THE UNIVERSITY OF MICHIGAN
COLLEGE OF LITERATURE, SCIENCE, AND THE ARTS
Computer and Communication Sciences Department

CELLULAR AUTOMATA MECHANICS

Tommaso Toffoli

The Logic of Computers Group

November 1977

Technical Report No. 208

with assistance from:

National Science Foundation
Grant No. MCS76-04297
Washington, D. C.

CELLULAR AUTOMATA MECHANICS

by

Tommaso Toffoli

Abstract

Cellular automata, whose study originated from problems of computation theory and biological modeling, appear to be a finitary counterpart of certain partial differential equations (namely, those of fields in homogeneous media) that play a fundamental role in theoretical physics. We investigate the capabilities of cellular automata as abstract dynamical systems and as paradigms of physical mechanics. In particular, we prove that the computing and constructing capabilities for which cellular automata are well known are preserved if one imposes the reversibility constraint. This result allows one to represent computing processes as taking place in an abstract "medium" that satisfies realistic phyical-like constraints. (It is well known that reversibility is indissolubly associated with many of the most important concepts of physics.)

We also show that cellular automata of a sufficiently small number of dimensions are amenable to a uniform physical implementation (without neglecting synchronization and initialization problems); we discuss certain experiments aimed at determining with what frequency one can expect the spontaneous appearance of elementary levels of organization in cellular-automaton statistical assemblies; and we investigate the relevance--as a computing resource--of the interconnection pattern of a sequential network.

ACKNOWLEDGMENTS

I wish to thank the many people who have helped me in this research. I am particularly indebted to John H. Holland, the chairman of my doctoral committee, for constant and warm encouragement, constructive criticism, and many ideas. Moreover, it was his long-standing interest in cellular automata as paradigms for fast computation that inspired my own interest in such structures.

I am greatly indebted to Arthur Burks and Richard Laing, for being so generous with their time, experience, and scientific wisdom, and to John Meyer, for valuable discussions and suggestions.

I am grateful to the Logic of Computers Group, which provided unique research facilities and a stimulating environment, and has been for me almost a second home for many years. I received help in many ways from all of its members.

I wish to express my gratitude to my wife Karan for managing to tame a very unruly list of references.

This research was supported in part by the Consiglio Nazionale delle Ricerche, Roma, Italy, and by the National Science Foundation, Grant No. MCS76-04297 , Washington, D.C.

FOREWORD

The word "mechanics" in the title of this work is deliberately ambiguous. On one hand, it refers to the mechanics of cellular automata treated as abstract dynamical systems. On the other hand, it implies that cellular automata can be used to advantage as paradigms of physical mechanics. Thus, cellular automata act as a bridge between certain aspects of mathematics and of physics, in a role analogous to that played by differential equations. In my opinion, they would loose much of their interest if this connection were ignored.

In April 1975, John Holland, Lutz Priese, Alvy Smith, and I were discussing the possible fate of cellular automata theory, and we started making not too flattering comparisons with a number of other "fads" for which a rapid spread of interest had been followed by almost total oblivion. Cellular automata has some of the symptoms of this disease, but we were willing to admit that our patient had perhaps stronger stamina than others. With my Ph.D. thesis at an early planning stage, I was, so to speak, the intern who was supposed to take care of the patient on that shift. My approach was (I had not taken the Hippocratic oath), What are cellular automata worth saving for?

Eventually, I decided on a method of investigation that had much of the flavor of the proven "branch-and-bound" optimization

method. Namely, draw a tree of investigation topics, estimate according to a priori information possible costs and returns of each initial branch, and start with the more promising branches. Repeat the process for these branches, concentrating on sub-topics, and feel free to go back to a temporarily abandoned branch whenever more precise estimates lead to a revision of priorities.

After a certain time, my tree had some of its branches blocked by exceedingly large demands on my current mathematical capabilities (in this context, costs depend very much on one's scientific background). On the other hand, one branch led very soon to unexpectedly good results. This is the one concerned with the connections between mathematics and physics. Such connections are many and deep rooted. Cellular automata play a particularly natural role as computational models of physics and, at the same time, as physical models of computation. In particular, reversibility, which appears to be a fundamental feature of the physical laws, can be imposed on cellular automata without jeopardizing their unique computation and construction capabilities.

These results about reversible cellular automata are just one fruit at the end of one branch. In presenting it to the readers, we hoped that they might be interested in knowing what the branch itself, the leaves, and--at least in overview--the whole tree looked like, so that they may start exploring other branches from a vantage point.

TABLE OF CONTENTS

CHAPTER 0

I N T R O D U C T I O N

0.0 Motivation

The processing power of computing systems used to be limited essentially by the number of active devices that one could effectively employ, as these represented the most expensive, bulky and unreliable portion of the whole apparatus. Today, alongside a dramatic reduction in size and a corresponding increase in reliability, active devices have attained very low fabrication costs. What if one tried to reduce also the cost of design and interconnection by using identical devices and laying them out, side by side, in a uniform array? In this way, it would seem that one could construct, at a fixed cost per unit element, computers of arbitrarily large size and, perhaps, correspondingly large computing power.

Is the implementation of such a project affected by any intrinsic limitations? Can one conceive of a substantially more efficient way to organize an arbitrarily large computer? Finally, are there any realistic problems whose treatment could make effective use of an extremely large number of computing elements with local and uniform interconnections?

A great deal of effort is currently being devoted to trying to solve in terms of parallel algorithms certain problems of actual interest, in the hope, of course, that the structure of such algorithms would eventually be matched by that of suitable parallel computers, thus permitting the fast treatment of problems of large size. One may be led to assume that, because of scale economy, the efficiency of such a match would improve or at least be preserved as problem size increases. Unfortunately, this is not true. Physical constraints of a general nature severely limit the class of parallel algorithms that can be efficiently implemented independently of problem size. In most cases, performance is eventually degraded to that of algorithms featuring a much lesser degree of parallelism. For this reason, it is important to inquire what abstract computing structures admit, at least potentially, of an efficient implementation as parallel computing systems of unboundedly large size. Cellular automata are likely candidates and, in any case, a convenient starting point.

0.1 Organization

In Chapter 1, we introduce cellular automata in a critical way, interpreting and making more precise the traditional definitions. In Sections 1.1 and 1.2 we stress the distinction between a cellular automaton, which is but a particular kind of description

of a dynamical system, and the dynamical system itself. This approach introduces a very clear distinction between certain "simulations," "blockings," "speed-ups," etc. described in the literature which merely lead to alternative descriptions of the same dynamical system (via coordinate transformations, as in Sections 1.5, 1.6, and 6.9) and other "simulations" that lead, via homomorphisms or other correspondences, to related but different dynamical systems (cf. Section 1.4). In Section 1.3 we study the tessellation group of a uniform automaton from the viewpoint of its compatibility with the geometry of physical space. We define the concept of "bounded embeddability," which is the formal counterpart of certain physical constraints on the structure of a computing network, and we motivate in terms of this concept the identification of cellular automata with those uniform automata whose tessellation group is free Abelian. In Section 1.5 we consider cellular automata from the viewpoint of their analogies with physical systems, and in Section 1.7 we discuss their role as computational models of physics.

In Chapter 2 we show that cellular automata of appropriate dimensionality are physically realizable as distributed computing networks by means of an implementation schema that is independent of the number of cells actually implemented. Section 2.1 deals with quantum-mechanic and relativistic constraints on the geometry of the computing network, while Sections 2.2 and 2.3 deal with energetic and thermodynamical constraints. Section 2.4 gives a "canonical" solution of the synchronization problem for an asynchronous network,

and Section 2.5 shows how initialization and read-out of arbitrarily large areas and blanking of the whole cellular array can be achieved in the context of the proposed synchronization schema.

Chapter 3 constitutes the nucleus of the present work. We discuss the relevance of reversibility in the analysis of dynamical systems (Sections 3.0 and 3.1) and review the literature concerned with reversible cellular automata (Section 3.2) In Sections 3.3 and 3.4 we prove that computation- and construction-universality are retained in reversible cellular automata. The proof is based on an effective construction whereby any cellular automaton is shown to be embeddable in a reversible one having one more dimension. The nature of the contrast between our result and some erroneous ones that had appeared in the literature is analyzed and clarified in Section 3.5. Section 3.6 discusses the relevance of reversible cellular automata for the modeling of "intelligent" behavior in a physical-like context.

In Chapter 4, we discuss from an experimental viewpoint the "numerical" simulation of cellular automata, and we give some statistical and pictorial evidence for the spontaneous appearance of recognizable--though very elementary--levels of organization in conditions approximating steady-state equilibrium. The originality of our approach consists in that, instead of using one selected cellular automaton expected to have particularly favorable characteristics in this respect, we surveyed the entire family (4096 members) of cellular automata defined by certain simple constraints (two-state cells, von Neumann neighborhood, and rotation-invariance). In this

way, one can directly verify that the capability to achieve elementary levels of self-organization in a relatively short time is a rather common property of cellular automata rather than a "rarity."

Many abstract results from dynamical-system theory (topological dynamics and ergodic theory) are potentially applicable to cellular automata. To facilitate this transfer, it is useful to recast the concepts of cellular automata theory into more abstract terms. Chapter 5 is a step in this direction. Essentially, we briefly explore the aspects of a characterization of cellular automata in topological terms.

Finally, in Chapter 6, we use cellular automata and related structures as a stylized representation of physical computation. This provides a very convenient background for discussing certain issues concerned with the measuring of computing resources involved in a computation. In particular, the interconnection structure of a network is explicitly treated as a computing resource (Sections 6.6-6.8). Moreover, the spacetime representation of a cellular automaton, as introduced in Section 6.8, is particularly appropriate for describing coordinate transformations that involve a linear combination of time and space coordinates (Section 6.9).

0.2 Notation and mathematical preliminaries

Sets. Union and intersection of sets are denoted, respectively, by $\cup$ and $\cap$. The cardinality, or size, of a set A will be denoted by $|A|$.

Common mathematical structures. Certain common mathematical structures consisting of a set together with certain operations tacitly associated with it will be denoted as follows:

N, the natural numbers (including zero),
Z, the integers,
Z_n, the ring of integers modulo-n,
R^n, the real numbers,
E^d, the d-dimensional Euclidean space,
C_n, the cyclic group of order n,
C_∞, the infinite cyclic group.

Logical operators and quantifiers. The operators "and" and "or" are denoted, respectively, by $\wedge$ and $\vee$. Similarly, the universal and the existential quantifier are denoted, respectively, by $\bigwedge$ and $\bigvee$. The range of such quantifiers is usually specified in an explicit way; e.g., $\bigwedge_{i=0}^{\infty}$ (or $\bigwedge_{i \in N}$) denotes quantification over the set of natural numbers. Implication and equivalence are denoted, respectively, by $\Longrightarrow$ and $\Longleftrightarrow$.

Indexed and ordered sets. Let A denote a finite set of size

r . For sake of enumeration, we shall often consider A ordered and identify it with the ordered set $\langle 0,1,\ldots,r-1\rangle$. Similarly, in enumerating a set of n-tuples from A^n we shall implicitly adopt the lexicographical ordering induced by that on A . In this context, the symbols $\oplus$ and $\ominus$ used as operators on A denote, respectively, addition and subtraction modulo-r.

In general, if a is an arbitrary n-tuple, a_i will denote its i-th component. The whole n-tuple will be written explicitly as $\langle a_1,\ldots,a_n\rangle$ or, in an abbreviated form, as $\langle a_i\rangle_{i=1}^{n}$, where i is used as a dummy index ranging from 1 through n . Similarly, the symbol $[a_i]_{i=1}^{n}$ will denote the n-tuple $a_1,\ldots,a_n$ written <u>without</u> the enclosing brackets. For example, $\langle a,b,c_1,c_2,c_3\rangle$ may be abbreviated as $\langle a,b,[c_i]_{i=1}^{3}\rangle$. Moreover, if I is an arbitrary index set of size n , $\langle a_i\rangle_{i\in I}$ will indicate the n-tuple whose elements have indices ranging over I . The implied ordering for I may be assigned arbitrarily, provided that the same assignment is used consistently in related formulas. Finally, where safe, the explicit reference to the index set I may be dropped; thus, we shall simply write $\langle a_i\rangle_i$ or $\langle a_i\rangle$ instead of $\langle a_i\rangle_{i\in I}$. When necessary, an additional index will be appended to the kernel as a <u>superscript</u>. Thus, $\langle\langle a_i^j\rangle_{j=1}^{n}\rangle_{i=1}^{m}$ will denote an m-tuple of n-tuples.

If both a and b are n-tuples of the form $\langle a_i\rangle$, $\langle b_i\rangle$, a+b will represent their <u>componentwise sum</u> $\langle a_i+b_i\rangle$, and -a the <u>opposite</u> of a , i.e., $\langle -a_i\rangle$.

The notation for n-tuples shall be extended to infinite

sequences, such as in $\langle a_i \rangle_{i=-\infty}^{+\infty}$, and, when clarity permits, to the elements of a more complex indexed product of sets (cf. below).

Indexed products of sets. The Cartesian product of A and B will be denoted by AxB or, when no ambiguity is possible, by AB. More generally, the product of an indexed collection of sets $\langle A_i \rangle_{i\in I}$ is denoted by $\bigtimes_{i\in I} A_i$ (the notation $\{A_i\}_{i\in I}$ for an indexed collection of sets is improper, though commonly used). If all of the A_i coincide with the same set A, their product can be written as A^I. We shall use the customary (though slightly improper) notation A^n for the product $\bigtimes_{i=1}^{n} A$.

Let $a\in A^I$. The element a is an indexed set. The component of a indexed by $i\in I$ is denoted by a_i or $\pi_i a$, where π_i denotes the projection operator with respect to i.

Functions. If f is a function of the form $f: A \to B$, A is the domain of f; B its codomain. The range of f is the set of all points of B that have a counterimage in A under f, i.e., the set of all values that f actually assumes for some values of the argument.

A transformation on A (or of A) is a function whose domain and codomain both coincide with A.

Unless required for clarity, a function symbol will be immediately juxtaposed to the argument, if the function is unary

(i.e., if it is treated as a function of only one variable). Thus, we write fa or $f\langle a_1,\ldots,a_n\rangle$ instead of $f(a)$ or $f(\langle a_1,\ldots,a_n\rangle)$. Also, we write fga for $f(g(a))$ when no confusion is possible.

Graphs. We shall be interested mainly in directed graphs. A graph is labeled when certain attributes (or symbols) are associated with its nodes; colored when certain attributes are associated with its arcs. We shall call discrete a finitely-generated group. The Cayley graph of a discrete group G [with respect to a given set X of generators for G] consists of the directed, colored graph whose nodes are in one-to-one correspondence with the elements of G and such that an arc $\langle p,q\rangle$ of color x exists iff $x\in X$ and $q = xp$. If this definition is modified so that X is an arbitrary subset of G, the graph thus obtained is called the group graph of G [with respect to X] (cf. [BART75]).

CHAPTER 1

C E L L U L A R A U T O M A T A

1.0 Preliminaries

Consider an indefinitely extended checkerboard, and place a number of tokens on it--no more than one for each square, or cell. This constitutes a configuration. Each cell is surrounded by eight neighbors. Construct a new configuration by applying the following rule, or transition function:

(i) Mark all cells that either contain a token and are surrounded by two or three tokens, or are empty and are surrounded by three tokens.

(ii) Remove all tokens, place new ones on the marked cells, and erase all marks.

By iterating the above process, a whole sequence of configurations, or trajectory, is obtained. Starting from suitable initial configurations, one obtains patterns of tokens that move, grow, and evolve in a way that may remind one of a population of biological organisms.

This elementary example of cellular automaton, devised by Conway[GARD70] and better known as the game "life," illustrates how extremely simple rules can be used to characterize very complex

behavior. In fact, with suitable encoding of variables and processes, one can even use the game "life" as a general-purpose computer (cf. [GARD71]).

Cellular automata were introduced by von Neumann ([VONN66], edited and completed by A. W. Burks), following a suggestion of Ulam, in order to provide a general framework for discussing complex structures capable of self-repair, self-reproduction, and other forms of behavior that are not conveniently expressed in a finite-automaton context. Holland[HOLL59,HOLL60] promptly showed their connection with parallel computation in iterative networks. The theory of cellular automata received a first consolidation by Burks[VONN66,BURK70], and considerable simplifications were soon introduced by Codd, Banks, and Smith[CODD68,BANK71,SMIT69]. On the whole, however, the theory developed in quite an unsystematic and fragmentary way, representing as often as not a part-time interest of some computer scientist or interdisciplinary biologist. As a consequence, cellular automata are probably better known in connection with certain occasional problems carrying picturesque names (the Garden of Eden[MOOR62], the Firing Squad[MOOR64], the French Flag[ALAD72], the Busy Beaver[GAJS75], and, of course, the game "life") rather than with more austere branches of traditional mathematics.

Following Yamada's and Amoroso's persistent attempts to give the theory a more acceptable status[YAMA69,YAMA70,YAMA71,AMOR72], several researchers started analyzing cellular automata by means of well-tested mathematical tools, such as abstract algebra, topology,

and measure theory (see, for example, [HARA72,SATO75,WILL75]). It seems clear from current work that many important concepts originally formulated in quite different contexts can be applied with success to answering questions in cellular automata theory. At the same time, one can notice a tendency to carry over from such contexts also concepts and problems that, though still applicable to cellular automata, do not seem to be particularly relevant.

The recent Tokyo conference on "Uniformly Structured Automata and Logic"[USAL75], while testifying that cellular automata are attracting the attention of an increasing number of workers, has not revealed any explicit motif capable of exerting a unifying influence on the subject. It is not even clear why cellular automata, which from a purely mathematical viewpoint are rather undistinguished structures, should be so "popular."

In our opinion, the importance of cellular automata lies in their connection with the physical world. In fact, they possess certain constructive properties that set them in a priviledged position as models of physical computation and, at the same time, as computational models of physics.

The characteristic features of cellular automata, which, for the moment, we shall present in an informal way using the terminology introduced for the game "life," are the following:

(a) Space (i.e., the checkerboard structure) is discrete.

(b) Time (i.e., the indexing of successive configurations)

is discrete.

(c) The state variables (i.e., the states of the individual cells) range over a finite set.

(d) The transition function is local (i.e., the next state of a cell depends only on the current state of the cell itself and its neighbors).

(e) Space is homogeneous (i.e., a pattern of tokens evolves in the same way independently of where it is placed on the checkerboard) and finite-dimensional (i.e., each cell can be simply addressed by means of an n-tuple of integers).

(f) Time is homogeneous (i.e., a pattern evolves in the same fashion independently of when it occurs).

Features (a), (b), (c), and (d) make it possible to study the elementary properties of cellular automata without requiring the introduction of elaborate mathematical tools (such as continuity, differential equations, etc.). In particular, they provide an effective means of constructing the trajectory originating from any given configuration. From this viewpoint, cellular automata display strong analogies with Turing machines. On the other hand, features strictly analogous to (d), (e), and (f) also appear in the formulation of the fundamental principles of physics. For these reasons, cellular automata have often been used as a plausible physical-like background for investigating certain general questions (e.g., self-reproduction) in which symbolic and mechanical aspects appear in a particular close relationship.

1.1 Dynamical systems

Starting from models of particular interest for the physicist, there has evolved in the course of years a vast literature of mathematical structures known collectively as "general dynamical systems." The emphasis on particular mathematical aspects of the theory and the degree of abstraction vary widely depending on the intended application.

DEFINITION 1.1.1 A [general] dynamical system U is a structure $\langle C,\tau\rangle$, where C is an arbitrary set called the phase space and τ is an arbitrary function of the form $\tau: C \to C$ called the dynamical map. Depending on the context, and according to the need to avoid ambiguities, an element of C may be called a state, a configuration, or a point of U.

The reader must be warned that there exist many nonequivalent definitions of "dynamical system." Ours is particularly convenient for a discussion having a finitary emphasis.

REMARK 1.1.1. Given a dynamical system $\langle C,\tau\rangle$, the iterates of τ, of the form τ^t ($t\in N$) (where $\tau^1 = \tau$ and $\tau^0 = I$) form, under composition, a one-parameter semigroup with identity (cf. [JACO63]). In certain cases, it is possible to define in a unique way also fractional powers of τ, of the form $\tau^{m/n}$, or, more generally,

powers of the form τ^t ($t \in R$), obtaining, in the latter case, a continuous one-parameter semigroup. Originally, systems of differential equations were used for defining such continuous semigroups of transformations[SIBI75]. (The word "continuous" here seems to refer to the real-number continuum as contrasted to a discrete set[GOTT55]).

REMARK 1.1.2. The dynamical map of a general dynamical system may be compared to the mechanical laws of an ordinary physical system. From this viewpoint, the semigroup parameter t may be identified with time; thus, it is customary to say, for instance, that a system governed by dynamical map τ evolves during time interval t from an initial state c to a final state $\tau^t c$. While the temporal aspect of a dynamical system is thus, so to speak, "built-in," additional structure is needed in order to extend the analogy with physics to spatial aspects: essentially, the dynamical map must commute with the elements of a suitable group of transformations on the phase space.

Specific classes of dynamical systems are introduced by imposing particular constraints on the cardinality of the phase space and the nature of the dynamical map. In general, it is difficult to decide whether a set of constraints specifies a single system (up to an isomorphism), a whole class of systems, or no system at all (this happens when the constraints are incompatible). These difficulties can be avoided to a great extent if a system is specified in a constructive way by supplying a presentation, i.e., a symbolic model, of it.

Cellular automata, formally defined in Section 1.2, constitute a class of such presentations. The models they provide may be interpreted as sequential networks consisting of identical, uniformly interconnected elements. The functional uniformity of such networks can be preserved--for cellular automata having a sufficiently small number of dimensions--as structural uniformity in a physical realization. The design and production economies that can be achieved by taking advantage of such uniformity make it possible to implement cellular automata on a very large scale. Moreover, the bounded interconnection length associated with geometrical uniformity makes it possible to operate such (physical) networks at a very high clock rate. Thus, the numerical simulation of the behavior of cellular automata can be carried out in a very efficient way.

For the above reasons, a dynamical system that admits of a cellular-automaton presentation can be easily subjected to extensive empirical investigation. On the other hand, it appears that a great deal of abstract knowledge about dynamical systems can be applied more or less directly to cellular automata (and consequently to a large class of parallel-computing mechanisms). In particular, the dynamical systems that admit of a cellular-automaton presentation possess, for this very reason, a number of well-characterized topological, measure-theoretic, and group-theoretic properties. Properties of such nature are analyzed very extensively in the literature (see, for example, [GOTT55,HALM56]). Thus, by extending his attention to the theory of dynamical systems, the researcher in cellular automata, who is often

motivated by interdisciplinary interests, can readily gain access to a number of useful results and techniques already developed by more specialized mathematicians.

In the remainder of this section, we shall briefly review a number of problems that arise when dealing with dynamical systems having a "large" phase space. There are important methodological differences between the study of finite and that of infinite automata.

Finite, denumerable, and uncountable systems. If the phase space C of a dynamical system is finite, the set C^C of all possible dynamical maps on C is finite, too, and any one of them can be explicitly specified by means of a presentation consisting of (i) a finite set of labels (i.e., finite words over an assigned finite alphabet) and (ii) a transition table consisting of ordered pairs of such labels. Dynamical systems having a finite phase space are studied in the theory of finite automata.

If C is denumerable, it is still possible to represent each element of C by means of a distinct label. However, such labels form an infinite set, and the transition table is, correspondingly, infinite. In this case, the best one can do in order to supply a constructive characterization of a system is give an effective algorithm for generating, on request, any entry of the table. The dynamical maps that can be assigned in this way form a small (denumerable but not recursively enumerable) subset of C^C. (At any rate, since C^C is nondenumerable, most of its elements are not

individually identifiable by means of a defining sentence of any kind, as such sentences form a denumerable set.) Systems having a denumerable phase space are studied in the theory of recursive functions.

Finally, if C is not countable, it is impossible even to represent with a distinct label each individual point of C, and the presentation of a particular system will, in general, use labels for certain distinguished sets of points rather than for individual points. In order to have a complete model, it must be possible to construct sequences of labels corresponding to more and more refined descriptions, so as to "converge" to any desired point. In this context, it is easy to understand why topology and measure theory are so important in the theory of dynamical systems.

Toplogy and measure-theory. Topologies and σ-algebras are essentially groupings of elements of a set into a family of distinguished subsets for which particular operations are defined. They allow one to refer to a point in the phase space in terms of its being a member of some of such subsets. From a constructive viewpoint, one may label certain of the distinguished subsets and represent operations on such subsets in terms of symbolic manipulation of labels. In order for such a labeling schema to be useful in constructing a presentation of a dynamical system $\langle C,\tau\rangle$, it is necessary that the labels associated with an element $c\in C$ (i.e., the names of the subsets to which c belongs) be related in some consistent way with those of its image τc. Typically, one requires that if two points

c , c' are "near" according to a certain labeling schema, also their images under τ be "near." These concepts can be defined precisely, and dynamical maps that are consistent in this sense are called continuous maps if the distinguished subsets form a topology, and measurable maps if they form a σ-algebra. Given the close similarity between the axioms for a topology and those for a σ-algebra, it is not surprising that in many applications the two labeling schemata can be used interchangeably. (For example, for many dynamical systems topological and measure-theoretic entropy concide[ADLE65].)

Since, as noted above, the cases of a finite and a denumerable phase space are specifically dealt with in, respectively, finite-automaton and recursive-function theory, the theory of dynamical systems is typically more interested in systems having an uncountable phase space. Cellular automata, though historically related to finite automata, and thus to problems of a discrete nature, have an uncountable phase space; therefore, their study naturally leads to using topological and measure-theoretic tools much in the same way as for other dynamical systems that are historically related to differential equations, and thus to problems of a continuous nature. We shall explicitly discuss certain topological aspects of cellular automata in Chapter 5. However, it must be noted that concepts such as "cell-state," "local map," "finite configuration," etc., introduced in the next few sections are substantially of a topological character. Topological arguments were first explicitly introduced in cellular automata theory by Richardson[RICH72]; measure-theoretic ones by

Willson[WILL75] (though cf. [MOOR62] for early hints).

State variables. Let us consider a finite automaton described by a given transition table--in which the automaton's states are treated as "points" without any internal structure. It is always possible to describe the same automaton by means of a sequential network. In so doing, one introduces some structure in the phase space. In fact, the network's delay elements represent the components, in a certain coordinate system, of the automaton's global state, and the combinatorial elements describe the dynamical map in terms of relationships between such components. While, in certain cases, by choosing suitable coordinates, one can achieve by means of a sequential network a greater economy of description than with a transition table (this happens when some coordinates are almost separable), in most cases the two descriptions have comparable length (i.e., most automata, in a straightforward enumeration schema, are "random"--in the sense of [CHAI66]). Thus, the use of a structured description may at times represent a convenience but does not play an essential role in dealing with finite automata.

For dynamical systems having an infinite phase space, the alternative between a structured and an unstructured description is not available. The use of coordinates (or equivalent tools) is necessary in order to present such systems in an effective way. (Note that labels as strings over a finite alphabet constitute a coordinatization of the phase space.) Informally speaking, we cannot describe one such system

without at the same time "interpreting" its operation in terms of a specific mechanism. For these reasons, the concept of "state variable," of which "component" (in a system of coordinates) is a special case, is an important one in the study of infinite systems.

DEFINITION 1.1.2. A function ϕ of the form $\phi: C \to Q$, where C is the phase space of a dynamical system M and Q is an arbitrary set, is called a state variable of M.

In most applications, one uses state variables whose range is much "smaller" than their domain (for an example, see Remark 1.2.1), since their typical function is indeed to allow one to speak of a large set (the phase space) in terms of smaller ones. Intuitively, a state variable defines an attribute (e.g., the velocity of a given particle) of the system's state, and its range represents the possible values for that attribute.

Given a dynamical system $M = \langle C, \tau \rangle$ and state variables $\phi: C \to P$, $\psi: C \to Q$, the dynamical map τ induces on $P \times Q$ a relation $\overrightarrow{\phi\psi}$ defined by

$$\bigwedge_{\substack{p \in P \\ q \in Q}} \left\{ p \overrightarrow{\phi\psi} q \iff \bigvee_{c, c' \in C} p = \phi c \wedge q = \psi c' \wedge c' = \tau c \right\} .$$

Intuitively, if the state of the system M is known in detail, i.e., if a configuration c is given, one can predict its next state $c' = \tau c$ with precision. However, if c is known only through an attribute $p = \phi c$, what can one say about attribute q at the next

instant? p and q are related by $\overrightarrow{\phi\psi}$, which can be thought of as a transformation of the available information. In spite of a relation of the form $\overrightarrow{\phi\psi}$ being, in general, nondeterministic, it is often possible to completely characterize τ by means of a suitable collection of such relations. In many applications, one uses relations of the form $\overrightarrow{\phi\psi}$ that are deterministic. An example is given by the local map of a cellular automaton (cf. Section 1.2), which uniquely specifies (in finitary terms) the dynamical map of a dynamical system having an uncountable phase space.

The projection operators associated with a phase space treated as a Cartesian product of sets are state variables which are continuous with respect to the product topology. In general, if a certain topology constitutes an appropriate labeling schema for the points of the phase space of a dynamical system, dynamical variables that are "well-behaved" (e.g., continuous) with respect to that topology are the most natural tools for discussing the behavior of that system.

1.2 Uniform and cellular automata

In this section, we formally define cellular automata and a number of related concepts. Our purpose here is not to achieve the utmost of conciseness or of mathematical "depth" (we shall make an attempt in this direction in Chapter 5), but to facilitate comparison with the terminology used by most authors. On the other hand, our definitions do not represent merely a "weighted average" of what is found in the literature. Rather, after considering a number of possible variants, we decided on a characterization of cellular automata that gives them a clear status as general models of physical computation (cf. Section 1.3, Chapter 2, and Chapter 5).

In particular, we have maintained and clarified (Section 1.3) the restriction of the tessellation array to one generated by an Abelian group[HOLL60]; we have dropped as substantially redundant (Section 1.5) the customary requirement for a quiescent state; and we have not accepted generalizations of the state alphabet to denumerable or uncountable sets[YAMA69,BART75], nor transition functions dependent on an external "input"[YAMA69] or nondeterministic transition functions[RICH72]. The fact that, in spite of the many proposed variants, an analysis based on physical motivations leads to a definition of "cellular automaton" that essentially coincides with the original one[VONN66] is a tribute to the practical insight of the

pioneers in the field, such as von Neumann, Burks, Holland, and Moore.

We start by defining uniform automata, of which cellular automata (Definition 1.2.6) are a particular case.

DEFINITION 1.2.1. A uniform automaton is a structure $\langle A,S,X,\lambda\rangle$, where A , the state alphabet, is a finite set; S , the tessellation group, is a discrete group (i.e., a finitely-generated one--note that this is more restrictive than a countable group, which need not be finitely-generated[KURO55]); X , the neighborhood template (also called neighborhood index[YAMA69]), is a finite subset of S , and λ , the local map, is a function of the form $\lambda: A^X \to A$.

To every uniform automaton $\langle A,S,X,\lambda\rangle$ we associate a dynamical system $\langle C,\tau\rangle$, as explained below.

From the abstract group S we construct two entities, namely, a set P of objects called cells and a set D of transformations on P , in such a way that $\langle P,D\rangle$ constitutes a regular representation of the group S [ZASS49]. The elements of D will be called displacements. P and S are constructed as follows. Let $\square$ be an arbitrary object, called the origin. Define $P = \{\langle\square,s\rangle \mid s\in S\}$. For all $s\in S$, let d_s be the function of the form $d_s: P \to P$ such that $d_s\langle\square,s'\rangle = \langle\square,s's\rangle$. Define D as the set of all such d_s .

STANDING CONVENTION 1.1.1. Clearly, for all $p \in P$, $d_{s'}{}_{s}p = (d_s d_{s'})p$. Since thus D forms, under composition, a group isomorphic to S, no ambiguity will result if one identifies the operator d_s with s itself and writes (using postfix notation) $\square s$ instead of $\langle \square, s \rangle$ $(= d_s \langle \square, 1 \rangle)$. (Intuitively, $\square s_1 s_2 \ldots s_i$ represents the cell that is reached by performing a sequence of moves $s_1, s_2, \ldots, s_i$ starting from the origin $\square$.) Also, when the choice of the origin is understood or irrelevant, one can denote $\square s$ simply by s. Nevertheless, the conceptual distinction between the displacement d_s and the cell $\square s$ --both of which, following custom, we shall henceforth denote by s when no confusion is possible--should be maintained. With the same warning, we shall likewise use S to denote both the set of cells P and the set of displacements D, for both of which there is an understood one-to-one correspondence with S. The reader will eventually appreciate the convenience of such notation.

Having constructed P and D, we proceed now to construct the phase space C and the dynamical map τ of the dynamical system $\langle C, \tau \rangle$ associated with the given uniform automaton. In this context, the elements of C will be called configurations and τ will be called the parallel map.

DEFINITION 1.2.2 (Configurations). The elements of the state alphabet A are called cell-states. A configuration is an assignment of cell-states to all cells, i.e., an arbitrary function of the form

$c: S \to A$. Thus C (the set of all configurations) $= A^S$. Cell-state cs , called the state of cell s in configuration c , will be denoted, more conveniently, by c_s .

REMARK 1.2.1. Since C is an indexed product of sets of the form $\bigtimes_{s\in S} A$, the state of cell s in configuration c (in other words, the s-component of c) is obtained by applying to that configuration the projection operator π_s (i.e., $\pi_s c = c_s$) . Such projection operators are typical examples of state variables of the dynamical system $\langle C,\tau\rangle$ (cf. Section 1.1). Traditionally, if $c_s = \pi_s c$, one may speak of "the state variable c_s" instead of, more correctly, "the state variable π_s ," thus naming a function by its result. However, one should not confuse a cell-state, which is a specific element of the state alphabet, with the state of a cell, which is a function that may take on different cell-state values depending on the configuration to which it is applied.

DEFINITION 1.2.3 (Neighborhood). The [input] neighborhood sX of a cell s is the set of cells $\{sx|x\in X\}$, where X $(\subset D)$ is the neighborhood template of the cellular automaton. Its elements are the [input] neighbors of s . An [input-]neighborhood configuration of s is an assignment of cell-states to each cell of the neighborhood of s , i.e., a function from sX to A . In particular, the neighborhood configuration of s in c , denoted by c_{sX} , is the restriction of configuration c to the neighborhood of s . In an analogous way one defines the output neighborhood of a cell, etc.

In this context, the term input is understood and may be dropped, while the term output must appear explicitly.

DEFINITION 1.2.4 (Parallel map). In conjunction with the neighborhood template X, the local map λ specifies the parallel map τ of the dynamical system $\langle C,\tau\rangle$ as follows:

$$\bigwedge_{c,c'\in C} \left(c' = \tau c \iff \bigwedge_{s\in S} c'_s = \lambda c_{sX} \right) .$$

(Briefly, τc is the configuration obtained from c by applying the local map λ to each neighborhood configuration of C.)

Informally, if configuration c represents the current state of a uniform automaton (at a certain moment, or at time step t), configuration $c' = \tau c$ represents its next state (at the successive moment, or at time step $t+1$). Thus, while λ is the transition function of an individual cell considered as a finite automaton having as inputs the states of its neighbors, τ is the transition function of the entire uniform automaton considered as a possibly infinite, autonomous sequential machine. While λ is a finitary construct, τ in general is not.

DEFINITION 1.1.5. Two cellular automata are equivalent if the associated dynamical systems are isomorphic. (Intuitively, equivalent cellular automata characterize--possibly in different terms--the same dynamical system.)

STANDING CONVENTION 1.1.2. As explained above, a uniform automaton is a mathematical structure characterized by many parameters. Since we shall have occasion to introduce several classes of uniform automata, the following conventions will simplify our task by automatically introducing the nomenclature required in each case both for the desired uniform automaton and for the associated dynamical system. On its first occurrence, the name of a uniform automaton will be accompanied by the name or the value (occasionally both) of certain of its paramenters, according to the template

$$M[C[A(r),S(d)],\tau[X(n),\lambda]] \ ,$$

where the place of a parameter specifies its meaning as follows:

M = uniform automaton
C = set of configurations
A = state alphabet
r = number of cell-states (i.e., size of A)
S = tessellation group
d = number of dimensions (i.e., rank of S), when applicable
τ = parallel map
X = neighborhood template
n = number of neighbors (i.e., size of X)
λ = local map

In dealing with a particular automaton, we shall omit from the template those parameters to which no explicit reference will be needed in the discussion. For instance, $M[C[A,],[(n=3),\lambda]]$ will introduce a uniform automaton M with set of configurations C , state alphabet A , number of neighbors $n = 3$, and local map λ , while the size of the state alphabet, the number of dimensions, and

the size of the neighborhood, as well as their name and that of the neighborhood template ond of the parallel map, are left unspecified.

DEFINITION 1.2.6. A cellular automaton is a uniform automaton whose tessellation group S is free Abelian. The rank of S is the number of dimensions of the cellular automaton.

The relevance of the concept of cellular automaton (as contrasted to the more general concept of uniform automaton) will be discussed in Section 1.3. Additional important concepts pertaining to uniform or cellular automata will be introduced in Section 1.4. In the remainder of this section, we shall introduce some notation peculiar of cellular automata.

When dealing with the tessellation group S of a cellular automaton, we shall use the additive notation customary for Abelian groups. Thus, 0 will represent both the origin cell and the null displacement; $s'+s$ the composition of the displacements s',s; $s+X$ the neighborhood of cell s; etc.

STANDING CONVENTION 1.2.3. Let d be the rank of S (being finitely generated, S has finite rank), and let $\{\sigma_i\}_{i=1}^{n}$ be a basis (i.e., a free set of generators) for S. Every element of S can be uniquely expressed as a linear combination of the form

$$s = s_1\sigma_1 + s_2\sigma_2 + \ldots + s_d\sigma_d$$

with integer coefficients. When the basis and its ordering are understood, s can be identified with the d-tuple $s_1,\ldots,s_d$. The integer s_i is called the i-coordinate of s. Such representation by means of d-tuples of coordinates will be used for both cells and displacements when notated as elements of S according to Standing Convention 1.1.1. In particular, the two identities

$$\begin{aligned} &\text{(a)} \quad \langle s_i\rangle+\langle s_i'\rangle = \langle s_i+s_i'\rangle \quad \text{and} \\ &\text{(b)} \quad \langle s_i\rangle-\langle s_i'\rangle = \langle s_i-s_i'\rangle \end{aligned} \tag{1.2.1}$$

can be interpreted as follows: In (a), displacement s' is added to cell s, yielding a new cell $s+s'$; in (b), displacement $s-s'$ is constructed as the difference between cells s and s'.

1.3 The tessellation group as a spatial category

In this section, we shall discuss certain geometrical aspects of the physical realization of uniform automata; dynamical aspects will be treated in Chapter 2.

We shall assume the reader to be familiar with sequential networks and their physical realization by means of digital circuitry. Briefly, a sequential network consists of a colored digraph called the structure graph, together with certain attributes associated with the graph's arcs and nodes. In the physical realiza-

tion here contemplated, each node is represented by a module containing, in conformity with the node's attributes, appropriate digital circuitry, and each arc is represented by a line capable of carrying, in conformity with the arc's attributes, an appropriate digital signal.

Note that the structure graph of a sequential network only specifies the topology of the modules' interconnection pattern, not its metric. However, there are a number of metric constraints that must be satisfied in a physical realization. Essentially (see Chapter 2), a specified clock rate imposes a lower bound on the volume of the modules and an upper bound on the length of the lines. For any finite sequential network, it is possible to avoid a conflict between the above two constraints by selecting a low enough clock rate. While this is true also for certain infinite sequential networks, it is not true for all of them.

As explained below, uniform automata constitute a class of infinite (except for a number of special cases) sequential networks. Thus, in the context of a physical realization, there arises the problem of determining what uniform automata have a structure graph that can be "embedded" in physical space in conformity with the above constraints.

With slight changes in terminology, a uniform automaton $M[[A(r),S],[X(n),\lambda]]$ can be identified with a sequential network as follows. The network's structure graph consists of the group graph

of S with respect to X (Section 0.2). Note that the attributes orientation and color are automatically associated with the arcs of a group graph. Orientation derives from each arc's being directed, while color derives from each arc's being associated, in the case of uniform automata, with a particualr element of the neighborhood template X . (Having domain in A^X , the local map λ can be thought of as a function of n variables-- n being the size of X . The coloring associates with each arc a well-determined argument of λ ; in a physical realization, this corresponds to specifying to which input port of a module a given line is connected.) In order to complete the characterization of M as a sequential network, one must associate additional attributes with the elements of the structure graph; namely, the state alphabet A (conceived as a set of signal states) is associated with each arc, and the local map λ (conceived as a switching function of n variables followed by a delay element capable of handling r-ary signals) is associated with each node.

In the present discussion, which is limited to geometrical aspects, (i) physical space will be idealized as a d-manifold with the Euclidean metric (this is consistent with experiment, over a scale large with respect to that of nuclear phenomena and small with respect to that of cosmological ones[OHAN76]). We shall denote such a manifold by E^d . Of course, for physical space, $d = 3$; however, we shall generalize to the case of $d \in N$. (ii) The physical realization of a uniform automaton M will be idealized as a set of points of E^d in a one-to-one correspondence with the nodes of the structure graph

of M. The above-mentioned metric constraints will be expressed by requiring that there exist a finite lower bound b on the distance between any two node images in E^d and a finite upper bound B on the distance between pairs of node images corresponding to nodes that are connected by an arc. Intuitively, the lower bound forbids arbitrarily dense packing of modules, while the upper bound forbids arbitrarily long propagation delays.

Because of the bounding constraints, the present embedding problem differs from the traditional one of algebraic topology, and is more closely related to the theory of group growth.

In algebraic topology, a graph G is embeddable in a manifold U if the geometric realization of G as a one-dimensional simplicial complex is homeomorphic to a subspace of U (refer to [WHIT73] for details). Informally, a graph is embeddable in U if its arcs can be "drawn" in U without intersecting. According to this definition, the graph of the free group over two generators (Figure 1.1a) is embeddable in E^2, while the complete bipartite graph $K_{3,3}$ (Figure 1.1b) is not; in fact, in the latter case crossover cannot be avoided. From a physical viewpoint, crossover presents no problems; in fact every countable graph is embeddable--in the algebraic-topological sense--in E^n, for $n \geq 3$. (A simple proof of this theorem can be found in [WHIT73]; note that that proof is stated for finite graphs but is valid also for countable ones.) On the other hand, the graph of Figure 1.1a clearly violates the bounding constraints when embedded

in E^d (for any $d \in N$), since, roughly speaking, it "fans out" at an exponential rate.

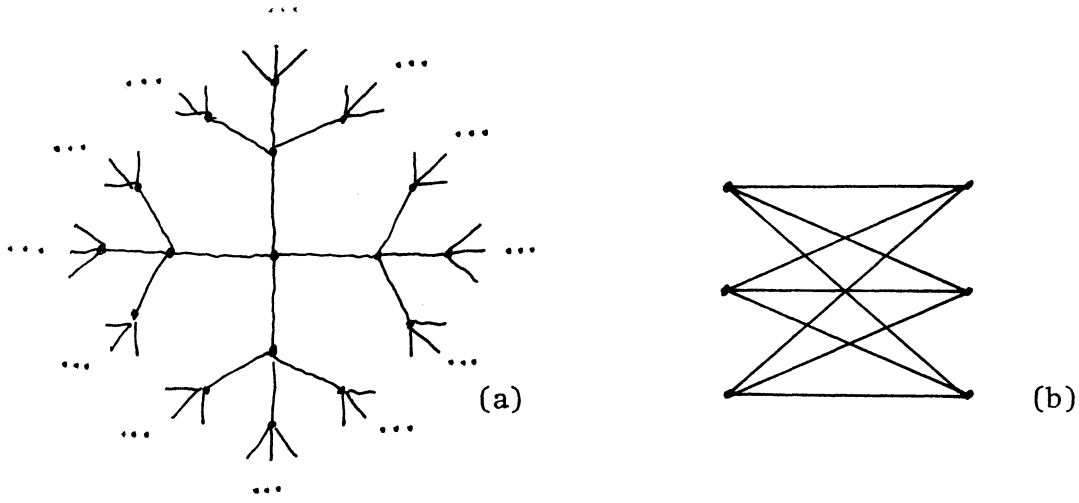


FIGURE 1.1. The graph of the free group over two generators (a) can be "drawn" in two dimensions, while the complete bipartite graph $K_{3,3}$ cannot without intersecting some arcs (b).

The theory of group growth studies the metric induced on a discrete group by the choice of a set of generators, as will be explained below in more detail. Substantially, such metric must be "close" to that of E^d in order for the Cayley graph of the group to be embeddable in E^d in conformity with the bounding constraints.

STANDING CONVENTION 1.3.1. The structure graph of a uniform automaton $M[[,S],[X,]]$ is connected only if X is a set of genera-

tors of S . Otherwise, the graph consists of a collection of distinct connected components (the automaton is laminated[YAMA69]). Such components correspond to disjoint portions of the sequential network, and, consequently, to independent dynamical systems. In such situation, S does not prescribe any definite topological relationship between nodes belonging to different components, and one can be introduced by adding to X other elements of S until a set of generators of G is obtained. In what follows, we shall always assume that X have been augmented in this way, and that, consequently, the structure graph of a uniform automaton be a Cayley graph.

