3 (5)	3 (4)	3 (4)	3 (4)
50	1 (2)	3 (5)	8 (14)	10 (17)	12 (20)
54	1 (2)	9 (16)	17 (32)	94 (179)	675 (1316)
72	1 (2)	5 (9)	5 (8)	5 (8)	5 (8)
76	1 (2)	3 (5)	3 (5)	3 (5)	3 (5)
90	1 (2)	1 (2)	1 (2)	1 (2)	1 (2)
94	1 (2)	15 (29)	230 (455)	3904 (7760)	$\gtrsim$ 20000
104	1 (2)	15 (29)	265 (525)	2340 (4647)	1394 (2675)
108	1 (2)	9 (16)	11 (19)	11 (19)	11 (19)
122	1 (2)	15 (29)	179 (347)	5088 (9933)	$\gtrsim$4000
126	1 (2)	3 (5)	13 (23)	107 (198)	2867 (5476)
128	1 (2)	4 (6)	6 (8)	8 (10)	10 (12)
132	1 (2)	5 (9)	7 (12)	9 (15)	11 (18)
146	1 (2)	15 (29)	92 (177)	1587 (3126)	$\gtrsim$ 20000
150	1 (2)	1 (2)	1 (2)	1 (2)	1 (2)
160	1 (2)	9 (15)	16 (24)	25 (35)	36 (48)
164	1 (2)	15 (29)	116 (227)	667 (1310)	1214 (2363)
178	1 (2)	11 (20)	15 (26)	19 (32)	23 (38)
182	1 (2)	15 (29)	92 (177)	1587 (3126)	$\gtrsim$ 20000
200	1 (2)	3 (5)	3 (5)	3 (5)	3 (5)
204	1 (2)	1 (2)	1 (2)	1 (2)	1 (2)
218	1 (2)	15 (29)	116 (227)	667 (1310)	1214 (2363)
222	1 (2)	5 (9)	7 (12)	9 (15)	11 (18)
232	1 (2)	11 (20)	15 (26)	19 (32)	23 (38)
236	1 (2)	3 (5)	3 (5)	3 (5)	3 (5)
250	1 (2)	9 (15)	16 (24)	25 (35)	36 (48)
254	1 (2)	4 (6)	6 (8)	8 (10)	10 (12)

Table 2. Characteristic polynomials $\chi^{(1)}(\lambda)$ for the adjacency matrices of state transition graphs for minimal DFA representing regular languages generated after one time step in the evolution of legal $k=2$, $r=1$ cellular automata. The nonzero roots of these polynomials determine the number of distinct symbol sequences that can appear in configurations generated by the cellular automaton evolution. The maximal root $\lambda_{\max}$ determines the limiting entropy of the sequences

Rule	$\chi^{(1)}(\lambda)$	$\lambda_{\max}$
0	$1-\lambda$	1.000
4	$-1-\lambda+\lambda^2$	1.618
18	$(1-\lambda-\lambda^2)(-1+\lambda-2\lambda^2+\lambda^3)$	1.755
22	$\lambda(1-\lambda)(2-2\lambda^2+6\lambda^3-3\lambda^4-5\lambda^5+10\lambda^6-5\lambda^7-3\lambda^8+6\lambda^9-2\lambda^{10}+2\lambda^{11}-3\lambda^{12}+\lambda^{13})$	1.917
32	$-1-\lambda+\lambda^2$	1.618
36	$1-\lambda+2\lambda^2-\lambda^3$	1.755
50	$1+\lambda+\lambda^2-\lambda^3$	1.839
54	$\lambda^3(1+\lambda^2)(1-\lambda+2\lambda^3-\lambda^4)$	1.867
72	$(1+\lambda-\lambda^2)(-1+\lambda-2\lambda^2+\lambda^3)$	1.755
76	$1+\lambda+\lambda^2-\lambda^3$	1.839
90	$2-\lambda$	2.000
94	$-\lambda(2-2\lambda+2\lambda^2-\lambda^3+2\lambda^4-5\lambda^5+13\lambda^6-16\lambda^7+10\lambda^8-3\lambda^{10}-\lambda^{11}+5\lambda^{12}-4\lambda^{13}+\lambda^{14})$	1.883
104	$\lambda(1-\lambda)(2-2\lambda^2+6\lambda^3-3\lambda^4-5\lambda^5+10\lambda^6-5\lambda^7-3\lambda^8+6\lambda^9-2\lambda^{10}+2\lambda^{11}-3\lambda^{12}+\lambda^{13})$	1.917
108	$\lambda^3(1+\lambda^2)(1-\lambda+2\lambda^3-\lambda^4)$	1.867
122	$-\lambda(2-2\lambda+2\lambda^2-\lambda^3+2\lambda^4-5\lambda^5+13\lambda^6-16\lambda^7+10\lambda^8-3\lambda^{10}-\lambda^{11}+5\lambda^{12}-4\lambda^{13}+\lambda^{14})$	1.883
126	$1-\lambda+2\lambda^2-\lambda^3$	1.755
128	$(-1-\lambda+\lambda^2)(1-\lambda+\lambda^2)$	1.618
132	$1-\lambda^2+2\lambda^4-\lambda^5$	1.785
146	$-\lambda(-2+4\lambda-6\lambda^2+4\lambda^3+\lambda^4-7\lambda^5+12\lambda^6-13\lambda^7+9\lambda^8-4\lambda^9+\lambda^{10}-2\lambda^{11}+5\lambda^{12}-4\lambda^{13}+\lambda^{14})$	1.887
150	$2-\lambda$	2.000
160	$(1-\lambda^2-\lambda^3)(1-\lambda^2+\lambda^3)(-1+\lambda-2\lambda^2+\lambda^3)$	1.755
164	$-\lambda(2-\lambda^2-2\lambda^4+5\lambda^5-9\lambda^6+14\lambda^7-9\lambda^8+2\lambda^9-6\lambda^{10}+5\lambda^{11}+3\lambda^{12}-4\lambda^{13}+\lambda^{14})$	1.915
178	$\lambda(1-\lambda^2+\lambda^5)(1-\lambda^2+2\lambda^4-\lambda^5)$	1.785
182	$-\lambda(-2+4\lambda-6\lambda^2+4\lambda^3+\lambda^4-7\lambda^5+12\lambda^6-13\lambda^7+9\lambda^8-4\lambda^9+\lambda^{10}-2\lambda^{11}+5\lambda^{12}-4\lambda^{13}+\lambda^{14})$	1.887
200	$1-\lambda+2\lambda^2-\lambda^3$	1.755
204	$2-\lambda$	2.000
218	$-\lambda(2-\lambda^2-2\lambda^4+5\lambda^5-9\lambda^6+14\lambda^7-9\lambda^8+2\lambda^9-6\lambda^{10}+5\lambda^{11}+3\lambda^{12}-4\lambda^{13}+\lambda^{14})$	1.915
222	$1-\lambda^2+2\lambda^4-\lambda^5$	1.785
232	$\lambda(1-\lambda^2-\lambda^5)(-1+\lambda^2-2\lambda^4+\lambda^5)$	1.785
236	$1-\lambda+2\lambda^2-\lambda^3$	1.755
250	$(1-\lambda^2-\lambda^3)(1-\lambda^2+\lambda^3)(-1+\lambda-2\lambda^2+\lambda^3)$	1.755
254	$(-1-\lambda+\lambda^2)(1-\lambda+\lambda^2)$	1.618

Tables 1, 2 and 3 give some properties of the sets $\Omega^{(t)}$ generated by a few time steps in the evolution of the 32 legal $k=2$, $r=1$ cellular automata[7]. These properties are deduced from the minimal DFA which describe the $\Omega^{(t)}$, obtained according to the construction of Sect. 2.

The minimal DFA corresponding to the trivial language $\Omega^{(0)}=\Sigma$ illustrated in Fig. 1.1(a) has just one state. The minimal DFA corresponding to the minimal regular grammars for more complicated languages have progressively more states.

7 Requests for copies of the *C* language computer program used to obtain these and other results in this paper should be directed to the author

Table 3. The length $L^{(t)}$ ov the shortest distinct blocks of site values newly-excluded after exactly t time steps in the evolution of legal $k=2$, $r=1$ cellular automata. The notation * indicates that the set of cellular automaton configurations $\Omega^{(t)}$ forms a finite complement language (finite number of distinct excluded blocks). The notation – signifies no new excluded blocks

Rule	$L^{(1)}$	$L^{(2)}$	$L^{(3)}$	$L^{(4)}$
0	1*	–	–	–
4	2*	–	–	–
18	3	11	12	13
22	8	7	11	9
32	2*	4*	6*	8*
36	3*	2*	–	–
50	3*	5*	9*	11*
54	5	9	9	7
72	3	3*		–
76	3*	–		–
90	–	–		–
94	5	7	11	11
104	8	8	8	7
108	5	4*		–
122	5	7	8	10
126	3*	12	13	14
128	3*	5*	7*	9*
132	4*	5*	6*	7*
146	6	6	8	8
150	–	–	–	–
160	5*	7*	9*	11*
164	9	9	8	9
178	5*	6*	7*	8*
182	6	6	8	8
200	3*	–	–	–
204	–	–	–	–
218	9	9	8	9
222	4*	5*	6*	7*
232	5*	6*	7*	8*
236	3*	–	–	–
250	5*	7*	9*	11*
254	3*	5*	7*	9*

The total number of states $\Xi^{(t)}$ in the minimal DFA that generates a set $\Omega^{(t)}$ provides a measure of the "complexity" of the set $\Omega^{(t)}$, considered as a regular language. $\Xi^{(t)}$ gives the size of the shortest specification of the set $\Omega^{(t)}$ in terms of regular languages: this shortest specification becomes longer as the complexity of the set increases.

Table 1 gives the "regular language complexities" $\Xi^{(t)}$ for the sets $\Omega^{(t)}$ generated at the first few time steps in the evolution of the legal $k=2$, $r=1$ cellular automata.