DEFINITION 1.3.1. Given a discrete graph $G = \langle P,L \rangle$, a function $\phi: G \to E^d$ is a bounded embedding of G in E^d if there exist two positive real numbers b and B (called, respectively, lower and upper bound) such that

$$\bigwedge_{p_1,p_2 \in P} \delta(\phi p_1, \phi p_2) \geq b \text{ , and}$$

$$\bigwedge_{p_1,p_2 \in P} \left(\{p_1,p_2\} \in L \iff \delta(\phi p_1, \phi p_2) \leq B \right),$$

where δ is the Euclidean metric on E^d . A discrete graph G is boundedly embeddable in E^d if it admits of a bounded embedding in E^d . A sequential network is boundedly embeddable in E^d if so is its structure graph.

STANDING CONVENTION 1.3.2. It is easy to prove that, if the Cayley graph of a group with respect to a finite set of generators is boundedly embeddable in E^d, then the Cayley graph of the same group with respect to another finite set of generators is also boundedly embeddable in E^d. For this reson, when no confusion is possible, we shall speak of "the Cayley graph of a group" (implying "with respect to some finite set of generators") and of "the bounded embedding of a group" (i.e., the bounded embedding of any Cayley graph of that group).

REMARK 1.3.1. Clearly, any finite network and any finite group are boundedly embeddable in E^d, for any $d \geq 1$.

DEFINITION 1.3.2 (cf. [MILN68]). Let S be a discrete group, and $X = \{ x_i \}_{i=1}^{n}$ a finite set of generators of S. The length [with respect to X] of $s \in S$ is the minimum ℓ such that $s = x_{i_1}^{\pm 1} \ldots x_{i_\ell}^{\pm 1}$ (briefly, the length of s is the length of the shortest word over the generators that expresses s). The length of the group identity is, by convention, 0. The growth function of S [with respect to X] is the function $\gamma: N \to N$ such that γm is the number of group elements having length less than or equal to m. Two growth functions γ, γ' are equivalent if there exist constants c, c' such that

$$\gamma m \leq \gamma'(cm) \quad \text{and} \quad \gamma' m \leq \gamma(c'm) .$$

The growth exponent of γ is defined as

$$\lim_{m\to\infty} \log\gamma m/\log m \ .$$

STANDING CONVENTION 1.3.3. All growth functions (with respect to different finite sets of generators) of a group are equivalent [ROSS76]. In particular, they have the same growth exponent. For this reason, when no confusion is possible, we shall speak of "the growth function of a group" (implying "with respect to some set of generators") and of "the growth exponent of a group" (i.e., the growth exponent of any growth function of that group).

In [MILN68], Milnor poses the following question: Is the growth function γm of a group necessarily equivalent to a power of m or to the exponential function 2^m ? In particular, is the growth exponent of γ always a well-defined integer or infinity? To date, these questions have been answered only in a fragmentary way[MILN68a, WOLF68,JUST71,BASS72]. On the other hand, a classification of groups on the basis of their growth function is important for deciding which uniform automata are physically realizable, since the growth function of a group is related to the bounded embeddability in E^d of its Cayley graph.

THEOREM 1.3.1. Let $G = \langle S,L\rangle$ be the Cayley graph of a group S . A necessary condition for G to be boundedly embeddable in E^d is that the growth exponent of S be $\leq d$.

Proof. Let ϕ be a bounded embedding of G in E^d , with upper and lower bounds respectively B and b . Given an arbitrary

node $\bar{s} \in S$, the sphere centered in $\phi\bar{s}$ and having radius Bm will contain all points ϕs such that the length of the group element $\bar{s}s^{-1}$ is $\leq m$. Thus, such a sphere will contain at least γm node images (where γ is the growth function of S). On the other hand, since the distance between any two node images in E^d is never less than b, such a sphere cannot contain more than $\frac{B^d}{b^d} m^d + Q_{d-1}(m)$ node images (where $Q_i(m)$ is a polynomial of degree $\leq i$ in m). Thus,

$$\gamma m \leq \frac{B^d}{b^d} m^d + Q_{d-1}(m) ,$$

and the growth exponent of S is $\leq d$.

Clearly, it is interesting to know whether the converse of Theorem 1.3.1 is true, i.e., whether

(CONJECTURE 1.3.1) a growth exponent $\leq d$ is a sufficient condition for bounded embeddability in E^d.

We shall prove this conjecture for a number of special classes of groups.

THEOREM 1.3.2. The free Abelian group of rank d is boundedly embeddable in $E^{d'}$, for any $d' \leq d$.

Proof. Consider the subspace E^d of $E^{d'}$, and associate in an obvious way (cf. Standing Convention 1.2.3) each element of the group with a point of E^d having integer coordinates. For the Cayley graph relative to the free generators of the group, such embedding

has bounds $b = B = 1$, and thus is bounded.

Thus, any d-dimensional cellular automaton is boundedly embeddable in $E^{d'}$ ($d' \geq d$). This justifies Definition 1.2.6 for "cellular automata" intended as prototypes of uniform systems embeddable in physical space.

THEOREM 1.3.3. If a group S is boundedly embeddable in E^d , so is the direct product SxF , where F is a finite group.

Proof. (For brevity, we shall give an informal proof.) Let n be the order of F . Given a bounded embedding ϕ of S in E^d with lower bound b , construct n copies of such embedding, respectively shifted along the same axis of E^d by amounts $0, \frac{b}{4n}, 2\frac{b}{4n}, \ldots, (n-1)\frac{b}{4n}$. Each such copy will be associated with an element of F . It is easy to verify that one obtains in this way a bounded embedding of SxF in E^d with the same upper bound and lower bound $b' = \frac{b}{2}$.

Since, in particular, a discrete Abelian group can be expressed as the direct sum of cyclic groups of prime-power or infinite order,

(COROLLARY 1.3.1) every discrete Abelian group is boundedly embeddable in E^d , for a suitable value of d .

In [HOLL 60], Holland defines "cellular automata" as uniform automata whose tessellation group is Abelian (not necessarily free Abelian). The above corollary shows that this definition stands, from the viewpoint of embeddability in physical space, in the same position as

ours. However, there is no substantial loss of generality if one restricts oneself to free Abelian groups. In fact, while the uniform automata based on free Abelian groups form a proper subset of those based on Abelian groups, the associated dynamical systems make up identical classes.

THEOREM 1.3.4. For every uniform automaton $M[C[A,S],\tau[X,\lambda]]$, where S is Abelian, there exists an equivalent cellular automaton $M'[C'[A',S'],\tau'[X',\lambda']]$.

Proof. Let $S = S_1+S_2$, where S_1 is free Abelian and S_2 is the direct sum of cyclic groups of finite order. Define $S' = S_2$, $A' = A^{S_2}$. Thus, $C' = A^{S_1+S_2} = A^S = C$. Define $X' = \{\pi_{S_1}x | x\epsilon X\}$, where $\pi_{S_1}: S \to S_1$ is a projection. Define $\lambda': A^{S_2+X} \to A^{S_2}$ in the following way:

$$\bigwedge_{s\epsilon S_2} \bigwedge_{a\epsilon A^{S_2+X}} \pi_s\lambda' a = \lambda\pi_{s+X}a , \qquad (1.3.1)$$

where $\pi_s: A^{S_2} \to A$ and $\pi_{s+X}: A^{s+X} \to A$ are projection operators. The componentwise equality (1.3.1) implies the identity of the Cartesian product of all its components, i.e., $\tau' = \tau$.

Intuitively, in the above proof, the modules of machine M' are constructed by formally grouping the modules of M into suitable supermodules whose internal structure is given by S_2 (a finite group) and whose interconnection pattern is given by S_1 (a free Abelian group). Since the structure of the whole network is not changed by this relabeling, its behavior is preserved. (This technique,

called blocking, is discussed, for instance, in [SMIT69].)

Theorem 1.3.4 can be immediately extended (the proof is identical, but multiplicative notation is preferred) to groups of the form $S = S_1 x S_2$, where S_1 is free Abelian and S_2 is an arbitrary finite group (cf. Theorem 1.3.3). It can be extended further, alongside with Theorem 1.3.3, to slightly more general products (called semi-direct products or splitting extensions[KURO55]) of two groups S_1 and S_2 as above. We do not know whether it can be extended to still more general extensions[KURO55] of a free Abelian group by a finite group.

In conclusion, we have explored but not completely delimited the range of validity of the following propositions (which thus may require further qualifications in order to be accepted as theorems).

PROPOSITION 1.3.1. A group is boundedly embeddable in E^d iff its growth exponent is $\leq d$.

PROPOSITION 1.3.2. A group is boundedly embeddable in E^d iff it is an extension of a free Abelian group of rank $\leq d$ by a finite group.

PROPOSITION 1.3.3. If a uniform automaton has a tessellation group that is boundedly embeddable in E^d, there exist an equivalent cellular automaton in no more than d dimensions.

1.4. Basic concepts and properties of cellular automata

To date, an extensive bibliography of works dealing with cellular automata and their applications would consist of a few hundred items. Those containing the most important concepts will be found cited, in the appropriate context, in the chapters of the present work. In this section, we shall recall only those concepts and properties that are necessary throughout the exposition.

Neighborhood. We shall often treat the neighborhood template as an ordered set, in order to make it easier to refer to individual elements of it, in particular, in defining the local map. When speaking of the output neighbors (as contrasted to the input neighbors) the word "output" will never be omitted. The group identity of S is a legitimate element of the neighborhood template, i.e., a cell can be counted as one of its neighbors (e.g., in the game "life" the neighborhood template strictly consists of nine elements).

There are two classes of neighborhood templates that appear frequently in the literature: the von Neumann neighborhood, which contains the identity element and those that differ from it by ± 1 in exactly one coordinate; and the Moore neighborhood, which contains the identity element and all those that differ from it by ± 1 in any number of coordinates. In two dimensions, these templates have the form, respectively, of a "cross" and a "square", as indicated below:

von Neumann

Moore

Another interesting neighborhood (used in Section 6.9) is the "speed-of-light" one, which contains only the elements which differ from the identity by ± 1 in exactly one coordinate, i.e.,

With such neighborhood, all information travels at the same speed of one cell per time step (the "speed of light").

Successors and predecessors. Let τ be the global map of a cellular automaton. The successor of a configuration c is the configuration τc. A predecessor of c is any configuration c' such that $\tau c' = c$. A configuration is Garden-of-Eden if it has no predecessors. By definition, the cellular automata that have no Garden-of-Eden configurations are those that have a surjective parallel map. A cellular automaton having a bijective parallel map is called reversible. Note that, since bijectivity implies surjectivity, reversible cellular automata have no Garden-of-Eden configurations.

A set of configurations is invariant if it is closed under τ, i.e., if it contains the successor of any of its configurations.

Trajectories. The sequence of configurations $\langle \tau^i c \rangle_{i=0}^{\infty}$ is called the trajectory originating from c. When restricted to finite confi-

gurations (cf. below), a trajectory is also called a propagation. In a reversible cellular automaton, the sequence $\overset{+\infty}{\underset{i=-\infty}{<}} \tau^i c>$, which is uniquely defined for every c , is called the orbit through c

Finite configurations. A cell-state $q \in A$ is quiescent if $q = \lambda \overset{n}{\underset{i=1}{<}} q>$ (where n is the number of neighbors and λ the local map). A cellular automaton is stable (cf. [BURK71]) if it admits of a quiescent state. If a cellular automaton M is stable, one may select a particular quiescent state as the blank state. A configuration is finite if it contains finitely-many cells in a nonblank state. (The blank configuration has all of its cells in the blank state.)

Owing to the finiteness of the neighborhood template, it is clear that the set C_f of finite configurations (for a cellular automaton where this concept is meaningful) is invariant. Thus, one may associate with the cellular automaton the dynamical system $<C_f,\tau>$ (instead of $<C,\tau>$) and concentrate one's attention on this system. Much of the early emphasis and terminology for cellular automata evolved in this context. Our dropping the requirement that a cellular automaton should have a quiescent state by definition does not imply that we intend to ignore the important role of quiescency in cellular automata. On the contrary, we show that by slightly generalizing the concept, every cellular automaton can be shown to have at least one "stable state" (Theorem 1.5.2). Such generalization preserves the algorithmic [BURK71] properties

of finite configurations (cf. Alternative Definition 1.5.1). Note that the set of finite configurations is countable. Thus, the study of the system $\langle C_f, \tau \rangle$ constitutes a link between cellular-automaton theory and recursive-function theory (cf. Section 1.1).

In the context of cellular automata having a blank state, the term "configuration" is often used informally to denote the pattern formed by the nonblank cells. Thus, one says that "a configuration constructs a copy of itself" if, starting from a configuration c containing a certain pattern, one eventually finds, in the trajectory originating from c , a configuration c' containing two copies of that pattern.

Computation and construction. In the context of the theory of computing, the reductionistic program exemplified by Turing machines is carried a step further in cellular automata, where passive (storage) and active (computing) functions can be carried out starting from simple, "undifferentiated" primitives connected in a uniform way. In fact, in appropriate cellular automata, by suitably assigning the initial state of a certain number of cells it is possible to specify structures that carry out recognizable computing and constructing tasks. It is well known[VONN66,CODD68] that there exist cellular automata that are computation- and construction-universal; briefly, it is possible to embed in such automata a structure that behaves as a universal Turing machine, and a structure that, on the basis of an encoded description, can

construct a variety of other structures, among which a copy of the original structure (inclusive of the description "tape") and a universal Turing machine. The relevance of such properties is amply discussed in [BURK70].

Simulations and embeddings. There are in the literature many examples of cellular automata that "simulate" other cellular automata or in which other cellular automata can be "embedded" (cf. [SMIT69]). The precise meaning of such terms varies depending on the context and the intended applications, and one must admit that there is a certain amount of confusion in the field. By recognizing that a cellular automaton is but a presentation of a dynamical system (Section 1.1), we introduce a way to clarify many issues. In fact, many cellular-automaton "simulations," "blockings," "speed-ups," etc. can be seen as alternative presentations of the same dynamical system corresponding to using different systems of spatial, temporal, or state-variable coordinates (cf. Sections 1.6, 6.9).

1.5 Physical-like aspects of the mechanics of cellular automata

We have shown, in Sections 1.1 and 1.3, that certain features of cellular automata can be interpreted quite naturally, in analogy with physics, in terms of spatial and temporal categories. (We may even say that such categories were deliberately introduced by defining cellular automata in a certain way.) Is it possible to extend further the analogy with physics?

Our question here is not whether cellular automata can be used as discrete computational models of particular physical systems. In fact, since there exist computation-universal cellular automata, such models can clearly be constructed to any degree of sophistication. Rather, we ask to what extent one can draw a parallel between the laws that govern the behavior of a cellular automaton and the general laws of physics, i.e., of those laws (admittedly, imperfectly known) that are presumed to be valid exactly for all physical systems under any set of initial conditions.

As an introduction, we shall discuss from the viewpoint of cellular automata two situations ("uniform wave" and "isolated particle") that involve particularly simple initial conditions, i.e., conditions that are specified by independently assigning the state of only a finite number of cells. For a uniform wave, one requires that such assignment be iterated in a regular fashion over the whole cell array; for an isolated particle, all cells whose state is not explicitly assigned are assumed to be in one and the same state.

In the following discussion, the nomenclature will implicitly refer to a cellular automaton of the form $M[C[A,S],\tau[X,\lambda]]$.

Intuitively, a wave is a phenomenon characterized by a definite spatial and temporal periodicity. In a cellular automaton, spatial periodicity is described in terms of a subgroup of the tessellation group S. We are interested in subgroups G such that $F=S/G$ is finite. Such subgroups, which have the same rank as S, will be called scaled subgroups of S. Informally, the cells of M, which are arranged in a regular pattern described by S, can be thought of a grouped into identical finite blocks; correspondingly, G describes the repetitive pattern formed by such blocks, while F describes the regular arrangement of cells within each block. More formally, if G is a scaled subgroup of S, S is a split extension of G by F (cf. [KURO55]); while S cannot, in general, be written as the direct product of G and F, yet, by choosing an arbitrary representative from each coset of S, every element $s\varepsilon S$ can be uniquely expressed as an ordered pair $s = \langle g,f\rangle$ (where $g\varepsilon G$ and $f\varepsilon F$) with the property that $\langle g,f\rangle + \langle g',f'\rangle = \langle g'',f+f'\rangle$. A configuration is "uniform" if in all blocks cells that occupy the same relative position are in the same state. Formally,

(DEFINITION 1.5.1) a configuration $c\varepsilon C$ is uniform with respect to a subgroup G of S if, for all $s\varepsilon G$, $sc = c$. If G is a scaled subgroup of S, c is finitely uniform.

The temporal periodicity of a wave is expressed in the following terms:

(DEFINITION 1.5.2) a configuration c is recurrent with period t if $\tau^t c = c$.

According to the following theorem, the equation $c^{i+1} = \tau c^i$ which characterizes the behavior of a cellular automaton always admits of solutions consisting of an oscillatory mode having a well-determined period and displaying a repetitive spatial pattern based on a finite template.

THEOREM 1.5.1. Given any scaled subgroup G of S , there exist a recurrent configuration that is uniform with respect to G .

Proof. Consider the uniform automaton M' obtained from M by identifying any two cells s , s' such that $s' = gs$, where $g \in G$. (Intuitively, this a wrapped-around version of M such that opposite edges of a block meet.) The labeling schema $s = \langle f,g \rangle$ defined above introduces a one-to-one correspondence between the configurations of M and those of M' that are uniform with respect to G . Such correspondence is preserved under τ . M' is a finite, input-free automaton; starting from any state, M' eventually enters a cycle of finite length t . Thus, for any state in this cycle, the corresponding configuration of M is recurrent with period t .

Note that there exists at least one such oscillatory mode for

any scaled subgroup G of S . In particular, when G coincides with S itself (and thus the blocks coincide with the individual cells), one obtains oscillatory modes in which, at any time step, all cells are in the same state. In such a situation, if one looked at the cellular automaton only at intervals having the same length t as the oscillation period, one would always see the same configuration (in other words, the equation $c^{i+t} = \tau^t c^i$ admits of constant solutions). Thus, anticipating a concept that will be formally defined in the next section, we have the following

THEOREM 1.5.2. Every cellular automaton is equivalent, up to a time-scale factor, to one having a quiescent state.

The above theorem shows that the customary defining stipulation that a "cellular automaton" have a quiescent state (cf., e.g., [VONN66,MOOR64,YAMA69]), is substantially redundant. In particular, all arguments that use the concept of finite configuration (Section 1.4) could be reworded in terms of the following slightly more general definition which applies to arbitrary cellular automata:

ALTERNATIVE DEFINITION 1.5.1. A configuration is finite if it can be obtained from a finitely uniform configuration by reassigning in an arbitrary way the state of a finite number of cells.

Finally, if M is reversible, every finitely uniform configuration is recurrent (cf. the proof of Theorem 1.5.1) and can be used, in the

sense of Theorem 1.5.2, as a quiescent configuration.

It is always possible to interpret an oscillatory mode that cycles through uniform configurations as a "wave" having a certain phase velocity, according to the following

(DEFINITION 1.5.3) a sequence of finitely uniform configurations $\langle c^i \rangle_{i=0}^{\infty}$ is a plane wave with phase velocity $v = \frac{s}{t}$ (note that v, as well as s, is a vectorial quantity, according to Standing Convention 1.2.3) if, for all i, $c^{i+1} = \tau c^i$ and $\tau^t c^i = s c^i$.

However, in this context, it does not make sense to speak of traveling waves having a well-defined group velocity, in analogy with waves in a continuous linear medium, until one agrees on a way of modulating such waves and determining whether the corresponding information is coherently transported in a certain direction--instead of being, for instance, dispersed or destroyed. (The case of an isolated particle, discussed next, can be interpreted as a special case of coherent transport of information by a modulated wave.)

In the study of physics, waves play two important roles. On one hand, independently of the linearity of the given problem, spatially periodic initial conditions constitute a special case in which an infinite system can be treated as if it were finite (cf. the wrap-around technique in the proof of Theorem 1.4.1). The corresponding solutions of the problem are usually much easier to derive than in the infinite case. On the other hand, if the problem involves linear differential equations of a kind very common in

physics, spatially periodic solutions constitute a basis (in the vector-space sense) for all solutions, owing to the superposition principle. Thus, by considering a sufficiently large number of periodic components, one can approximate the solution of the problem for any set of initial conditions. While, as discussed above, there is an immediate analogy in cellular automata for waves used in the former role, a satisfactory analogy for waves used in the latter role can be obtained only in cellular automata for which an appropriate form of the superposition principle holds. So far, our investigations in this direction have been very limited.

Let us return, for a moment, to the evolution in time of a recurrent, finitely uniform configuration. What happens if the state of a small number of cells in the initial configuration is perturbed in an arbitrary way? Substantially, one of the following three cases applies:

(i) The perturbation is reabsorbed, in the sense that the regular oscillatory pattern eventually reimposes itself throughout the whole cell array.

(ii) The perturbation diverges, in the sense that, as time passes, it spreads to more and more cells.

(iii) The perturbation eventually enters a steady state, in the sense that, while it may oscillate and/or move in a certain direction at a constant speed, it does not spread or die out.

The third case is clearly analogous to that of an isolated particle in physics.

(DEFINITION 1.5.4) a finite configuration c is called an (isolated) particle if, for some $s \in S$ and $t \in N$, $\tau^t c = sc$. The vectorial quantity $v = \frac{s}{t}$ is called the velocity of the particle.

Informally, one may speak of the size of a particle, i.e., the number of cells that are not in the quiescent state. Note, however, that this number may cyclically vary with period t. An example of an isolated particle is the "glider" in the game "life"[GARD70]. Other examples can be found in Chapter 4. The concept of particle and of collision between particles in cellular automata has been variously investigated by Zuse, Balzer, and Pilgrim, who have noted several analogies and differences with respect to the physical case[ZUSE67,BALZ67,PILG72]. It is clearly difficult to derive nontrivial properties for particles without first restricting one's attention to specific classes of cellular automata. However, one very general property should be noted that is neither trivial nor immediately evident.

For simplicity, consider a cellular automaton with the von Neumann neighborhood $\langle -1,0,1 \rangle$. In such an automaton, the maximum speed of transmission of information (the "speed of light") is one cell per time step. Consider a parallel map capable of supporting a variety of particles with different velocities (such maps can be easily constructed with techniques analogous to those used in solving the firing-squad problem). Such velocities are in any case rational numbers between 0 and 1 (we shall ignore the sign). Let s and

t be the smallest integers such that $v = \frac{s}{t}$. Intuitively, a particle having velocity v must contain a timing mechanism, i.e., a mechanism capable of counting off the integers s and t. Large values of s and/or t require a mechanism having a large number of states, and thus are compatible only with particles consisting of a large number of cells. In order to construct a particle whose velocity v differs from an assigned real value $\bar{v}$ by less than a given tolerance Δv, one is forced to select for the denominator t (and, consequently, for the particle itself) a size that is, roughly speaking, inversely proportional to the tolerance Δv. Therefore, a very precise specification of the speed of a particle entails in general a very imprecise localization of the particle itself. This property has a definite quantum-mechanical "flavor."

It should be clear from the above discussion that uniform waves and isolated particles, which are fundamental tools of descriptive physics (they are still, after all, essentially kinematic concepts), are available as well in cellular automata theory, in a form adapted to the discrete nature of such systems. In order to study specific dynamical analogies between cellular automata and physics, one would have to restrict one's attention to special classes of cellular automata.

In our opinion, it is too early to try to synthesize cellular automata that would satisfactorily mimic specialized features of physics, such as the structure of atoms. Nor does it appear fruitful, at least in a theoretically oriented research, to try to

summarily adapt the continuous schemata of classical mechanics to the discrete structure of cellular automata by using fixed-length integer registers for real variables, as done, for instance, by Balzer[BALZ67] (who, however, had more practical applications in mind). In Balzer's paper, cells contain finite-state registers that record the mass and the velocity of a particle. The free motion of a particle is represented by transferring at appropriate time intervals the contents of a cell to an adjacent one; in a collision, the registers of the colliding particles exchange information in a way that approximates the laws of conservation of energy and linear momentum. The schema attempts to reproduce Newtonian mechanics, but, since the registers may end up in an overflow condition, the resulting mechanics presents very odd discontinuities. Probably, in order to represent "elementary" particles with properties analogous to those encountered in physics it will be necessary to use aggregates containing a large number of cells. For the same particle, such number may vary according to the dynamical state of the particle. Intuitively, this corresponds to having variable-length registers. This approach would certainly produce particles that are "unconventional" from the classical viewpoint and whose behavior may display features reminiscent of relativity and quantum-mechanics. In any case, it is clear that, in the study of cellular automata mechanics, it is necessary to introduce relativistic considerations if one intends to consider particles of small size, which, as we have noted, must be either at rest or moving at a speed that is a

substantial fraction of the speed of light (cf. Section 6.8).

At the present stage, it is probably more fruitful to explore cellular-automaton analogies starting from general concepts of theoretical physics, rather than to pursue more specialized analogies. We shall briefly list some of the most obvious possibilities.

Homogeneity of space and time have a counterpart in the uniformity (with respect to the tessellation group) and the time-invariance of the laws that govern cellular automata.
The law of inertia is an immediate consequence of these properties. Consider an arbitrary configuration c, and suppose that, for some t and s, $\tau^t = sc$ (i.e., configuration c, after possibly undergoing a number of changes, reappears, after a time t, shifted by an amount s with respect to the original position). Then, for every $n \in N$, $\tau^{nt} = (ns)c$. Informally, if the configuration is moving at a speed $v = \frac{s}{t}$ at a certain moment, it will keep moving at that speed. The law of inertia applies, in particular, to an isolated particle (Definition 1.5.4). (Compare the above argument with [LAND60, p.5], where the law of inertia in physics is derived starting from the uniformity of space and time using the Lagrange principle.)

Cellular automata do not possess, in general, rotational invariance, or isotropy, which, in physics, seems to hold strictly (unlike mirror-reflection invariance, for which there are exceptions). However, Codd showed that the most conspicuous capabilities of cellular automata (namely, computation and construction universality) are preserved if one imposes the isotropy constraint[CODD68].

In Chapter 3, we show that such capabilities are preserved if one imposes reversibility of the parallel map. It should be noted that reversibility stands behind practically every conservation or variational principle of theoretical physics, and that empirical evidence is in favor of the strict reversibility of all physical phenomena on a microscopic scale.

Another feature that is encountered very frequently in physics is linearity, and certainly the study of linear cellular automata leads to many interesting applications[BART75]. However, from a microscopic viewpoint the physical world seems to be inherently nonlinear (though linear theories often supply convenient first-approximation models[ROBI77]). In cellular automata, linearity entails the loss of computation and construction universality[MEYE71]. Though the latter properties do not have an obvious formal counterpart in traditional physical theories, they seem to be somehow intimately connected with the nature of the physical world. For these reasons, we are of the opinion that imposing the constraint of (strict) linearity would result in cellular automata that are too weak to lead to satisfactory analogies with physics as a whole.

In spite of the discrete nature of the spatial and temporal coordinates in cellular automata (cf. Ulam's misgivings for finite systems[ULAM50]), it is possible to define for them space-time transformations analogous to the Lorentz transformations of special relativity (cf. Section 6.8). It is still open to investigation to determine whether, in correspondence with these coordinate

transformations, there are, at least for certain cellular automata, transformations of the dynamical variables that preserve the parallel map.

In classical physics, every degree of freedom is associated with a pair of dynamical variables (e.g., position and momentum, electric and magnetic field) that play a symmetrical role in the Hamiltonian formulation. The same pairs play a distinguished role in quantum mechanics. It is not clear where one should look for analogies of this duality in cellular automata. In this section, we have mentioned a certain complementarity between velocity and position determination for particles. Another possible analogy is suggested by the duplicate encoding of information introduced for synchronization purposes in Section 2.4.2.

In Section 3.6, we note that the embedding of an "intelligent" observer in a reversible cellular automaton may lead to a description of the "world," from the viewpoint of such an observer, that is affected by an indeterminacy comparable to that of quantum mechanics.

Finally, as noted also in Section 3.6, there is evidence that certain thermodynamical concepts may have a meaningful counterpart in the study of cellular automata.

In conclusion, it seems that cellular automata, while not necessarily the most appropriate tool for quantitative modeling, may represent a versatile playground for conceptual experimentation and qualitative modeling of the physical world.

1.6 Coordinate transformations

As soon as one considers using coordinates to describe a dynamical system (cf. Section 1.1), the problem arises of determining how the description changes when the system of coordinates itself is modified. Cellular automata are descriptions of dynamical systems. In fact, as explained in Section 1.2, there is exactly one (up to an isomorphism) dynamical system associated with each cellular automaton. In cellular automata, two "levels" of coordinates are used. On one hand, the set C of configurations is defined as the Cartesian product of a countable family (indexed by the tessellation group S) of identical copies of the state alphabet A. Thus, the state variables c_s ($s \in S$) make up a system of coordinates for C. Each of these coordinates has a finite range. On the other hand, the index set S is itself the Cartesian product of copies of the infinite cyclic group C_∞. Thus, each element of S can be thought of as a point in a d-dimensional space having coordinates $s_1,\ldots,s_d$. Each of these coordinates has a countable range. In summary, one has the following situation:

(i) Spatial coordinates. For a cell $s \in S$,

$$s = \prod_{i=1}^{d} < s_i > , \qquad (s_i \in C)$$

(a finite set of coordinates, each having countable range).

(ii) Dynamical coordinates. For a configuration $c \varepsilon C$,

$$c = \underset{s \varepsilon S}{< c_s >} , \qquad (c_s \varepsilon A)$$

(a countable set of coordinates, each having finite range).

In this section, we shall briefly review a number of cases of the following problem: Given the dynamical system associated with a cellular automaton, to determine how the local formulation of the dynamical laws (i.e., the local map together with the neighborhood template) and the dynamical coordinates transform in correspondence with a linear transformation of the spatial coordinates. (In general, this problem is ill-defined for nonlinear transformations, since it leads to a nonuniform description incompatible with the cellular-automaton schema.) The notation will refer to two cellular automata, M and M' , where M is the description of a dynamical system before the coordinate transformation and M' the description after the transformation. Note that, since the number of dimensions is not changed by a linear transformation, S' is isomorphic to S . By $D(M)$ we shall denote the dynamical system associated with M .

Uniform translation of coordinates. Consider a coordinate transformation of the form

$$s' = s + \overline{s} ,$$

(where $\bar{s}$ is an assigned element of S). It is clear that the transformations

$$\lambda' = \lambda ,$$

$$X' = X , \text{ and}$$

$$c'_s = c_{s-\bar{s}}$$

define an isomorphism between $D(M)$ and $D(M')$. Informally, under a uniform coordinate shift the dynamical laws are unchanged, and the value of the coordinates in the description of a configuration is affected by a shift of the same magnitude but opposite sign.

Scaling by an integer factor (also called blocking [SMIT69, YAMA71]). We shall sketch the general procedure with reference to the two-dimensional case. Consider the cell array and group the cells into adjacent blocks of size $m \times n$, as illustrated in Figure 1.2a (this corresponds to a coordinate transformation of the form $s'_i = \frac{1}{n_i} s_i$, where the n_i $(i=1,\ldots,d)$ are integers). In order to preserve the behavior of $D(M)$ in $D(M')$, i.e., using a cellular automaton having cells as in Figure 1.2b, each cell of M' must encode the contents of $m \times n$ cells of M, i.e., $A' = A^{mn}$. The new neighborhood template X' as well as the new local map λ' are constructed in an obvious way, by appropriately "lumping" the old ones. In particular, the Moore and the von Neumann templates are invariant with respect to this transformation. Note that, while the underlying dynamical system does not change, its description in cellular-automaton terms (in particular,

the local map) is substantially altered.

Fractional scaling. For simplicity, we shall consider only the one-dimensional case, but the extension to an arbitrary number of dimensions is immediate. Consider cellular automata M and $\overline{M}$, and suppose that they are transformed into M' by transformations of the form

$$s' = \frac{1}{n} s \qquad \text{and} \qquad s' = \frac{1}{m} \overline{s} \quad .$$

Then the coordinate transformation $\overline{s} = \frac{m}{n} s$ transforms M into $\overline{M}$. Thus, it is meaningful to speak of fractional scaling in the context of cellular automata, though such operation is not defined for all cellular automata.

Pure rotations and reflections. By permuting the coordinate axes (i.e., the generators of S) and suitably reordering the elements of the neighborhood template, the entries of the table that defines the local map, and the subscripts of the dynamical variables of the form c_s, one obtains cellular-automaton transformations corresponding to rotations (for even permutations) and rotation-reflections (for odd permutations).

Fractional rotations. It is easy to verify that fractional rotations (i.e., by angles that have a rational tangent) in combination with suitable scalings are always well defined for

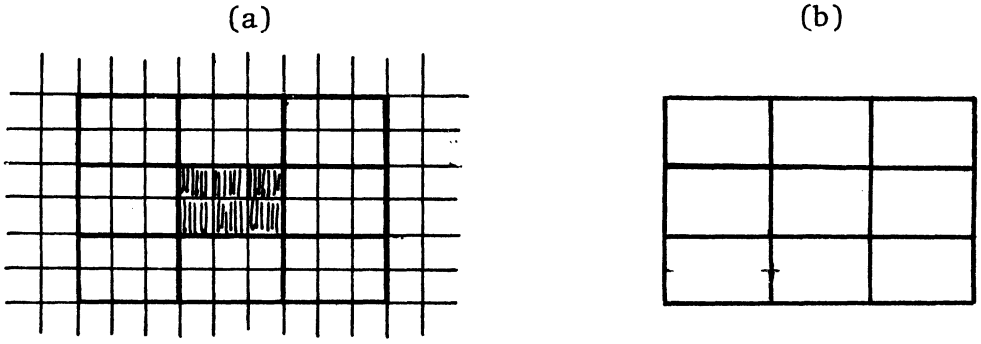


FIGURE 1.2. First the cells are grouped in blocks of size m n (like the shaded one in (a)). The Cartesian product of the state sets of these m n cells forms the state set of one cell in the new cellular automaton (b)

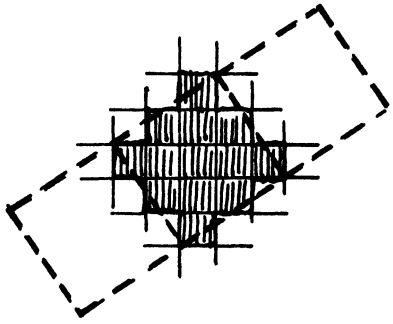


FIGURE 1.3. In a similar way, one can group cells into other repetitive patterns. Each block of such a pattern (shaded in figure) becomes a cell of a new cellular automaton (whose cells are suggested by the dashed lines).

cellular automata. Figure 1.3 illustrates a rotation by an angle whose tangent is $\frac{2}{3}$.

From the above considerations, it is evident that, if one compares the algebra of linear coordinate transformations for cellular automata with that for continuous mechanical systems, one encounters a number of limitations--which are substantially analogous to those encountered when one attempts to perform on the integers the same algebraic operations that are well-defined for real numbers. Most of the operations are still meaningful (and relevant in the context of many applications), but they are not defined in all cases.

In contrast to many other kinds of "simulation" of a cellular automaton by another (cf. [SMIT69]), the above transformations preserve the identity of the underlying dynamical system. Another transformation that substantially satisfies this property is time scaling. Given a dynamical system $U = \langle C,\tau \rangle$, consider the system $U' = \langle C,\tau^n \rangle$, where n is a positive integer. (Note that the operation that leads from U to U' is not a form of homomorphic modeling.) Intuitively, U' carries out the same task as U but is open for inspection only at longer intervals. Given a cellular-automaton description M for U, the construction of M' such that $U' = D(M')$ is immediate. We shall call U' isomorphic to U up to a time-scale factor n, and correspondingly, we shall call M' equivalent to M up to a time-scale factor. Time scaling is a linear transformation of coordinates, in a space-time representation of a

dynamical system. In such a representation (discussed, with reference to cellular automata, in Chapter 6) it is possible to define mixed linear transformations (i.e., ones involving both space and time coordinates) analogous to the Lorentz transformations of physics. Also these transformations preserve the identity of the underlying dynamical system. In general, one can say that the whole class of nonsingular linear transformations of spatial and temporal coordinates is meaningful in the context of cellular automata, though with certain restrictions related to the discrete nature of space and time in such structures.

1.7 Integration of the equations of motion

Cellular automata essentially constitute a finitary analog of differential equations of the form

$$\frac{\partial q}{\partial t} = f(q, \frac{\partial q}{\partial x}, \frac{\partial^2 q}{\partial x^2}, \ldots) \qquad (1.7.1)$$

(only one spatial dimension has been indicated, for simplicity) whose solutions describe how a spatially distributed quantity $q(x,y,z;t)$ evolves in time according to laws that are local and uniform. In order to express the solutions of (1.7.1), or trajectories, as explicit functions of time, one may attempt symbolic integration. In general, though, since no composition of elementary or other well-known functions may satisfy (1.7.1), one must make recourse to numerical integration, with the unavoidable truncation and round-off errors that this entails. Thus, both integration methods suffer from severe limitations. The first gives exact, global solutions, but is only applicable to a small class of equations, while the second is of general applicability, but yields local solutions, i.e., solutions that maintain a sufficiently good approximation only within a neighborhood of the point which represents the initial conditions.

Given the importance of studying the trajectories of certain

physical systems, various attempts have been made to overcome--or, at least, circumvent--the above limitations.

The British School. A most obvious line of attack is to try to extend the range of symbolic integration methods. In the nineteenth century, the so called British school undertook a systematic effort aimed at enlarging the class of "well-known" functions. In particular, they studied many special functions which are implicitly defined by differential equations arising from certain physical models. The corpus of information collected in this way soon grew to an unmanageable size. According to Moses, "the success of numerical analysis and the advent of digital computers have dealt a crushing blow to whatever remains of the programme of the British school. [However], ... it now appears possible that a programme similar to the British school's could be developed which avoids many of its difficulties. Instead of studying the particular properties of a limited class of functions, one would develop algorithms for generating properties directly from the differential equations that define them!"[MOSE72]

Poincaré and Birkhoff. In the sense indicated by Moses' suggestion, the British school's program for symbolic integration is somehow related to a different approach to the study of differential equations, pioneered by Poincaré[POIN80]. Given a differential equation associated with a physical system, in many applications

one is content with the knowledge of certain qualitative properties of its solutions (e.g., their behavior at infinity or in the vicinity of a singular point; whether the solution through a given point is bounded, or asymptotic to a known curve; etc.). To this purpose, it is convenient to globally consider the space of all possible system configurations and, in this space, the lines that make up the trajectories. The resulting picture is one of a continuous flow, in many ways analogous to that of a fluid. Not only is it interesting to determine how certain features of a flow (fixed points, sources and sinks, shear surfaces, vortices, etc.) are related to the form of the defining differential equations, but this is often more meaningful than considering individual solutions. Such an approach suggested the prototype of many concepts that were eventually more fully developed in topology, functional analysis, numerical analysis, and topological dynamics. The latter, in particular, owes much to the systematic efforts of Birkhoff[BIRK27]. More recently, such concepts have been extended to more general flows defined without making reference to differential equations. In particular, symbolic flows, or shift dynamical systems[HEDL69,SEAR73], are concerned with dynamical systems that are very close to cellular automata.

Discrete Mechanics. What Labudde and Greenspan (LG in what follows) unpretentiously call "a new numerical method"[LABU74] actually constitutes a definite departure from the conventional

approach. Traditionally, people have tried to represent a system as faithfully as possible by means of differential equations, and then, confronted with the difficulties of solving the latter, have often been forced to use very crude means for constructing approximate trajectories. In LG, the decision to approximate is taken right at the outstart, at the level of system representation rather than that of trajectory evaluation. This is done by deliberately associating with the original system, by means of suitable correspondence rules, one that is manifestly different--being intrinsically time-discrete. The recursion formulas that describe the new system are purely algebraic--they do not imply analytic or "infinitesimal" processes-- and, whether or not one can find a symbolic integral for them, they can be integrated exactly by numerical methods. Of course, the trajectories constructed in this way for the new system do not pretend to exactly duplicate those of the original system (though it should be noted that they converge to these in the limit $\Delta t \to 0$, where Δt is the elementary time interval for the discrete system). However, unlike customary numerical methods, which only indirectly and approximately reflect the physics of the situation, the recursion formulas derived by LG autonomously guarantee the physical consistency of the new trajectories (e.g., their satisfying certain conservation principles). For this reason, even using a large value of Δt many of the qualitative properties of the old system are preserved, and no degradation of the results occurs if the numerical integration is

protracted for a large number of time steps. In particular, the LG integration method is exactly, rather than only approximately, reversible.

In brief, instead of deriving approximate numerical solutions for a presumedly exact mathematical formulation of a physical system, in LG one derives exact solutions for an approximate--or, so to speak, "stylized"--version of the same system. In practice, though, the achievement of even the latter kind of exactness presents certain difficulties. While, by iterating the recursion formula, the value of the state variables for each point of the trajectory can be expressed as a function of the initial conditions by means of an explicit algebraic expression, such expression grows more and more complex as one moves away from the initial point, and there is no systematic way to simplify it (or, for that matter, to decide whether two such expressions have the same value). For this reason, in the applications the expression is numerically evaluated at each time step. Thus, while truncation errors are completely eliminated in the LG method, round-off errors are introduced at the expression-evaluation stage.

Cellular automata, which, in addition to discrete time, also feature discrete space and, for each point, a state variable ranging over a finite set, may be conceived as a further step in the direction outlined by LG, insofar as, by restricting the range of mathematical tools that are offered, they allow one to perform what is

essentially a form of numerical integration without accumulating round-off errors. In fact, the local map (which, as a recursive schema, replaces differential schema (1.7.1)) always produces a result that belongs to the same finite set as the input arguments. Thus, in the numerical computation of a trajectory, one can associate with each cell a fixed-length register in which, at any moment, the state of the cell is exactly encoded. If the initial state of the given system is represented by a finite configuration, every point of the system's trajectory will still be represented by a finite configuration, and, correspondingly, only a finite number of registers will be needed.

It must be noted that, unless one forces appropriate boundary conditions, the number of nonquiescent cells may increase as the system evolves along its trajectory. However, the rate of such increase is at most polynomial. Intuitively, this corresponds to the fact that at any moment any portion of the system may generate signals that travel outwards at a speed bounded by that of light. If such signals are "weak," or are not expected to be reflected back into the system, they can be ignored (this corresponds to imposing "absorbing" boundaries).

Since, in the present context, cellular automata associate a bounded amount of information with each volume element, their adequacy as surrogates of partial differential equations in physical modeling will ultimately depend on whether or not, under given initial conditions, physical systems are capable of storing an

unbounded amount of information in a given volume. The reader should be warned that this is a rather vague formulation of a difficult problem; in particular, there are quantum-mechanical limitations to the freedom with which initial conditions can be assigned and final conditions ascertained.

In conclusion, cellular automata are recursive schemata analogous to certain differential schemata which are fundamental in the formulation of the physical laws. In contrast to differential schemata, their numerical integration can be carried out exactly. Moreover, such integration process can be carried out in an extremely efficient way by parallel-computing mechanisms (cf. Chapter 2) which distribute in a natural way over physical space and time the abstract "space" and "time" of the recursive schema. The naturalness of this correspondence between abstract and physical features recommend cellular automata as an important paradigm in the theory of physical modeling.

CHAPTER 2

PHYSICAL IMPLEMENTABILITY

2.0 Preliminaries

A finite automaton can be described by means of a table which for each input/state pair specifies the corresponding next state. More conveniently, it can be expressed as a finite sequential network whose delay elements collectively encode the state of the given automaton and whose combinatorial elements collectively realize its transition function. In principle, it is always possible to implement a sequential network as an electronic circuit (or other physical apparatus), by systematically associating a suitable "gate" to each combinatorial element, a "flip-flop" to each delay, and an interconnecting line to each arc of the network. Once such functional correspondence is established, there is still enough freedom left in selecting the spatial arrangement of components (including the routing of the interconnecting lines) and the temporal parameters (e.g., the clocking rate) to permit taking into account technological limitations and, in any case, satisfying geometrical and physical constraints of a general nature.

Cellular automata cannot be expressed as finite sequential

networks. Their "natural" representation, and the only one with which we shall concern ourselves, is as infinite, uniform sequential networks, for which, of course, an effective physical implementation is not operationally meaningful. The real interest of such networks does not lie in their being actually infinite, but in displaying uniformity of structure over finite portions of any size (note the analogy with the tape of a Turing machine).

While there do not seem to be any intrinsical physical limitations (aside, of course, from cosmological ones, like the conceivably finite size of the universe) to the number of network elements one can implement, it is clear that, as long as one is interested in networks of arbitrary structure, design parameters cannot be specified independently of network size. (For instance, since electronic devices occupy a finite volume, as one constructs larger and larger networks it is not possible to keep all nodes within a fixed minimum distance from one another, and the clocking rate must be adjusted according to the maximum propagation delay.) However, the networks that represent cellular automata have a special structure which seems particularly favorable to large-scale implementation, since their interconnections are local and uniform. Is it possible to use size-independent design parameters in this case? In other words, is it possible to design once and for all a cell-module prototype such that an implementation of arbitrary size can be constructed by simply juxtaposing a sufficiently large number of such modules? This is equivalent to requiring that the uniformity precept

strictly hold not only for the switching functions associated with the network proper, but also for all auxiliary functions such as signal transmission, synchronization, power distribution, heat removal, initialization and read-out, etc. We shall call uniform a physical implementation schema (briefly, an implementation) that satisfies such conditions. In the following sections, we shall clarify under what assumptions and for what conditions cellular automata admit of a uniform physical implementation.

We shall restrict our attention to implementation schemata in which the tessellation group S is associated with physical space and the dynamical semigroup $\{\tau^i\}_{i=0}^{\infty}$ is associated with physical time.

REMARK 2.0.1. From a general standpoint, one should not rule out a priori the possibility of representing the tessellation group over nonspatial categories. For instance, one may think of associating cells with modes of vibrations of a finite cavity, cell-states with the energy levels of such modes, and the local map with some coupling between such modes. In this case, however, aside from the difficulty of realizing the required uniform coupling between modes, one would have to employ quanta of arbitrarily high energy as the number of implemented cells (and thus the frequency of the modes) increases. With respect to such high-energy quanta, the very abstraction of a cavity with well-defined smooth, elastic walls ceases to have a physical counterpart. Similar difficulties arise whenever one tries to constrain an unboundedly large amount of information in a finite volume (cf. next section).

2.1 Geometrical considerations

The quantitative considerations that follow are developed, for brevity, in an approximate fashion. Their purpose is to determine the physical dimensions and the orders of magnitude of certain limiting quantities, not their exact value.

Consider a physical implementation of a cellular automaton. Let n be the number of bits required to encode the state of a cell ($n \simeq \log_2 r$, where r is the size of the state alphabet); ℓ the distance between neighboring cells (thus, ℓ^3 is the volume of a cell); and t the clock interval corresponding to one time step. If the implementation is not uniform, ℓ and t may vary from cell to cell (the correct behavior can be maintained in spite of the variation of t ; cf. Section 2.4.1). However, in order to have a nonvanishing rate for the process of going from one configuration to the next for the whole array, t must be bounded by a maximum value $\overline{t}$. Each cell will process information at a rate $u = \frac{n}{t}$. According to [BREM67], such rate cannot exceed $\frac{mc^2}{h}$, where m is the total mass of the cell (the mass of the particles making up its structure plus the mass-equivalent of the energy in which signals are encoded), c is the speed of light, and h is Planck's constant. The above limitation derives from quantum-mechanical constraints. Let p $(= \frac{mc^2}{\ell^3})$ be the energy pressure in the cell. From the above limitation one obtains

$$p > nh/\ell^3\overline{t}. \qquad (2.1.1)$$

If, for technological reasons, one imposes an upper bound $\bar{p}$ on the pressure in the cells, one obtains from (2.1.1) a lower bound $\bar{\ell}$ for the diameter of a cell as follows

$$\bar{\ell} = \sqrt[3]{nh/\bar{p}\bar{t}} \quad . \tag{2.1.2}$$

On the other hand, in a time interval $\bar{t}$ signals can travel only a distance $c\bar{t}$. Thus, one has an upper bound

$$\bar{\bar{\ell}} = c\bar{t} \tag{2.1.3}$$

for the distance between neighboring cells. Conditions (2.1.2) and (2.1.3) are the physical counterpart of the two conditions that define bounded embeddability (Section 1.3). As explained in Section 1.3, the tessellation group of a d-dimensional cellular automaton is boundedly embeddable in physical space (identified with the manifold E^3) only for $d \leq 3$.

In conclusion, the requirements of bounded pressure and bounded clock rate (which must be satisfied in order to have a uniform implementation, as defined in Section 2.0) impose the limit $d \geq 3$ on the number of dimensions of a physically implementable cellular automaton.

2.2 Thermodynamical considerations. I

A further restriction on the number of dimensions of a uniformly implementable cellular automaton may be proposed on thermodynamical grounds. An amount of free energy of at least $\log_e 2kT$ is spent and an equal amount of thermal energy is released whenever one bit of information encoded in a physical system kept at temperature T is irreversibly erased[LAND61]. In cellular automata, which are generally irreversible, information erasure may take place at any time step. For a state-alphabet of size n, the maximum amount of information that can be encoded in a configuration is $\log_2 n$ bits per cell (cf. [ADLE65] for a precise definition of this concept for infinite configurations). In a hypothetical "worst case" where all cells erase at each time step that amount of information (or any amount greater than a fixed minimum) one would have to remove from each cell a steady flow of heat in order to maintain the system at a constant temperature, and supply each cell with a steady flow of free energy in order to preserve its functionality.

Let us consider the heat-removal problem first. In the above worst case, the thermal power generated by the information-erasing process in a large uniform implementation of a cellular automaton would be proportional to its volume (i.e., to the number of cells). On the other hand, the power lost by thermal irradiation would be

proportional to its effective surface (i.e., the surface not shaded by the other portions of the body). Thus, in order to achieve thermal equilibrium at a given temperature independent of size, the implementation must have a surface proportional to its volume. This implies that the uniform network cannot be more than two-dimensional.

How can one supply the required free energy necessary to keep such a two-dimensional network operating? According to the principle of detailed balance[REIF65], a body maintained at a certain temperature T emits from any portion of its surface, in any direction, and for any frequency and polarization angle, the same amount of radiation that it would absorb if placed inside a cavity whose walls were maintained at that temperature. Thus, the body loses energy by irradiating thermal power, whose distribution over certain parameters is completely specified by the temperature T and, to a certain extent, by the nature of the body. On the other hand, in order to maintain the body at that temperature, one may supply the required total power in any form whatsoever, for example, using thermal power having an arbitrarily assigned distribution over the above parameters. If the two distributions do not exactly match, the body will be able to exploit the difference between them so as to decrease its entropy. In particular, one can supply power to the body from a cavity of which one portion is kept at a temperature higher than T and the rest at a temperature lower than T . In order to satisfy the uniformity precept, one must also impose that, for every frequency and polarization angle, the corresponding component of the incoming radiation

strike each cell at the same angle. In practice, any radiating body or "antenna" surrounded by a black background and kept at a distance much greater than the diameter of the implemented network would approximately satisfy the uniformity requirement. (A good approximation of this kind would be found in interstellar space.)

2.3 Thermodynamical considerations. II

In regard to the above thermodynamical objections to three-dimensional cellular automata, one may wonder whether the worst case just considered, i.e., of each cell erasing a minimum amount of information at each time step, could ever occur. Intuitively, the amount of information per unit volume contained in the initial configuration of a cellular automaton is finite, and one cannot end up erasing more than that amount. Thus, in spite of irreversibility, the average rate of erasure per unit volume must approach zero as time progresses, leading to behavior that more and more closely approximates a reversible one. (Note that, owing to the finite speed of propagation of information, new information can enter a certain "volume" of a configuration at a rate that is at most proportional to the bounding "surface". For large "volumes," the rate of information replenishment is negligible.) The release of a finite amount of energy per cell during the whole lifetime of an implementation would only raise the temperature by a bounded amount, and thus, at least in principle, would

not lead to insurmountable operating difficulties.

Is it necessary to dissipate heat in those portions of the implementation of a cellular automaton where only reversible processes are taking place? The main difficulty with this approach is that, like energy, information is, in general, a quantity associated with a system as a whole, not with its individual parts. On the other hand, in order to avoid unnecessary dissipation of energy, the implementation mechanism would have to be able to determine by local means whether a certain amount of information is about to be destroyed or, so to speak, merely transferred to a different location (cf. [LAND61]). However, as explained in Section 3.2, no effective solution is known, in general, to an even more restricted problem, i.e., that of determining on the basis of its local map whether a cellular automaton is on the whole reversible. Thus, we do not know whether the heat production associated with information erasure is sufficient, per se, to rule out the implementability of an arbitrarily extended, three-dimensional cellular automaton.

However, there are other thermodynamical factors that weigh in the same direction. Let us consider the "best" case, i.e., that of a reversible cellular automaton whose individual cells perform reversible logical operations (see Technique 3.2.2). In this case, it is conceivable that suitable physical mechanisms (cf. superconductivity phenomena) would be able to perform the required operations without converting free energy into heat. By definition,

this would imply a strictly reversible mechanism. On the other hand, discretionary initialization and read-out are, by definition, incompatible with a reversible system. For such purposes, the system would have to operate in an auxiliary, irreversible "mode". From a thermodynamical viewpoint, every initialization act would release heat at the expense of externally supplied free energy, while every read-out act would convert some of the system's own free energy into heat. Clearly, in order to repeatedly initialize or read out arbitrary portions of the system, one would have to provide means of steadily supplying free energy and dissipating heat, and, as we have shown in the preceding section, this is impossible for a uniform implementation of a cellular automaton in more than two dimensions.

On the other hand, the additional energy-flow requirements for the initialization and read-out modes can be satisfied, for a two-dimensional implementation, by the same means suggested in the previous section in regard to the worst case operating mode. Thus, as far as energy flow is concerned, it is possible to provide the cellular automaton implementation with adequate input/output facilities (without which it could not be treated as a realistic "computing" device).

We conclude that the uniform implementation of two-dimensional cellular automata is consistent with thermodynamical principles, while that of three-dimensional cellular automata is not.

2.4 Synchronization

The term synchronization has been used, in regard to cellular automata and other parallel-computing structures, in two quite distinct contexts.

On one hand, there is the problem of guaranteeing that a certain system embedded in a cellular automaton behave in a desired way over a certain range of initial conditions. In particular, one may require that a certain number of components of that system eventually reach a specified state all at the same time step in order to synchronously perform a collective action. This is called the firing-squad problem[MOOR64] and has been extensively investigated [WAKS66,ROSE73,NGUY74,HERM74,SHIN74,KOBA75,ROMA76]. The firing-squad problem is essentially a "software" problem, in the sense that it is well-defined in the context of abstract cellular automata independently of how these may be implemented.

On the other hand, there is the "hardware" problem of constructing physical systems that recognizably reproduce the well-specified abstract behavior of a cellular automaton in spite of appreciable construction and operating tolerances. The implementation of large sequential networks presents special timing problems that are not easily solved, as in small ones, by placing the whole network under control of a single external clock. Thus, one must find other ways to guarantee that, for a cellular automaton, the

state of any cell at any time step be correctly computed as a function of the state of its neighbors at the previous time step. Here, we are interested in this aspect of synchronization, which too has been extensively investigated both in the context of digital circuitry[MCNA64, STUR68, BEIS74, KELL74, REYC 74] and of more abstract models[ROSE73,KELL75,LIEN76,LIPT76,GOLZ77]. In such works, emphasis, terminology, and range of applications vary widely. Basically, correct behavior in spite of timing tolerances can be achieved by redundantly encoding the states of a given synchronous network into a larger set of asynchronous-network states. Such redundancy is introduced in different ways by various authors, often implicitly and probably unknowingly (which contributes to a certain confusion in the field). We shall show that there is a simple, "canonical" way to achieve the required degree of redundancy. Briefly, starting from the state alphabet A, we shall define a (redundant) alphabet $AxA'x\Psi$, where set A', identical to A, allows a cell to prepare its own next state without "forgetting" the current one, and Ψ, the phase component, allows the cell to follow a definite protocol in exchanging information with its neighbors (cf. [LIPT76]). Such construction is applicable to any cellular automaton.

2.4.1 The role of delay elements

From a formal point of view, a sequential network is a particular kind of colored digraph (Section 1.3). It is well known that, without loss of generality, one can restrict one's attention to sequential networks consisting of only two types of nodes (two-input NAND-gates and two-way splitters) and two types of arcs (connecting lines and delay elements).

In terms of physical implementation, it is easy to visualize, for example, gates as threshold devices, splitters as amplifiers, and connecting lines as wires. On the other hand, it is not immediately clear what kind of physical devices should be associated with delay elements. Are they merely distinguished arcs off which one agrees to read the network state at regular intervals? Do they represent places where clock signals supplied by an external clock are made to interact with the signals flowing through the network proper? Are they supposed to contain an internal time reference and/or additional digital circuitry?

One often reads that the role of delay elements is merely to represent in a lumped form, for convenience of analysis, the various time delays to which, in a physical implementation, signals are inevitably subjected in traversing arcs and nodes. (It is certainly true that one can simplify the analysis of a physical network by

imagining that signals instantaneously propagate through gates, splitters and lines, and are delayed one unit of time whenever they pass through a delay element.) However, such an interpretation, while adequate for introducing a correspondence between physical time and the sequential behavior of an abstract network, constitutes an overly restrictive characterization of the role of delay elements. In particular, the expression "unit of time" seems to imply some means of guaranteeing that all delay elements actually use the same unit. This is very difficult to achieve in the implementation of large uniform networks and, on the other hand, is not at all necessary for the correct functioning of such networks.

The function of the delay elements is better understood in the following context. Physical sequential networks use continuous processes for representing events that are conceived in a discrete-state domain and a discrete-time domain. As state discreteness is achieved, in practice, by renormalizing to a certain reference voltage V_q any signal levels which fall within a certain voltage interval (V_q^-, V_q^+) (where q ranges over a discrete set of abstract states), in a similar way time discreteness is achieved by renormalizing to a certain reference time T_i any signal fronts which arrive within a certain time enterval (T_i^-, T_i^+) (where i ranges over a discrete set of abstract time steps). Thus, a delay element is generally implemented as a sample-and-hold device, which performs, in the time domain, a function similar to that of a clamp in the voltage domain.

Provided a suitable voltage interface is used, one can represent the same abstract state q by means of different reference voltages V_q in different portions of a network and still achieve correct results (this is done routinely in digital hardware). In an analogous way, with proper timing interface, it is possible to represent the same abstract time step i by a sampling time T_i that differs from place to place. The interface (or synchronization mechanism) must guarantee that the abstract order relation "argument-value" which holds between the state of a cell at a given time step and that of any one of its output neighbors at the next time step be translated into the correct temporal relation between the commands issued to the associated sample-and-hold devices.

Thus, if an abstract sequential network is interpreted as a blueprint for a physical circuit, the delay elements have two functions: (a) to represent sample-and-hold devices, and (b) to imply a timing mechanism which, in general, will have to be implemented as an auxiliary network underlying the original one and controlling the sample-and-hold devices.

In the binary-signal domain, sample-and-hold devices are realized by data (or D) flip-flops. These basically fall into two categories, as follows:

A. Edge-triggered. This flip-flop consists of a single binary storage register (Figure 2.1a). The information present at the input port is loaded in the register on the rising edge of the clock signal (i.e., a 0→1 transition), and appears at the output port immediately

thereafter. The register's previous content is destroyed.

B. Master-slave (MS). This flip-flop consists of two binary registers, master and slave, connected as in Figure 2.1b. On the rising edge of the clock signal, the master is loaded with the information present at the input port, while the slave is unaffected. On the falling edge of the clock signal, the content of the master is copied into the slave and appears at the output port.

Consider synchronous operation, where all flip-flops are controlled by a signal originating from a single clock. Since, for large networks, the clock signal must be distributed by an auxiliary network containing amplifier stages, the clock skew (i.e., the maximum difference between the time of arrival of the same clock pulse at different flip-flops) will increase with the size of the network,

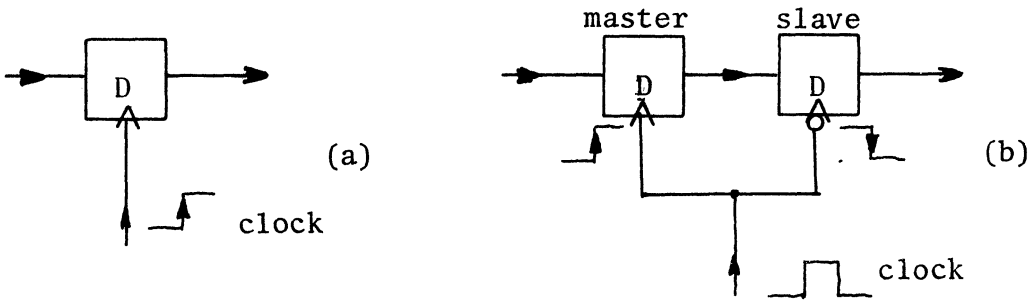


FIGURE 2.1. Edge-triggered (a) and master-slave (b) flip-flops. The first is operated by the leading edge of a pulse, while the second requires both a leading and a trailing edge.

since physical amplifiers are affected by delay noise (or jitter) as well as amplitude noise. On the other hand, if edge-triggered flip-flops are used, the clock skew must be less than the minimum propagation time of signals from the output of any flip-flop to the input of any other. (Otherwise, the signal present at one flip-flop's input may change before that flip-flop has had time to load it.) Thus, edge-triggered flip=flops are not compatible, in general, with networks of arbitrarily large size.

Always in the context of a centrally originated clock signal, timing specifications can be relaxed with MS flip-flops. In this case, in fact, in order to achieve the correct data transfer between flip-flops, it is sufficient that the interval between two clock pulses be larger than the sum of the clock skew and the maximum propagation time of data signals. In general, as the network size increases, one will have to reduce the clocking rate.

The use of timing pulses generated by a single clock is not compatible with the uniform implementation of a cellular automaton, since (a) the minimum clock period that avoids clock-skew problems is a function of the number of implemented cells, and (b) a single central clock requires a number of amplification stages that increases with the number of implemented cells.

In the following sections we shall consider distributed timing mechanisms that are compatible with the uniformity precept.

2.4.2 Phase-locked loops

In is clear from the above discussion that, in regard to the problem of synchronizing cellular automata by uniform means, one will have to restrict one's attention to local timing methods. Each cell will be provided with an internal time reference, and will be allowed to exchange timing information only with its immediate neighbors (cf. Figure 2.3 below).

Because of thermal noise and quantum indeterminacy[BREM67], the time references cannot be started and maintained at an identical value for all cells. Moreover, any correction introduced locally on the basis of feedback from the neighbors cannot take place before a measurable amount of drift has been detected. In such conditions, it is easy to verify that absolute synchronization (i.e., within one clock period for the whole network) is not achievable with local timing methods.

On the other hand, as noted in the previous section, absolute synchronization is not required a priori for a meaningful implementation. More specifically, one may observe that the sampling of input data and the corresponding change of output data take place, in an MS flip-flop, at quite separate and independent times. In particular, the slave register will display the flip-flop's state from some time before to some time after the master register is loaded with a new

state. Thus, one still obtains correct results if input sampling in the flip-flop's output neighbors takes place slightly "out-of-phase" with respect to sampling in the flip-flop itself. (The concept of phase is defined more precisely below.) In this context, the synchronization problem is reduced to guaranteeing that the phase difference between any cell and any of its neighbors be limited to a specified range. When this condition is satisfied, one has what is called relative synchronization or phase locking. The feedback mechanism through which phase locking is achieved is called phase-locked loop (PLL). Note that, in the case of relative synchronization, the total phase difference between two distant cells may amount to more than one cycle.

PLL's are used extensively in communications circuitry to achieve [relative] synchronization of independent oscillators (for a thorough discussion see [VITE66,BLAN76]). In its most basic form, a PLL consists of a master oscillator operating at a frequency $\bar{\omega}$ and a slave oscillator provided with a frequency reference ω_{ref} and operating at a variable frequency $\omega = \alpha\omega_{ref}$, where α is a parameter. (Here, the terms master and slave have no connection with MS flip-flops.) By comparing its own phase ϕ with the phase $\bar{\phi}$ of the master, the slave is able to continually adjust the parameter α, and thus its own frequency ω, in such a way as to maintain at a steady value the phase difference $\Delta\phi = \phi - \bar{\phi}$. For the circuit of Figure 2.2. the governing equation

$$\frac{d\phi}{dt} = [1-k(\phi-\bar{\phi})]\omega_{ref}$$

has the general solution

$$\Delta\phi - z = (\Delta\phi_0 - z)e^{-k\omega_{ref}t} \quad , \quad (z = \frac{1}{k}(1-\frac{\bar{\omega}}{\omega_{ref}})) \quad ,$$

where $\Delta\phi_0$ is the phase difference at time $t=0$. Thus, $\Delta\phi$ exponentially decays to the value z . In steady-state conditions, i.e., when the ratio $\frac{\bar{\omega}}{\omega_{ref}}$ drifts <u>slowly</u> with respect to the time constant $k\omega_{ref}$, ω will track $\bar{\omega}$ and $\Delta\phi$ will stay close to the value z .

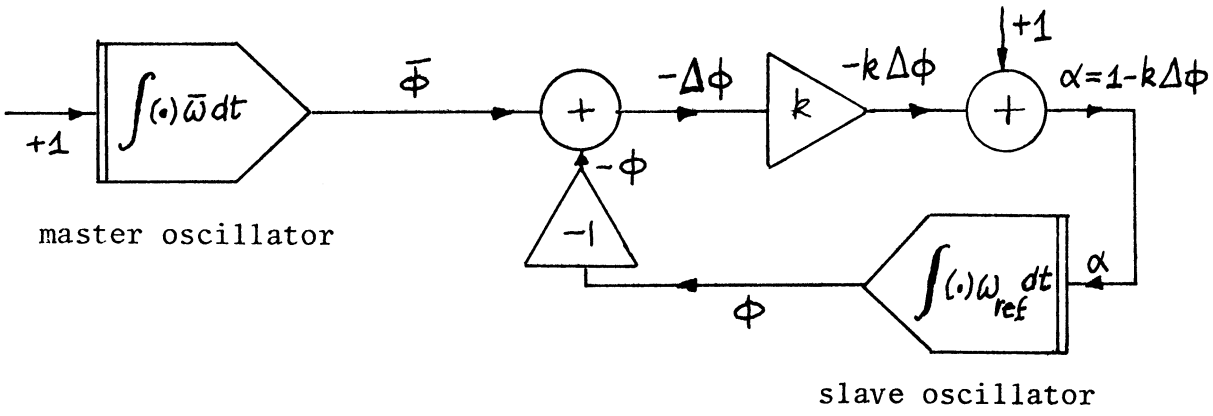


FIGURE 2.2. Elementary phase-locked loop. The frequency reference in the slave oscillator is constantly modified by a variable factor α in order to maintain this oscillator in phase with the master. The factor α depends on the phase difference between the two oscillators. The factor k determines the strength of the feedback. This is a linear first-order loop. In most applications, second-order loops are used.

The above schema represents an idealized case. In a physical implementation, the following two factors must be taken into account:

(i) In encoding the phase ϕ as a state variable $\psi = f(\phi)$, at least two sources of ambiguity are introduced. On one hand, on a microscopic scale, f contains a random noise component due to thermal noise and other mechanical imperfections. On the other hand, f maps a quantity ϕ which is defined over the whole time axis R into a finite interval T, usually by encoding ϕ modulo-T (i.e., f is a <u>periodic</u> function). Thus, ψ cannot "tell the difference" between two values of the phase that differ by a multiple of T. As long as the phase difference between master and slave is less than one cycle, and thus can be determined without ambiguity by the PLL mechanism on the basis of a difference in the state variable ψ, the oscillators are said to be <u>phase-locked</u>.

(ii) Because of the finite speed of signals, the phase of the master is known to the slave with a certain delay. Essentially, this introduces additional state variables in the PLL circuit and greatly complicates it dynamic analysis.

The circuit of Figure 2.2 is asymmetrical. One may design a symmetrical circuit containing two oscillators coupled in such a way that they will maintain phase locking in spite of being provided with different frequency references (or frequency references that are affected by a certain amount of noise). In principle, the schema can be extended to a whole array of oscillators; for instance, to oscillators associated with the cells of a cellular automaton and

used for synchronizing the exchange of information between such cells, as illustrated in Figure 2.3.

With reference to this figure, each cell consists of a data section and a timing section. In the data section, all lines carry r-ary

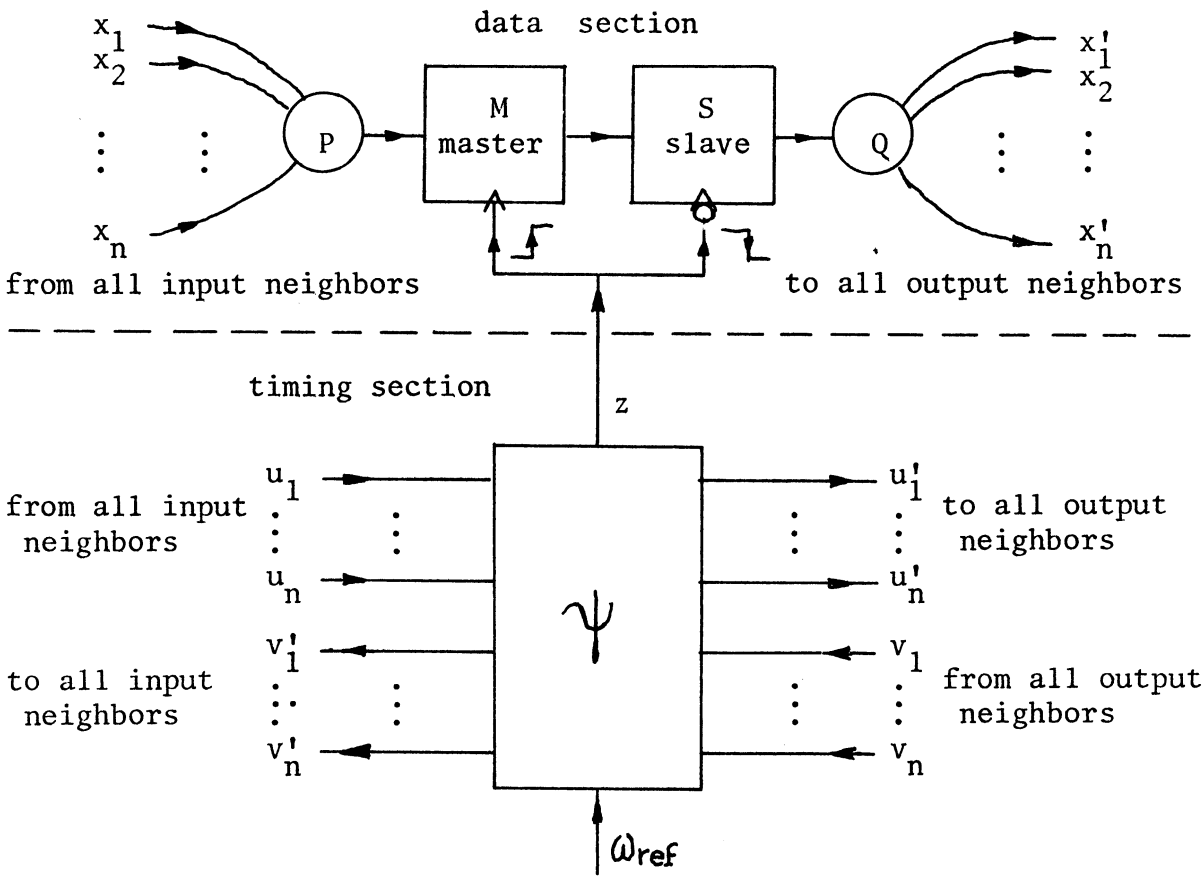


FIGURE 2.3. Basic cell structure in a cellular automaton synchronized by a PLL array. The data section is conventional and does not require comment. The timing section receives information about the phase of all input and output neighbors, and adjusts its own phase accordingly. At the same time, the timing section makes information about its phase available to all input and output neighbors, so that these also be able to adjust their phase. Binary signal z goes "high" and then "low" once for every timing cycle. The value of ω_{ref} may differ from cell to cell, and may be affected by noise.

signals (where r is the size of the cellular automaton's state alphabet A). M and S (the master and the slave) are r-ary registers, and together constitute the r-state generalization of a binary MS flip-flop (they may be thought of as consisting of a sufficiently large collection of binary flip-flops). Lines $x_1,\ldots,x_n$ are connected to the n input neighbors, while lines $x'_1,\ldots,x'_n$ are connected to the n output neighbors. On a $0\to1$ transition of the binary signal z, which is generated in the timing section, the master loads the state appearing at the input port p; on a $1\to0$ transition, the slave displays the same state at the output port q. Combinatorial network P implements the cellular automaton's local map λ, while Q is an n-way splitter.

The timing section is characterized by a quantity ψ called phase (properly, this is a state variable which is an ambiguous function of the phase, as noted above), which evolves in time describing a closed path through the phase space Ψ according to a governing equation of the form

$$\frac{d\psi_s}{dt} = f(\psi_s, \overset{n}{\underset{i=1}{<}} \psi_{s+X_i} >, \overset{n}{\underset{i=1}{<}} \psi_{s-X_i} >, \tau_{ref,s}) \qquad (2.4.2.1)$$

where ψ_s is the phase of cell s, ψ_{s+X_i} and ψ_{s-X_i} are the phases, respectively, of its input and output neighbors $s+X_i$ and $s-X_i$, and f represents an operator expression which may contain linear or nonlinear operators (e.g., differential or integral operators, convolutions, Dirac operators, etc.). Quantity $\tau_{ref,s}$ is an internal time reference. The functions u_i, v_i (and the lines

of the same name) encode whatever information about the phase is to be exchanged between cells, while z encodes the phase into a binary signal which controls the loading and displaying of cell-state data by the master and slave registers.

A network consisting of timing sections interconnected as in Figure 2.3 and governed by equations of the form (2.4.2.1) constitutes, at least formally, a generalization of the PLL schema to a uniform array. Are there any functions f for which such schema admits of phase-locked solutions? As far as we know, the analysis of phase-locked loops configured in infinite arrays has not been attempted. For simple choices of the function f, the mathematical machinery involved is similar to that used in studying the behavior of elastic membranes (thought of as collections of point masses governed by restoring forces). It is conceivable that stable solutions exist for uniform PLL arrays based on a continuous phase space Ψ and linear mechanisms such as are found in traditional PLL's. On the other hand, circuit tolerances which have little influence on PLL's containing few oscillators may play a much more important role in infinite arrays. If the variability of circuit components is explicitly considered in analyzing a PLL array, the task of studying the stability of its solutions becomes much more complicated. Fortunately, most of these difficulties can be avoided altogether, as shown in the next section, by considering extremely nonlinear mechanisms that are better described in terms of digital circuitry.

2.4.3 Phase interlocking

In this section, we shall consider a timing mechanism based on phase interlocking--a very specialized form of phase locking. While each cell is still provided with an internal timer, no information abut the timer's state (e.g., the currently elapsed fraction of the timer period) is exchanged between cells. The exchange of information is limited to a discrete component of the phase--typically, a single binary variable.

In what follows, we shall assume, for the timing section postulated in Figure 2.3, the structure illustrated in Figure 2.4. In addition to data paths, which carry information about the cell's state, a cell is connected to its neighbors by timing paths, which carry information about its phase. More specifically, a cell is connected to each of its output neighbors by means of an incoming line carrying notifications and an outgoing line carrying acknowledgments (these are both binary signals). Correspondingly, the cell is connected to each of its input neighbors by an incoming line carrying acknowledgments and an outgoing line carrying notifications. A value of 1 on a notification line signifies that the state of the issuing cell is available for sampling by the corresponding output neighbor, Similarly, a value of 1 on an acknowledgment line signifies that the issuing cell has read the state of the corresponding

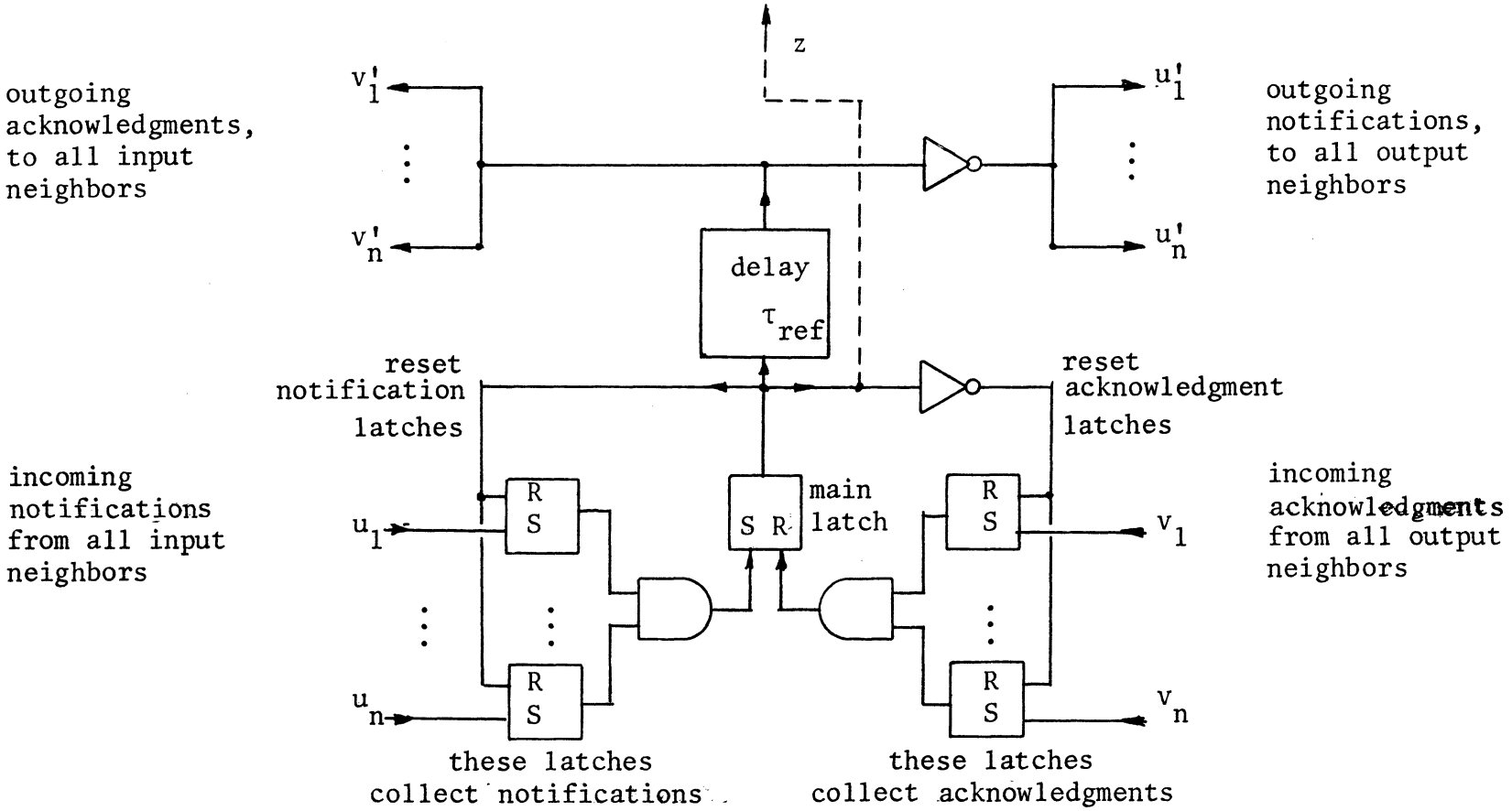


FIGURE 2.4. Basic timing mechanism for phase interlocking. When all notifications $u_1,\ldots,u_n$ have arrived, the main latch at the center of the figure is set. This clears all notification latches, loads the input data (clock signal z), and, after a delay τ_{ref}, posts acknowledgments. Similarly, when all acknowledgments have arrived, the main latch is reset. This clears all acknowledgment latches, transfers the data to the slave for display, and, after a delay, posts notifications. In this way, the action of each cell is interlocked with input and output neighbors.

input neighbor, which thus, as far as that cell is concerned, need not display such state any longer.

The timing cycle is invariably composed of four consecutive phases, according to the following schena:

A . Load input data, and wait a discrete amount of time (established by the timer) for the slave's input to attain its final value.

A' . Post acknowledgments and remove previously posted notifications. Wait for incoming acknowledgments. When all have arrived, go to phase B.

B . Display output data (i.e., the new state), and wait a discrete amount of time (established by the timer) for the input of the master of the neighboring cells to attain its final value.

B'. Post notifications and remove previously posted acknowledgments. Wait for incoming notifications. When all have arrived, go to phase A.

Phase A' [B'] may have arbitrary duration. In particular, if all incoming acknowledgments [notifications] are already present at the beginning of the phase, its duration reduces to a minimum dependent only on internal propagation delays. The four phases are completely characterized by the status of the timer (running/quiescent) and by the state of the main set/reset latch (whose inputs are

sensitive to a $0 \rightarrow 1$ transition), according to the following table:

PHASE	TIMER	LATCH
A	running	1
A'	quiescent	1
B	running	0
B'	quiescent	0

The output state of the latch controls the loading of cell-state data in the master and slave registers and, after being delayed by the timer, controls the posting of notifications and acknowledgments. The timer itself is represented, in Figure 2.3, by a true delay (i.e., the input signal appears at the output after a fixed time τ_{ref}). τ_{ref} may vary from cell to cell, but must be greater than the maximum propagation delay of cell-state data from register to register. On the other hand, internal propagation delays must be less than the minimum intercell propagation delays.

Clearly, the state of the slave of the input neighbors of any given cell is not allowed to change until that cell has been able to compute its own new state (trough P) and load it into the master. Thus, the sequencing of operations in neighboring cells is interlocked, and the interlocking extends recursively to the whole cellular automaton. If, by external means, the phase of a cell is held to constant value by latching the timer (this may be expedient for input/output operations--see Section 2.5.2), a "freezing" wave

originates from that cell and propagates through the whole array. When the phase is allowed to resume its normal evolution, a "thawing" wave is originated. In neither case is the correct sequencing of operations lost.

2.5 Initialization and read-out

A trajectory of a cellular automaton constitutes the solution of a recursion relation (the local map) for given initial conditions. In order to experimentally study such trajectories in a physically implemented cellular automaton, one must be able to set up particular initial conditions and observe the subsequent evolution of the automaton's state. To this purpose, the cellular automaton must be provided with means to interact with an external, localized structure, which we shall call the user.

In ordinary computers, where the time of propagation of signals through the whole structure is small, the user can be given essentially immediate access to every portion of the computer. In particular, the whole computer can be switched from the normal operating mode to an auxiliary input/output mode (and vice versa) within a single clock cycle. In such conditions, the two operating modes do not interfere with each other and can be treated independently. On the other hand, for distributed computing structures of arbitrarily large size, the user cannot be realistically assumed to be

ubiquitous, and a decision on the part of the user to read or alter the contents of a given cell will reach that cell only after a delay proportional to the distance between the cell and the user itself. During such time, the cell may have gone through a number of time steps. A satisfactory mechanism of interaction between the user and the cellular automaton must somehow allow the user to plan input/output operations in advance in such a way as to read or alter the state of a cell in correspondence to any specified time step. We shall describe such a mechanism (Section 2.5.1). In addition, it is desirable that the user be able to completely clear the cellular automaton (e.g., by setting all cells to a distinguished quiescent state), in order to effectively construct initial configurations that are completely specified--such as finite configurations. Owing to the finite speed of propagation of signals in cellular automata, such clearing operation--taken literally--would require an amount of time proportional to the "radius" of the implementation, thus violating the uniformity precept. However, we shall show (Section 2.5.2) that, with suitable uniform machinery, essentially the same result can be obtained by means of a short sequence of user-originated commands.

Thus, within the constraints of the uniformity precept, it is possible to provide a cellular-automaton implementation with initialization and read-out mechanisms that are adequate for meaningful experimentation.

2.5.1 External users

In conformity with the uniformity precept, any input/output provisions must be implemented in the same manner for all cells. On the other hand, a realistic user would not be able to exchange information with all cells at the same time. Thus, at any moment, a cell may or may not be in direct communication with the user. We shall imagine each cell provided with a set of "test points" where the user can insert a probe, and a set of "switches" that the user can manipulate and leave in a given position for an indefinite amount of time. Through the test points, the user can sense the state of a certain number of wires in the cell, while through the switches it can alter the state of the flip-flops or the timers (cf. Section 2.4.3), and interrupt or establish certain internal connections. In this way, the operation of the whole network is defined whether or not a user is actually performing input/output operations on any cells, and is also defined when more than one user is interacting with the network.

We shall not concern ourselves with the user's internal structure and computing capabilities. We shall only assume that at any moment the user be in contact with a single cell and be able to exchange information with it, much as a Turing machine communicates with its tape by means of a localized "head." From the viewpoint of

physical interpretation, the Turing machine head must be able to sense the boundary between adjacent tape squares, so that the program "move s squares to the right" can be carried out by counting how many times a boundary is traversed. A cellular automaton's cell-states are distributed over time as well as over space. Thus, the user, interpreted as a head scanning such spacetime structure, must be able to sense also the boundary between time steps (which is characterized, for each cell, by a certain transition of the timing mechanism, as discussed in Section 2.4.3). In particular, the user must be able to carry out programs such as "wait for t time steps." More generally, for any spacetime path described by the user in moving along the cellular automaton, the user must be able to determine the integral for both spatial displacements and phase differences, in order to associate with the cell-state encountered at a certain physical position and instant the corresponding abstract cell and time step. Such integrals must be independent of the path and unique up to a translation of space and time coordinates.

Because of the lack of global synchronism in the implementation schema that we are considering, the proof of the existence and unicity of the above phase integral is rather elaborate, though intuitively straightforward, and will be given separately[TOFF**]. In the same paper, we shall treat from a more formal viewpoint the constructions that are informally presented in the following sections. In order to discuss read-out and initialization problems, we shall assume that the user be provided with the following capabilities:

(a) To move along the cellular automaton, and to measure its own space and time displacements in terms of cell and time-step coordinates. (No assumptions are made on the speed of such movements, which can change from moment to moment and be arbitrarily high or low.)

(b) To sense and/or alter the state of the currently scanned cell.

(c) To latch or unlatch the timer contained in the cell (cf. following section).

(d) To sense and/or alter the phase of the currently scanned cell.

Such capabilities are physically plausible, and their formal counterpart is well defined. Capability (a) gives the user a frame of reference with respect to which events in the physical network can be associated with abstract cellular-automaton events. Capability (b) is, of course, essential for input/output operations. As we shall see below, capabilities (c) and (d) allow the user, in spite of his locality and limited mobility, to read out or initialize an arbitrarily large number of cells at the same time step by appropriately readjusting the timing pattern in such a way as to expand certain time steps over a sufficiently long interval of physical time.

2.5.2 Timer manipulation

As noted in Section 2.4.3, the function of the timers is to delay the timing signals in such a way that in no case these will arrive to a cell earlier than the corresponding data signals. On the other hand, the requirements of phase interlocking do not set any upper bound to the timers' period. Thus, a timer's nominal rate $\frac{1}{\tau_{ref}}$ can be replaced by a lower rate (in particular, by the zero rate) for an arbitrary length of time. For input/output purposes, we shall associate with each cell an additional flip-flop which can be set and reset by the user. When the state of such flip-flop is 1 the timer is unlatched, and operates at its nominal rate. When the state is 0 , the timer is latched and its rate is zero. In normal operating conditions, all timers are unlatched.

For simplicity of illustration, we shall consider a one-dimensional cellular automaton having the $\langle -1,0,1 \rangle$ neighborhood. For each cell, the phase evolves along the time dimension, repeatedly describing the invariable sequence A, A', B, B' (Figure 2.5a). The length of phases A and B is determined by the timer contained in each cell, and may differ from cell to cell because of differences in the value of the time references and/or because of superposed noise. During phases A' and B' each cell waits until the neighboring timers have reached the end of their periods

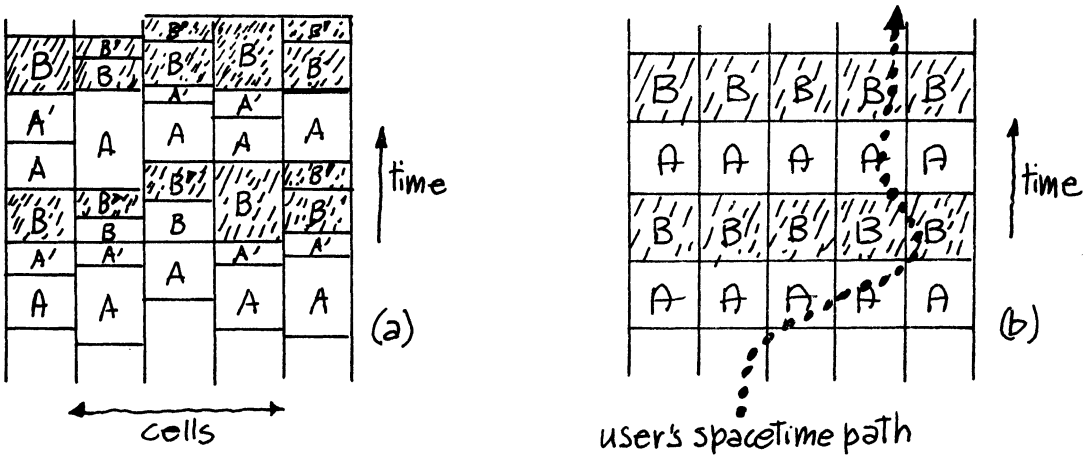


FIGURE 2.5. The spacetime evolution of the phase as in (a) is represented in the schematized way of (b) when no ambiguity is possible.