In all the cases given, $\Xi^{(t)}$ is seen to be non-decreasing with time. Cellular automata with only class 1 or 2 appear to give $\Xi^{(t)}$ which tend to constants after one or two time steps, or increase linearly or quadratically with time. Class 3 and 4 cellular automata usually give $\Xi^{(t)}$ which increase rapidly with time. In general,

$$1 \leqq \Xi^{(t)} \leqq 2^{k^{2rt}} - 1 . \tag{3.1}$$

The upper bound is found to be attained in several cases for $t = 1$; for larger t, $\Xi^{(t)}$ appears to grow at most exponentially with t.

All possible sequences of symbols occur in the trivial language Σ. In more complicated regular languages, only some number $N(X)$ of the k^X possible sequences of X symbols may occur. Each sequence which occurs corresponds to a distinct path in the minimal DFA graph for the language. (Note that all distinct paths in a DFA correspond to different symbol sequences; this need not be the case in a NDFA graph.) The number of such paths is conveniently computed using a matrix representation for the DFA.

Consider as an example the set $\Omega^{(1)}$ obtained by one time step in the evolution of the cellular automaton (2.1). The minimal DFA graph $\bar{G}$ for this set is given in Fig. 2.4, and may be represented by the adjacency matrix

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} . \tag{3.2}$$

The elements of M^X give the numbers $N(X)$ of possible length X paths in $\bar{G}$. For lengths from 1 to 10 these numbers are 2, 4, 7, 13, 24, 44, 81, 149, 274, and 504. In general, at least for large X,

$$N(X) \simeq \mathrm{Tr}[M^X] = \sum \lambda_i^X \sim \lambda_{\max}^X , \tag{3.3}$$

where the λ_i are the eigenvalues of M, and $\lambda_{\max}$ is the largest of them. These eigenvalues are determined from the characteristic polynomial $\chi(\lambda)$ for the minimal DFA adjacency matrix, given in the case of Eq. (3.2) by[8]

$$\chi(\lambda) = 1 + \lambda + \lambda^2 - \lambda^3 . \tag{3.4}$$

The largest (real) root of this characteristic polynomial (known as the "index" of the graph [19]) is given by the cubic algebraic integer

$$\begin{aligned} \lambda_{\max} &= [1 + \kappa + 4/\kappa] \simeq 1.83929 , \\ \kappa &= [(38 + \sqrt{1188})/2]^{1/3} \end{aligned} \tag{3.5}$$

The set of infinite configurations $\Omega^{(t)}$ generated by cellular automaton evolution may be considered to form a Cantor set. The dimension of this Cantor set is given by

$$s = \lim_{X \to \infty} \frac{1}{X} \log_k N(X) , \tag{3.6}$$

8 $1/\chi(\lambda)$ is related to the generating function for the sequence $N(X)$ (e.g. [19, Sect. 1.8])

and is equal to the topological entropy of the shift mapping restricted to this set (e.g. [20]). For any regular language, this entropy is given according to Eqs. (3.2) by [21]

$$s = \log_k \lambda_{\max} . \tag{3.7}$$

For the case of Eq. (3.2), the entropy is thus

$$s \simeq \log_2 1.83929 \simeq 0.87915 . \tag{3.8}$$

Table 2 gives the characteristic polynomials $\chi^{(1)}(\lambda)$ for the regular languages $\Omega^{(1)}$ obtained after one time step in the evolution of the 32 legal $k=2, r=1$ cellular automata, together with their largest real roots $\lambda_{\max}$. All the nonzero roots of the $\chi(\lambda)$ appear in the expression (3.3) for $N(X)$, and are therefore the same for all possible DFA corresponding to a particular regular language. (They may thus be considered "topological invariants.") Additional powers of λ may appear in the characteristic polynomials obtained from non-minimal DFA.

The characteristic polynomials $\chi(\lambda)$ such as those in Table 2 obtained from regular languages are always monic (the term with the highest power of λ that appears in them always has unit coefficient). The largest roots $\lambda_{\max}$ of the $\chi(\lambda)$ for regular languages are thus always algebraic integers (e.g. [22])[9], so that the entropies for regular languages are always the logarithms of algebraic integers. The minimal polynomial with $\lambda_{\max}$ as a root has a degree not greater than the size $\Xi^{(t)}$ of the minimal DFA for a regular language $\Omega^{(t)}$. This bound is usually not reached, since the characteristic polynomial $\chi(\lambda)$ is usually reducible, as seen in Table 2. Notice that in many cases, $\chi(\lambda)$ has several factors with equal degrees. (The factorizations of the $\chi(\lambda)$ are related to the colouring properties of the corresponding graphs [19]. Note that graphs corresponding to minimal DFA always have trivial automorphism groups.) Factors (other than λ^n) with smaller degrees appear to be associated with transient subgraphs in the minimal DFA graph.

The entropy (3.6) characterizes the number of distinct symbol sequences generated by cellular automaton evolution, without regard to the probabilities with which they occur. One may also define a measure entropy (e.g. [20])

$$s_\mu = - \lim_{X \to \infty} \sum_{i=1}^{k^X} p_i \log_k p_i \tag{3.9}$$

in terms of the probabilities p_i for length X sequences. Starting from an initial ensemble in which all symbol sequences of a given length occur with equal probabilities, the probability for a sequence i after t time steps is given by

$$p_i = \xi_i / k^{X+2rt} , \tag{3.10}$$

where ξ_i is the number of (length $X+2rt$) t-step preimages of the sequence i under the cellular automaton mapping Φ. This number is equal to the number of distinct paths through the NDFA graph analogous to g in Fig. 2.1 that yield the sequence i. It may also be computed from reduced NDFA graphs analogous to $\bar{g}$ of Fig. 2.2 by

9 The $\lambda_{\max}$ are always Perron numbers [23]. Any Perron number may be obtained from some regular language, and in fact also from some finite complement language [23]

including a weight for each path, equal to the product of weights giving the number of unreduced nodes combined into each node on the path.

The set of configurations generated by cellular automaton evolution always contracts or remains unchanged with time, as implied by Eq. (1.5). The entropies associated with the sets $\Omega^{(t)}$ are therefore non-increasing with time. Class 1 cellular automata are characterized by (spatial) entropies that tend to zero with time [2]. Class 2, 3 and 4 cellular automata generate sets of configurations with nonzero limiting spatial entropy. (Class 2 cellular automata nevertheless yield patterns essentially periodic in time, with zero temporal entropy.)

Some cellular automata have the special property that

$$\Phi\Sigma = \Sigma, \tag{3.11}$$

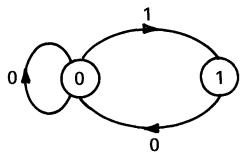
so that all possible configurations can occur at any time in their evolution, and the entropies of the $\Omega^{(t)}$ are always equal to one. Such surjective cellular automaton rules may be recognized by the presence of all k possible outgoing arcs at each node in a DFA representing the grammar of the set $\Omega^{(1)}$ obtained after one time step in their evolution. The finite maximum size $2^{k^{2r}}$ for such a DFA, constructed as in Sect. 2, ensures that this procedure (cf. [24–27, 2]) for determining the surjectiveness of any cellular automaton rule is a finite one[10].

Since there are k outgoing arcs at each node in the original NDFA analogous to Fig. 2.1 for any cellular automaton rule, the rule is surjective if in all cases these arcs carry distinct symbols (so that the NDFA is in fact a DFA). This occurs whenever the local cellular automaton mapping ϕ is injective with respect to its first or last argument (as for additive rules [29, 16] such as 90, 150 or 204 in Tables 1–3). However, at least when $k>2$ or $r>1$, there exist surjective cellular automata for which this does not occur [25, 30]. Since all surjective cellular automata must yield the same trivial minimal DFA, it is possible that a reversal of the minimization and subset algorithms discussed in Sect. 2 could be used to generate all NDFA analogous to Fig. 2.1 that correspond to surjective rules.

Surjective cellular automata yield trivial regular languages, in which all possible blocks of symbols may appear. Some cellular automata generate the slightly more complicated "finite complement" regular languages, in which a finite set of distinct blocks are excluded. (Such languages are equivalent to "subshifts of finite type" (e.g. [10, 11]).) An example of a finite complement language, illustrated in Fig. 1.1b, consists of all sequences from which the block of sites 11 is absent. To construct the grammar for a finite complement language in which blocks of length b are excluded, first form a graph analogous to Fig. 2.1, but with sequences of length $b-1$ at each node. Each arc then corresponds to a length b sequence, and may be labelled by the last symbol in the sequence. With this labelling, one arc carrying each of the k possible symbols emanates from each of the k^{b-1} nodes, so that the graph represents a DFA. Removing arcs corresponding to the excluded length b blocks then yields the graph for a DFA that recognizes the finite

10 The algorithm essentially involves testing whether a NDFA with k^{2r} states is equivalent to a NDFA that generates the trivial language Σ. This problem is known to be PSPACE-complete [28], and therefore presumably cannot be solved in a time polynomial in k^{2r}

a



b

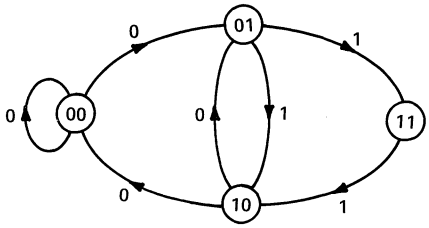


Fig. 3.1a and b. Non-deterministic finite automata (NDFA) corresponding to finite complement regular languages consisting of sequences of zeroes and ones in which **a** the block 11 is excluded, and **b** the block 111 is excluded. The graphs are constructed from analogues of Fig. 2.1 by dropping arcs corresponding to excluded blocks

complement language with these blocks absent. Examples of the resulting graphs for two simple cases are shown in Fig. 3.1.

The minimal DFA for a finite complement language with a maximal distinct excluded block of length b has at most k^{b-1} states, and at least b states. An excluded block is considered "distinct" if it contains no excluded sub-blocks. (Hence, for example, in the language of Figs. 1.1a and 3.1a, the excluded block 11 is considered distinct, but 110, 111 and so on, are not.)