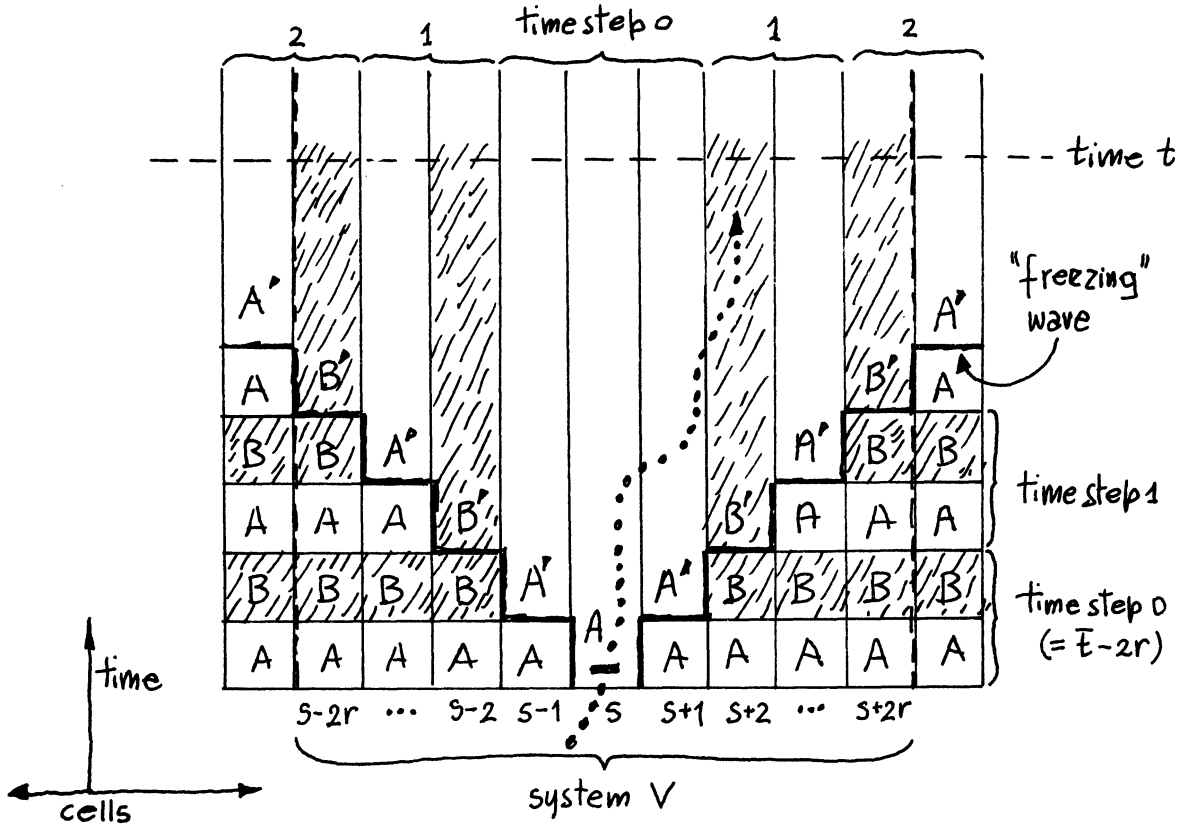


FIGURE 2.6. By suspending the activity of the timer of cell s at time step 0 ("▬"), the user originates a freezing wave that propagates at uniform speed in all directions. At time t, the activity of system V is frozen and available for inspection.

(cf. Section 2.4.3). In phase-locked conditions the phases of two adjacent cells can never differ by more than half cycle. Thus, in the spacetime diagram of Figure 2.5a, the A/A' and B/B' phases form continuous bands (clear and, respectively, shaded in the figure) that are approximately orthogonal to the time axis. In the following figures, we shall ignore those timing differences between cells that are not explicitly considered in the discussion. Thus, whenever convenient, the phase bands will be drawn straight and the waiting phases A' and B' will be implied by the boundary lines between A and B and, respectively, B and A , as in Figure 2.4b. The spacetime path described by the user will be indicated by a dotted line.

Suppose that the user intends to read out the state at time step $\bar{t}$ of a set V of cells contained within a sphere (a segment, in the one-dimensional case) having radius 2r and center in cell s . To this purpose, the user must be in contact with cell s no later than time step $\bar{t}$-r (= 0 in Figures 2.6 and 2.7). During phase A of such time step, the user shall latch the timer of s . A "freezing" wave will spread from that cell outwards, as adjacent cells successively find that their notifications and acknowledgments are not answered and enter a waiting state (A' and B' , in alternation). The wave will eventually reach and pass cells s+2r, s-2r . Thus, at a certain moment t all cells comprising set V will be latched in a waiting state, and can be examined by the user without any timing constraints. However, such cells will have halted at

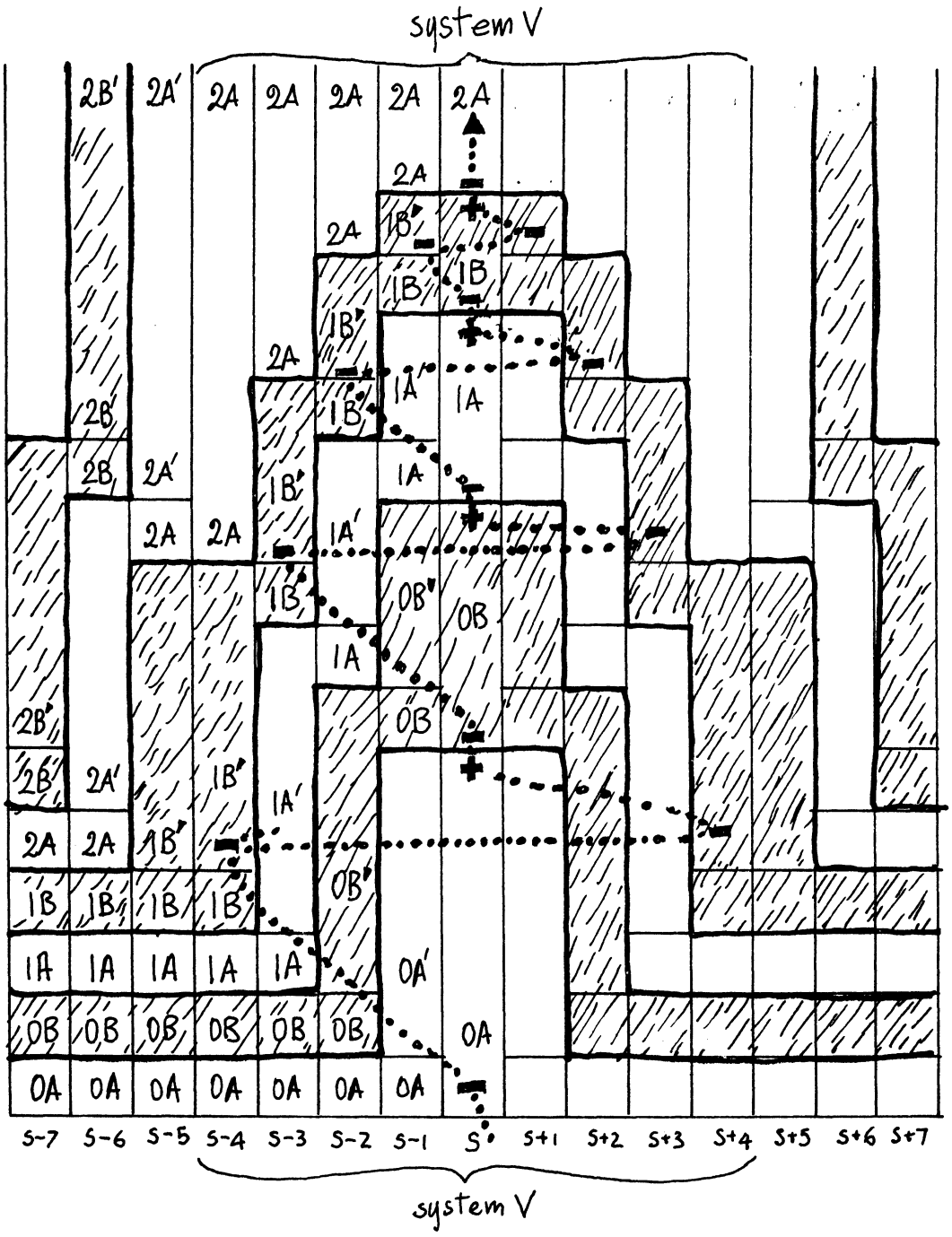


FIGURE 2.7. In order to be able to observe cells s-4,...,s+4 all at the same time step, the user describes the above spacetime path, latching ("-") and unlatching ("+") the timers in an appropriate sequence. Note that the "iso-phase" bands are distorted with respect to the isochrone lines, but are still continuous and distinct from one another.

different time steps, ranging from time step $\bar{t}-r$ for cell s to time step $\bar{t}$ for cells $s-2r$ and $s+2r$, and a cross-section of the system at physical time t will not represent a cross-section of the same system at any given time step. In order to achieve the latter situation, the following procedure must be followed (Figure 2.7).

After the freezing wave has reached cells $s-2r$, $s+2r$, the user shall move to the latter two cells and latch their timers. Then the user shall return to cell s, unlatch its timer, and latch it again after a half time step, i.e., as soon as the cell enters phase B. In this way, the whole "frozen" front contained between cells $s-2r$ and $s+2r$ exclusive will advance a half time step, and cells $s-(2r-1)$, $s+(2r-1)$ will reach time step $\bar{t}$. Following an analogous procedure, the user shall successively latch the timers in cells $s-(2r-1)$, $s+(2r-1)$, then $s-(2r-2)$, $s+(2r-2)$, etc., until all cells from $s-2r$ to $s+2r$ inclusive are latched. It is easy to verify that at this point all cells within the sphere will have reached phase A of the same time step $\bar{t}$. The user will then be able to observe at leisure the system's state as of time step $\bar{t}$, and possibly reset the system to an arbitrary initial state. Finally, in order to let the system resume its evolution, the user shall unlatch, in any order, the latched timers.

The above construction can be immediately extended to arbitrary cellular automata. Notethat the construction is not dependent on the user's maintaining a minimum velocity on moving along the

cellular automaton, but only on the correct sequencing of operations. Thus, timer manipulation provides an effective means of examining and initializing an arbitrary finite system embedded in a cellular automaton.

2.5.3 Blanking

Finite configurations (i.e., configurations in which all but a finite number of cells are in the blank state) play a distinguished role in the theory of cellular automata (cf. Section 1.4). In a physical implementation consisting of a possibly very large but finite number of cells, the study of finite configurations permits one effectively to ignore the existence of the "border" at least until the nonblank portion of a configurations attempts to "grow" past the border itself. In order to construct arbitrary finite configurations, it would be convenient for the user to be able to completely blank the entire cellular automaton, i.e., initialize all cells with the blank state. However, as noted in Section 2.5, any action taken by the user at a certain moment will propagate at best with a certain maximum speed, and the time required to blank the whole cellular automaton will be dependent on the size of the implementation. This violates the uniformity precept. On the other hand, the support of a given finite initial configuration cannot grow faster than allowed by the neighborhood structure--i.e., one neighborhood radius per time step.

Thus, in order to study the evolution of finite configurations it is sufficient to have a mechanism by which a "blanking" wave originated at a certain point propagates with a speed at least as large as the speed of propagation of cell-state signals. In such conditions, if the user constructs a finite configuration near the origin of the blanking wave, this configuration, as it grows, will always find itself surrounded by a layer of blank cells, and its evolution will be the same as if the whole array were blank. Such blanking mechanism does not violate the uniformity precept.

We shall describe a blanking mechanism that is but a extension of the local map and is controlled by the same timing pattern that controls the exchange of cell-state data.

Given the state alphabet A, let us introduce, for blanking purposes, an additional state component $B = \{BLANK, RESET, RUN\}$. The extended state alphabet will thus be $A' = A \times B$. State BLANK will propagate to all neighbors of a cell except those being in the RESET state, and will also force the A-component of the cell to the blank state. RESET will propagate to all neighbors except those in the RUN state. Finally, RUN will propagate to all neighbors except those in the BLANK state. RESET and RUN do not affect the A-component of a cell's state.

Assume that the whole cellular automaton is originally in the RUN state. At time steps t and $t+1$ the user forces cell s to the BLANK state; at $t+2$ and $t+3$ the same cell is forced to the RESET state, and at times $t+4$ and $t+5$ to the RUN state. (This

timing pattern applies to the Moore neighborhood; more complex neighborhood templates may require forcing longer runs of each auxiliary state.) It is easy to verify that, as the three layers of BLANK, RESET, and RUN states always maintain a finite thickness (in particular, a BLANK and a RUN belonging to the same blanking sequence never come into contact), such a blanking sequence achieves the desired result. Immediately after issuing the last RUN command, the user may start initializing with a nonblank configuration (cf. previous section) the growing blank area that has just been created. In order to avoid spurious interference of blanking-wave trains, blanking command sequences issued from different cells must be separated by a time interval proportional to the distance between such cells.

Chapter 3

R E V E R S I B I L I T Y

3.0 Preliminaries

The issue of reversibility vs irreversibility of basic processes in biological, physical, and mathematical systems has been for a long time a source of stimulating discussions (see [BRIL64, MORO68,HOBS71,BENN73] for recent examples). Problems connected with reversibility have appeared often in the literature of cellular automata since Moore introduced the concept of Garden-of-Eden configuration[MOOR62]. In particular, Burks explicitly considered backwards-determinism in cellular automata[BURK71], Amoroso and Patt established conditions for the surjectivity and injectivity of a cellular automaton's parallel map[AMOR72], and Richardson proved that an injective parallel map is bijective and that its inverse is still a parallel map[RICH72]. In a recent article, Di Gregorio and Trautteur examined several definitions of reversibility in cellular automata [DIGR75]. We have chosen to call reversible (Section 1.4) a cellular automaton whose parallel map is a bijective function. This definition agrees with the most restrictive (and most natural from the viewpoint of physical analogy) interpretation of the term.

In Section 1.5, we have discussed certain conceptual analogies between the laws that govern the behavior of cellular automata and those of physics. In view of the fundamental role played by reversibility in physics, it seemed to us reasonable to expect that the study of reversible cellular automata should lead to stronger and more detailed analogies. On the other hand, while universal computing and constructing capabilities had long been established for cellular automata in general (Section 1.4), several sources seemed to indicate an intrinsic weakness of cellular automata in terms of their behavioral repertoire, thus implicitly casting doubts on their capability to model complex physical situations. In particular, Moore wondered whether a cellular automaton can have a self-reproducing configuration without having erasable configurations[MOOR62], Burks conjectured that backwards-deterministic cellular automata may not be able to support universal computation[BURK71], and Amoroso and Patt observed that in a straightforward enumeration schema nontrivial bijective parallel maps appear to be exceedingly rare (even allowing for a very weak notion of "nontriviality")[AMOR72]. Moreover, Smith sketched a proof purporting to show that any computation-universal cellular automaton has an unsolvable bounding problem for configurations[SMIT68,SMIT69], and Aladyev, working on Smith's results, seemed to show that the existence of erasable configurations is necessary for universal computation[ALAD72,ALAD73]. However, such fears are unfounded. In fact, we shall prove that computation- and construction-universality are compatible with reversibility; actually,

every cellular automaton can be constructively embedded in a reversible one having one more dimension.

Of course, the reversibility constraint as applied to cellular automata does result in certain restrictions of behavior. Such restrictions (which, as stated above, do not affect computing power) appear to be analogous to those that are observed, as a consequence of the same constraint, in physical systems. Intuitively, as in physical systems irreversible processes require a source of free energy and a heat sink, thus in reversible cellular automata irreversible computation requires a source of predictable signals (such as those supplied by a quiescent environment) and a "sink" for computation by-products.

3.1 Reversible dynamical systems

To every dynamical system $\langle C,\tau\rangle$ one can associate the directed graph $G = \langle C,D\rangle$, called state-transition diagram, where D consists of all ordered pairs of the form $\langle c,c'\rangle$ such that $c' = \tau c$. Such a graph consists, in general, of a number of distinct connected components. Each component is either an infinite tree (as sketched in Figure 3.1a) or a cycle on which a number of finite or infinite trees are grafted (Figure 3.1b). If τ is bijective, the system's state-transition diagram G takes on a much simpler form; in fact, in this case, every connected component of G is either a path that is

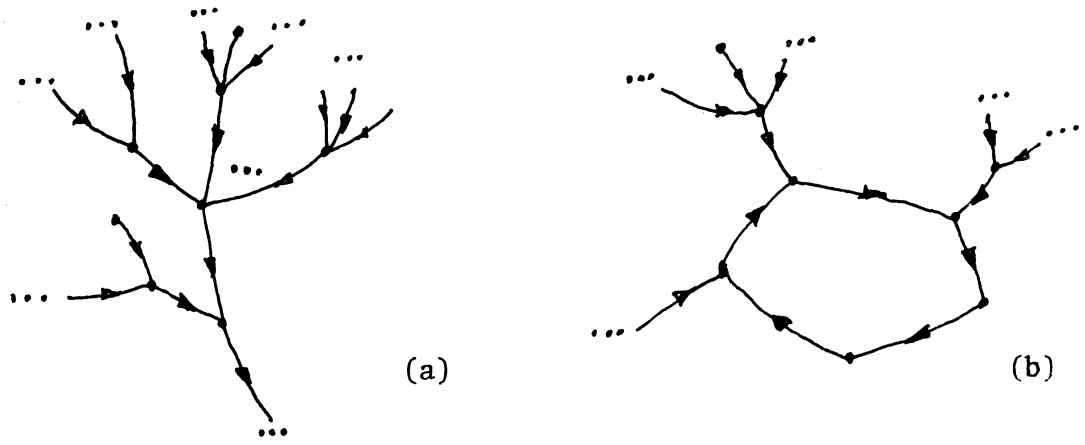


FIGURE 3.1. Each component of the state-transition diagram of a cellular automaton is either an infinite tree (a) or a cycle on which finite or infinite trees are grafted (b).

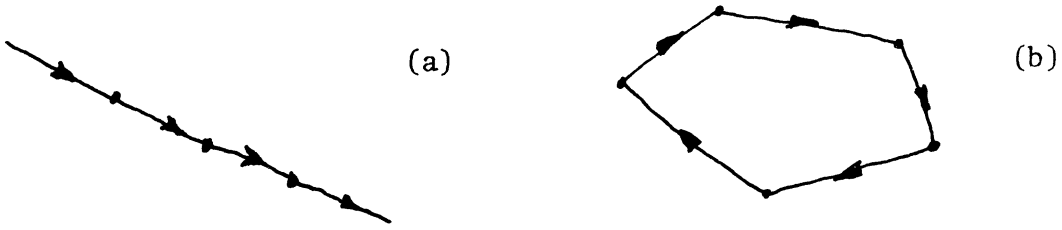


FIGURE 3.2. If the cellular automaton is reversible, each component is either a path infinitely extended on both sides (a), or a cycle (b).

infinitely extended on both sides (Figure 3.2a) or a cycle (Figure 3.2b). It is clear that the graph $\overline{G}$ obtained from G by inverting the direction of the arcs is the state-transition diagram of some dynamical system--namely, the system $\langle C,\tau^{-1}\rangle$. By inverting again the direction of the arcs one obtains, of course, the original system. For this reason, a system $\langle C,\tau\rangle$ whose dynamical map τ is bijective is called *reversible*, and the system $\langle C,\tau^{-1}\rangle$ is called the *reverse* of $\langle C,\tau\rangle$. (In this context, we shall call *reversible* also the bijective map itself.) Each of the connected components of the state-transition diagram of a reversible system is called an *orbit*.

For a dynamical system $M = \langle C,\tau\rangle$ there are in general many different ways of classifying, or *labeling*, the points of the phase space. (Here, we shall use the term "label" in a slightly freer way than in Section 1.1; in particular, we shall allow labels to range over a nondenumerable set.) From a constructive viewpoint, the task of integrating the equations of motion of M starting from given initial conditions (i.e., from a given point $c\in C$) consists in finding explicit labels for the points $\tau^i c$ ($i\in Z$, if M is reversible; $i\in N$, otherwise) *in an assigned labeling schema*. Given the dynamical map τ, the difficulty of the above task depends on the choice of the labeling schema. A peculiar property of *reversible* systems is that they always admit of *a labeling schema in which the integration of the equations of motion is trivial*. Essentially, it is sufficient to assign a distinct but otherwise arbitrary label to each orbit, and to sequentially number the individual points of each orbit

according to the order in which they appear in the orbit itself. Such sequential numbering can start from an arbitrary point of the orbit; if the orbit is cyclic, the numbering will be over the finite ring Z_n (where n is the length of the cycle); otherwise, over Z. Thus, one associates with each orbit (i) an orbit label r, (ii) a ring Z_{k_r}, and (iii) an arbitrary point of the orbit in correspondence with the zero element of the ring. In this way, each point $c \varepsilon C$ is uniquely identified by an ordered pair of the form $\langle r,s \rangle$, where r specifies the orbit and $s \varepsilon Z_{k_r}$ specifies the position of c on the orbit itself. The advantage of this labeling schema is that, if $\langle r,s \rangle$ is the label of a given point c, then the label of any point $\tau^i c$ $(i \varepsilon Z)$ is $\langle r,s+i \rangle$, where "+" denotes addition modulo-k_r. Clearly, in this labeling schema the integration of the equations of motion reduces to addition over an integer ring.

We shall illustrate the above concepts in the context of classical mechanics, with which the reader is assumed to have a cursory familiarity. Let us consider a time-independent mechanical system $\langle C,\tau \rangle$ having n degrees of freedom. (As explained in Section 1.1, in this case the powers of τ make up a continuous one-parameter semigroup. The group parameter t is identified with time.) A generic configuration c of the system can be described by means of $2n$ coordinates $q_1,\ldots,q_n,p_1,\ldots,p_n$, more concisely written as $\vec{q}$, $\vec{p}$. As c, which we shall treat as a function of time, evolves along an orbit, the quantities $\vec{q}$, $\vec{p}$ also vary with time. There exist, however, functions of these quantities whose values, expressed

in terms of time treated as an independent variable, remain constant during the motion and depend only on the initial conditions. Such functions are called the integrals of the motion (cf.[LAND60]). The number of independent integrals of the motion of the system is $2n-1$. In fact, the general solution of the equations of motion (which can be expressed, for instance, as n differential equations of the second order) contains $2n$ arbitrary constants. Of these, one can always be taken as an additive constant t_0 in time (since the system is time-independent). Eliminating $t+t_0$ from the $2n$ functions $\vec{q} = \vec{q}(t+t_0,r_1,\ldots,r_{2n-1})$, $\vec{p} = \vec{p}(t+t_0,r_1,\ldots,r_{2n-1})$, one can formally express the $2n-1$ constants $r_1,r_2,\ldots,r_{2n-1}$ (more briefly written as $\vec{r}$, a $(2n-1)$-dimensional vector) as functions of $\vec{q}$ and $\vec{p}$. In general, the value of these functions will change as c, considered as an independent variable, is arbitrarily moved over the phase space C. However, if the configuration c is moved along an orbit, i.e., in conformity with the equations of motion, these functions will yield constant values, which are thus characteristic of the orbit as a whole. One can replace the labeling schema $\langle\vec{q},\vec{p}\rangle$ for configurations by the schema $\langle\vec{r},t+t_0\rangle$. In the latter schema, as long as a configuration moves along an orbit, only the last component varies and its change coincides with the time elapsed since the system was in the initial conditions. Thus, in order to integrate the equations of motion for a time interval t starting from initial conditions $\langle s,t_0\rangle$ it is sufficient to replace t_0 by $t+t_0$.

EXAMPLE 3.1.1. Consider the motion of a particle in a plane under the influence of a central gravitational field. The system has two degrees of freedom. The dynamical state of the system at any moment can be expressed by giving the position (x and y) and the velocity ($\dot{x}$ and $\dot{y}$) of the particle. One can integrate the equations of motion by updating, instant by instant, the four quantities $x,y,\dot{x},\dot{y}$ in conformity with the differential equations that relate them. Alternatively, one can describe the state of the system by means of the following four quantities:

E, the energy of the orbit,

e, the eccentricity of the orbit,

θ, the orientation of the major axis, and

ϕ, the phase of the particle on the orbit.

The first three quantities are constant and represent three independent integrals of the motion. The fourth quantity obeys a single ordinary differential equation (such a quantity is called a cyclic coordinate) and can be expressed in such a way as to coincide with t modulo-T , where t is time and T the period of the orbit. Thus, in this labeling schema, the integration of the equations of motion reduces to addition modulo-T.

The reader must be warned that the above considerations lead, in a sense, to vacuous results (Poincaré uses the term "tautological" [POIN02]). In fact, the computational problems encountered in passing from a given labeling schema of the form $\langle \vec{p},\vec{q} \rangle$ to one of the form

$\langle\vec{p},\vec{q}\rangle$ and vice versa are often more difficult than those encountered in integrating the equations of motion directly in the $\langle\vec{p},\vec{q}\rangle$ schema. In general, in order to express an integral of the motion in terms of coordinates and momenta one requires functions that are not analytical or continuous (see [BRIL74] for a thorough discussion), and only energy (an integral of the motion shared by all isolated systems) is always a well-behaved function of such variables. However, the constancy of certain integrals of the motion "is of profound significance, deriving from the fundamental homogeneity and isotropy of space and time. The quantities represented by such integrals of the motion are said to be conserved, and have the important common property of being additive. Their values for a system composed of several parts whose interaction is negligible are equal to the sums of their values for the individual parts."[LAND60]

From the viewpoint of the present work, the relevant question is whether concepts such as "number of degrees of freedom, "integral of the motion," "conserved quantity," "cyclic coordinate," etc. are meaningful only in the context of physical mechanics or can be applied successfully to other kinds of dynamical systems such as cellular automata. In order to attempt this transfer of concepts, one has to give explicit consideration to many factors that in a more restricted context are usually taken for granted.

EXAMPLE 3.1.2. Consider the concept of "number of degrees of freedom of a system," which is (typically) defined as the number of

independent quantities which must be assigned in order to specify uniquely the system's spatial configuration. In such definition, a number of assumptions are tacitly made. First, by "quantities" one means real quantities (i.e., quantities expressible by a real number). Second, the system must be reversible (otherwise the number of degrees of freedom may vary with time). Third, the term "independent" is meaningless unless one considers, alongside the phase space C, also a particular dynamical map τ. In fact, it is well known that one can establish a one-to-one correspondence between R^n and R (for any integer n). Thus, a single real number is sufficient to specify uniquely the spatial arrangement of a number of particles. However, the correspondence between R^n and R is not, in general, transformed in a continuous way under the action of the dynamical map. If one adds the continuity constraint, then more than one real variable may be necessary to describe a point of the phase space. In brief, the concept of "number of degrees of freedom" is related to a particular topology on the phase space, i.e., that for which the dynamical map is a continuous function. If one uses such topological-continuity criteria in order to construct an analogous concept for cellular automata, one is led to a schema where one degree of freedom is associated with each cell (cf. Section 5.3), and the range of the dynamical variable associated with each degree of freedom coincides with the cellular automaton's state alphabet.

EXAMPLE 3.1.3. It is easy to verify (we shall do it in a trivial way) that the concept of "complete set of integrals of the

motion" is a meaningful one for cellular automata. Take, for instance, the collection of all dynamical variables of the form c_s (cf. Remark 1.2.1). By assigning a specific cell-state value to each variable, one obtains a particular configuration $\bar{c}$, which belongs to exactly one orbit. The collection of such values can be used as a label for that orbit, and any other configuration c belonging to that orbit can be uniquely identified (if the cellular automaton is reversible) by specifying the number of time steps that separates it, in a given time direction, from $\bar{c}$. Thus, one obtains a labeling schema where to each configuration of the form $c = \tau^i\bar{c}$ there is assigned a label of the form $\langle\bar{c},i\rangle$, where $\bar{c}$ identifies the orbit and i the position on the orbit. Clearly, the value of the s-component of the fixed point $\bar{c}$ does not change as the point c evolves on the orbit. Thus, the collection of quantities $\langle c_s \rangle_{s\in S}$ constitutes a set of integrals of the motion. Such set is <u>complete</u> (it uniquely specifies an orbit) but redundant (there is more than one way to specify the same orbit).

From the above examples, it should be clear that many of the most important concepts of theoretical mechanics are specific to physical systems only insofar as these are assumed to be reversible. Therefore, if satisfactory analogies for such concepts are to be found in the theory of cellular automata, they must be sought within the more restricted context of <u>reversible</u> cellular automata.

3.2 Reversible cellular automata

Very little has been written about reversible cellular automata. A local map that shifts the contents of each cell one or more positions, say, to the right obviously characterizes a reversible cellular automaton. So does a local map that replaces every cell-state by another one according to a given permutation schema. Any composition of maps of the above kinds again leads to reversible behavior. In all these cases, one has to do with neighborhood templates that consist of only one element; thus, aside from a possible shift of coordinates at each time step, each cell constitutes an isolated system. (For this reason, the corresponding parallel maps are called trivial in [AMOR72].) In systematic search for nontrivial reversible parallel maps, Yamada and Amoroso[YAMA70] found none for one-dimensional, two-state cellular automata with neighborhood template of size $n<4$, while for $n=4$ Patt[PATT71] found exactly eight maps (of which four are compatible with a quiescent state) out of a total of 2^{16}. In [AMOR72], Amoroso and Patt give a general procedure for deciding whether a given local map yields a reversible parallel map, but only for the one-dimensional case. Their hopes that such procedure could be generalized to more than one dimension has not materialized yet. In the same paper, they give a decision procedure for surjectivity, also for the one-dimensional case. It has been proved (cf. [GOLZ76]) that surjectivity is, in

general, undecidable for cellular automata having more than one dimension. This may well turn out to be the case also for bijecvity (i.e., reversibility). The analogous problem in the continuous case (i.e., whether a system of partial differential equations has all of its solutions nonsingular) is also far from satisfactorily solved [MACK63]. At any rate, given the relative rarity (cf. next paragraph) of nontrivial reversible maps, a decision procedure such as that of [AMOR72] is of little help, by itself, as a tool for discovering new ones.

A necessary condition for reversibility of the parallel map is that the local map (and all of its iterates) be balanced, i.e., that all cell-states appear the same number of times as entries in the local-map table[PATT71, MARU76]. (This result was foreshadowed by Moore[MOOR62].) By using this constraint, one can drastically restrict the scope of the search. However, direct construction methods are more efficient. In [TOFF75], we present a technique for generating nontrivial reversible parallel maps. This technique is briefly illustrated below. Immediately following, we shall illustrate another construction technique of presumably much greater interest.

TECHNIQUE 3.2.1 (Guarded-context permutations). Consider the four reversible maps exhibited in [AMOR72]. Observe that these maps coincide with their inverses and, being nontrivial, lead to oscillations of period 2 (cf. [DIGR72]). The mechanism of such oscillations is easily explained. On one hand, all that each of those maps does is perform a state-alphabet permutation (here, exchange 0 for 1

and vice versa) if and only if the target cell is surrounded by a certain neighborhood-state context. For the four maps, such contexts are, respectively,

σ_1	σ_2	σ_3	σ_4
0*10	1*01	01*0	10*1

.

On the other hand, the context is chosen in such a way that a change made according to the above rule never creates or destroys an occurrence of the context elsewhere in the array. Since the "stimulus" context is preserved by the map and since permutations are bijective functions, it is clear that the above two conditions are sufficient to guarantee the reversibility of the parallel map. It is easy to realize such conditions in cellular automata of arbitrary dimensionality and state-alphabet size[TOFF75]. Moreover, this technique (as well as the following one) automatically produces the inverse along with the direct map. (Note that, while it is known that the inverse of a parallel map is still the parallel map of a cellular automaton[RICH72], no general method is known for constructing it.)

TECHNIQUE 3.2.2 (Reversible primitives). The present technique permits one to construct reversible distributed automata (viz., sequential networks) that are finite or infinite, uniform or nonuniform, as desired. Here we shall give only an informal illustration. In Section 3.4 we shall use this technique for a particular application and formally prove that the cellular automata thus obtained are indeed reversible. Enough machinery is supplied with that proof for the

reader to be able to extend it to the general case.

Intuitively, if a cell makes its own content available to the output neighbors for processing, and such neighbors are not able to appropriately coordinate their action, some of the information supplied by the cell may end up being destroyed or duplicated--both being irreversible processes. In order to insure reversibility, one may restrict the processing of information only to predetermined "safe" cases, as done in the preceding technique. Another solution is for the cell to hand over to each of its output neighbors an independent piece of information, so that interneighbor conflicts on how to dispose of that information can be avoided. At this point, it is sufficient that each output neighbor reversibly process the information thus gathered from its own input neighbors.

More formally, we shall consider sequential networks consisting, as usual, of delay elements and switching elements (for simplicity, we shall consider only binary signals). Each arc will contain exactly one delay element, and each node will consist of a switching element having as many output as input lines, and thus input set and output set of the same size. We shall treat the input [output] state of a node as a vector whose components are the states of the individual input [output] lines. In order to insure reversibility, we shall use only switching elements that establish a one-to-one correspondence between the set of values for the input vector and that for the output vector. Consider, for example, a switching element λ having three input lines and three output lines. Thus, input set and output

set both consist of eight elements. λ is required to realize a permutation over $\{0,1\}^3$. For instance, λ could be defined by the table

$$\xrightarrow{\lambda}$$

$\langle p_1,p_2,p_3\rangle$	$\langle q_1,q_2,q_3\rangle$
0 0 0	0 0 1
0 0 1	0 1 0
0 1 0	1 0 0
0 1 1	1 1 0
1 0 0	0 1 1
1 0 1	1 0 1
1 1 0	0 0 0
1 1 1	1 1 1

Analogous considerations apply to each node. In order to verify the reversibility of the behavior of a network consisting only of nodes of this kind (Figure 3.3), consider the collective state D ($=\langle D_1,D_2,\ldots\rangle$) of its delay elements,

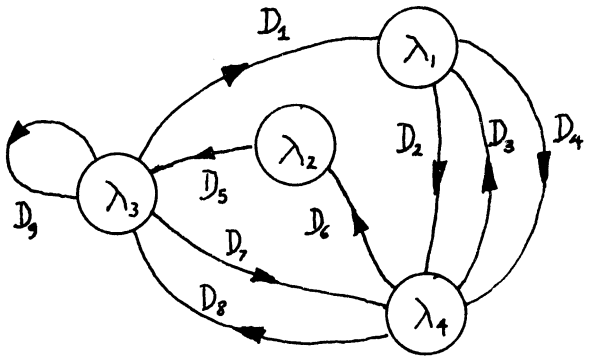


FIGURE 3.3. A network consisting only of reversible primitives. Each arc contains exactly one delay element (not indicated in the figure).

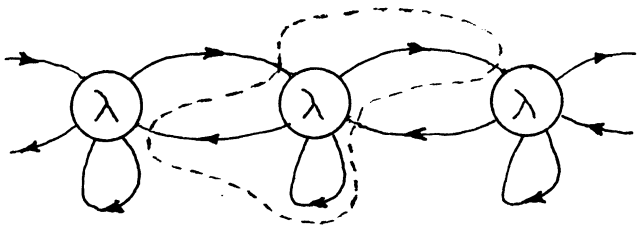


FIGURE 3.4. A uniform, reversible sequential network, Each node, together with the delays implicitly associated with each output arc, can be interpreted as a cell of a reversible cellular automaton.

and compute their next state D' . It is clear that D can be univocally reconstructed starting from D' by using the permutation table associated with each node (such as the one given above) "from right to left." Thus, the network's global transition function (which transforms D into D') is bijective.

If one uses Technique 3.2.2 with the additional constraint that the nodes be identical and uniformly connected, one obtains a reversible uniform automaton. One can visualize each cell as consisting of one switching element together with the delay elements that appear on its output arcs, as, for instance, in Figure 3.4, where (assuming binary signals on all arcs) the state alphabet is the set $A = \{0,1\}^3$, of size 8 , the neighborhood template is the set $X = \{-1,0,1\}$, and the local map, of the form $\lambda: A^X \to A$, depends only on three of the nine components of its domain A^X $(=\{0,1\}^9)$ and actually reduces to a permutation over A .

By suitably assigning the initial state of a number of cells (and assuming that all other cells are initially in a well-determined state) one can embed in a cellular automaton structures designed to perform certain computing or constructing tasks. The range of capabilities that can be "programmed" in such embedded structures ultimately depends on the nature of the cells themselves. To what extent is this range restricted if one uses reversible cellular automata?

Computation involves the evaluation of functions, and functions are, in general, many-to-one maps. It is apparently paradoxical that one should be able to realize many-to-one maps in a computing medium

governed by a one-to-one transition function. In our opinion, it is essentially this problem that induced many authors (cf. Section 3.0) to express doubts about the computing capabilities of reversible cellular automata. The paradox is resolved by observing that, in a distributed computing medium, the value of a function that one may be interested in computing is not necessarily represented by the state --at a certain instant--of the whole medium, but only by the state of a certain portion of it. More precisely, in interpreting in terms of computing activity the behavior of a dynamical system, one is interested in the relationship between certain state variables defined on the system's phase space (cf. Section 1.1). One such variable may be represented, for instance, by the state at time t of a certain cell s in which the argument of the desired function is encoded, while another state variable may be represented by the state at time $t+t'$ of a cell s' where the value of the function is read off. The relation between such two state variables need not be bijective (or, for that matter, deterministic) even if the system's dynamical map is bijective. Therefore, one cannot rule out a priori nontrivial computing capabilities for reversible cellular automata.

Indeed, in the next three sections we shall prove the computation- and construction-universality of certain such automata. To this purpose, we shall use the following approach. Instead of exhibiting a particular universal computer/constructor in an ad hoc cellular automaton (as done, for instance, in [VONN66, CODD68]), we shall show how to embed any cellular automaton in a reversible one having one

more dimension. Like other embedding techniques that have appeared in the literature (cf. [SMIT69]), ours obeys certain constructive constraints (quiescent state, straightforward encoding and decoding etc.) which guarantee that the computing capabilities one arrives at are attributable to the cellular automaton itself rather than to spurious contributions originating from the embedding technique.

3.3 Embedding cellular automata in reversible ones

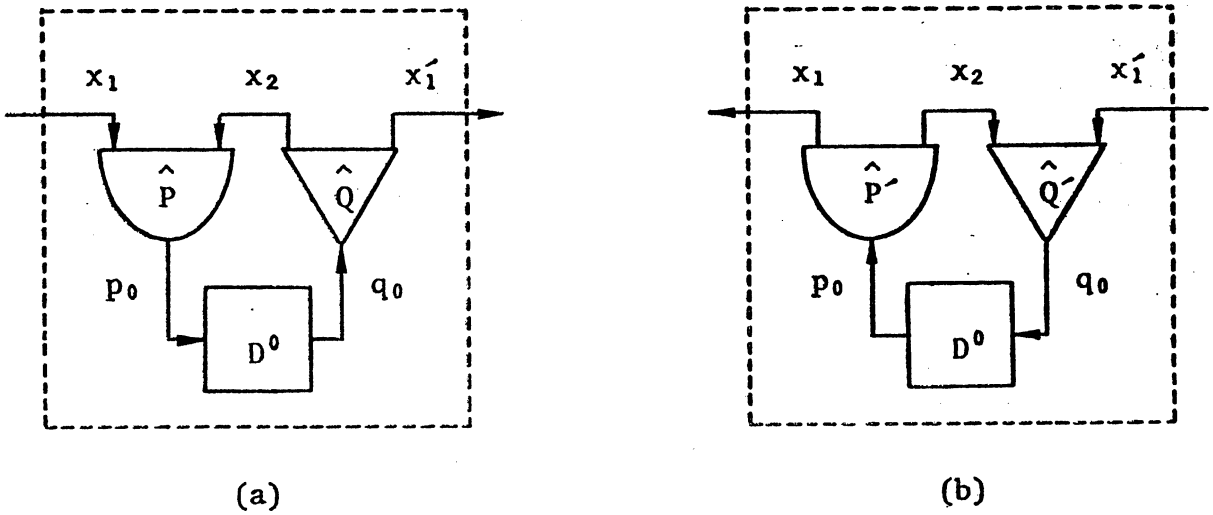
In this section we informally illustrate our embedding technique by means of a simple example. The discussion is intended as a guide to the general procedure developed in the next section.

Let us consider a one-dimensional cellular automaton $\hat{M}$ with state alphabet $\{0,1\}$ (this is the cell-state set), neighborhood index $<-1,0>$ (a cell's state at time $t+1$ depends on that of the cell itself and that of its left neighbor at time t), and local map $\hat{\lambda}$ defined by the following table: $\{00\to0, 01\to0, 10\to0, 11\to1\}$ (this is the transition function for an individual cell). Let $\hat{\tau}$ be the parallel map of $\hat{S}$ (i.e., the corresponding transition function for the whole cellular automaton). For convenience, we shall represent the cellular automaton by an iterative sequential network consisting of elementary networks as in Figure 3.5a connected in an infinite row. The output state of unit delay D^0 represents the state of a cell, the truth table of AND-gate $\hat{P}$ coincides with the

local-map table, and signal splitter $\hat{Q}$ distributes the cell's state to the output neighbors (in this case, the cell itself and its right neighbor).

Let us see what would happen if we tried to run the cellular automaton backwards in time. For this purpose, we would need to reverse the direction of signal flow on each arc of the network and to reverse the behavior of each node, exchanging inputs with outputs, as in Figure 3.5b. Obviously, two difficulties would arise:

(a) When operated "in reverse", signal splitter $\hat{Q}$ would become a device $\hat{Q}'$ that attempts to merge signals. If the incoming signals x_2 , x_1' did not agree there would be a contradictory assignment for q_0 , i.e., the backwards behavior of $\hat{Q}$ would be overspecified.



(a) (b)

FIGURE 3.5. The original network consists of cells as in (a) connected in an infinite row. In (b), we attempt to "reverse" the behavior of such cell.

(b) On the other hand, the "reverse" version $\hat{P}'$ of AND-gate $\hat{P}$ would have more than one choice for outputs x_1 , x_2 when $p_0 = 0$, i.e., the backwards behavior of $\hat{P}$ would be underspecified.

Clearly, in order to insure reversibility, one must modify the given cellular automaton. To this purpose, we shall augment the tables which specify $\hat{P}$ and $\hat{Q}$ (Figure 3.7a) with the aim of obtaining invertible combinatorial functions, as, for instance, P and Q of Figure 3.7b. P and Q are clearly invertible, since they define a permutation; their inverses are, respectively, $\overline{P}$ and $\overline{Q}$ (Figure 3.7b).

How shall we connect the arcs corresponding to the newly introduced input and output variables (respectively, x_0 , q_1 and p_1 , p_2 of Figure 3.5b), if we want the resulting cellular automaton to reproduce, under certain conditions, the behavior of the original one? We shall introduce an additional spatial dimension and associate to each cell a new input neighbor and a new output neighbor lying along this dimension, as in Figure 3.6a (to be precise, the effective neighborhood extends farther, since signals on the q_1-x_1' path "turn the corner" and encounter a delay only in the next adjacent cell); the new arcs p_1 , p_2 will be routed through two additional delays, D^1 and D^2 , in order to avoid instantaneous propagation of signals over an infinite distance.

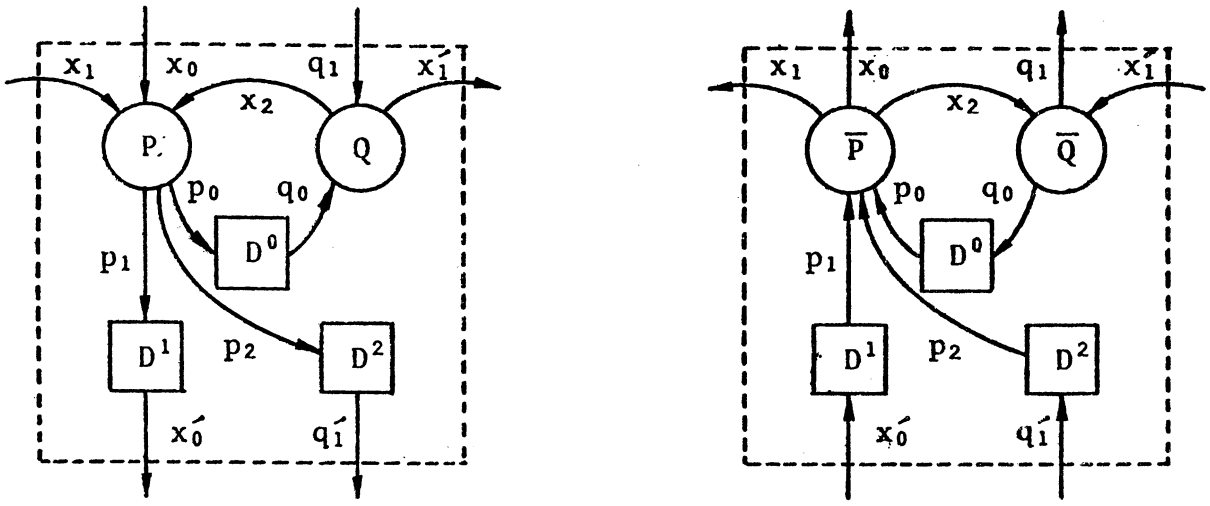


FIGURE 3.6. (a) The expanded, reversible network derived from the original one (Figure 3.5a), and (b) its reverse.