Any path through the minimal DFA graph for a regular language of length greater than $\Xi^{(t)}$ must contain a cycle, which retraverses some arcs. If no symbol sequence of length less than $\Xi^{(t)}$ is excluded, then no sequence of any length can therefore be excluded, and the corresponding language must be trivial. If some symbol sequences of length less than $\Xi^{(t)}$ are excluded, but no distinct sequences with lengths between $\Xi^{(t)}$ and $2\Xi^{(t)}$ are excluded, then no longer distinct sequences can be excluded, and the corresponding language must be a finite complement one. If further distinct excluded blocks with lengths between $\Xi^{(t)}$ and $2\Xi^{(t)}$ are found, then an infinite series of longer distinct excluded blocks must exist, and the language cannot be a finite complement one.

The language of Fig. 2.4, generated by the evolution of the $k=2$, $r=1$ cellular automaton with rule number 76, is a finite complement one, in which 111 is the only distinct excluded block. The language of Fig. 2.5, obtained after one time step in the evolution of rule number 18, is not a finite complement one. The block 111 is the shortest excluded in this case. But the distinct length 7 block 1101011 is also excluded, as are the two distinct length 8 blocks 11001011 and 11010011, three distinct length 9 blocks (11010100011, 110001011, 110010011), four distinct length 10 blocks, and so on.

The length $L^{(t)}$ of the shortest excluded block in a language $\Omega^{(t)}$ generated by cellular automaton evolution (denoted X_c in [2]) is in general given by the shortest

distance from the start node in the corresponding DFA graph to an "incomplete" node, with less than k outgoing arcs. If the cellular automaton rule is not surjective, then

$$0 < L^{(t)} \leqq \Xi^{(t)} . \tag{3.12}$$

Whenever cellular automaton evolution is irreversible, the set of configurations $\Omega^{(t)}$ generated contracts with time, and progressively more distinct blocks are excluded. One may define $L^{(t)}$ to be the length of the shortest newly-excluded block at time step t in the evolution of a cellular automaton. The values of $L^{(t)}$ obtained in the first few time steps of evolution according to the 32 legal $k=2, r=1$ cellular automaton rules are given in Table 3. In most cases, $L^{(t)}$ is seen to increase with time, indicating that progressively finer subsets of Σ are excluded, and qualitatively reflecting the increase of $\Xi^{(t)}$. In general, however, $L^{(t)}$ need not increase monotonically with time. A length l block is excluded after t time steps if there is no initial length $l+2rt$ block that evolves into it. A length l block is newly excluded at time step t if and only if no length $l+2r$ blocks allowed at time step $t-1$ evolve to it, but at least one length $l+2r$ block newly excluded at time step $t-1$ would evolve to it. The length $L^{(t)}$ of the shortest newly excluded block at time t is thus bounded by

$$L^{(t)} \geqq L^{(t-1)} - 2r . \tag{3.13}$$

Table 3 includes several cases for which the lower bound is realized.

The sets of infinite symbol sequences $\Omega^{(t)}$ generated by cellular automaton evolution are characterized in part by the numbers and lengths of allowed and excluded finite blocks which appear in them. A further characterization may be given in terms of the number $\Pi(p)$ of infinite sequences with (spatial) period p that appear. This number is related to the number of distinct cycles in the minimal DFA graph for $\Omega^{(t)}$. Cycles are considered distinct if the sequences of symbols that appear in them are distinct. The enumeration of cycles thus requires knowledge of the arc labelling as well as connectivity of the DFA graph.

Just as the number of finite blocks $N(X)$ for all X may be summarized in the characteristic polynomial $\chi(\lambda)$, so also the number of periodic configurations $\Pi(p)$ may be summarized in the zeta function (e.g. [10, 11])

$$\zeta(\lambda) = \exp\left(\sum_{p=1}^{\infty} \Pi(p) \lambda^p / p \right) . \tag{3.14}$$

For all regular languages $\zeta(\lambda)$ is a rational function of λ [31]. For the special case of finite complement languages,

$$\zeta(\lambda) = 1/\chi(\lambda) . \tag{3.15}$$

A finite procedure may be given [32] to compute $\zeta(\lambda)$ for any regular language.

4. Evolution of Finite Time Sets

Tables 1–3 gave several properties of the sets of configurations generated by a finite number of steps in the evolution of legal $k=2, r=1$ cellular automata. This section

discusses these results, identifies several types of behaviour, and considers analogies with classes of cellular automaton behaviour defined by dynamical systems theory means [2].

In the simplest cases, the set $\Omega^{(t)}$ generated by a cellular automaton evolves to a fixed form after a small number of time steps T (the case of surjective cellular automata, with $\Omega^{(t)}=\Sigma$ for all t, is considered separately). The minimal DFA corresponding to $\Omega^{(t)}$ for all $t \geqq T$ are then identical, and the values of $\Xi^{(t)}$ and $\chi^{(t)}(\lambda)$ are thus constant. (Notice that $\Xi^{(t)}=\Xi^{(t+1)}$ does not necessarily imply $\Omega^{(t)}=\Omega^{(t+1)}$, as seen for rule 36 in Table 1.) In addition, for $t \geqq T$, no more distinct blocks of sites are excluded. Such behaviour occurs in the trivial case of rule 0, under which all initial configurations are mapped to the null configuration after one time step. It also occurs for many other rules: one example is rule 76, discussed in Sects. 2 and 3. All the examples of this behaviour in Tables 1–3 have $T=1$ (e.g. rule 76) or $T=2$ (e.g. rule 108). In the trivial case of rule 0, only a single configuration (the null configuration) can appear when $t \geqq T$. More complicated single configurations are sometimes generated, represented by minimal DFA consisting of a single cycle. In most cases (such as rule 76), however, $\Omega^{(T)}$ contains an infinite number of configurations. However, it appears that even in these cases, all configurations occur on finite cycles: each configuration is invariant under the cellular automaton mapping, or some finite iteration of it. (A result given in Sect. 5 then shows that the $\Omega^{(T)}$ must form finite complement languages in these cases.) This implies that changes in the initial state for such cellular automata propagate a distance of at most rT sites. A small initial change can thus ultimately affect a region no larger than $2rT$ sites. Such cellular automata must therefore exhibit class 1 or 2 behaviour [2].

For a second set of cellular automata, the form of the minimal DFA does not become fixed after a few time steps, but exhibits a simple growth with time, maintaining a fixed overall structure. The $L^{(t)}$ for such cellular automata typically increases linearly with time, and $\Xi^{(t)}$ increases as some polynomial function of t (linear or quadratic for legal $k=2$, $r=1$ rules). Rule 128 gives an example of this behaviour. Under this rule $111 \rightarrow 1$, but all other neighbourhoods map to 0. Any initial sequence of ones thus decreases steadily in length by one site on each side at each time step. After t time steps, any pair of ones must be separated by at least $2t+1$ zeroes; all blocks of the form 10^j1 for $1 \leqq j \leqq 2t$ are thus excluded. The first few languages $\Omega^{(t)}$ in the sequence generated by successive time steps in the evolution of rule 128 are shown in Fig. 4.1. The minimal DFA are seen to maintain the same overall structure, but include a linearly increasing number of nodes at each time step. The characteristic polynomials corresponding to these DFA are given by

$$\chi^{(t)}(\lambda)=(1-\lambda^t+\lambda^{t+1})(-1-\lambda^t+\lambda^{t+1}), \tag{4.1}$$

yielding a set entropy which tends to zero at large times, roughly as $1/t$. Rule 160 provides another example in which the minimal DFA maintains the same overall structure, but increases in size with time. In this case, sequences of the form $1[(0 \vee 1)0]^j(0 \vee 1)1$ for all $j \leqq t$, are excluded after t time steps, and the size $\Xi^{(t)}$ of the corresponding minimal DFA grows quadratically with time.

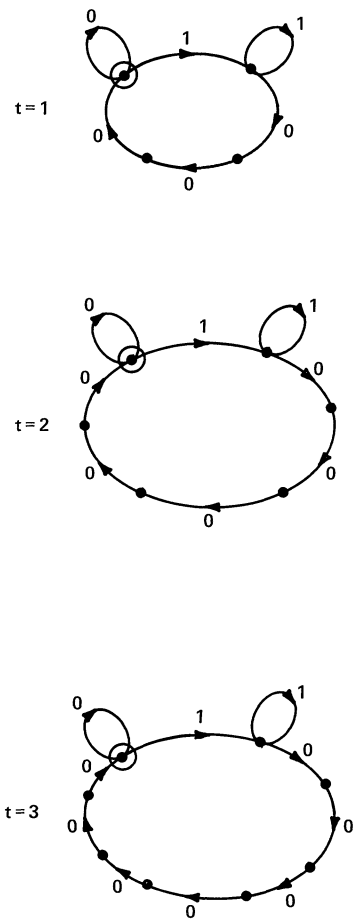


Fig. 4.1. Minimal deterministic finite automata (DFA) corresponding to the regular languages $\Omega^{(t)}$ generated in the first few time steps of evolution according to cellular automaton rule 128. The DFA maintain the same structure, but increase in size with time. They correspond to finite complement languages, with all blocks of the form 10^j1 excluded for $1 \leqq j < 2t$

Many cellular automata generate sets $\Omega^{(t)}$ whose corresponding minimal DFA become much more complicated at each successive time step, and appear to exhibit no simple overall structure.