$\hat{P}$

x_1	x_2	p_0
0	0	0
0	1	0
1	0	0
1	1	1

$\hat{Q}$

q_0	x_2	x_1'
0	0	0
1	1	1

(a)

P

x_0	x_1	x_2	p_0	p_1	p_2
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	0	1	1

$\overline{P}$

Q

q_1	q_0	x_2	x_1'
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	1

$\overline{Q}$

(b)

FIGURE 3.7. The elements $\hat{P}$ and $\hat{Q}$ of the original cell (figure 3.5a) are specified by the tables in (a). In (b), such tables have been augmented in order to insure reversibility. Note that the boxed portion of the new tables coincides with that of the original tables. Read from left to right, the tables in (b) specify the network elements P and Q (cf. Figure 3.6a). read in the opposite direction, they specify the elements $\overline{P}$ and $\overline{Q}$ used in the reverse network (cf. Figure 3.6b).

In this way, we obtain a two-dimensional iterative network consisting of cells as in Figure 3.6a. This network is reversible (i.e., at each moment the state of the whole network has exactly one predecessor). If τ is the function that associates with each network state its successor, the inverse function $\bar{\tau} = \tau^{-1}$ is realized by the reverse network consisting of cells as in Figure 3.6b.

Thus, starting from a one-dimensional, irreversible cellular automaton $\hat{M}$ with parallel map $\hat{\tau}$, we have constructed a two-dimensional, reversible cellular automaton M with parallel map τ. Now, we show that, under a suitable correspondence rule, M simulates $\hat{M}$ in a sense explained below.

Let $\hat{c}$ be an arbitrary initial configuration of $\hat{M}$, i.e., an assignment of states to all cells in $\hat{M}$. The corresponding initial configuration c of M is constructed by using the same state assignment in initializing the D^0 delays of an arbitrary row ρ of M, and by setting all other delays of M to state 0. It is easy to verify that the behavior of M, restricted to the state of the D^0 registers of row ρ, exactly reproduces the behavior of $\hat{M}$.

In this example, we have achieved the goal of reproducing the behavior of cellular automaton $\hat{M}$ in a portion of a suitably initialized reversible cellular automaton M. Note that the above construction will work for any truth table $\hat{P}$ compatible with a quiescent state (the first row in both tables $\hat{P}$ and $\hat{Q}$ of Figure 3.7a contains all zeroes--thus state 0 is self-maintaining, or

quiescent), not merely with an AND function. Presently, we shall show how the construction can be adapted to cellular automata that do not admit of a quiescent state. In the next section, in addition to working in a more formal setting, we shall generalize the construction to cellular automata having an arbitrary number of dimensions and of neighbors, an arbitrary state-alphabet size, and an arbitrary local map.

The effectiveness of the simulation of $\hat{M}$ by M is based on the fact that, for an appropriate choice of initial conditions, the selected row ρ of M in which $\hat{M}$ is embedded is fed with constant signals (all 0, in the example) coming from above through the x_0, q_1 arcs (Figure 3.6). In such conditions, the cells of ρ use only the upper part of the tables of Figure 3.7b, as required for a correct simulation. Moreover, the initial condition whereby the top half plane (see Figure 3.8) is loaded with all 0's is self-sustaining, since the $\langle 0,0,0 \rangle$ cell state is quiescent (and, with the given neighborhood, no signals come from below).

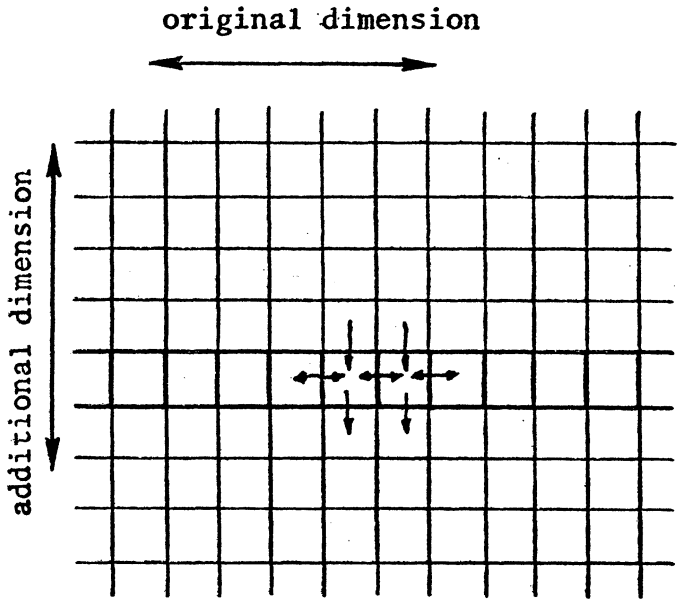


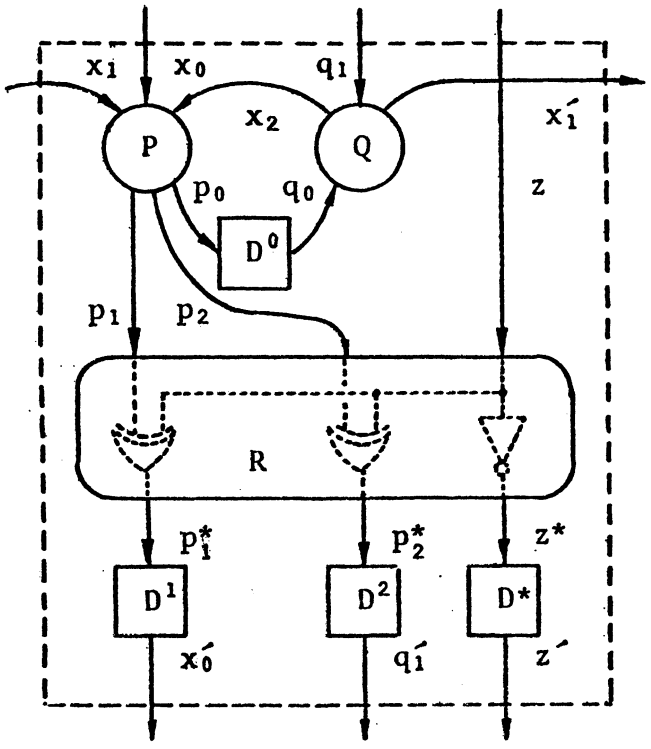
FIGURE 3.8. The simulation of the original cellular automaton takes place in a selected row ρ of a two-dimensional cellular automaton. The half-space above is initialized with all zeros.

Suppose, now, that the cellular automaton's local map is replaced by the following: {00→1,01→1,10→1,11→0} (NAND function), which admits of no quiescent state. Going through the construction as before, we obtain for P the following table:

P →

x_0	x_1	x_2	p_0	p_1	p_2
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	1	1	0
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	0	0	1
1	1	0	0	1	0
1	1	1	1	1	1

If a row of cells as in Figure 3.6a is initialized with all 0 , and supposing that all its x_0 , q_1 inputs are fed with 0 , the delays D^0 of that row will all oscillate in synchronism between states 0 and 1 , with a cycle of length 2 . In order to have the required output (all 0) from arcs x_0' , q_1' , we shall add to the basic cell a modulo-2 counter consisting of delay D* and an inverter, and shall XOR its output with the p_1 , p_2 lines, as shown in Figure 3.9. Again, one can verify that the iterative network consisting of



(a)

R

z	p1	p2	z*	p1*	p2*
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	1	1	0
0	1	1	1	1	1
1	0	0	0	1	1
1	0	1	0	1	0
1	1	0	0	0	1
1	1	1	0	0	0

(b)

FIGURE 3.9. In (a), delay D* and combinatorial function R have been added to the cell of Figure 3.6a. R , defined by the table in (b), is invertible. It may be visualized, as indicated in (a), as a network containing an inverter (which transforms delay D* into a modulo-2 counter) and two XOR-gates (which, under suitable conditions, guarantee a constant output from x_0' , q_0' in spite of the oscillations of D^0).

cells as in Figure 3.9 is reversible, and that, with the same initial conditions as before, the simulation proceeds correctly. The whole upper half-plane of Figure 3.8, initialized with all zeroes, will now oscillate between 0 and 1 .

3.4 Computation- and construction-universality

The embedding procedure illustrated in this section closely follows the example given in Section 3.3. Here, however, the procedure is defined in a formal way and developed in its full generality. In particular,

(1) The given cellular automaton's state alphabet may have an arbitrary number r of elements. Correspondingly, the iterative network which realizes it will work with r-ary (instead of binary) logic, and all its lines and delays will handle r-ary signals.

(2) A cell may have an arbitrary number n of neighbors. Consequently, $\hat{P}$ will be a combinatorial function of n variables and $\hat{Q}$ an n-way splitter (cf. Figure 3.10).

(3) The cellular automaton may have any number of dimensions d . However, aside from the fact that the embedding will be realized in d+1 dimensions, the number of dimensions d need not explicitly appear in the formulas that define the embedding procedure.

The labeling of nodes and arcs will closely follow that

introduced in Section 3.3. In particular, starting from an iterative sequential network with cells as in Figure 3.10, we shall construct a reversible one, with cells as in Figure 3.11a together with its reverse, with cells as in Figure 3.11b. P is an invertible combinatorial function whose input and output are both (n+1)-tuples, while both input and output of Q are n-tuples.

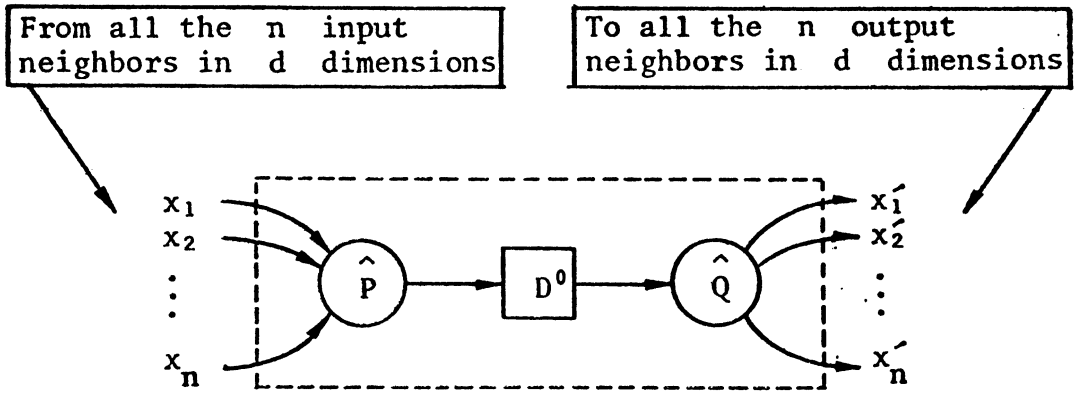


FIGURE 3.10. The original cellular automaton is represented by a d-dimensional iterative network with cells as above. No attempt has been made to indicate the relative position of the input and output neighbors in the d-space. In particular, some of the x' may feed back into some of the x (if the cell is a neighbor of itself). $\hat{P}$ coincides with the local map λ, while $\hat{Q}$ is an n-way splitter.

Since the functions P, Q and the delays $D^0, \ldots, D^n$ are invertible and, moreover, every feedback loop includes a delay, the whole (d+1)-dimensional network consisting of cells as in Figure 3.11 is reversible. Moreover, with a suitable initialization, the D^0 delays of a d-dimensional hyperplane of this network will reproduce the behavior of the original cellular automaton.

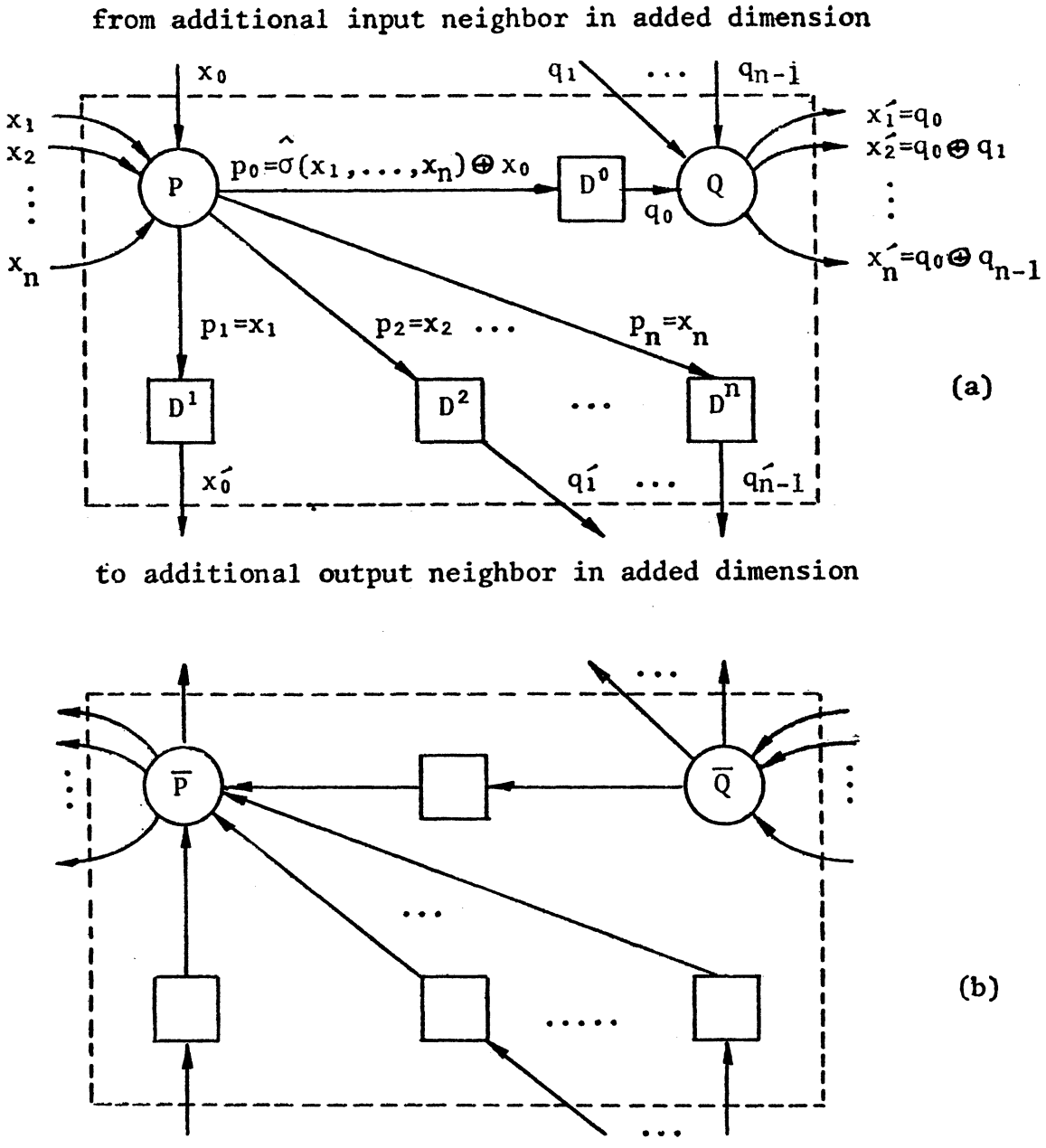


FIGURE 3.11. (a) A cell of the expanded, reversible network in d+1 dimensions. Inputs $x_1,\ldots,x_n$ and outputs $x'_1,\ldots,x'_n$ are connected with neighbors in the original d dimensions, as in Figure 3.10. In (b), we show a cell of the reversed network. The direction of all arcs has been reversed, and P and Q have been replaced, respectively, by the inverse functions $\overline{P}$ and $\overline{Q}$.

(DEFINITION 3.4.1) given an arbitrary finite set A (for instance, a cellular automaton's state alphabet) and positive integers m, n, a combinatorial function F is a mapping of the form $F: A^m \to A^n$. A permutation P on A^n is a bijective combinatorial function $P: A^n \to A^n$. Let $a = \langle a_1,\ldots,a_n \rangle$, $b = \langle b_1,\ldots,b_n \rangle$, and $b = Pa$; then P_i, the i-component of P, is defined by $b_i = P_i a$.

Note that a permutation having as argument an n-tuple from A^n does not permute the elements of the n-tuple; rather, it replaces the whole n-tuple with another chosen from A^n according to a bijective substitution rule. If a permutation and its inverse are applied in succession, the result will be the original n-tuple with each component exactly in the same order. Thus $P^{-1}P$ may be decomposed into a set of n identity functions each one operating independently on a component of the n-tuple. Trivial though it may seem, this result, synthesized in the following lemma, will prove extremely useful in our construction.

LEMMA 3.4.1. Let P be a permutation on A^n and $\overline{P} = P^{-1}$. Let $a \varepsilon A$. The following identity holds:

$$\bigwedge_{i=1}^{n} \overline{P}_i Pa = a_i .$$

Proof.

$$\overline{P}Pa = a \implies \left\langle \overline{P}_i Pa \right\rangle_{i=1}^{n} = \left\langle a_i \right\rangle_{i=1}^{n} \implies \bigwedge_{i=1}^{n} \overline{P}_i Pa = a_i .$$

In the remainder of this section we specify in detail the required embedding procedure. Given an arbitrary cellular automaton $\hat{M}$, we shall construct two new automata, M and $\overline{M}$. Theorem 3.4.1 will show that $\overline{M}$ is the reverse of M and that, therefore, both are reversible. Theorem 3.4.2 will show that, with a suitable correspondence rule, M simulates $\hat{M}$ (cf. Definition 3.4.3).

DEFINITION 3.4.2. Given an arbitrary cellular automaton $\hat{M}[\hat{C}[A(r),\hat{S}(d)],\hat{\tau}[\hat{X}(n),\hat{\lambda}]]$, we define two new cellular automata, $M[C[A^{n+1},S(d+1)],\tau[X(2n+1),\lambda]]$ and $\overline{M}[C[A^{n+1},S(d+1)],\overline{\tau}[\overline{X}(2n+1),\overline{\lambda}]]$. The neighborhood templates X, $\overline{X}$ and the local maps λ, $\overline{\lambda}$ are specified, respectively, in Construction 3.4.1 and Construction 3.4.2.

For a configuration $c \varepsilon C$, the states c_s of cell s in c is an element of A^{n+1}. The $n+1$ components of c_s are denoted by $c_s^0,\ldots,c_s^n$. If M is represented as an iterative network with cells as in Figure 3.11a, the components of c_s are identified with the output states of the $D^0,\ldots,D^n$ delays associated with the corresponding cell. In a similar way, the components of a cell's state are associated with the D delays of Figure 3.11b.

The neighborhood template $\hat{X}$ of $\hat{M}$ is an n-tuple of displacements (n is the number of neighbors). In turn, each displacement is a d-tuple of integers (d is the number of dimensions). In symbols,

$$\hat{X} = \underset{i=1}{\overset{n}{<}} \hat{X}_i > = \underset{i=1}{\overset{n}{<}} \underset{j=1}{\overset{n}{<}} \hat{X}_i^j >> \quad .$$

In the neighborhood template X of M each displacement will be a (d+1)-tuple of integers (since we have added one dimension). Intuitively, to the n neighbors in the d original dimensions we shall add n+1 neighbors in the additional dimension; namely, the cell immediately above a given cell s (cf. (3.4.1a) below) and the n cells immediately above the n original neighbors of s (3.4.1c). This is specified formally in Construction 3.4.1 below.

Note. In what follows, the expression $\underset{i=m}{\overset{n}{[}} a_i]$ will be used as an abbreviation for the string "$a_m, a_{m+1}, \ldots, a_n$".

CONSTRUCTION 3.4.1. The neighborhood index X of M is defined as the ordered set $\underset{i=0}{\overset{2n}{<}} \underset{j=0}{\overset{d}{<}} X_i^j >>$ whose elements are specified by

$$X_0 = \langle 1, \overset{d}{\underset{j=1}{[}} 0]\rangle \tag{3.4.1a}$$

$$X_i = \langle 0, \overset{d}{\underset{j=1}{[}} X_i^j]\rangle \tag{3.4.1b}$$

$$X_{i+1} = X_i + X_0 \tag{3.4.1c}$$

(for $i=1,2,\ldots,n$)

The neighborhood index $\overline{X}$ of $\overline{M}$ is constructed by taking the opposite of each element of X, i.e., $\overline{X} = -X$.

Referring to Figure 3.11a, we see that the delays $D^0,\ldots,D^n$ are immediately affected by inputs $x_1,\ldots,x_n$ (thence the assignment (3.4.1b) above) and by input x_0 (thence (3.4.1a)). Moreover, inputs $q_1,\ldots,q_{n-1}$ immediately affect the outputs $x_1',\ldots,x_n'$ without any interposed delays. This gives each cell n additional neighbors; thence (3.4.1c).

CONSTRUCTION 3.4.2. Let $\hat{\lambda}$ be the local map of cellular automaton $\hat{M}$ having state alphabet A and number of neighbors n (cf. Definition 3.3). By definition, $\hat{\lambda}$ is an arbitrary combinatorial function of the form $\hat{\lambda}: A^n \to A$. To $\hat{\lambda}$ we associate two permutations, P and Q, respectively, on A^{n+1} and A^n, defined by

$$P_h\Big(\overset{n}{\underset{i=0}{<}} x_i>\Big) = \begin{cases} \hat{\lambda}\Big(\overset{n}{\underset{i=1}{<}} x_i>\Big) \oplus x_0 & \text{(for } h=0\text{)}, \\ x_h & \text{(for } h=1,\ldots,n\text{)}; \end{cases} \tag{3.4.2a}$$

$$Q_h\Big(\overset{n-1}{\underset{i=0}{<}} q_i>\Big) = \begin{cases} q_0 & \text{(for } h=1\text{)}, \\ q_0 \oplus q_{h-1} & \text{(for } h=2,\ldots,n\text{)} . \end{cases} \tag{3.4.2b}$$

and their inverses, respectively $\overline{P}$ and $\overline{Q}$, explicitly written as follows

$$\overline{P}_h\Big(\overset{n}{\underset{i=0}{<}} p_i>\Big) = \begin{cases} p_0 \ominus \hat{\lambda}\Big(\overset{n}{\underset{i=0}{<}} p_i>\Big) & \text{(for } h=0\text{)}, \\ p_h & \text{(for } h=1,\ldots,n\text{)}; \end{cases} \tag{3.4.2c}$$

$$\overline{Q}_h\Big(\overset{n}{\underset{i=1}{<}} x'_1>\Big) = \begin{cases} x'_1 \\ x'_{h+1} \ominus x'_1 \end{cases} \tag{3.4.2d}$$

P and Q are the invertible combinatorial functions indicated in Figure 3.11a (similarly, $\overline{P}$ and $\overline{Q}$ appear in Figure 3.11b). It is easy to verify that, as long as the $x_0, q_1, \ldots, q_{n-1}$ inputs are held to state 0, P_0 reproduces the local map $\hat{\lambda}$ and Q acts as an n-way signal splitter, i.e., that

(LEMMA 3.4.2)

$$P_0\Big<0, \overset{n}{\underset{i=1}{[}} x_i]>\Big) = \hat{\lambda}\Big(\overset{n}{\underset{i=1}{<}} x_i>\Big) \quad \text{and} \quad Q\Big<q_0, \overset{n-1}{\underset{i=1}{[}} 0]>\Big) = \overset{n}{\underset{i=1}{<}} q_0> .$$

CONSTRUCTION 3.4.3. Let X be the neighborhood template defined in Construction 3.4.1 and P, Q the combinatorial functions defined in Construction 3.4.2. Then the local map λ of M is specified by

$$c'_s = \lambda \overset{2n}{\underset{i=0}{<}} c_{s+X_i} > = P<c^1_{s+X_0}, \overset{n}{\underset{j=1}{[}} Q_j<c^0_{s+X_j}, \overset{n}{\underset{k=2}{[}} c^k_{s+X_j+X_0}]>]> \quad . \tag{3.4.3a}$$

In a similar way, given $\overline{X}$, $\overline{P}$, and $\overline{Q}$, the local map $\overline{\lambda}$ of $\overline{M}$ is specified by

$$c''_s = \overline{\sigma} \overset{2n}{\underset{i=0}{<}} c'_{s+X_i} > = \tag{3.4.3b}$$

$$= <\overline{Q}_0 \overset{n}{\underset{j=1}{<}} \overline{P}_j\, c'_{s+\overline{X}_j}>, \overline{P}_0\, c'_{s+\overline{X}_0}, \overset{n-1}{\underset{j=1}{[}} \overline{Q}_j \overset{n}{\underset{k=1}{<}} \overline{P} c'_{s+\overline{X}_0+\overline{X}_k}>]> \quad .$$

Intuitively, given the cell structure of Figure 3.11a and the neighborhood relationship of Construction 3.4.1, in Construction 3.4.3 we have specified the interconnections between nodes (combinatorial functions and delays) of the whole uniform network.

X and λ [respectively, $\overline{X}$ and $\overline{\lambda}$] uniquely specify the parallel map τ of cellular automaton [the map $\overline{\tau}$ of $\overline{M}$].

THEOREM 3.1. Cellular automaton M is reversible, and $\overline{M}$ is its reverse.

Proof. In order to prove that τ is invertible it is sufficient to show that $\bar{\tau}\tau = 1$. Then τ is injective and, according to [RICH72], also bijective. (Alternatively, one could verify that also $\bar{\tau}\tau = 1$.)

By substituting c' of (3.4.3a) in (3.4.3b) we obtain

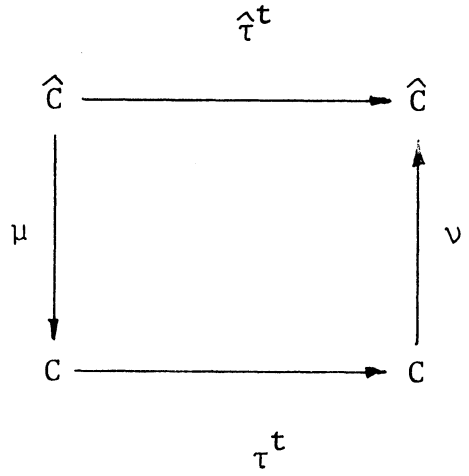
$$
\begin{aligned}
c''_s = \\
= \langle \bar{Q}_0 \overset{n}{\underset{i=1}{<}} \bar{P}_i \, P\langle c^1_{s+X_0-X_i}, \overset{n}{\underset{h=1}{[}} Q_h\langle c^0_{s-X_i+X_h}, \overset{n}{\underset{k=2}{[}} c^k_{s-X_i+X_h-X_0}]\rangle]\rangle\rangle, \\
\bar{P}_0 P\langle c^1_s, \overset{n}{\underset{h=1}{[}} Q_h\langle c^0_{s-X_0+X_h}, \overset{n}{\underset{k=2}{[}} c^k_{s+X_h}]\rangle]\rangle, \qquad (3.4.4) \\
\overset{n-1}{\underset{i=1}{[}} \bar{Q}_i \overset{n}{\underset{j=1}{<}} \bar{P} \; P\langle c^1_{s-X_j}, \overset{n}{\underset{h=1}{[}} Q_h\langle c^0_{s-X_0-X_j+X_h}, \overset{n}{\underset{k=2}{[}} c^k_{s-X_j+X_h}]\rangle]\rangle\rangle]\rangle \; .
\end{aligned}
$$

By repeated use of Lemma 3.4.1, and considering that $\bar{P}P = I$, $\bar{Q}Q = I$ (I denotes the identity function), (3.4.4) is eventually transformed into

$$
c''_s = \langle c^0_s, c^1_s, \overset{n}{\underset{i=2}{[}} c^i_s]\rangle = c_s \; .
$$

Thus, for every cell s, $c''_s = c_s$, i.e., $c = \bar{\tau}\tau c$. Therefore τ is bijective and $\bar{\tau}$ is its inverse.

DEFINITION 3.4.3. Let $\hat{M}$, M be cellular automata with, respectively, parallel maps $\hat{\tau}$, τ and configuration sets $\hat{C}$, C. M simulates $\overline{M}$ ($\hat{M}$ is embeddable in M) if there exist an injective encoding map $\mu: \hat{C} \to C$ and a decoding map $\nu: C \to \hat{C}$ such that, for all $t \geq 0$, $\tau^t \hat{C} = \nu \tau^t \mu \hat{C}$, i.e., if the following diagram is valid (cf. [SMIT69])



A pair of functions μ and ν having the above properties are said to define an embedding of $\hat{M}$ into M.

REMARK 3.4.1. In order to obtain an embedding that is realizable by means of an effective procedure one should appropriately restrict the choice of the functions μ and ν. Smith[SMIT69] requires that μ and ν be in some sense "simpler" than $\hat{\tau}$ and τ, obviously intending that no extension of the computing power of M should be hidden in μ and ν. A thorough analysis of these problems would lead to a very lengthy discussion (cf. e.g., [GOLZ75]). On the other hand, the encoding and decoding maps that we shall use in Theorem 3.4.2 below are quite safe in this respect, as they consist of straightforward copying procedures; in technical terms, they are, respectively, an extension and a projection with respect to fixed components of the set of configurations (treated as the Cartesian product of an infinite number of copies of the state alphabet).

THEOREM 3.4.2. An arbitrary cellular automaton having d dimensions is embeddable in a reversible one having $d+1$ dimensions.

Proof. Let $\hat{M}$ be an arbitrary cellular automaton, and M the corresponding reversible one of Definition 3.4.2. Let us consider first the case of $\hat{M}$ having a quiescent state 0. Recall that a cell $s \in \hat{S}$ is a d-tuple of coordinates of the form $\langle s_1,\ldots,s_d\rangle$. Similarly, a cell of S is a $(d+1)$-tuple. Let $\hat{c}$ denote a configuration of $\hat{M}$, and c one of M. It is easy to verify that the following maps μ, ν specify an embedding of $\hat{M}$ in M:

$$c = \mu\hat{c} \Longleftrightarrow c^{i}_{\langle s_0,s_1,\ldots,s_d\rangle} = \begin{cases} \hat{c}_{\langle s_1,\ldots,s_d\rangle}, & \text{for } i=0, s_0=0, \\ 0 & \text{otherwise} \end{cases}$$

(μ initially blanks the whole automaton M except for the d-dimensional hyperplane $s_0 = 0$ whose D^0 delays are initialized with a copy of $\hat{c}$);

$$\hat{c} = \nu c \iff \hat{c}_{<s_1,\ldots,s_d>} = c^0_{<0,s_1,\ldots,s_d>}$$

(ν copies back the state of the D^0 delays of hyperplane $s_0 = 0$ into $\hat{c}$). Note that μ and ν map finite configurations into finite ones.

If $\hat{M}$ has no quiescent state, we could modify our construction as shown, for a particular case, at the end of Section 3.3. However, we can prove the theorem much more briefly by means of the following argument.

Choose an arbitrary state $q \in A$ and consider the configuration c whose cells are all in state q. There is an $m \leq |A|$ such that $\hat{\tau}^m c = c$. We can construct a new cellular automaton $\hat{M}'$ with parallel map $\hat{\tau}' = \hat{\tau}^m$ such that $\hat{M}'$ simulates $\hat{M}$ with a speed-up factor of m (cf. [SMIT69]). By construction, q is a quiescent state of $\hat{M}'$ (cf. Section 1.4), and we can proceed to Definition 3.4.2 starting from cellular automaton $\hat{M}'$ instead of $\hat{M}$. Note that computation- and construction-universality are not affected by such speed-up.

It is well known that computation- and construction-universal cellular automata do exist. Therefore, by Theorem 3.4.2,

(COROLLARY 3.4.1) there exist computation- and construction-universal reversible cellular automata.

Our embedding of a cellular automaton in a reversible one is achieved at the cost of increasing the number of dimensions by one. The following problems arise:

(1) Whether an arbitrary cellular automaton can be embedded in a reversible one having the same number of dimensions.

(2) Whether computation- and construction-universal cellular automata exist at all in one dimension.

We conjecture that problem (1) has a negative answer. On the other hand, certain fragmentary findings seem to suggest a positive answer to problem (2).

3.5 Reversibility and the bounding problem for configurations

A computation-universal cellular automaton can play host to a universal Turing machine and, consequently, share its heritage of unsolvable problems. Observing that it is not decidable in general whether a Turing machine will restrict its activity to a bounded amount of tape, both Smith[SMIT69] and Aladyev[ALAD72] concluded that any computation-universal cellular automaton has an unsolvable

bounding problem for configurations (i.e., that it is not decidable whether the diameter of a configuration will eventually outgrow any bounds). Yet, as Smith pointed out, the Turing machine embedded in a portion of the automaton (by means of encoding and decoding functions analogous to those of Section 3.4) may halt while other portions of the automaton indefinitely expand their activity. Thus, one might suspect that the bounding problem for configurations is, in general, independent of the halting problem for an embedded Turing machine.

This turns out to be the case. Indeed, in Corollary 3.5.1 we prove the existence of computation-universal cellular automata in which every configuration has an unbounded propagation. For such automata, the bounding problem for configurations is clearly solvable, The existence proof may be extended to reversible cellular automata (Corollary 3.5.2).

We are in a better position, now, to understand the reason for the contrast between a proposition of Aladyev--that every computation-universal cellular automaton has a Garden-of-Eden configuration--and Corollary 3.4.1. Aladyev's proposition was based on a false assumption (that every computation-universal cellular automaton has an unsolvable bounding problem for configurations[SMIT68],[ALAD72], which is in direct contradiction with Corollary 3.5.1 and, consequently, with Corollary 3.5.2) and turns out to be false (being in direct contradiction with Corollary 3.4.1).

DEFINITION 3.5.1. (The present definitions, adapted, in the main, from [SMIT69], are meaningful only for cellular automata having

a blank state.) We recall that the distance between two cells s , s' is $\max_i(|s_i-s'_i|)$, i.e., the maximum difference between homonymous coordinates. (Other common definitions of distance, as, for instance, that based on the "city-block" metric, may be used as well without affecting the following definitions.) The diameter of a configuration is the maximum distance between pairs of nonblank cells.

Given parallel map τ , the propagation of a configuration c is the sequence $\langle \tau^t c \rangle_{t=0}^{\infty}$, in what follows simply written as $\langle c \rangle$. A propagation is bounded if the corresponding sequence of configuration diameters is bounded. The bounding problem for configurations of a cellular automaton M is to determine for any configuration c of M whether c is bounded. A cellular automaton M is propagation-unbounded if every configuration except the blank one has an unbounded propagation.

We shall prove that an arbitrary cellular automaton is embeddable in one that is propagation-unbounded (in the same number of dimensions). The following informal argument will introduce the technique used.

Consider the one-dimensional cellular automaton K[[{blank,signal},],[X,λ]] where X is the von Neumann neighborhood <-1,0,1> and the following local map: {<blank,blank,blank>↦blank,... any other entries...↦signal} . Any signal appearing in a configuration of K will indefinitely propagate right and left. Thus, K is clearly propagation-unbounded. One can easily construct a cellular

automaton having analogous properties in any number of dimensions. Moreover, an arbitrary neighborhood template having at least two distinct elements is sufficient to permit unbounded propagation.

Given an arbitrary cellular automaton $\hat{M}$ and a suitable K with the above properties, our intention is to "couple" K to $\hat{M}$ weakly enough so that both the computing capabilities of $\hat{M}$ and the propagation unboundedness of K are preserved in the resulting cellular automaton M. M will have a state alphabet which is the Cartesian product of that of $\hat{M}$ and of K. Thus, a configuration of M will have two components; the $\hat{M}$-component will reproduce the behavior of M, while the K-component will take care of propagating signals. The "coupling" is achieved by having any nonblank cell in the $\hat{M}$-component inject a new signal in the K-component. Thus, unless a configuration is all blank in both components, signals will appear in its K-component (if not already present at the outset).

CONSTRUCTION 3.5.1. Given an arbitrary cellular automaton $\hat{M}[[A,],[\hat{X}(n),\hat{\lambda}]]$ with blank state $0\epsilon A$, we define a new cellular automaton $M[[AxB,],[X(n+2),\lambda]]$, where B is the set $\{0,1\}$ and X and λ are constructed as below.

The first n components of X coincide with those of $\hat{X}$. The remaining two, X_{n+1} and X_{n+2}, are two arbitrarily chosen distinct displacements; if $\hat{X}$ contains at least two distinct components, X_{n+1} and X_{n+2} can be identified with them.

The state c_s of a cell s in a configuration c of M consists of two components, $c_s^{\hat{M}} \in A$ and $c_s^K \in B$. Correspondingly, the local map λ of M is defined by

$$c'_s = \langle c'^{\hat{M}}_s, c'^K_s \rangle = \lambda \left\langle \bigwedge_{i=1}^{n+2} c_{s+X_i} \right\rangle ,$$

where

$$c'^{\hat{M}}_s = \hat{\lambda} \left\langle \bigwedge_{i=1}^{n} c^{\hat{M}}_{s+X_i} \right\rangle , \tag{3.5.1a}$$

$$c'^K_s = \begin{cases} 0, & \text{if } \bigwedge_{i=n+1}^{n+2} c_{s+X_i} = \langle 0,0 \rangle \\ 1, & \text{otherwise} . \end{cases} \tag{3.5.1b}$$

In agreement with (3.5.1a), (3.5.1b), cell-state $\langle 0,0 \rangle$ is selected as the blank state of M.

The local map λ of M can be visualized as having two components. The $\hat{M}$-component of λ only deals with the $\hat{M}$-component of a configuration, and exactly reproduces the given map $\hat{\lambda}$ (the two added neighbors are ignored). The K-component of λ forces a 1 in the K-component of the state of a cell s if the state of either neighbor $s+X_{n+1}$, $s+X_{n+2}$ is not blank (i.e., has not a 0 in <u>both</u> components).

THEOREM 3.5.1. Any stable cellular automaton is embeddable in a propagation-unbounded one.

<u>Proof</u>. Let $\hat{M}$ be the given automaton, and M be defined as in Construction 3.5.1. It is easy to verify that the following

encoding and decoding functions, respectively μ and ν (merely copying routines) define an embedding of $\hat{M}$ in M :

$$c_s = \mu\hat{c}_s \Leftrightarrow \begin{cases} c_s^M = \hat{c}_s \ , \\ c_s^K = 0 \ ; \end{cases} \tag{4.2a}$$

$$\hat{c}_s = \nu c_s \Leftrightarrow \hat{c}_s = c_s^M \ , \tag{4.2b}$$

(where $\hat{c}$ and c denote, respectively, a configuration of $\hat{M}$ and one of M). Moreover, it is clear from 3.5.1b) that any nonblank configuration of M has an unbounded propagation.

COROLLARY 3.5.1. There exist computation- and construction-universal cellular automata whose bounding problem for configurations is solvable.

Proof. Referring to Theorem 3.4.1, if the original automaton $\hat{M}$ has the required computing capabilities, these are not affected by its embedding in M.

THEOREM 3.5.2. Any stable cellular automaton is embeddable in a propagation-unbounded, reversible one.

Proof. Let $\hat{\hat{M}}$ be the given cellular automaton, and $\hat{M}$ the propagation-unbounded one obtained from $\hat{\hat{M}}$ by means of Construction 3.5.1. Let M be the reversible cellular automaton obtained from $\hat{M}$ according to Definition 3.4.2. Given any finite nonblank configuration c of (not merely one constructed by means of the encoding function

μ of Theorem 3.4.2), consider the greatest integer k for which the hyperplane $s_0 = k$ is not all in the blank state. Since a cellular automaton is a translation-invariant structure, we may assume, without loss of generality, that $k = 0$. By using the decoding function ν of Theorem 3.4.2, we obtain a configuration $\hat{c} = \nu c$ of $\hat{M}$. Since propagation $\langle\hat{c}\rangle$ is unbounded (M is propagation-unbounded), so is propagation $\langle c\rangle$. Therefore M is propagation-unbounded.

COROLLARY 3.5.2. There exist computation- and construction-universal, reversible cellular automata whose bounding problem for configurations is solvable.

3.6 Conclusions

We have shown that nontrivial irreversible processes such as those required for universal computation can be explicitly represented in a homogeneous medium governed by a reversible set of laws. In Section 3.4, this is achieved by means of a straightforward embedding procedure whereby the original process is represented in a hyperplane of a structure having one more dimension. If one concentrates one's attention only on that particular hyperplane (i.e., if one considers the evolution of the state variables represented by particular projection operators), one observes irreversible phenomena; the

"information content," so to speak, of the process gradually decreases. The whole structure, however, remains reversible, as, at any moment, the information still remaining in the hyperplane, together with that diffused through the remainder of the space, is exactly sufficient to reconstruct the system's past history and, therefore, its initial state. Such "backtracking" can be successfully done for any configuration--not only for those considered in the proof of Theorem 3.4.2. Thus, our results are much stronger than, for instance, those of [BENN73], where universal computation is achieved by proceeding on the reversible portion of a carefully selected phase-space path in a dynamical system that is not, on the whole, reversible.

Intuitively, the selected hyperplane is able to carry out irreversible computations as long as it is fed with constant signals from the "upper" half-space (cf. Figure 3.8) and is allowed to dispose of the by-products of the computation by relinquishing them to the "lower" half-space. (Isotropic behavior can be achieved by using a more complex local map.) This situation is analogous to that encountered in thermodynamics, where--assuming that all microscopic processes are strictly reversible--one can obtain irreversible macroscopic processes only as long as a source of free energy and a heat sink are available. While free energy consists of "predictable" signals that, when steered by a suitable mechanism, can be used to force a portion of the system into a desired state, thermal energy consists of "random" signals which, because of their impredictability, cannot be exploited by an a priori assigned mechanism.

Note that in a suitable irreversible cellular automaton the correctness of the computation performed by an embedded Turing machine can be guaranteed when only a small area surrounding the head and the nonblank portion of the tape is quiescent. Instead, in a reversible cellular automaton such computation in general can proceed correctly for t time steps only if the whole area surrounding the machine for a radius vt (where v is the "speed of light") is initially quiescent. Intuitively, in a reversible cellular automaton a higher degree of predictability of the environment is needed in order for an embedded structure to have predictable behavior.

The approach used in Section 3.4 can be easily extended to computing structures that are more general than cellular automata. In particular, it can be extended to a kinematic model of computation such as that used in [LAIN75]. In general, biological-like problems concerned, e.g., with self-reproduction, self-description, evolution, adaptation, and intelligence can be treated in terms of structures making use of reversible mechanisms.

The simultaneous occurence of computation-universality, uniformity of structure, and reversibility in reversible cellular automata opens a number of interesting perspectives. It is well known that in many thermodynamical and quantum-mechanical problems it is necessary to consider the state of the observer in the description of an experiment. A model capable of supporting a wide range of computing and constructing capabilities makes it possible to explicitly characterize the observer and its interaction with the

observed system. Such a detailed characterization has largely been neglected in the past, possibly because of the lack of satisfactory tools. Conceivably, adequate tools for the purpose may be provided by reversible cellular automata, where one can represent, in one and the same physical-like medium, intelligent machines capable of interacting with their environment and the environment itself.

CHAPTER 4

STEADY-STATE EQUILIBRIUM

4.0 Preliminaries

In this chapter, we shall discuss a set of exploratory experiments that were aimed at providing some insight in the variety of behavior possible in cellular automata. In particular, we shall present some empirical--sometimes pictorial--evidence on how different cellular automata approach a steady state.

As suggested in Section 1.1 and explained in more detail in Chapter 5, it is possible to treat the set of configurations of a cellular automaton as a measurable space and its parallel map as a measurable map on such space. The parallel map of a cellular automaton is not, in general, measure-preserving. Intuitively, if one starts with an ensemble of configurations and repeatedly applies the parallel map, the ensemble's distribution will evolve in a certain way and, possibly, many configurations will drop out of the distribution altogether. The problem arises whether the distribution itself or, at least, some of the parameters that characterize it, will tend to a steady state.

This problem is of particular interest in those applications of cellular automata in which one considers spatially localized

systems (e.g., finite set of cells) capable of carrying out recognizable computing or constructing activities, surrounded by a relatively uniform and quiescent environment. This viewpoint is particularly evident in many of the early papers on cellular automata as well as in more recent ones concerned with pattern recognition. Especially when such systems are large, it may be convenient (or necessary) to introduce an element of uncertainty in their specification, according, for instance, to one or more of the following cases:

(i) The cellular automaton is implemented as an array of physical computing elements whose behavior is not entirely reliable. In this situation, if one attempts to construct a system in such an array, either by direct initialization or by means of another system already operating in the array, the system that is effectively constructed may differ in structure and behavior from the one that was intended.

(ii) A spatially localized system is subjected to unpredictable disturbances originating from an environment that is not completely uniform.

(iii) The system itself is incompletely specified. For instance, a system may be described as initially consisting of "a singly connected 'patch' of cells in a certain state, surrounded by a background of cells in another state," without specifying the size and shape of the "patch."

(iv) The relative position of two or more spatially localized systems (which are otherwise specified in detail) is known only approximately. Thus, the mechanism of an eventual interaction between such systems is not foreseeable in detail.

Analogous considerations may be extended to systems whose parameters are not associated with localized features. Systems of this kind, more similar to a superposition of waves than to an assembly of particles, are discussed in [BART75]. In all cases, one would like to be able to make at least qualitative predictions about the evolution of such systems.