Figure 4.2 shows the minimal DFA obtained after one and two time steps in the evolution of rule 126. No simple progression in the form of these minimal DFA is seen. $\Omega^{(1)}$ is a finite complement language, with only the block 010 excluded, yielding a characteristic polynomial

$$\chi^{(1)}(\lambda) = 1 - \lambda + 2\lambda^2 - \lambda^3, \tag{4.2}$$

giving $\lambda_{\max} \simeq 1.7549$. After two time steps, an infinite sequence of distinct blocks is excluded, starting with the length 12 block 011101101110. The corresponding

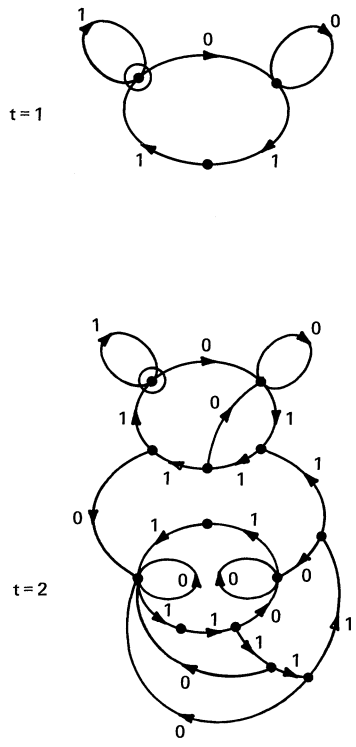


Fig. 4.2. Minimal deterministic finite automata corresponding to the regular languages generated in the first two time steps of evolution according to the class 3 cellular automaton rule 126. A considerable increase in complexity with time is evident, characteristic of cellular automata which can exhibit class 3 behaviour

characteristic polynomial is

$$\chi^{(2)}(\lambda) = -1+\lambda-\lambda^2+2\lambda^3-4\lambda^4+\lambda^5+3\lambda^6-5\lambda^7 \\ +3\lambda^8-3\lambda^9+5\lambda^{10}-6\lambda^{11}+4\lambda^{12}-\lambda^{13}, \tag{4.3}$$

with $\lambda_{\max} \simeq 1.7321$. The minimal DFA for $\Omega^{(3)}$ has 107 states, and the shortest newly-excluded block is 1011100011101 (length 13). $\Xi^{(t)}$ increases rapidly with time. After four time steps, the shortest newly-excluded blocks are 10111000011101, 10111000001110 and its reversal (length 14), and $\Xi^{(4)} = 2876$.

Figures 2.5 and 4.3 give the minimal DFA obtained after one and two time steps in the evolution of rule 18. A considerable increase in complication with time is again evident. After one time step, the shortest of an infinite number of distinct excluded blocks is 1101011 (length 7); after two time steps, the shortest newly-excluded block is 10011011001 (length 11); after three time steps, it is 110010010011 (length 12), and after four time steps it is 1001000010011 (length 13). In this case, as

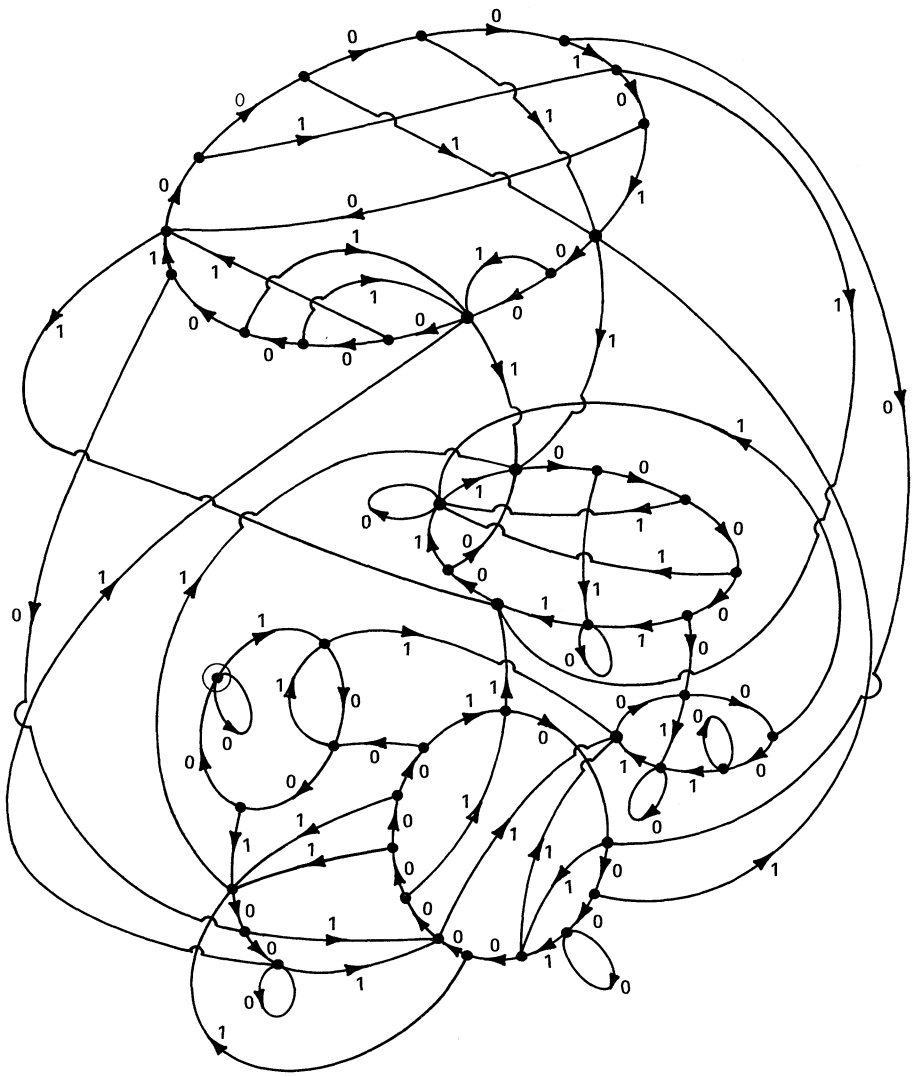


Fig. 4.3. Minimal deterministic finite automaton (DFA) corresponding to the regular language generated after two time steps of evolution according to the class 3 cellular automaton rule 18. The minimal DFA for $t = 1$ is shown in Fig. 2.5. Rapidly-increasing complexity is again evident. The DFA illustrated here has 47 states

for rule 126, $L^{(t)}$ is found to increase monotonically over the range of times investigated. Progressively larger neighbourhoods of the start state are therefore left unchanged in the corresponding minimal DFA. However, as discussed in Sect 3, $L^{(t)}$ need not increase with time, but must in general only satisfy the inequality (3.13). Rule 22 provides an example in which $L^{(t)}$ decreases with time. The minimal DFA for $\Omega^{(1)}$ in this case is shown in Fig. 4.4; the shortest excluded

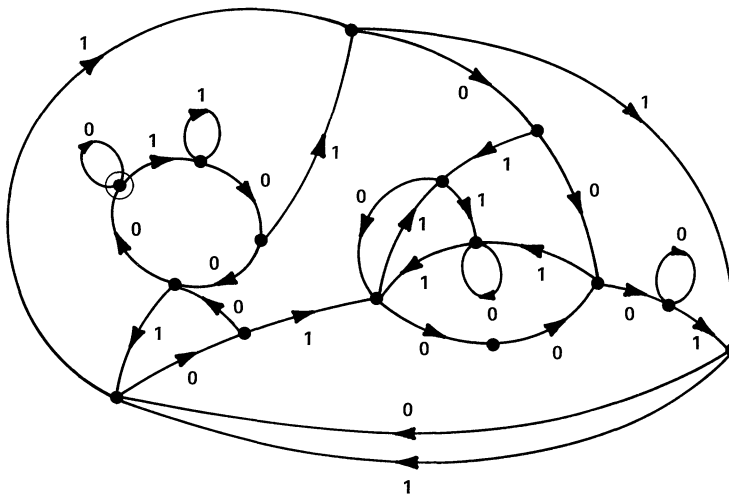


Fig. 4.4. Minimal deterministic finite automaton (DFA) corresponding to the regular language $\Omega^{(1)}$ obtained after one time step in the evolution of the class 3 cellular automaton rule 22. The DFA has all 15 possible states. The shortest excluded block in $\Omega^{(1)}$ has length 8, and corresponds to the shortest path from the encircled start state to the one "incomplete" node in the DFA graph

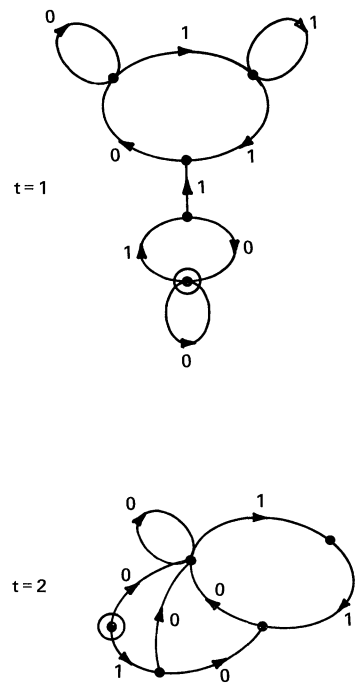


Fig. 4.5. Minimal deterministic finite automata corresponding to the regular languages $\Omega^{(t)}$ generated in the first two time steps of evolution under rule 72. $\Omega^{(1)}$ is an infinite complement regular language, with the infinite sequence of distinct blocks 111, 1101011, 11001011, ... excluded. $\Omega^{(2)}$ is a finite complement language, with only the blocks 010 and 111 excluded

blocks are 10101001 and 10010101 (length 8). After two time steps, the blocks 1110101 and 1010111 (length 7) are also excluded. The shortest newly-excluded blocks after three time steps are 01000010101, 01000110101, 10000010101 and their reversals (length 11). After four time steps, the shortest newly-excluded blocks are 010110011 and 110011010 (length 9), realizing the equality in (3.13).

Rule 126 provides an example in which the set generated after one time step is a finite complement language, but the sets generated at subsequent times are not. Rule 72 exhibits the opposite behaviour[11], as shown in Fig. 4.5. After one time step, it yields a set in which the infinite sequence of distinct blocks 111, 1101011, 11001011, ... are excluded (as in $\Omega^{(1)}$ for rule 18). After two times steps, however, the block 010 is also excluded. The exclusion of this single block implies exclusion of the infinite set of blocks excluded from $\Omega^{(1)}$. The resulting set thus corresponds to a finite complement language. In general, it can be shown that if a cellular automaton evolves to a finite complement language limit set, then it must do so in a finite number of time steps [34].

The sets $\Omega^{(t)}$ generated by most cellular automata never appear to become simpler with time. One exception is rule 72, in which the number of arcs in the minimal DFA for $\Omega^{(2)}$ is less than in that for $\Omega^{(1)}$. In most cases, the regular language complexity $\Xi^{(t)}$ appears to be non-decreasing with time. In fact, whenever the set of configurations generated continues to contract with time, a different regular language must be obtained at each time step. Since there are a limited number of regular languages with complexities below any given value (certainly less than $2^{k\Xi^2}$), the complexity must on average increase at least slowly with time in this case.

Table 1 suggests that a definite set of cellular automata (including rules 18, 22 and 126) yield regular language complexities $\Xi^{(t)}$ that grow on average more rapidly than any polynomial in time (perhaps exponentially with time). Many of the cellular automata in this set generically exhibit class 3, chaotic, behaviour, suggesting that rapidly-increasing $\Xi^{(t)}$ are a signal for class 3 behaviour in cellular automata.

In a few cases, such as rule 94, $\Xi^{(t)}$ increases rapidly with time, but almost all initial configurations are found to give ultimately periodic behaviour. Nevertheless, special initial conditions (in this case, those in which successive pairs of sites have equal values) can yield chaotic behaviour. Since the set $\Omega^{(t)}$ includes all configurations that ever occur, it includes those that give chaotic behaviour, even though they occur with vanishingly small probability. Presumably these configurations would not affect a probabilistic grammar for the set $\Omega^{(t)}$ that included only nonzero probability configurations. But the $\Xi^{(t)}$ for the grammars discussed here appear to increase rapidly with time whenever any set of configurations in the cellular automaton yield class 3 behaviour.

Some exceptional cases are surjective class 3 cellular automata, such as the additive rules 90 and 150, in which every possible configuration can be generated at any time. The complexity of these and other cellular automata could perhaps be measured by constructing a grammar for the set of possible space-time patterns generated in their evolution. Such a grammar could presumably be characterized

11 A more complicated example of this behaviour was given in [33]

in terms of computers with memories arranged in a two-dimensional lattice (cf. [35])[12].

The local rules ϕ for the 32 legal $k=2, r=1$ cellular automata of Tables 1–3 are all distinct. Yet in many cases sets of configurations with the same structure or properties are found to be generated. In some cases, there may exist bijective mappings which transform configurations evolving according to one cellular automaton rule into configurations evolving according to another rule. Several properties of the sets $\Omega^{(t)}$ are invariant under such mappings. One example is the set of non-zero roots of the characteristic polynomials $\chi^{(t)}(\lambda)$. While after one time step several of the cellular automata in Tables 1–3 yield the same sets of configurations $\Omega^{(1)}$, there are few examples of complete equivalence between pairs of cellular automaton rules. One simple example is rules 146 and 182, which are related by interchange of the roles of 0 and 1.

5. Some Invariant Sets

Section 2 showed that the set of configurations generated after a finite number of steps in the evolution of any cellular automaton forms a regular language. Sections 3 and 4 discussed some properties of such sets. This section and the next one consider the limiting sets of configurations generated after many time steps of cellular automaton evolution.

For all configurations A that appear in the limit set for a cellular automaton, there must exist some configuration A' such that $A=\Phi^t A'$ for any t. Any set of configurations invariant under the cellular automaton rule therefore appear in its limit set. This section considers some simple examples of invariant sets; Sect. 6 gives some comments on the complete structure of limit sets for cellular automata.

Periodic Sets

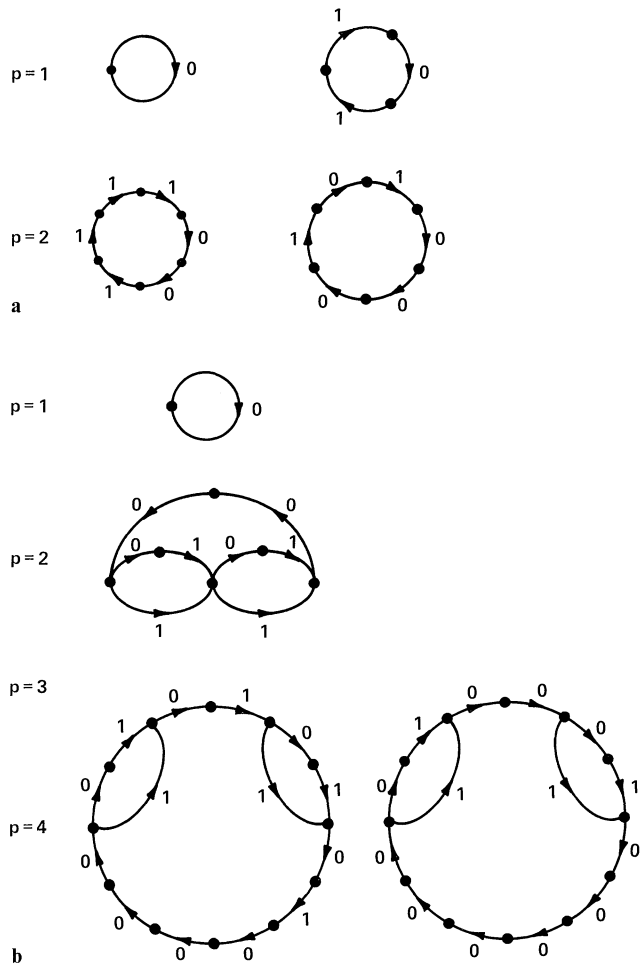
A simple class of invariant sets consist of configurations periodic with time under cellular automaton evolution. Such sets are found to form finite complement languages.

Consider the set of configurations that are stable (have temporal period 1) under a cellular automaton rule with $k=2$ and $r=1$. The set of such configurations is exactly those which contain only neighbourhoods $\{a_{i-1}, a_i, a_{i+1}\}$ for which

$$\phi[a_{i-1}, a_i, a_{i+1}]=a_i\,. \tag{5.1}$$

Only the finite set of distinct three-site blocks that violate (5.1) are forbidden, so that the complete set forms a finite-complement language, with a maximum distinct excluded block of length 3. A NDFA that generates the set of stable configurations is represented by a graph analogous to Fig. 3.1 in which only those

12 This paper concentrates on one-dimensional cellular automata. Such cellular automata potentially correspond most directly with conventional formal languages. Two and higher dimensional cellular automata show some differences. For example the set of configurations obtained after a finite number of time steps in their evolution need not form a regular language and may in fact be nonrecursive [36, 51]



arcs satisfying (5.1) are retained. The minimal grammar for this set is obtained by constructing the minimal equivalent DFA, as described in Sect. 2.

The procedure generalizes immediately to arbitrary cellular automaton rules, and to sets of configurations with any finite period (cf. [37]). The distinct excluded blocks in the finite complement languages corresponding to sets of configurations with period p have maximum length $2pr+1$.

Figure 5.1 shows the minimal grammars for sets of configurations with various periods under the $k=2$, $r=1$ cellular automata with rule numbers 90, 18 and 22. The grammars are represented by graphs containing several disconnected pieces, each corresponding to a disjoint set of configurations.

Figure 5.1a suggests that only a finite number of configurations, all spatially periodic, are found with each temporal period in the surjective cellular automaton rule 90. For this and other surjective cellular automata whose local mappings ϕ are injective in their first and last arguments, the number of distinct configurations

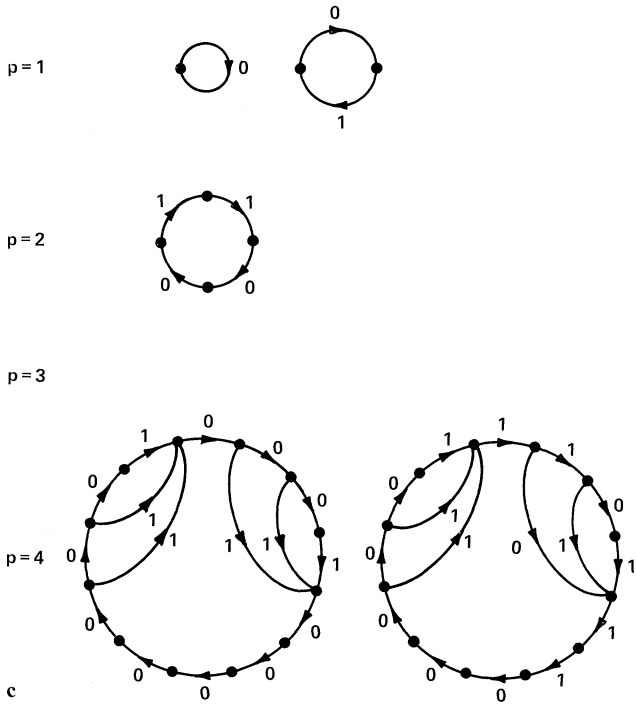


Fig. 5.1a–c. Minimal deterministic finite automata corresponding to sets of configurations with (temporal) periods exactly p under cellular automaton rules **a** 90, **b** 18 and **c** 22

with any period p is always finite, and is exactly $k^{\mathbf{h}p}$, where **h** is the invariant entropy of the cellular automaton mapping (**h** = 2 for rule 90)[13]. This result follows from the fact that the complete space-time pattern generated by the evolution of such a cellular automaton is completely determined by any patch of site values with infinite temporal extent, but spatial width **h** (typically equal to $2r$). Moreover, any possible set of site values may occur in this patch. If the complete space-time pattern is to have period p, then so must the patch; but there are exactly $k^{\mathbf{h}p}$ possible patches with period p. (For large p, this result is as expected for any expansive homeomorphism (e.g. [10, 11]).)

In general, the sets of configurations with a particular periodicity under a cellular automaton rule are infinite, as illustrated for rules 18 and 22 in Figs. 5.1b and 5.1c. Presumably there are sets of this kind with arbitrarily large periods. These infinite sets are nevertheless finite complement languages. For example, for the set of configurations with period two under rule 18, only the distinct blocks 111, 1011, 1101 and 10101 are excluded. It is common in class 3 cellular automata to find configurations with almost every possible period; for class 4 cellular automata, only some periods are typically found.