The same problem may be approached from a slightly different viewpoint. The evolution of a configuration, expressed in terms of the state of the individual cells as affected by the local map, may be compared to that of a physical system expressed in terms of its "atomic" constituents (at a given level of detail) and of the corresponding "atomic" laws. For a given cellular automaton and assigned initial statistical distribution of cell-states in a configuration, it may happen that, as a configuration evolves in time, certain contiguous arrangements of cell-states eventually tend to be found with a much greater probability than others. In turn, such arrangements--which we may think of as"molecules"--may combine in various ways to form larger aggregates with an organization similar to that of macromolecules, polymers, and crystals. More generally, for many cellular automata it is to be expected that, at different aggregation levels, certain classes of variously characterized

structures be more stable (i.e., more probable, in the above sense) than others. Indeed, in the cellular automata with which we have epxerimented, such situation occurs with a relatively high frequency. Considering the ease with which the local map of a cellular automaton can be tailored to fit particular applications[CODD68], it is conceivable that even very simple cellular automata may display a variety of stable "materials" sufficient for the construction of a wide range of relatively permanent macroscopic structures. From this viewpoint, experimentation with cellular automata may indicate how such structures can be combined in order to construct reliable mechanisms capable of performing an identifiable task.

Von Neumann's approach of directly synthesizing at the microscopical level certain structures capable of self-reproduction is certainly extremely valuable as an existence proof of a wide range of capabilities for cellular automata. However, such structures are stable and viable only in a totally controlled environment, and require detailed specifications that are quite intolerant of random deviations from the plan. On the other hand, one may conceive of cellular automata capable of supporting peculiar varieties of thermodynamics and chemistry comparable, in some sense, to those of natural systems, and possibly an evolutionary "biology" of macroscopic structures based on mechanisms more flexible than those of von Neumann's original self-reproducing automata. (Incidentally, this approach is quite consistent with von Neumann's longer-term goals[VONN66].)

An experimental verification of this hypothesis presents conceptual and practical difficulties whose import is difficult to judge. If we consider the analogy whereby cells correspond to physical features at an elementary-particle scale (10^{-15} cm) and the speed of propagation of information (one neighborhood radius per time step) corresponds to the speed of light, then, in the same scale, macroscopic structures corresponding to simple biological organisms would have a diameter on the order of 10^{10} cells, and a system large enough to support a minimum of biological-like evolution would have to be of a much larger size--at least, say, 10^{12} cells across.

One cannot expect to implement cellular automata of such dimensions with the currently foreseeable technologies. What is more, unless the implementation were so miniaturized that the cells themselves actually had atomic dimensions (in which case it is questionable whether one would have unlimited choice in selecting the the local map) the time required for observing such hypothetical biological-like phenomena would be impractically long. However, while natural systems obey laws over which we have no choice, there exist a very large number of cellular automata to choose from, even if we restrict our attention to very simple ones. In this context, a reasonable research goal would be to try to identify those cellular automata in which, in some sense, the span between microscopical and macroscopical features encompassed a much shorter size range.

A precise formulation of the above goal would require, to say the least, a clarification of the concept of "macroscopic," which

seems to be connected both with the degree of ignorance, on the observer's part, of the microscopic initial and boundary conditions, and with the degree of reversibility of the microscopic laws. Here, we do not intend to deal with this issue in any detail. Suffice it to say that, while classical thermodynamics is based on the assumption that the microscopic laws of physics are reversible, and can thus take advantage of a number of conservation principles, cellular automata are not, in general, reversible. The results of Chapter 3 suggest that irreversible cellular automata can be treated in terms of reversible ones containing a portion that is not in statistical equilibrium with its environment. Therefore, one may presume that even equilibrium conditions for irreversible cellular automata should be dealt with by means of arguments analogous to that of nonequilibrium thermodynamics; thence the difficulty of the problem.

Before attempting to develop a general statistical theory for cellular automata, it seems wise to take a humbler attitude and collect a number of empirical data that may help suggest an eventual line of attack.

4.1 Cellular automata as statistical assemblies

The statistical theory of ensembles is applicable to arbitrary systems having a certain number of degrees of freedom and characterized by a Hamiltonian function (classical mechanics) or a

Hamiltonian operator (quantum mechanics). Since an ensemble is "an imagined infinite multitude of replicas of a given system"[FURT 70] each independently affected by random disturbances-either in the initial conditions or in its evolution, depending on the point of view--the empirical study of ensembles would require the observation of a large number of replicas of a given system. This situation can be greatly simplified if the system possesses a high degree of regularity, e.g., if it consists of a number of identical elementary subsystems whose mutual interactions obey identical laws. In this case, instead of considering the statistical distribution of system states across a whole collection of such systems (i.e., an ensemble), one may consider the statistical distribution of elementary-subsystem states across a single instance of the system. Such a regular system is called an assembly, and consists, in the classical or quantum-mechanical case, of "a finite multitude of real identical particles in space, which may interact with each other by forces."[FURT 70]

From an empirical point of view, assemblies are clearly much more convenient to experiment with than ensembles. Finite uniform automata fit well the statistical-assembly scheme, as they consist of a finite collection of identical, uniformly interacting elementary subsystems. It seems reasonable to conjecture that many of the statistical properties of a cellular automaton would be approximately reproduced in a finite version of it of sufficiently large size. (By finite version of a cellular automaton M having tessellation group $S = C_{\infty}^{d}$ we mean a finite uniform automaton obtained from M by

replacing S by a direct product of **finite** cyclic groups, of the form $C_{\ell_1} x \ldots x C_{\ell_d}$). In the remainder of this chapter, when no ambiguity is possible, we shall use the term "cellular automaton" also for any such finite version.

The present experiments involve cellular automata treated as statistical assemblies. As an initial state for such assemblies we consider a configuration having a given distribution of cell-states. The temporal evolution of this configuration was simulated on a general-purpose computer and observed (visually, on a CRT screen, and by means of programs that monitored certain selected parameters) for an adequate number of time steps. In particular, we looked for symptoms (like cyclic behavior, convergence of certain statistical parameters to a stationary level, etc.) that the assembly had reached an approximately steady state.

The experiments were performed, in similar conditions, on a large number of cellular automata that differed only in their local map. More emphasis was placed on obtaining a global, though mostly qualitative, outlook on how different cellular automata approach a steady state, rather than on the detailed study of the behavior of any one particular cellular automaton.

4.2 Practical considerations

In planning extensive experiments with cellular automata, it

is necessary to keep in mind a number of practical factors. Cellular automata are very naturally implemented as uniform arrays of digital integrated circuits. (Different ways of realizing the local map by means of microprogrammed logic, reprogrammable memories, or control pulses are discussed in [TOFF72].) However, the development of such specialized circuits requires a considerable investment of resources, and preliminary experimentation is better carried out as a much cheaper simulation in a general-purpose computer. In this case, the time required for performing a transition (i.e., one application of the parallel map) is approximately proportional to the number of cells, while in a parallel-computing array this time is essentially independent of array size. On the other hand, in a computer simulation it is much easier to superpose on the machinery of the local map a number of auxiliary functions for initializing, monitoring, and documenting the automaton's activity. In particular, for two-dimensional cellular automata of moderate size, it is possible to display on a CRT screen, time step after time step, a graphic representation of the current configuration.

The following discussion applies to simulation on a general-purpose computer. Let us consider a <u>family</u> of N two-dimensional cellular automata having state alphabet of size r and neighborhood template of size n. We shall refer to finite versions consisting of a square array of size ℓ by ℓ (i.e., the tessellation group is $C_\ell \times C_\ell$). For an experiment in which, for each member of the family, the evolution of the initial configuration is followed for t time

steps, the total computing time T that is required is on the order of

$$T \sim Nt\ell^2 f(n,r)\tau \ , \tag{4.2.1}$$

where τ is the length of the machine cycle and $f(n,r)$ is the number of cycles required for computing the next state of a single cell starting from the state of its neighbors. For large r, $f(n,r)$ is approximately proportional to $n \cdot \log r$. However, as long as a cell-state can be encoded in a single machine word, we can take, approximately,

$$f(n,r) = an \ , \tag{4.2.2}$$

where a is a small integer. For a typical general-purpose computer, we may assume $a \sim 3$ and $\tau \sim 10^{-6}$sec. A reasonable value for T (the length of the experiment) is of the order of a few hours ($\sim 10^4$sec). Replacing such values in (4.2.1), we obtain, for n, N, ℓ^2, and t, the constraint

$$nN\ell^2 t \ < \ 3 \cdot 10^9 \ . \tag{4.2.3}$$

To the above essentially technological constraints we shall add some related to the experiment's goals. Let δ be the radius of the neighborhood template. If one wants to approximate to some extent the statistical properties of an infinite cellular automaton, ℓ (the array's side) must be chosen much greater than δ (the range of the "forces" between cells). If one wants to approximate steady-state

conditions, t must be chosen much greater than ℓ/δ (i.e., the signals originating from each cell must have time to traverse the cellular automaton several times in order to achieve sufficient "mixing"). Thus, we require

$$\ell >> \delta \quad \text{and} \tag{4.2.4a}$$

$$t >> \frac{\ell}{\delta} \quad . \tag{4.2.4b}$$

Finally, for a neighborhood template having the approximate shape of a compact disc,

$$n \sim 4\delta^2 \quad . \tag{4.2.5}$$

The above constraints do not leave a large margin for the selection of a suitable family of cellular automata. For our experiments, on one hand we wanted to use cellular automata that were not so simple as to be utterly trivial. On the other hand, after choosing a certain simply characterized family, we intended to perform the same experiments on <u>all</u> members of the family, in order to avoid bias in a comparative analysis. The number of distinct local maps that can be defined for the class of r-state, n-neighbor cellular automata is r^{r^n}, as in the following tabulation (Table 4.1):

n =	r = 2	3	4
2	16	$19{,}683_{12}$	$4 \cdot 10^{9}$
3	256	$7.6 \cdot 10^{12}$	$3.4 \cdot 10^{38}$
4	$65{,}000_{9}$	$4.3 \cdot 10^{38}$	
5	$4.2 \cdot 10^{19}$		
6	$1.8 \cdot 10^{38}$		
7	$3.2 \cdot 10^{38}$		

TABLE 4.1. The table lists the values of the function r^{r^n} which represents the number of distinct local maps that can be defined for the class of r-state, n-neighbor cellular automata.

As observed by Smith[SMIT69] (also cf. [YAMA69]), a d-dimensional cellular automaton with fewer than d+1 neighbors can be resolved into a collection of (d+1)-dimensional independent laminations. Thus, the simplest irreducibly 2-dimensional cellular automaton requires a neighborhood template consisting of at least three neighbors arranged, for instance, on an L-shaped pattern (a linear arrangement of three neighbors in two dimensions again brings about a laminar decomposition of the cellular automaton).

Comparing the above data with constraints (4.2.3), (4.2.4), it is easy to verify that, as long as the selection of a family of cellular automata is based only on the number of cell-states and the

number of neighbors, any choice with $n > 3$ (and, of course, $r > 1$) would yield a family that is too large for an exhaustive experimental investigation. On the other hand, the choice $n = 3$ yields cellular automata that are exceedingly simple. Moreover, the L-shaped neighborhood template leads to unidirectional flow of information in either dimension. Without additional constraints, a large and more symmetrical neighborhood (for instance, the von Neumann neighborhood with $n = 5$ or the Moore neighborhood with $n = 9$) would yield an extremely large family of cellular automata. For these reasons, following Codd's suggestion[CODD68] we decided to impose the rather natural (from a physical viewpoint) constraint that the local map be rotationally invariant. In this case, for $n = 5$ (von Neumann neighborhood) and $r = 2$, the local map, which is defined on $2^5 = 32$ points, is <u>independently</u> defined on only 12 points. The corresponding family of cellular automata consists of 2^{12} (= 4096) members.

The choice $N = 4096$, together with the choice $\ell = 24$ (which is about the maximum size that can be displayed by the display subroutines of CESSL[BREN70] that we intended to use) satisfies constraints (4.2.3) and (4.2.4). In fact, with

$n = 5$ (von Neumann neighborhood)

$\ell = 24$ (linear size of the array)

$N = 4096$ (number of members of the family),

constraint (4.2.3) yields

$t < 250$ (average length of the simulation for each member of the family)

Observing that

$\delta = 1$ (radius of the von Neumann neighborhood)

we have

$$24 >> 1$$

for constraint (4.2.4a) and

$$250 >> \frac{24}{1}$$

for constraint (4.2.4b), so that such constraints are substantially satisfied.

For the above reasons, we decided to restrict our observations to the class of two-dimensional cellular automata having two states per cell, the von Neumann neighborhood, and a rotationally invariant map. The domain of the local map, consisting of 32 distinct neighborhood configurations, is partitioned into 12 equivalence classes by the rotational-invariance constraint, as shown in Table 4.2. Since, in this case, a particular local map is obtained by assigning a 0 or a 1 to each of these neighborhood-configuration classes, we shall use such an ordered 12-tuple of 0's and 1's as a label for the corresponding cellular automaton, with the ordering

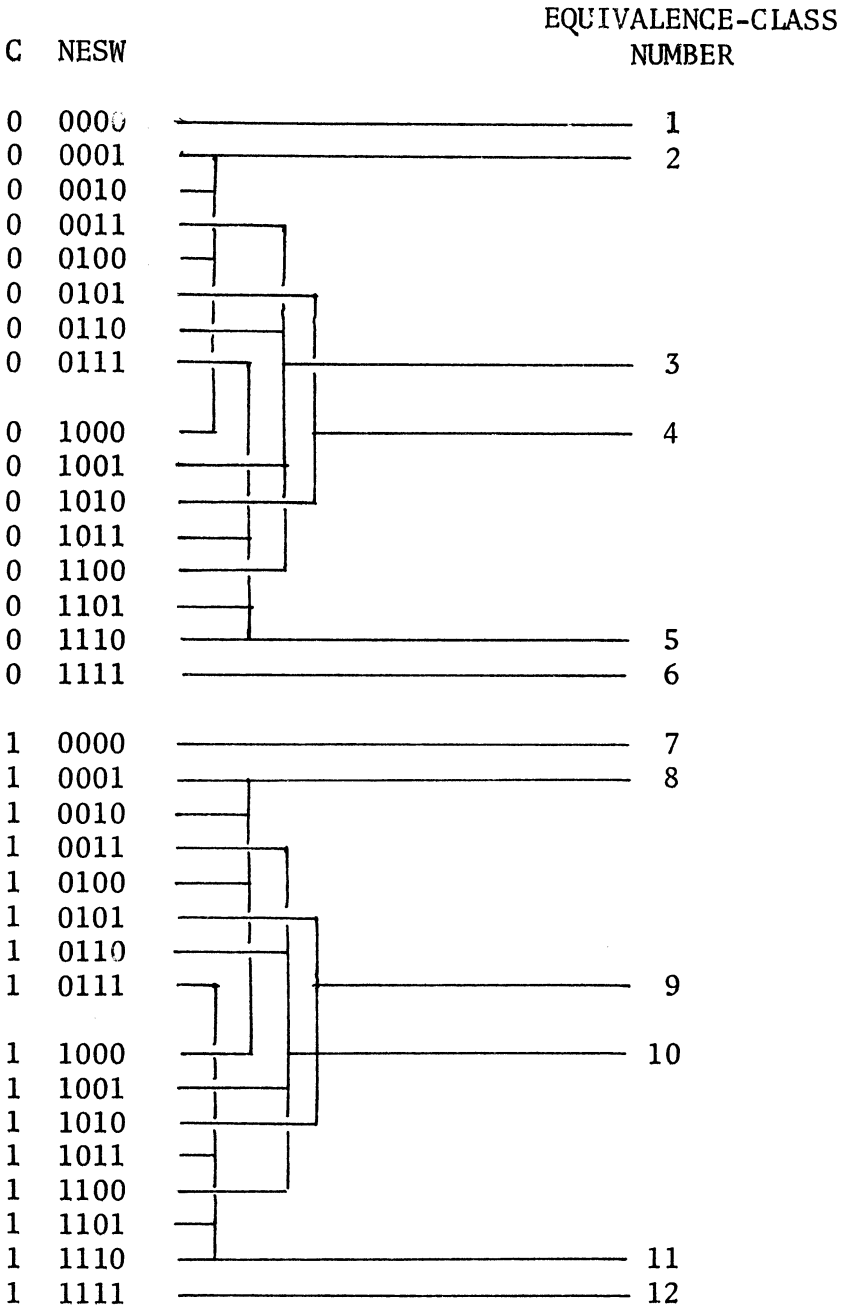


TABLE 4.2. The 32 neighborhood configurations for a two-state, von Neumann-neighborhood cellular automaton are grouped into 12 equivalence classes. This corresponds to identifying nighborhood configurations that differ only by a rotation. In the heading of the column on the left, the letters denote, respectively, "center" (C), "north" (N), "east" (E), "south" (S), and "west" (W), corresponding to the following relative position:

N
WCE
S

given in Table 4.2. Except where noted, the experiments employed a 24-by-24 cell version of each cellular automaton of the family.

The simulation was performed on an IBM/1800 system while display and interactive commands were handled by a DEC/PDP-7 driving a DEC/338 graphic display system. Higher-level software functions were written in the CESSL language[FRAN71], while, for sake of efficiency, the local-map subroutines were written in IBM/1800 Assembler language. The above hardware/software facilities were developed by the Logic of Computers Group (Comp. Comm. Sci. Dept., University of Michigan), and are particularly convenient for experimentation of the present kind.

4.3 Summary of the experiments

Essentially, our experiments consisted in constructing, for each member of the family of cellular automata discussed in the preceding section, the trajectory originating from a "random" configuration. The same initial configuration was used in all cases; this was randomly generated in such a way as to obtain half 1's and half 0's (Figure 4.1). Each trajectory was followed for a number of time steps that was determined as discussed below. Certain of the simulation data were used in real time; in particular, in order to determine if the trajectory had entered a cycle. Other data were analyzed and compared only at the end of the simulation runs.

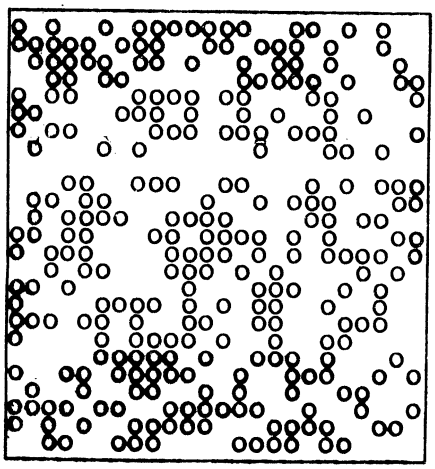


FIGURE 4.1. The random initial configuration used in all simulations. The configuration consists of 50% zeros (represented by blanks in the figure) and 50% ones (circles).

According to the figures given in the preceding section, practical limits imposed an average of about 250 time steps for the length of the simulation for each cellular automaton of the family. On one hand, it was not clear a priori whether this would be sufficient to effectively observe the approach to a steady state in a satisfactorily large number of cases. On the other hand, if we had a reliable practical criterion for determining when a sequence of configurations had reached, at least approximately, a steady state, we would be able to interrupt that simulation run ahead of time and use the remaining time steps to protract the simulation of "harder" cases. Of course, the trajectory of an arbitrary finite input-free automaton must ultimately enter a cycle, possibly preceded by a transient. We shall call characteristic time of a given initial configuration the length of the transient plus that of the cycle

itself. An upper bound for the characteristic time is given by the total number of available states. In our case, with 24x24 cells and two states per cell, such an upper bound is 2^{576} --a rather large number. (Because of the uniformity constraint, a cellular automaton cannot, in general, be made to follow a trajectory containing all possible configurations; however, it is difficult to explicitly compute a lower, more realistic upper bound that would account for this fact.) A preliminary test on a small sample of cellular automata of the family showed that in most cases the characteristic time was much shorter, displaying a rather broad distribution with a peak corresponding to about 9 time steps. We resolved, then, to test for the occurrence of cycles as the simulation proceeded, and to interrupt the simulation of each cellular automaton as soon as a cycle was detected. With the additional time made available in this way, it was possible to push the exploration of the other trajectories to 1024 time steps.

In order to detect a cycle in the most straightforward way, one would have to "save" the sequence of configurations as they are generated, and compare the current configuration with all the previous ones. While, in this way, one can recognize the cyclic behavior already at the end of the very first cycle, the method is inefficient in terms of computer memory and time. In practice, we stored in disk memory every 32nd configuration, for back-up purposes, and kept in core memory a very compact (1:576 reduction) hash-coded version of each configuration of the sequence. A test for

possible cycles was performed first on this hash-coded sequence and verified, when successful, by reconstructing a portion of the trajectory from an appropriate back-up configuration. Though more efficient, this method allowed us to detect cyclic behavior only at the end of the second cycle, Since the length of cycles was, on the average, much less than that of transients, this did not constitute a problem. Within the maximum number of 1024 time steps allotted to each simulation run (cf. the above paragraph), the cyclic portion of the trajectory was identified in more than 60% of the cases.

It is clear that for those trajectories which were ascertained to have entered a cycle during the computer simulation, nothing more about the steady state could be learned by protracting the simulation, and indeed just recording on permanent storage one configuration of the cycle was sufficient for reconstructing, later on, the whole cycle. On the other hand, for those trajectories that had not shown evidence of "settling," recording on permanent storage the last configuration observed during the simulation was sufficient for resuming the simulation at a later time, if desired. For these reasons, at the end of each simulation run the following data were recorded

(i) if no cycle had been detected,

-hard copy of the 1024th configuration;

(ii) if a cycle had been detected,

-hard copy of the first recurrent configuration,

-length of the transient, and

-length of the cycle .

The distribution of cell-states in the steady state (cf. Section 4.4) was computed from such hard copies.

If one ignores the limitations deriving from working with finite--indeed, rather small--versions of cellular automata, any of the configurations that make up a cyclic sequence can be considered a typical representative of the steady state that a cellular automaton reaches starting from a configuration having an assigned distribution of cell-states. It should be noted that, even when the initial configuration is constructed by randomly assigning a cell-state independently to each cell, there may appear in the steady state a definite correlation between the state of adjacent cells. As an example, let us consider cellular automaton #1114 (the 12 binary digit that identify one of the 4096 cellular automata of the family are represented, for convenience, by four octal digits) in the trajectory starting from the configuration of Figure 4.1. After a transient of 9 time steps, the automaton enters a cycle of length 2 , as illustrated in Figure 4.2. It is clear that merely reporting the steady-state distribution of cell-states for this cellular automaton (4.5% 1's and 95.5% 0's) would convey little information about the nature of the steady state. It is easy to verify that, in this particular cellular automaton, the steady-state distribution of cell-states is strongly dependent on the initial configuration, while

the strong correlation between the state of adjacent cells, which is evident from Figure 4.2, would be preserved under a great variety of initial conditions.

For the above reasons, the analysis of the simulation data (we emphasize again that these experiments were intended to be merely orientative) was carried out along two distinct lines. In the first (Section 4.4) we discuss a few statistics based on the entire set of simulation runs (4096 cellular automata). In the second (Section 4.5) we analyze in more detail, from an intuitive viewpoint, the individual behavior of a small number of cellular automata. These were selected, as representative of certain particular features, out of a randomly chosen sample of 64 cellular automata of the family whose evolution had been monitored on the CRT screen and interactively experimented with.

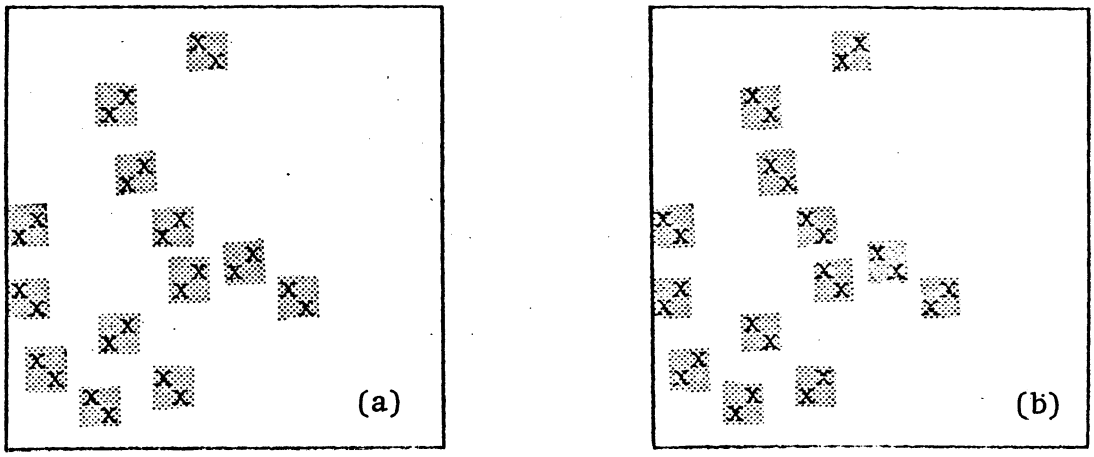


FIGURE 4.2. In the steady state, this automaton oscillates between configuration (a) and (b). Crosses represent 1's, blanks 0's; Shading has been added for contrast. Note that the 1's have coalesced into many disjoint instance of a single pattern consisting of a 2x2 squares where the 1's alternate between the two diagonals (x_x and $_xx$).

4.4 Brief statistics

Cycle length. Approximately 60% of the observed automata had completed at least two cycles (after a leading transient) within 1024 time steps. The cycle-length distribution thus observed is given in Table 4.3. Note that 20% of the cases have a cycle of length 1 (the automaton is "frozen") and that over half have cycle length ≤ 6 . Odd cycle lengths are practically missing. There is a strong preference for cycles that are multiples of small integers. The general impression is that, in most cases, the cyclic behavior is due to a combination of short local loops involving few adjacent cells. This impression is substantially confirmed by the analysis of Section 4.5, where we also discuss the possible influence of the array size on the cycle-length distribution.

Transient length. The transient-length distribution is illustrated in Figure 4.3. The length of the observed transient is not so critically dependent on the particular local map as the cycle length, since the same automaton may display a very different transient length depending on the initial conditions but would probably end up in a cycle of the same length. Thus, the transient-length distribution is much smoother than that of the cycle length. There is a single strong peak around 7 time steps, and after that the

CYCLE LENGTH	COUNT	CYCLE LENGTH	COUNT
1	835	66	2
2	770	72	1
3	2	80	1
4	313	84	11
5	1	90	1
6	187	120	4
8	44	132	1
10	8	140	2
12	134	156	2
14	6	168	3
15	1	180	1
16	3	204	1
18	1	210	4
20	12	228	1
24	42	240	3
28	3	252	1
30	15	264	1
40	2	280	1
42	13	330	1
48	4	336	1
56	2	360	1
60	17	420	2

TABLE 4.3. Cycle-length distribution. The table lists the number of cellular automata of the family that were observed to enter a cycle of a certain length. The distribution is very irregular, with sharp peaks and practically no occurrences of cycles of odd length (aside from length 1 , which represents a completely "frozen" configuration). Approximately 2/3 of the cellular automata were observed to enter a cycle. The remaining 1/3 (1635 cases) do not appear in the table.

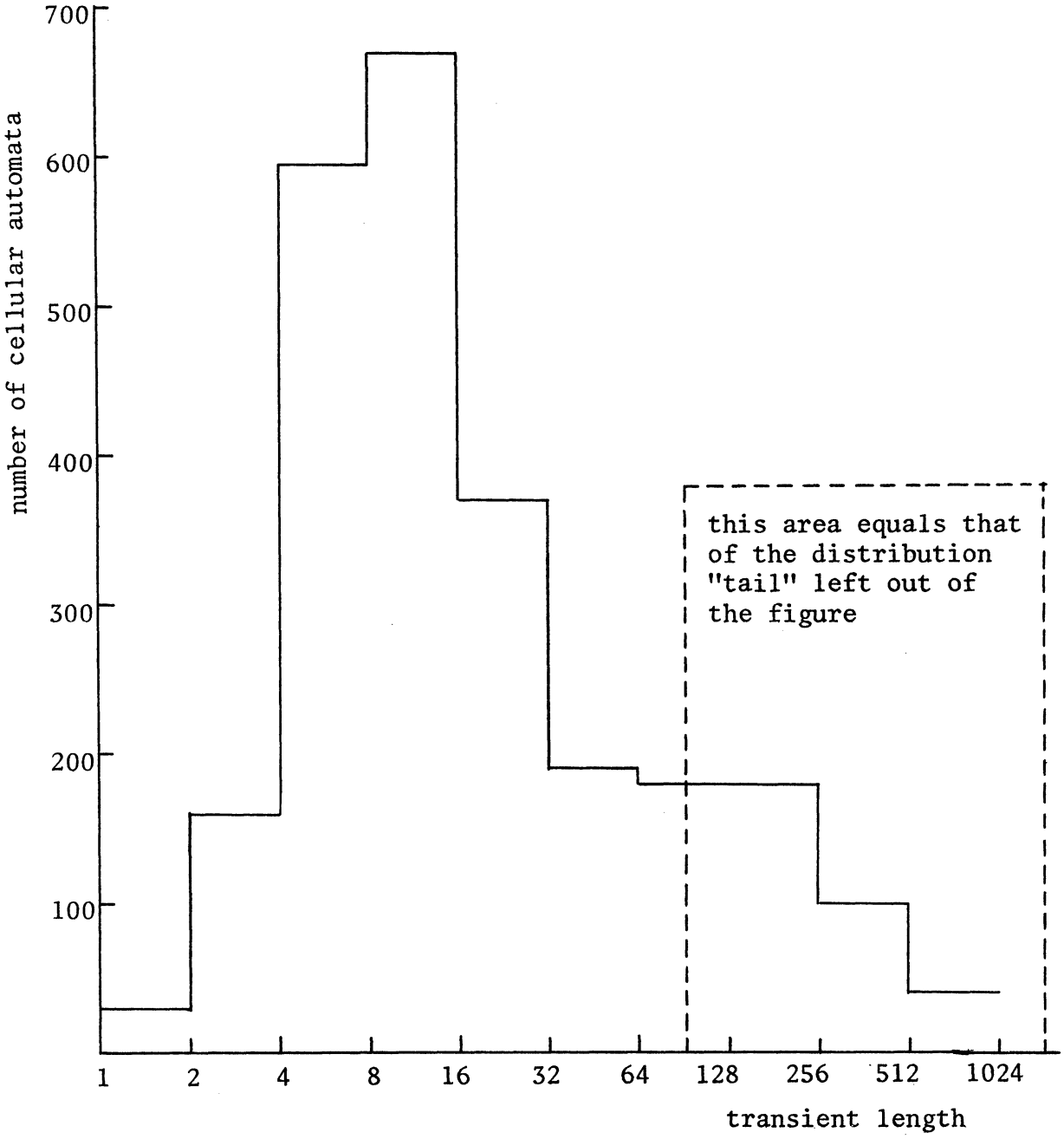


FIGURE 4.3. Each vertical column of the histogram represents the number of cellular automata of the family that were observed to traverse, before entering a cycle, a transient of length falling between 1 (inclusive) and 2 (exclusive), 2 and 4, 4 and 8, etc.

decay is very rapid.

Cell-state distribution. The distribution of 0's and 1's in the "steady state" is plotted in Figure 4.4, where curve C refers to the cellular automata that were observed to have entered a cycle and curve N refers to the remaining ones. Extrapolating from intermediate data, it seems likely that if the experiment had been protracted for many more time steps the peak in curve N would have been less pronounced, while the peak at the center of curve C would have been relatively higher. We suggest the following interpretation for the graphs. If at any moment the number of cells in the 0 state (or in the 1 state) is so great that the cells in the other state are practically isolated (extreme left or extreme right in the curves of Figure 4.4), the interaction between the latter cells is so low that, in most cases, they freeze, die out, or enter a short loop, so that the steady state display a very short cycle. Also, a short cycle is likely to appear if the map is balanced (center peak of the C curve); in this case, in fact, there are great chances of having a reversible map (cf. Section 3.2), and in these very simple cellular automata most reversible maps are trivial, leading either to a uniform shifting motion or to single-cell oscillations. On the other hand, the absence of short cycles entails that each portion of the array must be able to encode long counting sequences, which, of course, are impossible to obtain if one of the two symbols is in scarce supply. Thus, in the N curve ("no cycle observed" implies that the actual cycle is very long), the

distribution is significantly different from zero only in its central portion.

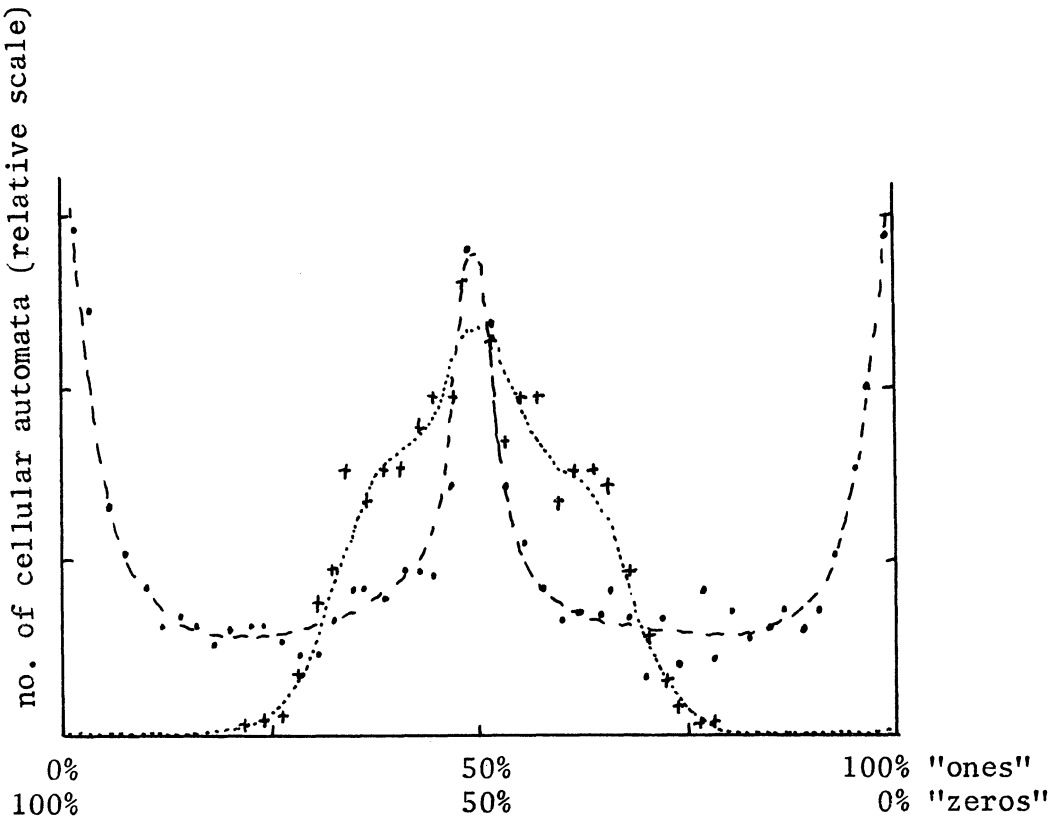


FIGURE 4.4. Cell-state distribution. Curve C refers to the cellular automata of the family that were observed to enter a cycle; curve N to those for which no cycle had been observed at the end of the simulation. The ordinate of a point is proportional to the number of cellular automata having a certain percentage of "ones" in the configuration at the beginning of the first cycle (C) or in the last observed configuration (N).

4.5 Representative cases

In the course of the experiments, we observed in detail the evolution of a number of cellular automata of the family (cf. Section 4.3) in order to investigate from a heuristic viewpoint what properties would be more useful in discussing their steady-state behavior. As stated before, all such cellular automata were started on the same randomly-generated configuration of Figure 4.1. Here, we shall give a pictorial description of some of the more representative cases, under the following headings:

(1) Complete erasure,

(2) Partial erasure followed by freezing,

(3) Atoms,

(4) Molecules,

(5) Polymers,

(6) Oscillators,

(7) Crystals,

(8) Nonisotropic areas.

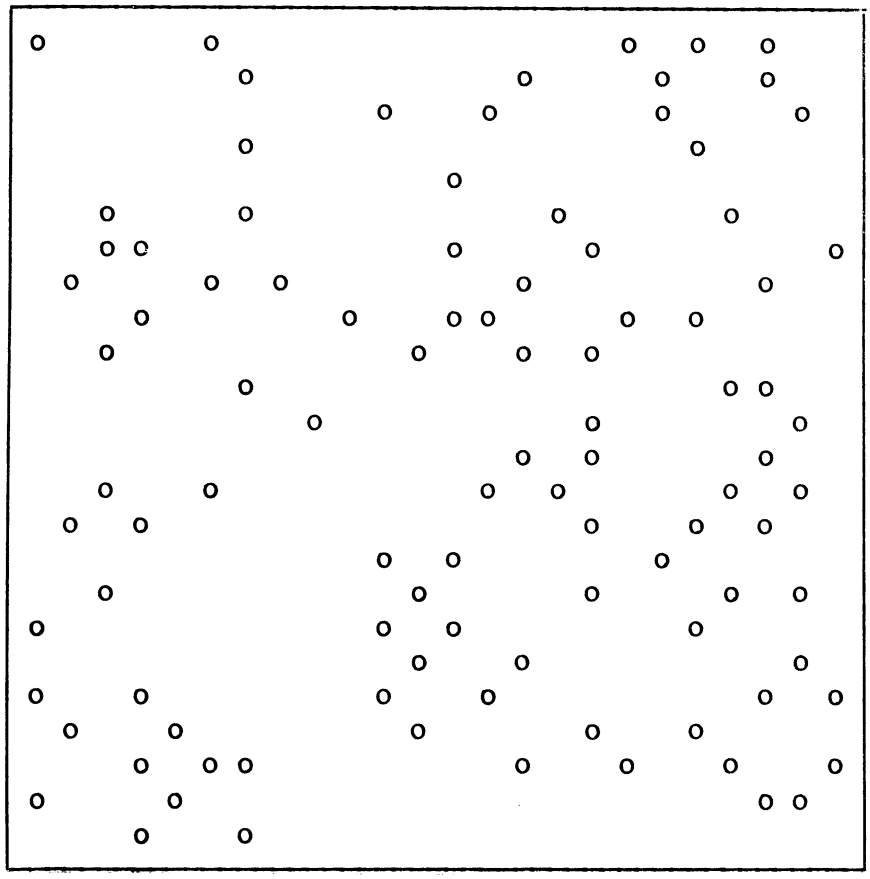
Of course, the suggestive chemical terms are not meant to imply other than a very superficial analogy.

The present observations may be compared with certain positive theoretical results obtained by Holland concerning the spontaneous emergence of several levels of aggregation in systems

similar to cellular automata[HOLL76]. The cellular automata observed by us are very simple (none are computation-universal[CODD68]) and are not expected to support, in the steady-state equilibrium, levels of aggregation much higher than those exemplified below. For more complex cellular automata, it would be worthwhile to explore the quasi-steady states achieved over very different time scales, in order to determine in what conditions significantly higher levels of aggregation may obtain.

(1) Complete erasure. Of the cellular automata that we observed in detail, many ended up, after a brief transient (7 time steps on the average), in configurations containing all 0's or all 1's. These represent, of course, particularly trivial cases, and are mostly associated with a very unbalanced local map, i.e., one whose table contains very few 0's or very few 1's. (Recall from Section 3.2 that having a balanced map, i.e., one in which all cell-states are evenly represented, is a necessary condition for reversibility, while these cellular automata display a high degree of erasure, i.e., of irreversibility.) It is fair to assume that almost all trajectories in such cellular automata would converge to one and the same all-blank (or all-"black") configuration.

(2) Partial erasure followed by freezing. On the other hand, there are cellular automata such as that of Figure 4.5 that end up in a stable configuration after a minimum amount of erasure. In these cellular automata, conditions in a portion of a configuration have



CA # 0060

STEP 2
TRANSIENT 1
CYCLE 1

FIGURE 4.5. The initial configuration almost immediately settles into a stable state after a relatively small amount of erasure.

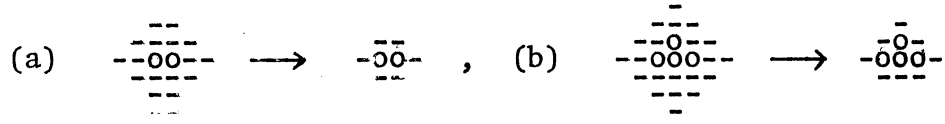
NOTE. In this and in the following figures the symbols used for each cell denote the cell's state during the last two time steps, according to the following schema:

" " (blank)	state was 0 in both time steps
"_"	a 1 just turned into a 0
"x"	a zero just turned into a 1
"o"	state was 1 in both time steps

little influence on remote portions, i.e., any exchange of information is mostly local. The resulting final configuration is a very rapidly reached compromise between the cell-state assignment of the initial configuration and the selective action of the local map which, acting on a very limited scope, "weeds out" certain local patterns of cell-states. With reference to Figure 3.1b (Section 3.2), these automata have a cycle of length one and very short trees, while those of (1) above have again a cycle of length one but very long trees. Cases (3) and (4) below represent an intermediate situation.

(3) Atoms. When the selection imposed by the local map is more stringent than in (2) above, after a certain amount of processing only certain few patterns survive. This is the case of Figure 4.6, where it seems that the only stable pattern is that consisting of two adjacent 1's. Figure 4.7 illustrates a similar case where the steady state admits of more than on kind of stable pattern.

(4) Molecules. One can systematically enumerate all stable patterns of a cellular automaton by applying the local map to finite configurations of increasingly large size. For example, for cellular automaton #0522 one would get, among others, the following "fixed points:"



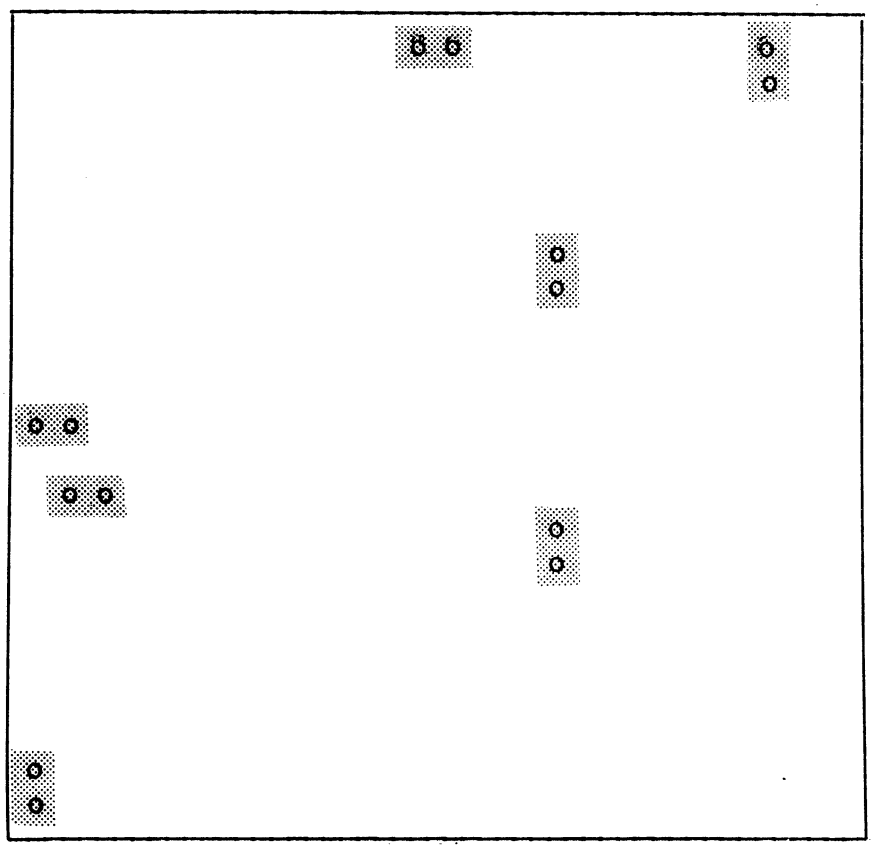
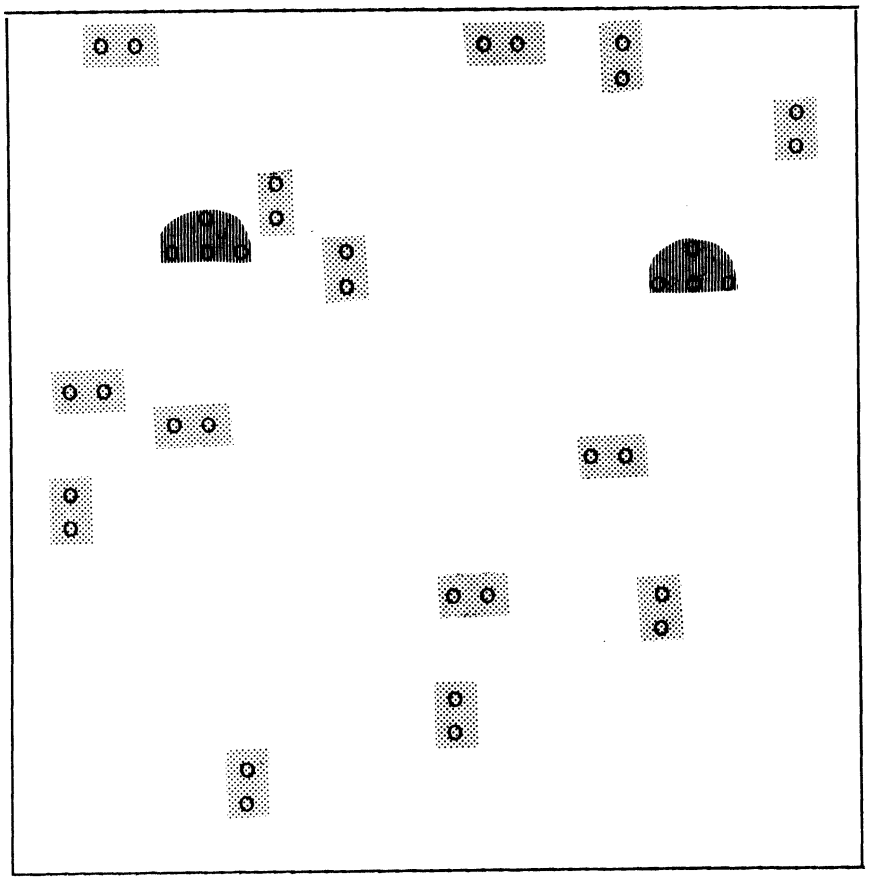


FIGURE 4.6. Extensive erasure and selection take place before the automaton reaches a stable state. Only a variety of pattern--or "atomic species"--is stable.