13 The actual configurations with particular periods may be found by methods analogous to those used in [29] for the complementary problem of determining the temporal periods of configurations with given spatial period

Periodic configurations form a small subset of all the configurations in the limit sets for cellular automata. Their entropy nevertheless provides a lower bound on the entropy of the complete limit sets. For rule 90, the set of periodic configurations has zero entropy, yet the complete limit contains all possible configurations, and thus has entropy 1. For rule 18, the period 2 set has entropy $\simeq 0.4057$ (given as the logarithm of the largest root of $\lambda^3-\lambda-1$), while the period 4 set has entropy $\simeq 0.1824$ ($\lambda^6-\lambda-1$). For rule number 22, the period 4 set has entropy $\simeq 0.3219$ ($\lambda^5-\lambda^4+\lambda^3-\lambda^2-1$). Since irreversible cellular automaton mappings are contractive, the entropy of the set obtained after a finite number of time steps gives an upper bound on the entropy of the complete limit set. Using results from Table 3 one then finds

$$\begin{aligned} 0.4057 \lesssim s^{(\infty)}_{[18]} \lesssim 0.8114\,, \\ 0.1824 \lesssim s^{(\infty)}_{[22]} \lesssim 0.9390\,. \end{aligned} \tag{5.2}$$

Simulation Sets

The complete invariant sets for many cellular automata Φ are very complicated. Parts of these invariant sets may however have a simpler structure, and may consist of configurations for which Φ "simulates" a simpler cellular automaton rule. Thus for example stable configurations under Φ may be considered as those for which Φ "simulates" the identity mapping.

One class of configurations for which a cellular automaton rule Φ_1 may simulate a rule Φ_2 are those obtained by "blocking transformations." Each symbol in the possible configurations of Φ_2 is replaced by a length b_X block of symbols in Φ_1, and each time step in the evolution of Φ_2 is simulated by b_T time steps of evolution under Φ_1. Thus, for example, rule 18 simulates rule 90 under the ($b_X=2, b_T=2$) blocking transformation 00→0, 01→1 [1, 38, 5]. The evolution of an arbitrary configuration under rule 90 is thus simulated by the evolution under rule 18 of a configuration consisting of the digrams 00 and 01. But since rule 90 is surjective, all possible configurations correspond to an invariant set. Thus configurations containing only 00 and 01 digrams form an invariant set for rule 18. The entropy of these configurations is 1/2, so that

$$0.5 \leqq s^{(\infty)}_{[18]} \lesssim 0.8114\,. \tag{5.3a}$$

Rule 22 simulates rule 90 under the (4,4) blocking transformation 0000→0, 0001 →1, implying that

$$0.25 \leqq s^{(\infty)}_{[22]} \lesssim 0.9390\,. \tag{5.3b}$$

A cellular automaton rule may simulate other rules with the same values of k and r under different blocking transformations (cf. the simulation network given in [5]). Some rules, apparently only surjective ones such as rule 90, simulate themselves, and thus correspond to fixed points of the blocking transformation. In other cases, one rule may simulate another under several distinct blocking transformations. For example, rule 18 simulates rule 90 under both 00→0, 01→1, and 00→0, 10→1, while rule 22 simulates rule 90 under any permutation of 0000 →0, 0001→1. One may consider the sets of blocks appearing in these blocking

transformations to represent different "phases." An initial configuration then consists of several "domains," each of which contains blocks of one phase. The domains are separated by "walls." For rule 18, these walls appear to execute random walks, and annihilate in pairs, yielding progressively larger domains of a single phase [38]. The simulation of rule 90 by rule 18 may thus be considered "attractive" [3]. For rule 22, no such simple behaviour is observed.

Blocking transformations yield a particular class of configurations, corresponding to simple finite complement languages. Other classes of configurations, specified by more general grammars, may also yield simulations. (An example occurs for rule number 73, in which configurations containing only odd-length sequences of 0 and 1 sites simulate rule 90.) In addition, a set of configurations evolving under one rule may simulate an invariant set of configurations evolving under another rule.

6. Comments on Limiting Behaviour

Section 2 showed that after any finite number of time steps, the set of configurations $\Omega^{(t)}$ generated by any cellular automaton forms a regular language. Some cellular automata yield regular languages even in the infinite time limit; others appear to generate limit sets corresponding to more complicated formal languages. Cellular automata which exhibit different classes of overall behaviour appear to yield characteristically different limiting languages.

As discussed in Sect. 4, some cellular automata in Tables 1–3 yield regular languages which attain a fixed form after a few time steps. The limit sets for such cellular automata are thus regular languages. In fact, except for surjective rules, the limit sets found appear to contain only temporally periodic configurations, and are therefore finite complement languages. These cellular automata exhibit simple large time behaviour, characteristic of classes 1 and 2.

Rule 128 provides a more complicated example, discussed in Sect. 4. After t time steps, any pair of ones in configurations generated by this rule must be separated by at least $2t$ sites. The complete set of possible configurations forms a finite complement regular language, with a minimal DFA illustrated in Fig. 4.1 whose size $\Xi^{(t)}$ increases linearly with time. After many time steps, almost all initial configurations evolve to the null configuration. However, even after an arbitrarily long time, configurations containing just a single block of ones may still appear. A block of n ones, flanked by infinite sequences of zeroes, is generated after any number of time steps t from a block of $n+2t$ ones. Such configurations therefore have exactly one predecessor under any number of time steps of the cellular automaton evolution. They thus appear in the limit set for rule 128, although if all initial configurations are given equal weight, they are generated with zero probability. Once generated, their evolution is never periodic. An increasing number of distinct blocks are excluded from the successive $\Omega^{(t)}$ obtained by evolution under rule 128. The set of configurations generated in the infinite time limit does not, therefore, correspond to a finite complement language. Nevertheless, the set does form a regular language, shown in Fig. 6.1. While the set contains an infinite number of configurations, its entropy vanishes, as given by the limit of Eq. (4.1).

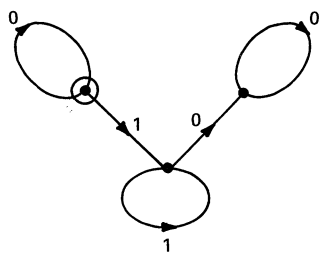


Fig. 6.1. The deterministic finite automaton representing the regular language corresponding to the limit set for cellular automaton rule 128. This infinite complement regular language is obtained as the infinite time limit of the series of finite complement regular languages illustrated in Fig. 4.1. It contains an infinite number of configurations, but has zero limiting entropy

Several rules given in Tables 1–3 exhibit behaviour similar to rule 128: they generate (finite complement) regular languages whose minimal grammars increase in size linearly or quadratically with time, but in the infinite time limit, yield regular language limit sets. These limit sets contain one or a few periodic configurations, together with an infinite number of aperiodic configurations, generated from a set of initial configurations of measure zero. The sets have zero entropy, and do not correspond to finite complement languages. (Only trivial finite complement languages can have zero entropy.) All the class 1 cellular automata (except for the trivial rule 0) in Tables 1–3 exhibit such limiting behaviour. The generation of limit sets corresponding to regular languages that are not finite complement languages appears to be a general feature of class 1 cellular automata.

Tables 1–3 suggest the result, discussed in Sect. 4, that cellular automata capable of class 3 or 4 behaviour give rise to sets of configurations represented by regular languages whose complexity increases rapidly with time. The limit sets for such cellular automata are therefore presumably not usually regular languages. If a finite description of them can be given, it must be in terms of more complicated formal languages.

Any language that can be described by a regular grammar must obey the regular language "pumping lemma" (e.g. [7]). This requires that it be possible to write all sufficiently long symbol sequences α appearing in the language in the form $\alpha_1\alpha_2\alpha_3$ so that for any n the symbol sequence $\alpha_1\alpha_2^n\alpha_3$ also appears in the language. (This result follows from the fact that any sufficiently long sequence must correspond to a path containing a cycle in the DFA. This cycles may then be traversed any number of times, yielding arbitrarily repeated symbol sequences.) The sets generated after a finite number of time steps in cellular automaton evolution always obey this condition: arbitrary repetitions of the string α_2 are obtained by evolution from initial configurations containing arbitrarily-repeated sequences evolving to α_2.

It is possible to construct cellular automata for which the regular language pumping lemma fails in the large time limit, and which therefore yield non-regular language limit sets. In one class of examples [39, 34], there are pairs of localized structures which propagate with opposite velocities from point sources. After t time steps, such cellular automata generate configurations consisting roughly of

repetitions of sequences

$$(10^j20^j1) \quad j \leqq t . \tag{6.1}$$

In the infinite time limit, arbitrarily long identical pairs of symbol sequences thus appear. The limit sets for such cellular automata are therefore not regular languages. Instead it appears that they correspond to context-free languages.

The pumping lemma for regular languages may be generalized to context-free languages. Any sufficiently long sequence in a context-free language must be of the form $\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5$ such that $\alpha_1\alpha_2^n\alpha_3\alpha_4^n\alpha_5$ is also in the language for any n. The possibility for separated equal length identical substrings is a reflection of the non-local nature of context-free languages, manifest for example in the indefinitely large memory stacks required in machines to recognize them.

Limiting sets of configurations of the form (6.1) that violate the regular language pumping lemma nevertheless obey the context-free language pumping lemma, and thus correspond to context-free languages.

The correspondence between sets of infinite cellular automaton configurations and context-free languages is slightly more complicated than for regular languages. In all cases, the cellular automaton configurations correspond to infinite symbol sequences generated according to a formal grammar. For regular languages, it it also possible to construct finite automata which recognize words in the language, starting at any point. The necessity for a stack memory in the generation of context-free languages makes their recognition starting at any point in general impossible. Infinite configurations generated by context-free grammars must thus be viewed as concatenations of finite context-free language words. Only at the boundaries between these words is the stack memory for the machine generating the configuration empty, so that sequences of symbols may be recognized. Configurations generated by context-sensitive and more complicated grammars must be considered in an analogous way.

If the limit set for a cellular automaton is a context-free language, whose generation requires a computer with an indefinitely large stack memory, then one expects that the regular language sets obtained at successive finite time steps in the evolution of the cellular automaton would require progressively larger finite size stack memories. If the limiting context-free grammar contains say Q (nonterminal) productions, then there are $O(Q^t)$ possible stack configurations after t time steps, and the set of configurations obtained may be recognized by a finite automaton with about Q^t states. In addition, the context-free pumping lemma is satisfied for repetitions of substrings of length up to about t. Regular languages that approximate context-free languages for t time steps should have comparatively simple repetitive forms. The regular languages of Fig. 4.1 generated at finite times by rule 128 have roughly the expected form, but their limit is in fact a regular language. The absence of obvious patterns in the regular grammars such as Figs. 4.2–4.4 generated by typical class 3 cellular automata after even a few time steps suggests that the limiting languages in these cases are not context-free. They are presumably context-sensitive or unrestricted languages.