NOTE. In this and in the following figures, shading, in correspondence with the discussion in the text, has been added for emphasis



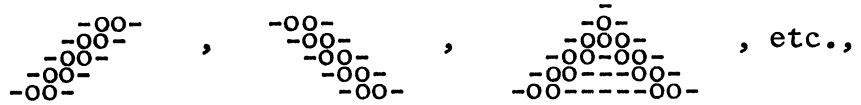
CA # 0527

STEP 8
TRANSIENT 7
CYCLE 1

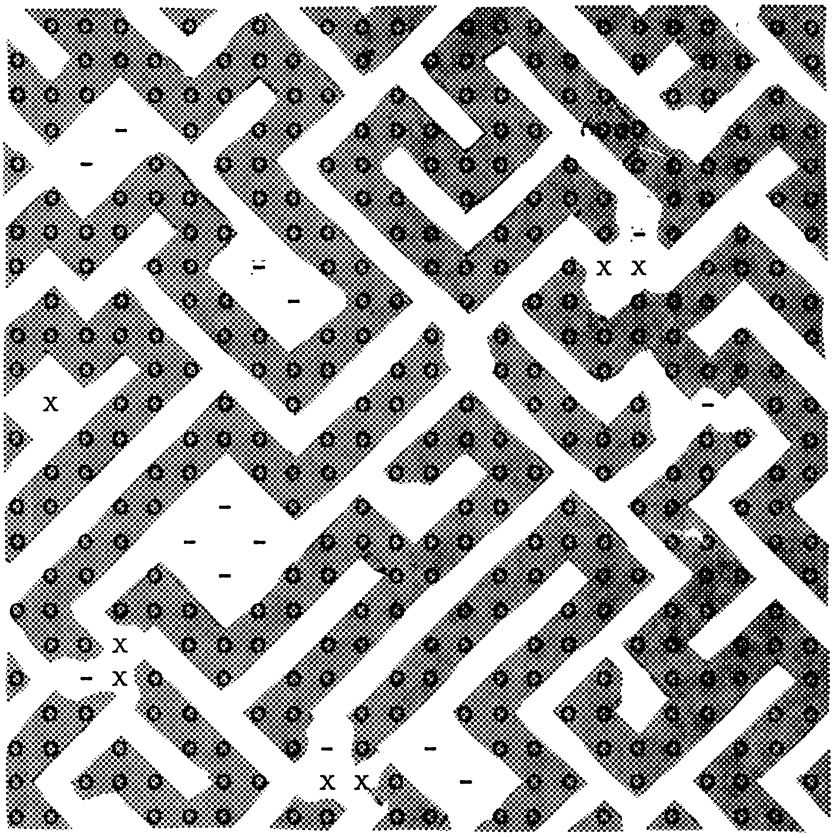
FIGURE 4.7. Two kinds of atom are present.

(where "-" represents a 0 and "o" represents a 1). Considering that the environment consisting of all 0's is stable in this cellular automaton, mappings (a) and (b) above guarantee that patterns like oo and ooo are stable in such environment. The problem arises whether there are mappings of this kind that can be "joined" in such a way as to produce larger stable patterns; in other words, whether for certain cellular automata stable patterns obey some simple generative grammars. Intuitively, patterns like (a) and (b) above can be imagined as "noble gas" atoms that are stable only in isolation; on the other hand, one may look for patterns that are stable when juxtaposed to occurrences of the same or of different patterns. In this way, one can rather naturally extend to (at least certain) cellular automata concepts analogous to those of "valence," "molecule," "polymer," "crystal," etc. In what follows, we shall keep in mind this analogy without bothering putting such terms in quotes.

(5) Polymers. Figure 4.8 supplies a good example of polymers. Here in an environment of all 0's, patterns of the form



are self-maintaining when appropriately juxtaposed (i.e., so that overlapping 0's and 1's match). In this way, one obtains polymer chains of arbitrary length.



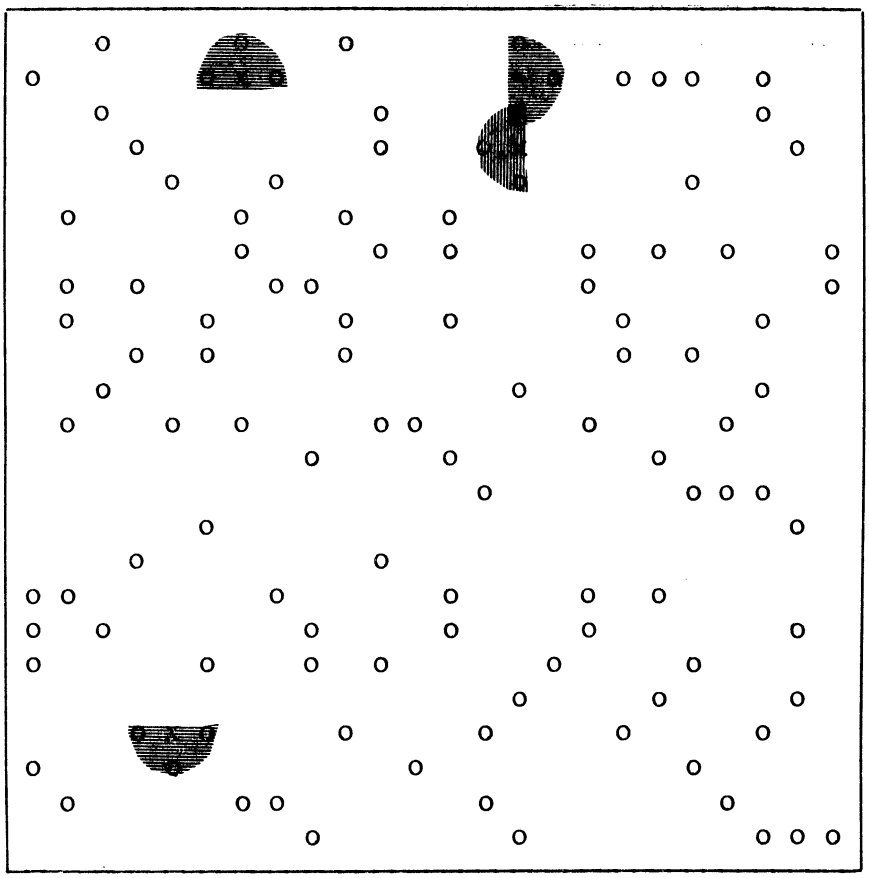
CA #5426

STEP 73
TRANSIENT 61
CYCLE 12

FIGURE 4.8. Here, most of the space is covered by long, zig-zagging polymer chains.

(6) Oscillators. In addition to stable patterns, one may observe cyclically stable patterns (i.e., patterns that evolve cyclically and after a number of time steps reproduce the original arrangement of cell-states). For instance, in Figure 4.9 there appear isolated atoms that are oscillating with a period of two time steps; such atoms can be joined into larger molecules. The cellular automaton of Figure 4.10 displays three distinct atomic species, of which two are of the oscillating type.

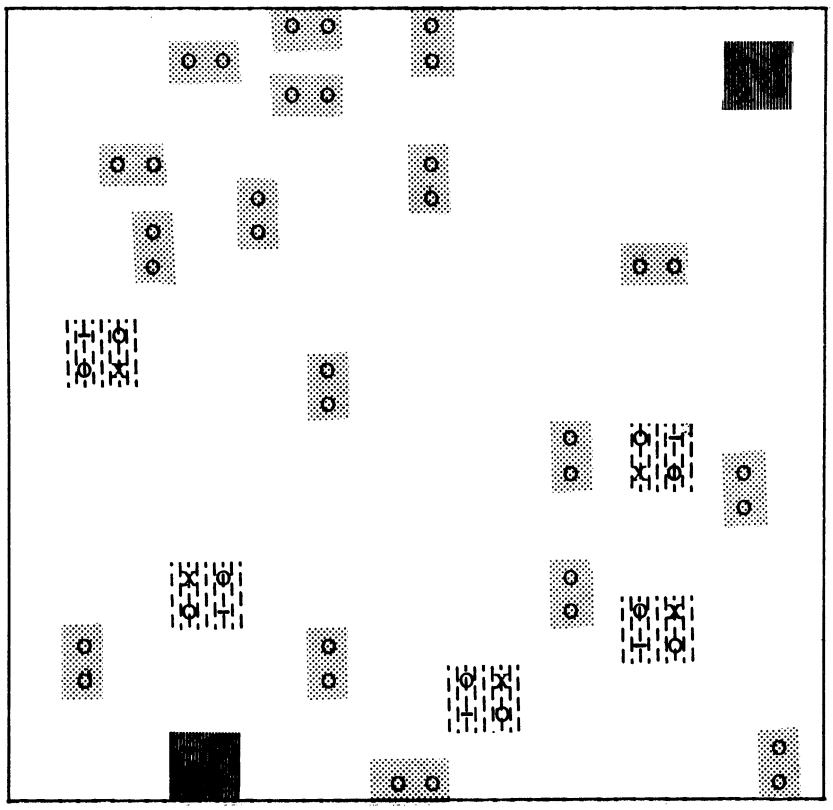
(7) Crystals. In addition or in alternative to isolated atoms or one-dimensional polymer chains one may have stable or cyclically stable structures that display spatial periodicity in both dimensions. We shall call such structures crystals. A trivial example of a crystal is given by an extended area whose cells are all in the quiescent state. In general, the elementary cell of a crystal in a cellular automaton is characterized by both a spatial period (in each dimension) and a temporal period (cf. Section 1.5). Thus, a configuration may contain different portions or domains all organized according to the same crystalline pattern and separated by fracture lines where there is a discontinuity either in the spatial or the temporal phase of a cycle. The crystal of Figure 4.11 is rotationally invariant, has temporal cycle of length 1 and a spatial cycle of length 2. Thus, only spatial-phase discontinuities (clearly showing in the figure) are possible. Note that the crystal also contains some stable and cyclically stable inclusions. A similar case appears in



CA #0271

STEP 8
TRANSIENT 6
CYCLE 2

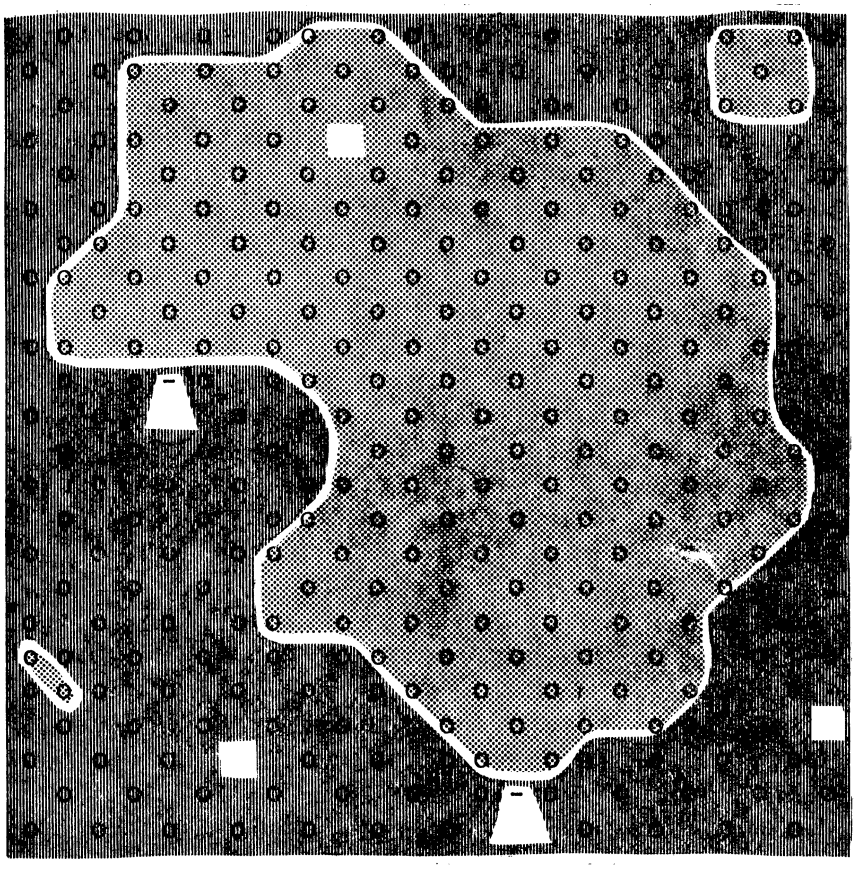
FIGURE 4.9. Here, atoms of one kind may exist in isolation or join to form large molecules. The atoms themselves are oscillating with a period of two time steps.



CA #1022

STEP 19
TRANSIENT 17
CYCLE 2

FIGURE 4.10. Three atomic species appear, of which two are oscillating.



CA #1465

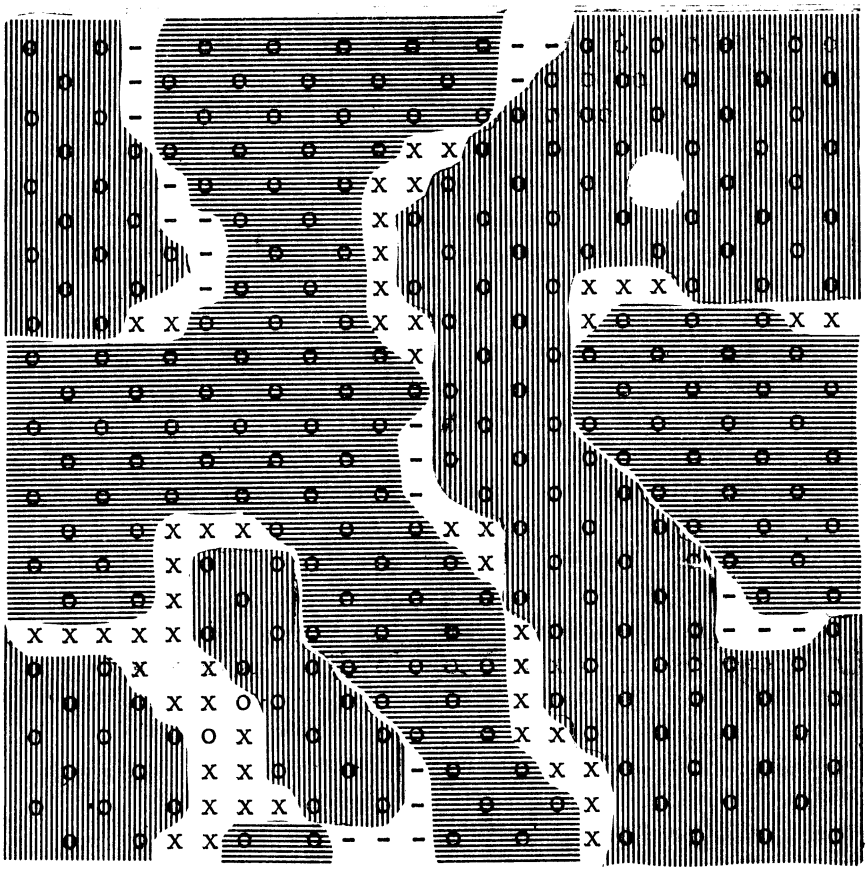
STEP 30
TRANSIENT 28
CYCLE 2

FIGURE 4.11. Several domains of the same crystalline pattern appear. The fracture line represents a discontinuity of the spatial phase. Note a certain number of inclusions.

Figure 4.12. Figure 4.13 gives an example of discontinuity in the temporal phase (the temporal cycle has length 2). Finally, in Figure 4.14 one observes an oriented crystal, with a fracture line betwen domains having different orientation.

(8) Nonisotropic areas. The fracture line betwen different domains may be very thin, as in Figure 4.11, or may have a substantial width, as in Figures 4.12 and 4.14. In the latter case, if the adjacent domains belong to two different crystalline patterns (instead of out-of-phase domains of the same crystalline pattern), the fracture-line zone constitutes a nonisotropic area which may display micro-patterns having a definite orientation with respect to the adjacent domains. This case, illustrated in Figure 4.15, represents a rudimentary attempt at a higher aggregation level.

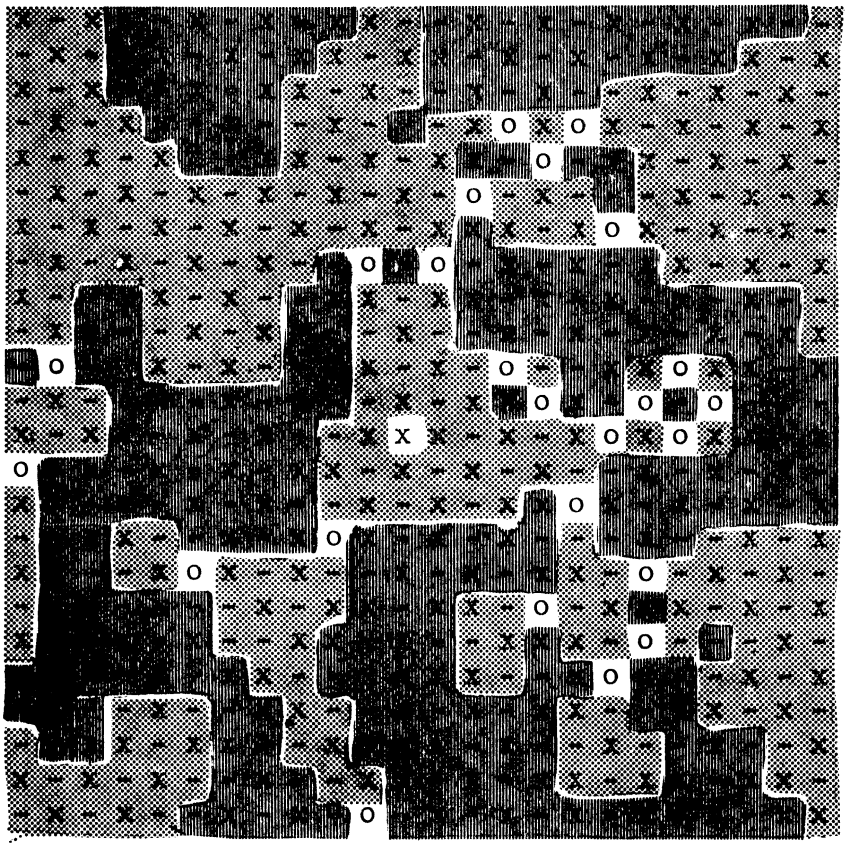
Of the cellular automata that we have observed, in more than half the steady state can be conveniently described in terms of molecules, polymers, crystals, etc., as in the above discussion. The others failed to reach a recognizable steady state during the allotted observation time, or reached one that, instead of containing stable or cyclically stable structures, displayed apparently random motions more reminiscent of gases or fluids or of travelling waves. Of course, this ratio may change for different families of cellular automata. However, it seems clear that the study of cyclically stable, isolated or repetitive, patterns should play an important role in the understanding of the macroscopic behavior of cellular automata.



CA #3464

STEP 63
TRANSIENT 57
CYCLE 6

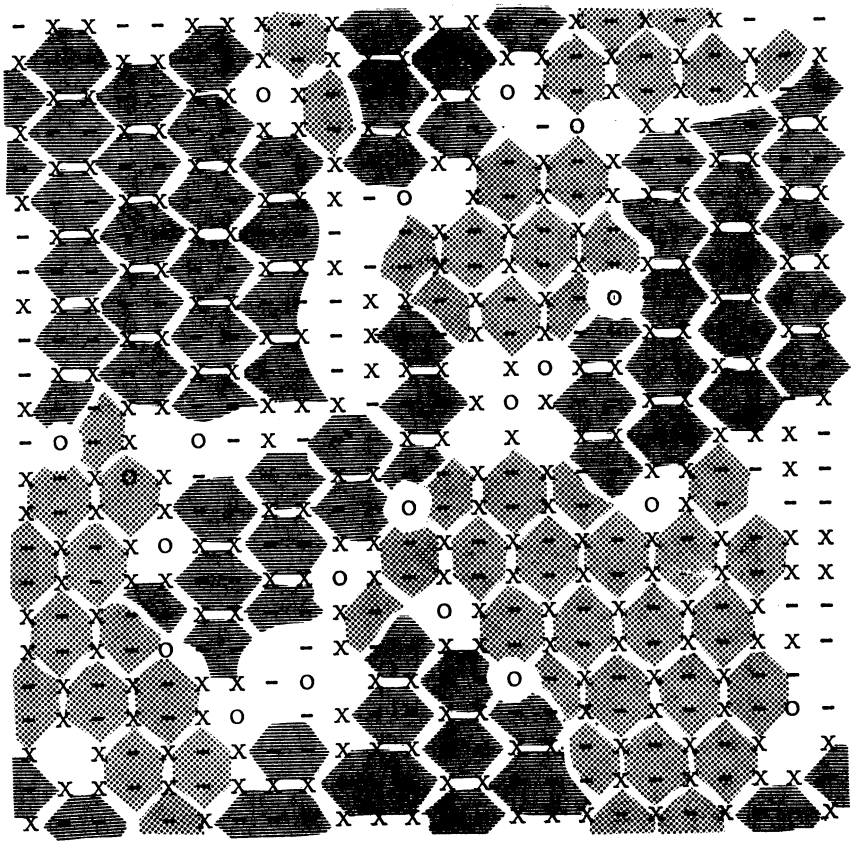
FIGURE 4.12. As in Figure 4.10, one observes several domains of the same crystalline structure. The fracture line has an appreciable width, i.e., the spatial phases passes through an incoherent transient before adjusting to a new value.



CA #5710

STEP 10
TRANSIENT 8
CYCLE 2

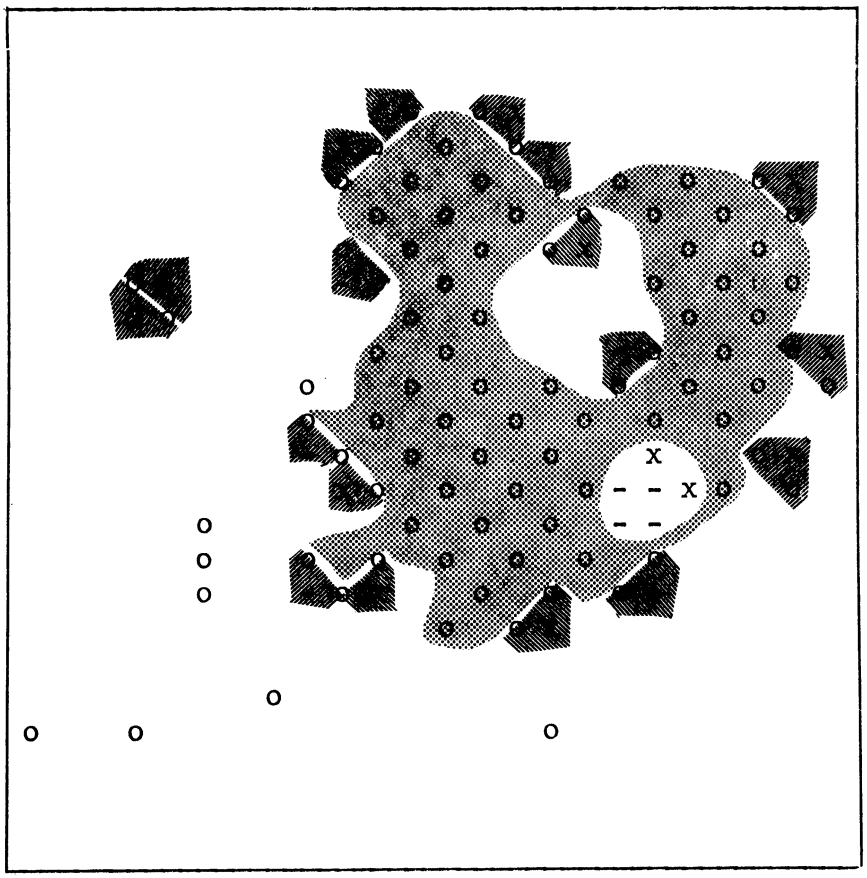
FIGURE 4.13. The crystalline structure oscillates with a period of two time steps. Domains separated by a fracture line are one time step out of phase.



CA #1643

STEP 421
TRANSIENT 419
CYCLE 2

FIGURE 4.14. Here the crystalline structure is spatially oriented. One can see domains with different orientations.

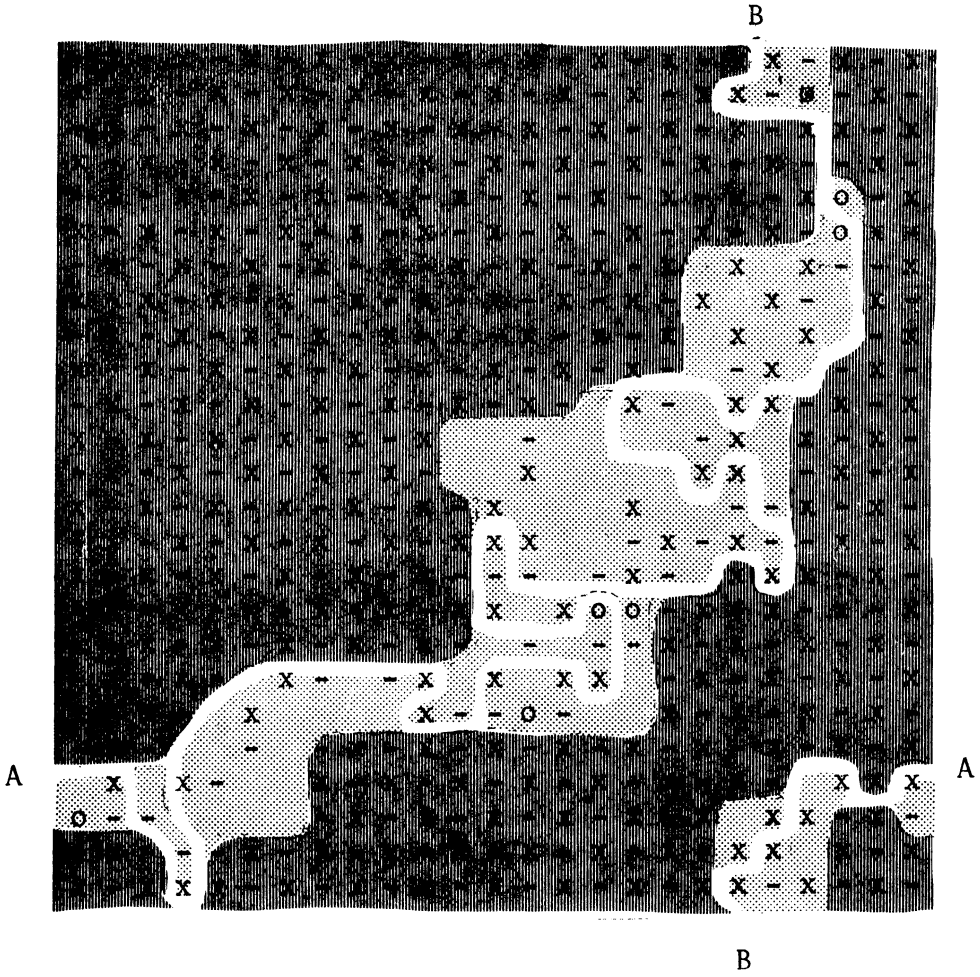


CA #1470

STEP 289
TRANSIENT 287
CYCLE 2

FIGURE 4.15. Domains belonging to two different crystalline structures. Note the consistently oriented formations on the fracture line making up an asymmetric "membrane."

One may wonder, at this point, how much the above results are dependent on the particular size chosen for the cellular automata of these experiments. Of course, for instance, molecular species that in an implementation having infinite size would appear with very low frequency may not show up at all in an experiment with a small-size implementation. Moreover, wherever uniform oscillatory modes ("waves") would appear in the steady state, one may assume that a particular implementation size would select in favor of waves of certain periods (e.g., submultiples of the automaton's "side") and against other periods. We conducted a small number of experiments in which the size of the implementation was varied. In almost all cases, essentially the same steady-state behavior was observed for each cellular automaton. However, there were cases where size played a critical role. For instance, in a 24x24 array cellular automaton #2302 reaches a steady state consisting of a single crystalline domain (with possible inclusions but no fracture lines). The crystal has a spatial period of 2 in either dimension. In a 25x25 array, there is no way for the even period of the crystal to match the odd size of the array, and a fracture line appears. Curiously, this fracture line is very unstable and keeps shifting its meanders much as a river erodes its banks (Figure 4.16). When two curves come into contact the "river" breaks through and a portion of one domain is transfered to the other bank and eventually "assimilated" by the other domain. Such relatively complex behavior is quite remarkable for a cellular automaton of such small neighborhood and state alphabet.



CA #2382

STEP 2258
NO CYCLE YET

FIGURE 4.16. The "river" which separates the two domains continually shifts its course. The light-gray banks indicate portions that have not yet been fully assimilated by the adjacent domain. As the cellular automaton is "wrapped-around," the "river" consist of a single connected loop, as indicated by the A-A, B-B markers.

CHAPTER 5

TOPOLOGICAL ASPECTS

5.0 Preliminaries

The ease with which important questions of cellular automata theory were answered as soon as Richardson attacked them with topological methods ([RICH72]; also cf. [AMOR75]) suggests that topological properties are fundamental in such theory. Besides, a study of such properties would permit seeing cellular automata in perspective, so to speak, within the wider context of dynamical-system theory, which is essentially founded on topological concepts. A complete characterization of cellular automata in topological terms is not available yet; however, such a characterization has been provided for shift dynamical systems, which coincide with cellular automata of a particularly simple kind. At any rate, the topological characterization of cellular automata is centered on three aspects that can be easily interpreted in intuitive terms: (i) topological continuity of the parallel map (small changes in a configuration produce small changes in its successor; this corresponds to the finiteness of the neighborhood template), (ii) commutativity of the parallel map with respect to the elements of the tessellation group (this corresponds to translation invariance), and (iii) completeness

of the set of configurations (this corresponds to using cell-states in the most "efficient" possible way for encoding configurations).

In the following sections, we shall define the "natural" topology for cellular automata, and we shall discuss a number of issues connected with their topological characterization.

5.1. The product topology

Let S be an arbitrary countable set and A an arbitrary finite set of size r. Let C be the Cartesian product, indexed by S, of a countable family of copies of A, i.e.,

$$C = \bigtimes_{s \in S} A = A^S .$$

An element of C is called a configuration [of S over A]. In order to maintain a terminology compatible with that of Section 1.2, we shall call cells the element of S and cell-states those of A. However, in this section we shall not assume any particular property for S aside from its countability.

C can be thought of as the set of all functions of the form $c: S \to A$. The value of a function $c \in C$ at a point $s \in S$ will be denoted by c_s, called the state of cell s in configuration c, or, briefly, the state of s. A set of functions such as C is

a function space when associated with a given topology. As we do not intend to give the cell-state set any particular structure, we shall assume for A the discrete topology D, whose open sets are all the subsets of A itself. Considering that C is the product of sets all identical to A, the most natural choice for a topology on C is the product topology U, defined as follows. For any $s \in S$, one extends the discrete topology on A to a topology U_s on C by identifying U_s with the set of all counterimages of the open sets of D with respect to the projection operator π_s, i.e.,

$$U_s = \{Y \mid Y = \{c \mid c_s \in P\} \wedge P \subset A\} .$$

The collection of all U_s (for $s \in S$) forms the defining subbase of the product topology U. (Note that each of the U_s is a finite open cover of C). Since the defining subbase consists of all functions which associate an arbitrary open subset $P \subset A$ with an arbitrary point $s \in S$, the product topology U is sometimes referred to as the point-open or pointwise topology.

The product topology U supplies a convenient way of expressing certain concepts concerning configurations in cellular automata. In many applications, it is useful to consider only a finite "portion" of a configuration (such "portions" are called, for example, configurations in [MOOR62], restrictions in [CODD68], partial configurations in [SMIT69], and patterns in [RICH72]. The notion of "portion" can be expressed in topological terms as follows. Let us call [finite] stencil any finite set of cells. Two configu-

rations c,c' agree on a stencil w iff $\bigwedge_{s\in w} c_s = c'_s$. "Agreement on w," denoted by $\equiv_w$, is clearly an equivalence relation. Each of the equivalence classes associated with the relation $\equiv_w$ is called a pattern on w. Each element of a pattern is called an instance of that pattern. Intuitively, a pattern is characterized by an assignment of cell-states to all points of a stencil w. Any configuration whose projection on w coincides with that assignment is an instance of the correponding pattern. Thus, a pattern can be considered interchangeably either as a class of configurations that agree on a certain stencil, or as the restriction of a configuration to that stencil, in agreement with the intuitive notion of "portion" of a configurations.

Note that, when endowed with the product topology, the set of configurations of a nontrivial cellular automaton (i.e., one having dimensionality $d \geq 0$ and state-alphabet size $r \geq 1$) is homeomorphic to the Cantor set[VAID60], which is characterized by being a totally disconnected, perfect, compact, metric space[WILL70].

It is perhaps appropriate to emphasize that the topology introduced in this section, which is defined on the set of configurations, is totally unrelated to that discussed in [MYLO71], which is defined on the set of cells. While the latter deals with the set of cells as a geometrical space, the topology discussed here deals with the set of configurations as a phase space, i.e., as a set of dynamical states rather than geometrical points.

The "natural" σ-algebra for cellular automata is defined in a way very similar to the "natural" topology (cf. Section 1.1). As a starting point, instead of the discrete topology on A, consisting of all subsets of A, one considers the finite partition of A consisting of all singletons of A. This partition is extended to S by considering, for any $s \in S$, the partition V_s such that

$$V_s = \{Y | Y = \{c | c_s = p\} \wedge p \in A\} .$$

The collection of all V_s generates the product σ-algebra V on C. As one obtains a topological space by associating to C the product topology, thus one obtains a measurable space by associating to C the product σ-algebra. The similarity of the two approaches is stressed in [SEAR73]. One can define a probability measure on the state alphabet A by assigning the same probability $\frac{1}{r}$ to each of the r cell-states. Such an assignment automatically defines a probability measure on C. Such probability measure is used, for instance, in [WILL75].

5.2 Metrics compatible with the product topology

We have chosen the discrete topology as the "natural" one for A and the corresponding product topology as the "natural" one for the product space C. Note that the product topology is symmetric with

respect to its factors, in the sense that its definition is not affected by any particular ordering of the factors. In the same context, the "natural" metric for A is the trivial metric d, defined by

$$d(a_1,a_2) = 0 \text{ (for } a_1 = a_2 \text{)}$$
$$1 \text{ (for } a_1 \neq a_2 \text{)} ,$$

which is compatible with the discrete topology on A. The question arises whether such metric can be extended (as a new metric D) to the product space C in such a way that D is compatible with the product topology and symmetric with respect to its factors. Theorem 5.2.1 below shows that this is not possible. On the other hand, it is possible to define a whole collection of topologically equivalent metrics, all compatible with the product topology, each of which is, so to speak, "biased" in favor of certain terms of the product. For instance, for any enumeration $\langle s_i \rangle_{i=1}^{\infty}$ of the set of cells S, the metric

$$D(c,c') = \sum_{i=1}^{\infty} \frac{1}{2^i} d(c_{s_i},c'_{s_i})$$

generates the product topology. Another topologically equivalent metric that is encountered in the context of symbolic flows is described in [SIBI75,SATO75]. The fact that topological metrics on the set of configuration are not symmetric, and thus do not respect the uniformity policy that is at the base of cellular automata theory, suggest that perhaps metric considerations are not

as "natural," in this context, as topological ones.

In what follows, let A denote a finite set containing at least two elements (denoted by 0 and 1, respectively). A will be treated as a topological space with the discrete topology. Let C denote the indexed product $\underset{i\in I}{\times} A$, where I is a countable index set. C will be treated as a topological space with the product topology.

DEFINITION 5.2.1. A metric D on C is symmetric [with respect to I] if it is invariant under any permutation on I. More explicitly, given any permutation p on I, let $c(= \underset{i\in I}{<} c_i>)$, $d(= \underset{i\in I}{<} d_i>)$ be arbitrary elements of C, and let

$$c' = \underset{i\in I}{<} c_{p(i)}> \ , \quad d' = \underset{i\in I}{<} d_{p(i)}> \ .$$

The metric D on C is symmetric if, for any p,

$$D(c,d) = D(c',d') \ .$$

THEOREM 5.2.1. If a metric D on C is symmetric, then D is not compatible with the product topology.

Proof. Consider the infinite sequence $\underset{i=0}{\overset{\infty}{<}} c^i>$ of configurations of C defined as follows

$$c^0 = <1,0,0,0,0,\ldots> \ ,$$

$$c^1 = <0,1,0,0,0,\ldots> \ ,$$

$$c^2 = <0,0,1,0,0,\ldots> \ , \text{ etc.}$$

c^0 and c^1 are distinct. Let $h = D(c^0,c^1) \neq 0$. Note that any pair $\langle c^i,c^j\rangle$ $(i\neq j)$ can be transformed into $\langle c^0,c^1\rangle$ by a suitable permutation of the index set I . Thence, if D is symmetric,

$$\bigwedge_{i\neq j} D(c^i,c^j) = h \ .$$

Thus, the distance between any two elements of the sequence is bounded from below. On the other hand, the topological space C (with the product topology) is a complete, compact metric space (cf. preceding section), and every infinite sequence in it contains a convergent subsequence. Therefore, any symmetric metric on C induces a topology that is distinct from the product topology.

REMARK 5.2.1. We have noted that there exist nonsymmetric metrics that are derived from the trivial metric and are compatible with the product topology. At the same time, there exist symmetric metrics derived from the trivial metric that are not compatible with the product topology. Such is, for instance, the metric D defined by

$$D(c,c') = \max_S(d(c_s,c'_s)) \ .$$

5.3 Topological characterization of cellular automata

Displacements, introduced in Section 1.2 as operators on the set of cells, can be interpreted as operators on the set of configurations. The action of a displacement $\bar{s} \in S$ on a configuration $c \in C$ is defined by

$$c' = \bar{s}c \iff \bigwedge_{s \in S} c'_s = c_{-\bar{s}+s} \quad .$$

A property of the parallel maps on C is that they commute with every element of the tessellation group S. In other words, the paralled map of a cellular automaton is--by construction--translation invariant. (For brevity, we shall say that a transformation τ on C commutes with S if it commutes with every element of S.) However, commutativity with respect to S is not sufficient to characterize the class of parallel maps, as shown by the following

EXAMPLE 5.3.1. Let $C = B^S$, where $B = \{0,1\}$ and $S = C_\infty$. Let c^0 be the configuration consisting of all zeros, and c^1 that consisting of all ones. Consider the function μ that maps a configuration $c \in C$ into c^0 if $c = c^0$, and into c^1 otherwise. Clearly, μ commutes with S. However, the next state (under transformation μ) of each cell depends on the current value of all cells. Thus, μ cannot be expressed in terms of a local map acting on a finite neighborhood, and, consequently, is not a parallel map.

A second property shared by all parallel maps is continuity with respect to the product topology. The sequential continuity of the parallel map of a cellular automaton was proved in [RICH72]. Moreover, continuity (which implies sequential continuity) can be proved very easily, using [LIPS65,Theorem 7.2], by noting that the counter-image of any one-cell pattern is a pattern (and thus an open set). Topological continuity is analogous to (and for the real-line topology it coincides with) the familiar "ε-δ" property encountered in elementary calculus. Intuitively, a function is continuous if small changes in the argument lead to small changes in the value of the function. For cellular automata, a "small change" is a change in the state of a small number of cells. Two configurations c^1, c^2 that differ only in the state of one cell give rise to successors $c^{1'}, c^{2'}$ that differ in at most a finite (and, because of translation invariance, bounded) number of cells. In physical terms, this can be interpreted as saying that the speed of propagation of information (the "speed of light") in cellular automata is bounded.

If a set C can be written as a Cartesian product indexed by S, as in Section 5.1, and a map $\tau: C \to C$ commutes with S, then the continuity of τ with respect to the product topology is not only necessary but also sufficient to guarantee that the dynamical system $\langle C, \tau \rangle$ can be described by a cellular automaton (sufficiency too is proved in [RICH72]). More generally, let C be a set, S a discrete free Abelian group of transformations on C, and μ a map from C to C. A topology T on C is symmetric [with respect

to S] if, for any open set $Y \subset T$, $sY \subset T$ (for all $s \in S$) . Suppose that μ commutes with S and is continuous with respect to a symmetric topology T on C . Such conditions are not sufficient to characterize the parallel map of a cellular automaton, as shown by the following

EXAMPLE 5.3.2. Given a cellular automaton [C[A(r>1),S(d>0)], τ] with a blank state, consider the set C' obtained from C by deleting all infinite configurations. Such deletion yields, starting from the product topology U on C , a topology U' on C' which is also symmetric with respect to S . Let τ' be the restriction of τ to C' . It is easy to verify that τ' commutes with S and is continuous with respect to U' . However, there is no cellular automaton whose associated dynamical system is $\langle C', \tau' \rangle$, since C' is denumerable while the set of configurations of a cellular automaton is always noncountable (except for certain trivial cases where it is finite).

Intuitively, in the dynamical system associated with a cellular automaton the state of each cell can be used as an independent coordinate in specifying a point of the phase space (thus, one can associate a degree of freedom with each cell). Instead, in the system $\langle C', \tau' \rangle$ certain values of a coordinate are "forbidden" as soon as the value of certain other coordinates is specified. In other words, the labeling schema that encodes a point of the phase space as an array of cell-states is underutilized, in the sense that many possible codes do not correspond to any state of the system.

What additional constraints must be satisfied by a topology T and a function μ as above in order for μ to be the parallel map of some cellular automaton? As far as we know, a characterization of cellular automata in these terms has not appeared in the literature. However, a number of recent papers (e.g., [KEYN69,GOOD70,RYAN72, SEAR73]) consider essentially this problem in the similar context of symbolic dynamics. In particular, [GOOD70] gives a characterization for shift dynamical systems, which coincide with cellular automata of a very simple kind.

CHAPTER 6

COMPUTATION COMPLEXITY AND COMPUTING POWER

6.0 Preliminaries

Intuitively, a "powerful" computing machine is one that can perform a complex task. However, it is difficult to agree a priori on whether one task is more complex than another until an attempt has been made to specify a computing environment where both tasks can be performed. If one characterizes such an environment by the kind of computing resources that are available, then task complexity can be measured in terms of the amount of resources that are needed in order to perform a given task.

We shall propose the following

THESIS 6.0.1. Cellular automata of appropriate dimensionality characterize the kind of computing resources that are available in a physical computing environment.

That the computing resources offered by cellular automata are indeed available in a physical environment is proved in Sections 2.1-2.3.

Thesis 6.0.1 states also the converse, i.e., that in such an environment substantially no other resources are available. This amounts to saying that, given a very large physical computer satisfying certain limiting technological parameters (nature of the materials used, degree of miniaturization, heat removal capacity, etc.), the same computer can be simulated by a cellular-automaton implementation that satisfies the same limiting parameters, and in such a way that volume and clock rate of the implementation differ from that of the original computer by numerical factors that are independent of size.

Since the concept of physical computation is not formally defined, we shall not make an attempt to "prove" Thesis 6.0.1. Nevertheless, in favor of it we shall offer the following intuitive argument. Given a physical computer, consider a fine-meshed three-dimensional grid that divides the computer into small volume elements. Each volume element will contain some piece of computing machinery, such as a gate or a length of wire. One can imagine a cellular-automaton cell that can be programmed, by suitably setting some of its state components, in such a way as to simulate the behavior of any one of those pieces of machinery. In such simulation, a volume element of the original computer is replaced by a cell, and the speed of signals in the computer (which is bounded) is replaced by the one-cell-per-time-step speed. Thus, by scaling time and volume by fixed factors, an arbitrary physical computer can be replaced by one structured like a cellular automaton.

In the following sections, we shall show how one can derive

from the above thesis a consistent schema for measuring certain aspects of computation complexity, and we shall compare such schema with other well-known ones that measure different aspects of computation complexity.