The entropies of regular languages are always logarithms of algebraic integers [as in Eq. (3.5)]. Context-free languages may, however, have entropies given by logarithms of general algebraic numbers (whose minimal polynomials are not

necessarily monic). The enumeration of words in a formal language may be cast in algebraic terms by considering the sequence of words in the language as a formal power series satisfying equations corresponding to the production rules for the language (e.g. [40]). For the simple regular language ((0*)10) (repetition understood) of Fig. 1.3b, with production rules

$$u_0 \to s_0 u_0 , \quad u_0 \to s_1 u_1 , \quad u_1 \to s_0 u_0 , \tag{6.2}$$

(where the terminal symbols s_0 and s_1 represent 0 and 1 respectively), the corresponding equations are

$$u_0 = s_0 u_0 + s_1 u_1 + 1 , \quad u_1 = s_0 u_0 + 1 . \tag{6.3}$$

Solving for u_0 as the start symbol one obtains

$$u_0 = (s_1 + 1)/(1 - s_0 - s_1 s_0) . \tag{6.4}$$

The expansion of this generating function (accounting for the non-commutative nature of symbol string concatenation) yields the sequence of possible words in the language. Replacing all terminal symbols in the generating function by a dummy variable x, the coefficient of x^n in its expansion gives the number of distinct symbol sequences of length n in the language. The asymptotic growth rate of this number, and thus the entropy of the language, are then determined by the smallest real root of the (monic) denominator polynomial. The generating function for any regular language is always a rational function of x. For a context-free language, however, the equations analogous to (3.8) are in general non-linear in the u_i. At least for unambiguous languages, the positions of the leading poles in the resulting generating functions obtained by solving these simultaneous polynomial equations are nevertheless algebraic numbers [41].

There is a finite procedure to find the minimal regular grammar that generates a given regular language, as described in Sect. 2. No finite procedure exists in general, however, to find the minimal context-free or other grammar corresponding to a more complicated language. The analogue of the regular language complexity is thus formally non-computable for context-free and more complicated languages. This is an example of the result that no finite procedure can in general determine the shortest input program that generates a particular output when run for an arbitrarily long time on some computer (e.g. [42]). Explicit testing of successively longer programs is inadequate, since the insolubility of the halting problem implies that no upper bound can in general be given for the time before the required sequence is generated. Particular simple cases of this problem are nevertheless soluble, so that, for example, the minimal grammars for regular languages are computable.

The entropies for regular and context-free languages may be computed by the finite procedures described above. The entropies for context-sensitive (type 1) and unrestricted (type 0) languages are, however, in general non-computable numbers [43]. Bounds on them may be given. But no finite procedure exists to calculate them to arbitrary precision. (They are in many respects analogous to the non-computable probabilities for universal computers to halt given random input.) If

many class 3 and 4 cellular automata do indeed yield limit sets corresponding to context-sensitive or unrestricted languages, then the entropies of these sets are in general non-computable.

The discussion so far has concerned the generation of infinite configurations by cellular automaton evolution. One may also consider the evolution of configurations in which nonzero sites exist only in a finite region. Then for class 3 cellular automata with almost all initial states, the region of nonzero sites expands linearly with time. (Such expansion is guaranteed if, for example, $\phi[1, 0, \ldots, 0] = 1$ and so on.) For class 4 cellular automata, the region may expand and contract with time. One may characterize the structures generated by considering the set of finite sequences generated at any time by evolution from a set of finite initial configurations. For class 3 cellular automata, this set appears to be no more complicated than a context sensitive language, while for class 4 cellular automata, it may be an unrestricted language. Notice that the set generated after a fixed finite number of time steps always corresponds to a regular language, just as for infinite configurations. (The regular grammar for these finite configurations consists of all paths with the relevant length that begin and end at the 00...0 node of the NDFA analogous to Fig. 2.1.)

Consider the language formed by the set of sequences of length n generated after any number of time steps in the evolution of a class 3 cellular automaton from all possible initial configurations with size n_0[14]. This language appears to be at most context-sensitive, since a word of length n in it can presumably be recognized in a finite time by a computer with a memory of size at most n. In its simplest form, the computer operates by testing configurations generated by evolution from all k^{n_0} possible initial states. Since the configurations expand steadily with time, the evolution of each configuration need be traced only until it is of size n; the required configuration of length n is either reached at that time, or will never be reached.

In a class 4 cellular automaton, evolution from an initial configuration of size n_0 may yield arbitrarily large configurations, but then ultimately contract to give a size n configuration. No upper bound on the time or memory space required to generate the size n configuration may therefore be given. The problem of determining whether a particular finite configuration is ever generated in the evolution of a class 4 cellular automaton from one of a finite set of initial configurations may therefore in general be formally undecidable. No finite computation can give all the structures of a particular size ultimately generated in the evolution of a class 4 cellular automaton.

The procedure for recognizing finite configurations generated by class 3 cellular automata, while finite in principle, may require large computational resources. Whenever the context-sensitive language corresponding to the set of finite configurations cannot be described by a context-free or simpler grammar, the problem of recognizing words in the language is PSPACE-complete with respect to the lengths of the words (e.g. [28]). It can thus presumably be performed

14 This is analogous to but distinct from the problem of finding all initial configurations which ultimately evolve to a particular complete final configuration, such as the null configuration (cf. [2, 44, 45])

essentially no more efficiently than by testing the structures generated by the evolution of each of the k^{n_0} possible finite initial configurations.

As well as considering the evolution of finite complete configurations, one may also consider the generation of finite sequences of symbols in the evolution of infinite configurations. Enumeration of sets of length n sequences that can and cannot occur provide partial characterizations of sets of infinite configurations. However, even for configurations generated at a finite time t, such enumeration in general requires large computational resources. A symbol sequence of length n appears only if at least one length $n_0 = n + 2rt$ initial block evolves to it after t time steps. A computation time polynomial in n and t suffices to determine whether a particular candidate initial block evolves to a required sequence. The problem of determining whether any such initial block exists is therefore in the class NP. One may expect that for many cellular automata, this problem is in fact NP-complete. (The procedure of Sect. 2 provides no short cut, since the construction of the required DFA is an exponential computational process.) It may therefore effectively be solved essentially only by explicit simulation of the evolution of all exponentially-many possible initial sequences.

In the limit of infinite time, the problem of determining whether a particular finite sequence is generated in the evolution of a cellular automata becomes in general undecidable. For a cellular automaton with only class 1 or 2 behaviour, the limit set always appears to correspond to a regular language, for which the problem is decidable. But for class 3 and 4 cellular automata, whose limit sets presumably correspond to more complicated formal languages, the problem may be undeciable. (The problem is in general in the undecidability class Π_1 [46]; the set of finite sequences that occur is thus recursively enumerable, but not necessarily recursive.) Even when the general problem is undecidable, the appearance of particular finite sequences in the limit set for a cellular automaton may be decidable. The fraction of particular sequences whose appearance in the limit set is undecidable provides a measure of the degree of unpredictability or "computational achievement" of the cellular automaton evolution (presumably related to "logical depth" [47]).

7. Discussion

This paper has taken some preliminary steps in the application of computation theory to the global analysis of cellular automata. Cellular automata are viewed as computers, whose time evolution processes the information specified by their initial configurations. Many aspects of this information processing may be described in terms of computation theory. The intrinsic discreteness of cellular automata allows for immediate identifications with conventional computational systems; but the basic approach and many of the results obtained should be applicable to many other dynamical systems.

Self-organization in cellular automata involves the generation of distinguished sets of configurations with time. These sets are described as formal languages in computation theory terms. Each configuration corresponds to a word in a language, and is formed from a sequence of symbols according to definite

grammatical rules. These grammatical rules provide a complete and succinct specification of the sets generated by the cellular automaton evolution.

Section 2 showed that, starting with all possible initial configurations, the sets generated by a finite number of time steps of cellular automaton evolution always correspond to regular formal languages. Such languages are recognized by finite automata. These finite automata are specified by finite state transition graphs; words in the languages correspond to all possible paths through these graphs. The (limiting) set entropies of such regular languages are then given as logarithms of the algebraic integers corresponding to the largest eigenvalues of the incidence matrices for their state transition graphs.

In genral, several different finite automata or regular grammars may yield the same regular language. However, it is always possible to find a simplest finite automaton, or set of grammatical rules, which correspond to any particular regular language. This simplest finite automaton provides a canonical representation for sets generated by cellular automaton evolution, and its size (number of states) gives a measure of their "complexity." The larger the "regular language complexity" for a set of configurations, the more complicated is the minimal set of grammatical rules necessary to describe it as a regular language.

Section 4 suggests the general result that the regular language complexity is non-decreasing with time for all cellular automata. This result gives a quantitative characterization of progressive self-organization in cellular automata. It may give a first indication of a generalization of the second law of thermodynamics to irreversible systems.

Entropy may be estimated from experimental data by fitting parameters in simple models which reproduce the data. Extraction of regular language complexities from experimental data requires the identification of maximal (regular language) patterns in the data, or the construction of a minimal (finite automaton) model that generates the data. Given perfect data (and an upper bound on the regular language complexity), a direct method may be used (e.g. [48]). In practice, it will probably be convenient to construct stochastic finite automata which provide probabilistic reproductions of the available data (cf. estimates for the structure of Markovian sources (e.g. [49])).

Dynamical systems theory methods were used in [2] to identify four general classes of cellular automaton behaviour. Sections 4 and 6 suggested computation theory characterizations of these classes. The limit sets for cellular automata with only class 1 or 2 behaviour are regular languages. For most class 3 and 4 cellular automata, the regular language complexity increases steadily with time, so that the set of configurations obtained in the large time limit does not usually form a regular language. Instead (at least for appropriate finite size configurations) the limit sets for class 3 cellular automata appear to correspond to context-sensitive languages, while those for class 4 cellular automata correspond to general languages.

Regular languages are sufficiently simple that their properties may be determined by finite computational procedures. Properties of context-free and more complicated languages are, however, often not computable by finite means. Thus, for example, the minimal grammars for such languages (whose sizes would

provide analogues of the regular language complexity) cannot in general be found by finite computations. Moreover, for context-sensitive and general languages, even quantities such as entropy are formally non-computable.

When cellular automaton evolution is viewed as computation, one may consider that the limiting properties of a cellular automaton are determined by an infinite computational process. One should not expect in general that the results of this infinite process can be summarized in finite mathematical terms. For sufficiently simple cellular automata, apparently those of classes 1 and 2, however, it is nevertheless possible to "short cut" the infinite processes of cellular automaton evolution, and to give a finite specification of their limiting properties. For most class 3 and 4 cellular automata, no such short cut appears possible: their behaviour may in general be determined by no procedure significantly faster than explicit simulation, and many of their limiting properties cannot be determined by any finite computational process. (Such non-computable limiting behaviour would be an immediate consequence of the universal computation capability conjectured for class 4 cellular automata, but does not depend on it.)

Non-computability and undecidability are common phenomena in the systems investigated in pure mathematics, logic and computation. But they have not been identified in the systems considered in theoretical physics. In many physical theories one can in fact imagine constructing complicated systems which behave, for example, as universal computers, and for which undecidable propositions may be formulated. Cellular automata (and other dynamical systems) may be considered as simple physical theories. This paper has suggested that in fact even simple, natural, questions concerning the limiting behaviour of cellular automata are often undecidable (except for very simple systems such as those corresponding to class 1 and 2 cellular automata). One may speculate that undecidability is common in all but the most trivial physical theories. Even simply-formulated problems in theoretical physics may be found to be provably insoluble.

Undecidability and non-computability are features of problems which attempt to summarize the consequences of infinite processes. Finite processes may always be carried out explicitly. For some particularly simple processes, the consequences of a large, but finite, number of steps may be deduced by a procedure involving only a small number of steps. But at least for many computational processes (e.g. [28]), it is believed that no such short cut exists: each step (or each possibility) must in fact be carried out explicitly. It was suggested that this phenomenon is common in cellular automata. One may speculate that it is widespread in physical systems. No simple theory or formula could ever be given for the overall behaviour of such systems: the consequences of their evolution could not be predicted, but could effectively be found only by direct simulation or observation.

Acknowledgements. I am grateful to A. Aho, C. Bennett, J. Conway, D. Hillis, L. Hurd, D. Lind, O. Martin, M. Mendes France, J. Milnor, A. Odlyzko, N. Packard, J. Reeds, and many others for discussions. A preliminary version of this paper was presented at a workshop on "Coding and Isomorphisms in Ergodic Theory," held at the Mathematical Sciences Research Institute, Berkeley (December 8–13, 1983). I thank M. Boyle, E. Coven, J. Franks, and many of the other participants for their comments. Some of the results given above were obtained using the computer mathematics system SMP [50].

References

1. Wolfram, S.: Statistical mechanics of cellular automata. Rev. Mod. Phys. **55**, 601 (1983)
2. Wolfram, S.: Universality and complexity in cellular automata. Physica **10**D, 1 (1984)
3. Packard, N.H.: Complexity of growing patterns in cellular automata, Institute for Advanced Study preprint (October 1983), and to be published in Dynamical behaviour of automata. Demongeot, J., Goles, E., Tchuente, M., (eds.). Academic Press (proceedings of a workshop held in Marseilles, September 1983)
4. Wolfram, S.: Cellular automata as models for complexity. Nature (to be published)
5. Wofram, S.: Cellular automata. Los Alamos Science, Fall 1983 issue
6. Beckman, F.S.: Mathematical foundations of programming. Reading, MA: Addison-Wesley 1980
7. Hopcroft, J.E., Ullman, J.D.: Introduction to automata theory, languages, and computation. Reading, MA: Addison-Wesley 1979
8. Minsky, M.: Computation: finite and infinite machines. Englewood Cliffs, NJ: Prentice-Hall 1967
9. Rozenberg, G., Salomaa, A. (eds.): L systems. In: Lecture Notes in Computer Science, Vol. 15 Rozenberg, G., Salomaa, A.: The mathematical theory of L systems. New York: Academic Press 1980
10. Guckenheimer, J., Holmes, P.: Nonlinear oscillations, dynamical systems, and bifurcations of vector fields. Berlin, Heidelberg, New York: Springer 1983
11. Walters, P.: An introduction to ergodic theory. Berlin, Heidelberg, New York: Springer 1982
12. Weiss, B.: Subshifts of finite type and sofic systems. Monat. Math. **17**, 462 (1973); Coven, E.M., Paul, M.E.: Sofic systems. Israel J. Math. **20** 165 (1975)
13. Field, R.D., Wolfram, S.: A QCD model for e^+e^- annihilation. Nucl. Phys. B **213**, 65 (1983)
14. Smith, A.R.: Simple computation-universal cellular spaces. J. ACM **18**, 331 (1971)
15. Berlekamp, E.R., Conway, J.H., Guy, R.K.: Winning ways for your mathematical plays. New York: Academic Press, Vol. 2, Chap. 25
16. Lind, D.: Applications of ergodic theory and sofic systems to cellular automata. Physica **10**D, 36 (1984)
17. de Bruijn, N.G.: A combinatorial problem. Ned. Akad. Weten. Proc. **49**, 758 (1946); Good, I.J.; Normal recurring decimals. J. Lond. Math. Soc. **21**, 167 (1946)
18. Nerode, A.: Linear automaton transformations. Proc. Am. Math. Soc. **9**, 541 (1958)
19. Cvetkovic, D., Doob, M., Sachs, H.: Spectra of graphs. New York: Academic Press 1980
20. Billingsley, P.: Ergodic theory and information. New York: Wiley 1965
21. Chomsky, N., Miller, G.A.: Finite state languages. Inform. Control **1**, 91 (1958)
22. Stewart, I.N., Tall, D.O.: Algebraic number theory. London: Chapman & Hall 1979
23. Lind, D.A.: The entropies of topological Markow shifts and a related class of algebraic integers. Ergodic Theory and Dynamical Systems (to be published)
24. Milnor, J.: Unpublished notes (cited in [2])
25. Hedlund, G.A.: Endomorphisms and automorphisms of the shift dynamical system. Math. Syst. Theor. **3**, 320 (1969);
    Hedlund, G.A.: Transformations commuting with the shift. In: Topological dynamics. Auslander, J., Gottschalk, W.H. (eds.). New York: Benjamin 1968
26. Amoroso, S., Patt, Y.N.: Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures. J. Comp. Syst. Sci. **6**, 448 (1972)
27. Nasu, M.: Local maps inducing surjective global maps of one-dimensional tessellation automata. Math. Syst. Theor. **11**, 327 (1978)
28. Garey, M.R., Johnson, D.S.: Computers and intractability: a guide to the theory of NP-completeness. San Francisco: Freeman 1979, Sect. A10
29. Martin, O., Odlyzko, A.M., Wolfram, S.: Algebraic properties of cellular automata. Commun. Math. Phys. **93**, 219 (1984)
30. Hedlund, G.: Private communication

31. Manning, A.: Axiom A diffeomorphisms have rational zeta functions. Bull. Lond. Math. Soc. **3**, 215 (1971);
Coven, E., Paul, M.: Finite procedures for sofic systems. Monat. Math. **83**, 265 (1977)
32. Franks, J.: Private communication
33. Coven, E.: Private communication
34. Hurd, L.: Formal language characterizations of cellular automata limit sets (to be published)
35. Rosenfeld, A.: Picture languages. New York: Academic Press (1979)
36. Golze, U.: Differences between 1- and 2-dimensional cell spaces. In: Automata, Languages and Development, Lindenmayer, A., Rozenberg, G. (eds.). Amsterdam: North-Holland 1976
Yaku, T.: The constructibility of a configuration in a cellular automaton. J. Comput. System Sci. **7**, 481 (1983)
37. Grassberger, P.: Private communication
38. Grassberger, P.: A new mechanism for deterministic diffusion. Phys. Rev. A (to be published)
Chaos and diffusion in deterministic cellular automata. Physica **10** D, 52 (1984)
39. Hillis, D., Hurd, L.: Private communications
40. Salomaa, A., Soittola, M.: Automata-theoretic aspects of formal power series. Berlin, Heidelberg, New York: Springer 1978
41. Kuich, W.: On the entropy of context-free languages. Inform. Cont. **16**, 173 (1970)
42. Chaitin, G.: Algorithmic information theory. IBM J. Res. Dev. **21**, 350 (1977)
43. Kaminger, F.P.: The non-computability of the channel capacity of context-sensitive languages. Inform. Cont. **17**, 175 (1970)
44. Smith, A.R.: Real-time language recognition by one-dimensional cellular automata. J. Comput. Syst. Sci. **6**, 233 (1972)
45. Sommerhalder, R., van Westrhenen, S.C.: Parallel language recognition in constant time by cellular automata. Acta Inform. **19**, 397 (1983)
46. Rogers, H.: Theory of recursive functions and effective computability. New York: McGraw-Hill 1967
47. Bennett, C.H.: On the logical "depth" of sequences and their reducibilities to random sequences. Inform. Control (to be published)
48. Conway, J.H.: Regular algebra and finite machines. London: Chapman & Hall 1971
49. Shannon, C.E.: Prediction and entropy of printed English. Bell Syst. Tech. J. **30**, 50 (1951)
50. Wolfram, S.: SMP reference manual. Computer Mathematics Group. Los Angeles: Inference Corporation 1983
51. Packard, N.H., Wolfram, S.: Two dimensional cellular automata. Institute for Advanced Study preprint, May 1984
52. Hopcroft, H.: An $n \log n$ algorithm for minimizing states in a finite automaton. In: Proceedings of the International Symposium on the Theory of Machines and Computations. New York: Academic Press 1971
53. Hurd, L.: Private communication

Communicated by O. E. Lanford

Received December 12, 1983; in revised form April 17, 1984