6.1 Computability

According to Chaitin, "computation complexity differs from recursive function theory in that, instead of just asking whether it is possible to compute something, one asks exactly how much effort is needed to do this."[CHAI70] To speak of "how much" implies, of course, that one accepts a measuring schema, i.e., a procedure for constructing the desired object (or a model thereof) by repeated use of a given unit or, more generally, of assigned primitives.

It may happen that different measuring schemata--or the same schema using different sets of primitives--yield comparable results (for instance, one can consistently measure all lengths in inches or in centimeters). It is indeed such invariance which leads one to think that the quantity measured by a certain measuring schema is, in some sense, meaningful or relevant. Thus, to restrict ourselves to computational problems, Church's thesis--that the "effectively calculable" functions should be identified with the general recursive functions[CHUR36]--was formulated after realizing that at least two

independently suggested schemata (λ-definability[CHUR32] and general recursiveness[GOED34]; also cf. those listed in [KLEE67]) defined the same class of functions. For all of those schemata, one can think of a measured quantity called computability that takes on the value 0 or 1 according to whether a function can or cannot be constructed starting from certain primitives.

6.2 Randomness, or definition complexity

The computational "difficulty" of a function can be further quantified. Chaitin and Kolmogorov independently discovered a schema for measuring a quantity called randomness, associated with the amount of effort that is needed to define a function[CHAI66,KOLM65]. In this case, one typically uses as primitives the instruction set of a computation-universal programming language. The randomness of a function is given by the length (i.e., the number of instructions) of the shortest program that can compute it. While computability is concerned with whether certain primitives are at all sufficient to define a function, randomness asks how many occurrences of such primitives are needed to define it. It is true that the measuring schema for randomness gives different results when used with different primitive sets; however, it can be shown that these results differ by a bounded additive term. Thus, the difference is negligible when the randomness is high, and, in this sense, randomness is a well-defined

quantity.

6.3 Minimal look-up networks and evaluation complexity

We ask next how much effort is required in order to actually evaluate--instead of merely define--a function.

While both computability and randomness are attributes of a function as a whole, the amount of computing effort will depend, in general, also on the particular value of the argument. In order to avoid, for the moment, this difficulty, we shall consider combinatorial functions, i.e., functions whose domain and codomain are both finite sets. Without loss of generality, we may restrict our attention to the case where arguments and values are, respectively, m-tuples and n-tuples over the set $\{0,1\}$, with m and n fixed for a given function (Figure 6.1). In this case, one can always

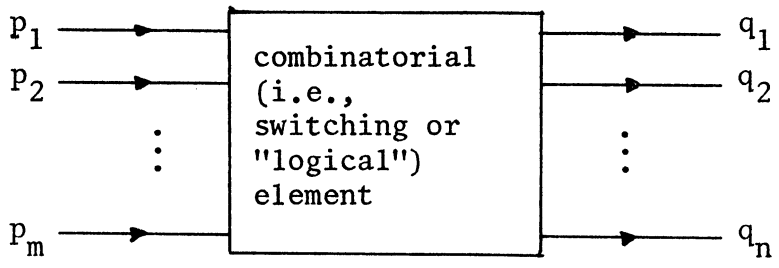


FIGURE 6.1. Combinatorial function represented as a combinatorial network having m input lines and n output lines.

represent the computation process as a table look-up. To be explicit, one can use the disjunctive normal form of the function as a blueprint for a combinatorial network of the following form (Figure 6.2):

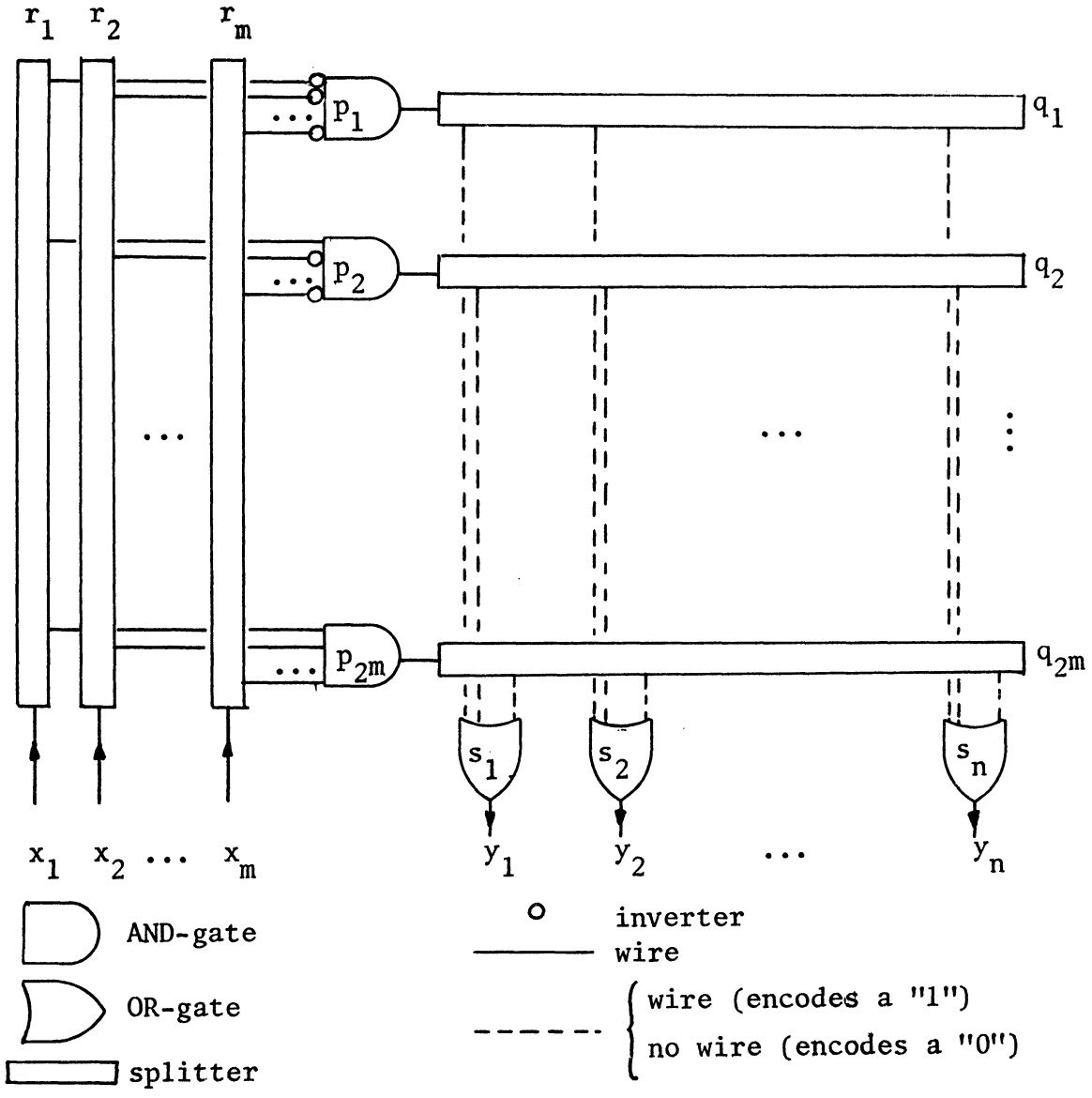


FIGURE 6.2. Gates $p_1,\ldots,p_{2^m}$ are set up as an address decoder, and activate one and only one of the splitters $q_1,\ldots,q_{2^m}$ for any value of the input m-tuple. The corresponding output n-tuple is coded for by a selection of dashed lines (in the actual network, each of the dashed lines would be replaced by a wire for a 1 in the output, and by no wire for a 0) and read off by the OR-gates $s_1,\ldots,s_n$.

In order to realize with only two levels of gating (as in Figure 6.2) any combinatorial function, an unboundedly large number of different primitives would be needed, since gates with an arbitrarily large fan-in as well as splitters with an arbitrarily large fan-out must be available. However, bounded fan-in and fan-out can be achieved by increasing the number of levels. It is well known that any combinatorial function can be realized by a combinatorial network that uses, for instance, only two-input NAND-gates, two-way splitters, and connecting wires.

The minimum number of occurrences of such primitives that is required to realize a given combinatorial function can be taken as a definition of the function's evaluation complexity (as contrasted with its definition complexity or randomness). Such a minimal-count network can be interpreted as a particularly sophisticated look-up procedure which has been optimized in view of the most economical use of a given kind of resources (namely, gates and splitters), independently of the amount of effort that the design of the network may entail.

Of course, by choosing a larger set of primitives, one could obtain a reduced complexity count. However, in analogy with the situation for randomness, different primitive sets yield results that differ by a bounded multiplicative term. (In order to prove this, it is sufficient to observe that any element from a finite universal set of combinatorial-network primitives can be synthesized from a finite number of elements from another universal set). In this sense,

evaluation complexity is a well-defined quantity.

6.4 Algorithm-constrained look-up networks

We are well aware that the above situation, based, as it is, on the minimum amount of computation that is required in order to evaluate a function, is a rather unrealistic one. To use Chaitin's words,

> "In a typical scientific application, the computer may be used to analyze statistically huge amounts of data and produce a brief report in which a great many observations are reduced to a handful of statistical parameters. We would view this in the following manner. The same final result could have been achieved if we had provided the computer with a table of the results, together with instructions for printing them in a neat report. This observation is, of course, ridiculous for all practical purposes. For, had we known the results, it would not have been necessary to use a computer. This example, then, does not exemplify those aspects of computation that we will emphasize.
>
> Rather, we are thinking of such scientific applications as ... calculating the apparent positions of the planets as observed from the earth over a period of years. A small program incorporating the very simple Newtonian theory for this

> situation will predict a great many astronomical observations. In this problem there ... [is] only a program that contains, of course, a table of the masses of the planets and their initial positions and velocities." [CHAI66]

In other words, in most situations of interest the function to be evaluated is not given explicitly, by means of a table, but implicitly, by means of an algorithm. How shall one appraise the cost of computation in this case? Clearly, one is interested in some quantity associated with the given algorithm, not merely with the function itself. For simplicity, we shall consider first algorithms that can be realized by a finite automaton allowed to run for a given number of steps. The function's argument will be encoded in the automaton's initial state (the automaton is input-free). Upon reaching any one of a selected set of terminal states, the automaton shall "latch" on such state. At the end of the allotted time, the function's value for that argument is read off the terminal state actually reached.

Of course, lest we again expose ourselves to Chaitin's objections, we cannot assume explicit knowledge of the automaton's state-transition diagram. On the other hand, it is legitimate to translate the algorithm into a finite sequential network whose structure is thus explicitly given and whose behavior, only implicitly known, corresponds to the behavior of the required automaton.

Suppose one has obtained in this way, say, the network of Figure 6.3, which is supposed to start from a given initial state for

the delay elements and to operate for t time steps.

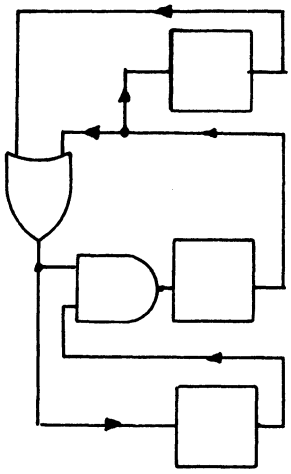


FIGURE 6.3. Finite network which realizes a certain algorithm when started in the appropriate initial conditions. The square boxes represent delay elements.

From the viewpoint of actual computation, the same network elements are used over and over, in time, to perform different steps of the algorithm. If one wants to keep track of each usage, one can unfold the network along the time dimension, as shown in Figure 6.4.

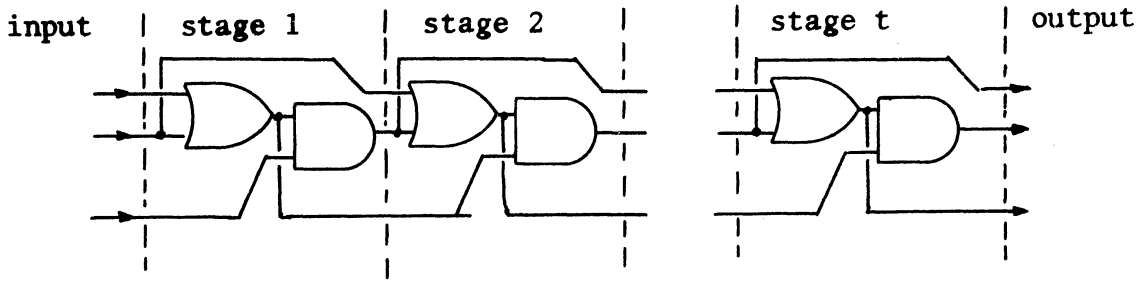


FIGURE 6.4. The same network as in Figure 6.3, but unfolded along the time dimension in order to explicitly show the repeated usage of each computing element. The new network is purely combinatorial and loop-free.

Of course, the so called "delay" elements have been removed, since their only function was to regulate the repeated usage of the other elements of the network (cf. Section 2.4.1). The new network, which is purely combinatorial, is loop-free by construction. Note that now time, instead of being implicitly represented by the delay elements, is explicitly represented by the iteration of the original network; thus, in evaluating a function for a certain value of the argument, signals travel through the new network only once. Any particular choice of the input (which replaces the initial state of the sequential network) completely determines the signal that will pass through a wire (we shall call this the state of the wire). However, while the network displays an iterative pattern consisting of identical stages whose structure is assigned a priori, the state of the wires for any given input is not iterated, in general, from stage to stage. From the viewpoint of the computer's user, the pattern formed by the state of the wires is "impredictable."

In contrast with a network synthesized directly from a look-up table, the construction of a combinatorial network like that of Figure 6.4 does not require any prior knowledge of the function's values. One the other hand, such a network will require, in general, many more elements than a minimized one. In a sense, this is the price one has to pay in order to get something "new" from a computation. Note that one again realizes, with such a network, a particularly sophisticated look-up mechanism. This time, instead of minimizing the number of computing elements, one has minimized the cost of

design by restricting oneself to the ready-available plan represented by the given algorithm. Since the structure of the network corresponding to a given algorithm is static and explicitly given (aside from simple transliteration rules that can be codified once and for all), one can count the number of occurrences of primitives in it. According to the definition of measuring schema given in Section 6.2, we recognize, in the reconstruction of an algorithm in terms of combinatorial primitives, a way of measuring the computing effort that the algorithm entails.

6.5 Nonterminating algorithms

The above measuring schema appears "naive" in the sense that, by counting also the network nodes that occur after the algorithm has halted, it may grossly overestimate the amount of resources that are effectively required. Moreover, the schema only works for algorithms that compute combinatorial functions, while, in general, the theory of computation deals with algorithms that accept inputs of arbitrary size and may require an unbounded number of time steps. On the other hand, any refinement of the measuring schema in this direction would imply some unspecified mechanism which should continuously "monitor" the computation in order to throw in additional resources when needed and divert any unused resources to another task when the halting state is reached. In this case, should one include in the gate count the

additional switching machinery required for the dynamic resource-allocation mechanism? And who would monitor the "monitor?" It is clear that, if this refinement process is carried all the way, one must end up with a computing schema able to incorporate also the computer's user.

A reasonable way out of the above dilemma can be found, for instance, in the approach used by Blum[BLUM67], also in the context of computation complexity. There, the conceptual availability of a universal Turing machine U together with the associated unbounded tape is replaced by the availability, for any d and n , of the combinatorial function

$$y = f_{d,n}(x)$$

which, given an arbitrary string x of length d , computes a string y such that, if x is the initial state of the tape submitted to U , y is the state of the same tape after n time steps. This is just a more pedantic, but, from a finitary viewpoint, more satisfactory, way of couching the concept of effective calculability. (In physical terms, this corresponds to the situation where the user identifies a computation by encoding the input in a certain number of physical variables and specifying , in a way independent of the particular value of the input argument, what other variables should be treated as an output.) Therefore, without loss of generality, we can restrict our attention to combinatorial functions, provided that we also supply a way of constructing, for any value of d and n , a

combinatorial network that realizes $f_{d,n}$, without requiring for the construction any explicit knowledge of the function's values. This we shall do in Section 6.7.

6.6 Interconnection as a computing resource

Our considering networks constrained to implement a given algorithm in order to evaluate a certain function, instead of just limiting our attention to minimized networks, removes the objections raised, for instance, by Lawler against combinatorial primitives as a basis for a meaningful complexity measure [LAWL71]. However, we should like to raise a new objection ourselves. In identifying the effort entailed by an algorithm with the number of nodes of a certain network, we have completely neglected the possible significance of their interconnection pattern. Does this not contribute to the intuitive "complexity" of a computation? Alternatively, would we not have to increase the number of nodes if we were restricted to using only an assigned set of interconnection patterns? After all, the theory of computation complexity is based on concepts whose prototypes (storage size, computation time, etc.) occur as physical constraints to computation, and a network's interconnection pattern is likewise subjected, in the real world, to physical constraints (cf. Section 1.3). Since little has been written on this subject in the computation-complexity literature, we shall briefly outline an

approach to this problem consistent with the above treatment.

6.7 Uniform combinatorial networks

In order to make explicit the "cost" (or "value") of interconnection as a computing resource, we shall follow the generally accepted policy of limiting such a resource and observing the effects that this produces on the utilization of the other resources. Namely, we shall consider networks that are uniformly interconnected, so that no computation complexity would be "hidden" in the choice of a specific interconnection pattern. Intuitively, this is equivalent to restricting to a single element the set of interconnection "primitives." In order to simplify the discussion, we shall restrict to a single element also the set of computing primitives, i.e., we shall consider networks consisting of identical elements. Our plan is to consider computations that are carried out in these networks, and to measure the cost of a computation in terms of the number of network nodes that are required.

Note that a sequential network consisting of identical, uniformly interconnected elements is a uniform automaton (as defined in Section 1.2). Thus, in order to take into account the contribution given to evaluation complexity by the interconnection pattern of a network, we are led to the study of computation as performed in uniform automata. Following a procedure analogous to that described

in Section 6.4, one can unfold along the time dimension the sequential network which represents a uniform automaton, thus generating a combinatorial network consisting of an iteration of identical stages (one for each time step), as indicated in Figure 6.5, which represents the combinatorial network obtained from a one-dimensional cellular automaton. We shall call uniform a sequential network that can be obtained in this way. Note that the uniformity group of such a network is $S \cdot C_\infty$, where S is the tessellation group of the given uniform automaton and C_∞ is the additional uniformity component introduced by the iteration process.

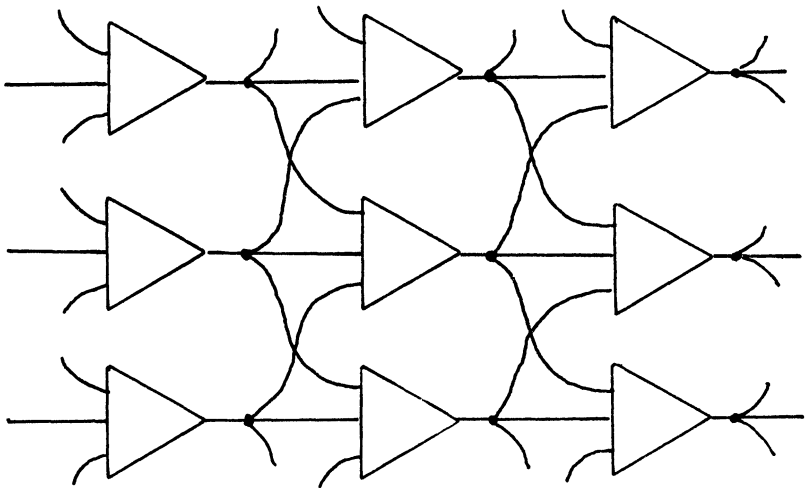


FIGURE 6.5. Uniform combinatorial network generated by a one-dimensional cellular automaton. Each column represents the cellular automaton at a given time step, while each row represents the same cell used over and over in time. The arcs correspond to a neighborhood template of the form $\langle -1,0,+1 \rangle$.

Once the underlying uniform automaton has been assigned, the structure of the corresponding (time-unfolded) combinatorial network is completely specified. The only variables that can be manipulated in order to perform different computations in such network are (i) the number of stages (cf. Section 6.4), (ii) the set of output lines off which the output value is going to be read, and (iii) the values of the input signals. A portion of the network consisting of a certain number of stages (corresponding to operating the uniform automaton for as many time steps) has in general an infinite number of input and output lines. However, given the finite speed of propagation of information in uniform automata, the state of an assigned finite set of output lines is affected only by the state of a well-defined finite subset of input lines, as illustrated in Figure 6.6. Thus, by selecting the number of stages and the output lines, one automatically defines a portion of the network having well-defined input and output lines and capable of computing a certain combinatorial function. Since there exist computation-universal uniform automata, it is easy to verify that, by appropriately "programming" the value of certain input lines and treating the value of certain output lines as a function of the remaining input lines, one can use combinatorial networks to compute arbitrary combinatorial functions, in particular, the $f_{d,n}$ functions defined in Section 2.5. In this sense, uniform combinatorial networks are computation-universal.

In general, it is not necessary to use a portion of the network having "trapezoidal" shape as in Figure 6.6. Any surface

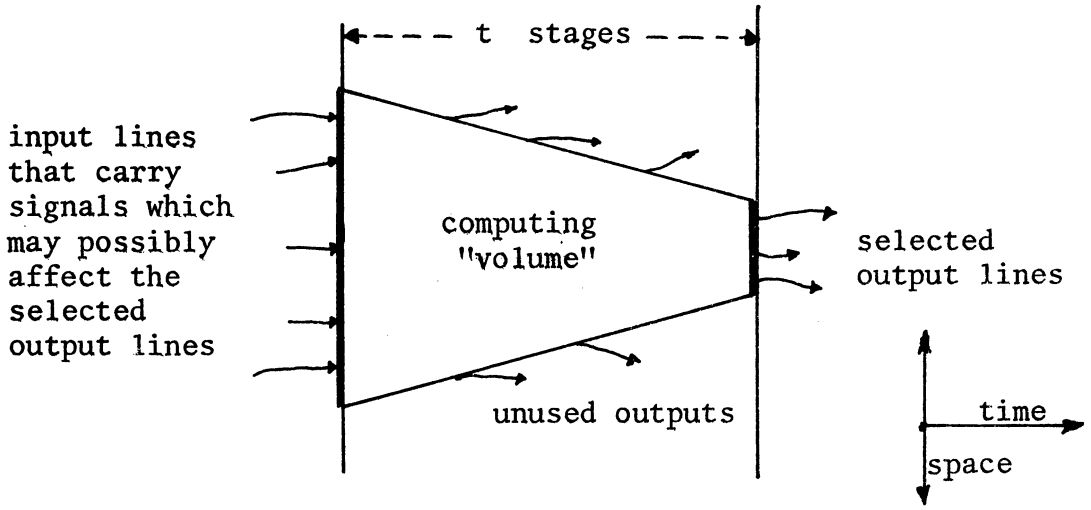


FIGURE 6.6. t stages of a combinatorial network such as that of Figure 6.5. By selecting certain output lines, one automatically determines the portion of the network and the input lines that may possibly affect the output. The slope of the sides of the trapezoid corresponds to the "speed of light"--i.e., one neighborhood radius per time step.

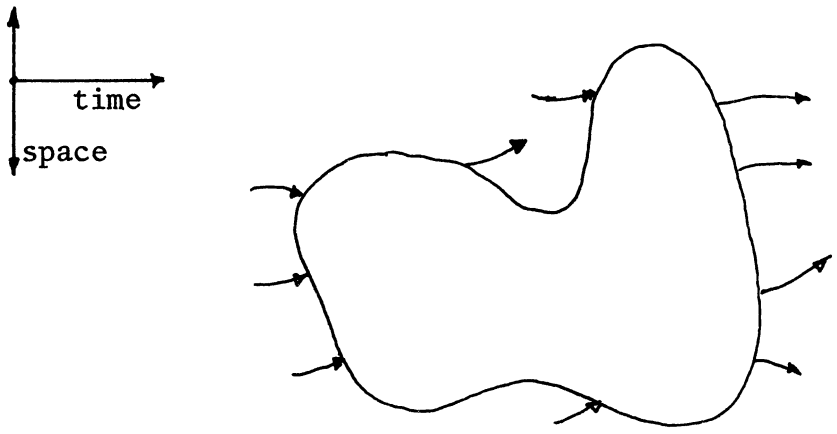


FIGURE 6.7. Given an arbitrary uniform combinatorial network (which is supposed to underly the figure), one can define a computation by delimiting (by means of a closed "surface") a portion of the network itself. A distinguished subset of the output lines will represent the output of the computation, while the input lines represent the input of the computation.

enclosing a finite volume, as sketched in Figure 6.7, has well-defined input and output lines and identifies a particular combinatorial network. The number of nodes contained in such volume can be taken as a measure of the amount of computing resources with which that combinatorial function can be realized. One can constrain the network to compute a function by means of a particular algorithm, by suitably assigning the value of the "programmed" input lines. Thus, the measuring schema of Section 6.4 can be extended to networks satisfying a fixed interconnection pattern.

The above considerations remain valid if one uses cellular automata instead of more general uniform automata. In this case, one obtain a measure of computation cost that is compatible with physical constraints.

6.8 Physical aspects of computing resources

Let us apply the concepts introduced in the preceding section to uniform combinatorial networks generated by three-dimensional cellular automata. In this case, according to Thesis 6.0.1, each "portion" of the network corresponds to a volume of spacetime. The selection of particular network arcs to be associated with the input [output] of a computation corresponds to specifying the spacetime surface which the input [output] signals will cross in entering [leaving] that volume. Thus, the amount of resources available to the

computation is proportional to the volume of spacetime containing the causal links (i.e., the network arcs) involved in the computation, and to the mass/energy pressure (i.e., the packing density of network nodes) in that volume. The product of these these two physical quantities has the dimension of action. Thus, Thesis 6.0.1 implies that the natural unit for measuring the amount of resources expended by a computation is the quantum of action h, i.e., Planck's constant. Note that action is invariant under relativistic transformations. Thus, our proposed measure of "computing resources" has the same value in different inertial frames.

6.9 Lorentz transformations in cellular automata

In Section 6.7, the dynamical semigroup of a cellular automaton appears in an explicit form as a symmetry subgroup of a uniform combinatorial network. This permits one to deal with certain coordinate transformations that involve both space and time and whose formulation would be awkward if stated directly in cellular-automaton terminology.

Here, we shall show in an informal way that transformations analogous to the Lorentz transformations of relativity are meaningful for the dynamical systems associated with cellular automata.

Consider, for simplicity, a one-dimensional cellular automaton M with neighborhood template $\langle -1,+1 \rangle$. The combinatorial network generated by M (cf. Figure 6.5) has the form illustrated in Figure

6.8a, where s and t indicate, respectively, the space axis and the time axis. Note that the network consists of two connected components, each having the form of Figure 6.8b. We shall consider only one component, as the other is identical. Let us group the nodes of the network into blocks, as indicated in Figure 6.9. Each of the blocks has the internal structure depicted in Figure 6.10a and (since it does not contain any delay elements) can be replaced by a single node, as in Figure 6.10b. After this transformation, the network of Figure 6.8b will appear as in Figure 6.11a, or--after merging parallel arcs--as in Figure 6.11b.

It is clear that one can label the nodes of the new network using the axes s',t' indicated in Figure 6.11b by dashed lines. This labeling corresponds to treating the new network (which performs the same computation as the old one) as generated by a new cellular automaton M' having neighborhood template <-1,+1> . Note that, in general, M' will be different from M . In other words, the form of the dynamical laws of the given system is not invariant with respect to "Lorentz" transformations. However, the "speed of light," as measured in the new frame of reference, is the same as in the old one.

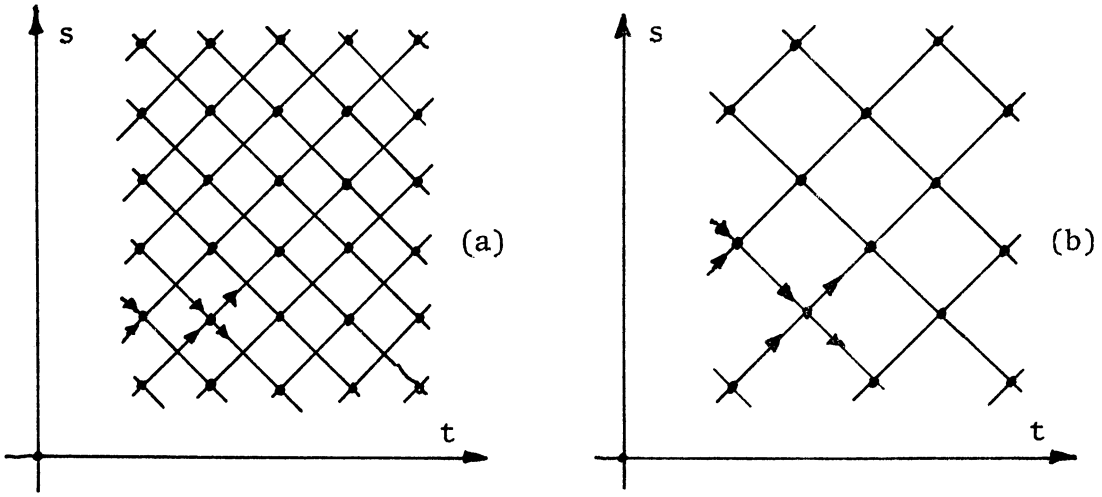


FIGURE 6.8. The uniform combinatorial network generated by a cellular automaton having neighborhood template <-1,+1> is represented in (a). Such network consists of two distinct connected components, each having the form illustrated in (b). There is no interaction between the two components. s and t represent, respectively, the space axis and the time axis.

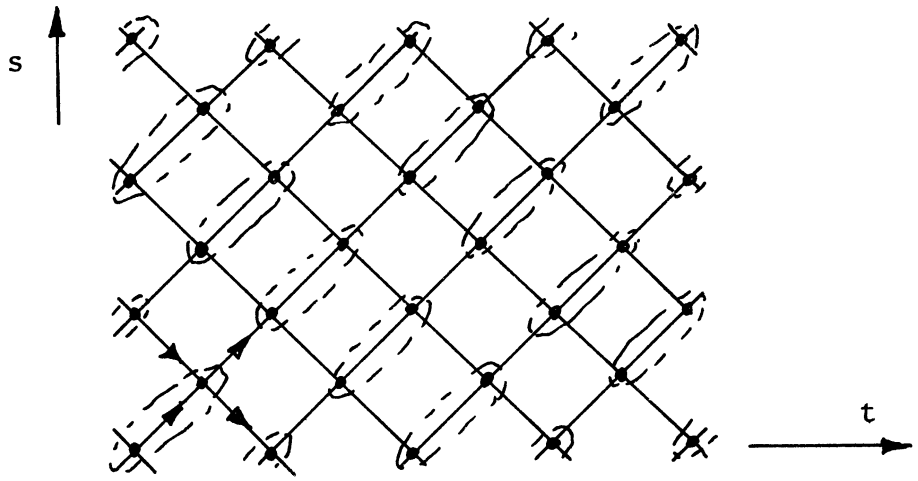


FIGURE 6.9. The nodes of the network illustrated in Figure 6.8b are grouped in blocks of two, as a preliminary to a linear transformation of coordinates involving both space and time.

FIGURE 6.10. Each of the blocks of Figure 6.9 has the internal structure indicated in (a). The functions of the two nodes can be lumped into a single combinatorial element (b).

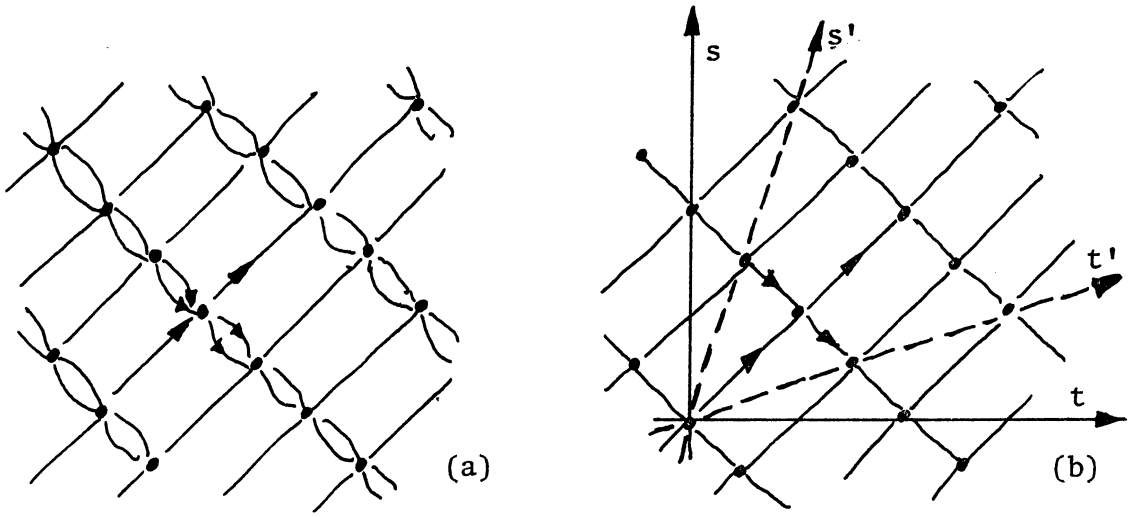


FIGURE 6.11. After the lumping operation of Figure 6.10, the network of Figure 6.9 will have the aspect shown in (a). Parallel arcs can be merged, obtaining the network shown in (b) . Note that the structure group of this network is identical to that of Figure 6.8b. Thus, the network of (b) can be obtained by unfolding along a time axis indicated by t' a cellular automaton whose cells are spatially arranged with the orientation indicated by s' .

CONCLUSIONS AND PERSPECTIVES

Mathematics has long emancipated itself from the requiremennts that its abstract constructs have a physical counterpart; nevertheless, its modes of expression (collectively called computing), bound as they are to a physical medium, necessarily remain subject to physical constraints. Only recently have people begun investigating the nature of these constraints and their relevance not only to practical computing but also to the very foundations of mathematics. In order to study this problem from a formal viewpoint, mathematicians have constructed abstract concepts that are more or less closely related to the objects and the operations of computing, thus obtaining diverse theories of computing. However, it must be emphasized that the subject matter of computing belongs to the domain of the natural sciences.

While any physical experiment can be interpreted as a computation, a theory of computing need not involve itself in all the details of the actual physical laws. In fact, many distinct physical mechanisms are, at a certain level of schematization, isomorphic (roughly speaking, they induce the same formal relationship between input and output variables), and can be described by a single paradigm. In order to be manageable, a comprehensive theory of computing should by and large avoid introducing a formal counterpart for too large a variety of isomorphic mechanisms; at the same time,

it should strive to retain all those aspects of physics that are expected to play an essential role in determining what modes of expression are effectively available to mathematics.

In this context, cellular automata constitute a substantial improvement over Turing machines. In the real world, a given system may act as a symbolic structure in a given context and as a functional structure in a different one. In the first case, the mechanical properties of the structure's support are accidental and largely irrelevant; in the second, they are an integral part of the structures's specifications. (Note that, for the same system, the two roles may alternate as the context changes.) On the other hand, in a Turing machine the functional component (i.e., the head) and the symbolic component (the tape) are intrinsically heterogeneous, and one cannot use, as it were, material form the environment (the infinite tape) in order to augment the real-time processing capabilities of the active subject (the finite head). Cellular automata are much less "anthropomorphic" (note that Turing explicitly presented his machines as an idealization of a human computer). Within a physically realistic framework, they provide a self-contained theory in which many "material" as well as symbolic activities that are usually conceived as belonging to different abstraction levels (such as "computation," "construction," "evolution," etc.) can be explicitly represented as taking place in a portion of one and the same unbounded, homogeneous dynamical medium.

This approach is relatively novel and may offer the key to

many questions that are not easily treated in the more limited context of recursive-function theory. From current work in computation complexity, programming languages, etc., it is evident that there is much more to computing than mere computability.

> "What is a machine? What is a computable process?" asks Dana Scott in his Turing Award Lecture[SCOT77]. "How (or how well) does a machine simulate a process? Programs naturally enter in giving descriptions of processes. The definition of the precise meaning of a program then requires us to explain what are the objects of computation (in a way, the statics of the problem) and how they are to be transformed (the dynamics).
>
> So far the theories of automata and nets, though most interesting for dynamics, have formalized only a portion of the field, and there has been perhaps too much concentration on the finite-state and algebraic aspects. It would seem that the understanding of higher-level program features involves us with infinite objects and forces us to pass through several levels of explanation to go from the conceptual ideas to the final simulation on a real machine. These levels can be made mathematically exact if we find the right abstractions to represent the necessary structures.
>
> The experience of many independent workers with the method of data types as lattices (or partial orderings) under an information content ordering, and with their continuous mappings, has demonstrated the flexibility of this approach in providing definitions and proofs, which are clean and without undue dependence on implementations. Nevertheless much remains to be done in showing how abstract conceptualization can (or cannot) be actualized before we can say we have a unified theory." (italics mine)

(Note that the mappings mentioned in the above quotation are continuous with respect to a topology for data structures that is analogous to that discussed in Chapter 6 for cellular automata.)

The results already available from cellular automata theory suggest that a comprehensive theory of computing can be founded (and other evidence suggests that it ought to be founded) on an appropria-

te schematization of physics. In order to determine whether the cellular-automaton schematization is indeed appropriate, the following questions must be investigated in much greater detail than in the present work.

(1) What physical-like constraints (in addition to those already implicit in their definition) must cellular automata obey to be consistent with their intended role as models of the computational aspects of nature?

(2) What correspondence should be used for associating objects and phenomena in cellular automata with objects and phenomena in nature? For example, what counterparts, if any, are to be found in cellular automata for energy, momentum, gravitation, etc.?

(3) To what extent can the general mathematical methods of physics (e.g., harmonic analysis, calculus of variations, etc.), which were originally conceived for continuous problems, be applied to cellular automata?

To us, question (3) above seems at this moment the most interesting and the most likely to bear immediate fruit. In fact, the similarity of setting (dynamic causality in space- and time-coordinates) makes many of the concepts of theoretical physics immediately transferable to cellular automata in spite of certain technical differences Thus, the analogy with physics both suggests a great number of nontrivial problems for cellular automata theory and indicates the way to their solution. (It may be remarked that our results on reversibility owe much to physical intuition.)

The fact that reversibility is compatible with universal computing and constructing capabilities represents critical evidence for the belief that the cellular-automaton schematism is able to capture--in a domesticated and thus very tractable form--the essentials of physics. If this is true, cellular automata represent a nec ultra warning (cf. our thesis of Section 6.0) which may constructively influence the development of extremely large scale computing hardware and the direction of research in parallel computing. As Turing machines epitomize what can be "done," cellular automata epitomize how that can be done.

REFERENCES

ADLE65 Adler R.L., Konheim, A.G., Andrew M.H., "Topological Entropy," Trans. Amer. Math. Soc., 114 (1965), 309-331.

ALAD72 Aladyev V., "Computability in Homogeneous Structures," Izv. Akad. Nauk. Estonian SSR, Fiz.-Mat.21 (1972), 80-83.

ALAD73 Aladyev V., "Some Questions Concerning the Nonconstructibility and Computability in Homogeneous Structures," Izv. Akad. Nauk. Estonian SSR, Fiz.-Mat.22 (1973), 210-214.

AMOR72 Amoroso S., Patt Y.N., " Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tessellation Structures," J. Comput. System Sci. 6 (1972), 448-464.

AMOR75 Amoroso S., Cooper G., Patt Y.N., "Some Clarifications of the Concept of a Garden-of-Eden Configuraion," J. Comput. System Sci.10 (1975), 77-82.

BALZ67 Balzer R.M., "An 8-State Minimal Time Solution to the Firing Squad Synchronization Problems," Inform. Contr.10 (1967), 22-47.

BANK71 Banks E.R., "Information Processing and Transmission in Cellular Automata," Ph.D. Thesis, M.I.T. (1971).

BART75 Barto A.G., "Cellular Automata as Models of Natural Systems," Tech. Rep. NO. 183, Logic of Computers Group, Comp. Comm. Sci. Dept, Univ. Michigan (1975).

BASS72 Bass H., "The Degree of Polynomial Growth of Finitely Generated Nilpotent Groups," Proc. London Math. Soc. 25 (1972), 603-614.

BEIS74 Beister J., " A Unified Approach to Combinatorial Hazards," Trans. IEEE C-23 (1974), 566-575.

BENN73 Bennett C.H., "Logical Reversibility of Computation," IBM J. Res. Develop. 6 (1973), 525-532.

BIRK27 Birkhoff G.D.,Dynamical Systems, Amer. Math. Soc. Colloq. Publ., 9 (1927).

BLAN76 Blanchard A., Phase-Locked Loops--Application to Coherent Receiver Design, John Wiley & Sons, (1976).

BLUM67 Blum M., "A Machine-Independent Theory of the Complexity of Recursive Function," J. ACM 14 (1967), 322-336.

BREM67 Bremermann H.J., " Quantitative Aspects of Goal-Seeking, Self-Organizing Systems," in: F.M. Snell(ed.), Progress in Theoretical Biology, 1, 59-77, Acad. Pres.(1967).

BREN70 Brender R.F.," A Programming System for the Simulation of Cellular Spaces," Tech. Rep. No.25, Concomp., Univ. Mich. (1970).

BRIL64 Brillouin L., Scientific Uncertainty, and Information, Acad. Press (1964).

BURK70 Burks A.W.(ed.), Essays on Cellular Automata, Univ. Illinois Press, (1970).

BURK71 Burks A.W., "On Backwards-Deterministic, Erasable, and Garden-of-Eden Automata," Tech. Rep. No.012520-4-T, Comp. Comm. Sci. Dept., Univ. Mich., (1971).

CHAI66 Chaitin G.J., "On the Length of Programs for Computing Finite Binary Sequences," J. ACM 13 (1966), 547-569.

CHUR36 Church A., "An Unsolvable Problem of Elementary Number Theory," Amer. J. Math., 58 (1936), 345-363.

CHUR36a Church A., Rosser J.B., "Some Properties of Conversion," Trans. AMS 39 (1936), 472-482.

CODD68 Codd E.F., Cellular Automata, Acad. Press, (1968).

DIGR75 Di.Gregorio S., Trautteur G., "On Reversibility in Cellular Automata," J. Comput. System Sci. 11 (1975), 382-391.

FRAN71 Frantz D.R.,Brender R.F., "The CESSL Programming Language," Tech. Rep. No.012520-5-T, Comp. Comm.Sci. Dept., Univ. Mich., (1971).

FURT70 Furth R.H., Fundamental Principles of Modern Theoretical Physics, Pergamon Press, (1970).

GAJS75 Gajski D., Yamada H.M., "A Busy Beaver Problem in Cellular Automata," Proc. Symp. Uniformly Structured Automata and Logic, 171-183, IEEE (1975).

GARD70 Gardner M., "The Fantastic Combinations of John Conway's New Solitaire Game 'Life'," Scientific American 223:4 (1970) 120-123.

GARD71 Gardner M., "Mathematical Games," Scientific American 224:3 (1971), 106-109.

GOED34 Godel K., "On Undecidable Propositions of Formal Mathematical Systems," mimeographed lecture notes, Institute for Advanced Study, Princeton, N.J., (1934).

GOLZ76 Golze U., "Some New Differences Between 1-and 2-Dimensional Cellular Spaces," in: A. Lindenmayer and G. Rozenberg (ed.), Automata, Languages, Development, 369-384, North-Holland (1976).

GOLZ77 Golze U., "(A-) Synchronous (Non-)Deterministic Cell Spaces Simulating Each Other," Tech. Rep. 60, Inst. Math., Tech. Univ. Hannover, (1977).

GOOD70 Goodwyn L.W., "A Characterization of Symbolic Cascades in Terms of Expansiveness and Topological Entropy," Math. System Theory, 4 (1970), 157-159.

GOTT55 Gottschalk W.H., Hedlund G.A., "Topological Dynamics," Amer. Math. Soc., Providence, R.I., (1955).

HALM56 Halmos P.R., Lectures on Ergodic Theory, Chelsea Publ. Co., New York, (1956).

HARA72 Harad M., Noguchi S., "Algebraic Properties of the Iterative Automaton," Trans. Inst. Electr. Comm. Eng. Jap. 55D (1972), 767-774 (in Jap.). Eng. Transl. in Systems-Computers-Controls (Scripta Nipponica III) 3:6 (1972), 38-46.

HEDL69 Hedlund G.A., " Endomorphisms and Automorphisms of the Shift Dynamical System," Math. System Theory, 3 (1969), 320-375.

HERM74 Herman G.T., Liu W., Rowland S., Walker A., "Sychronization of Growing Cellular Arrays," Inform. Control, 25 (1974), 103-122).

HOBS71 Hobson A., Concepts in Statistical Mechanics, Gordon and Beach Sci. Publ., New York, (1971).

HOLL59 Holland J.H., "A Universal Computer Capable of Executing An Arbitrary Number of Subprograms Simultaneously," Proceedings of the East Joint Computer Conference (1959), 108-112 (Also in [BURK70, 264-276]).

HOLL60 Holland J.H., "Iterative Circuit Computers," Proceedings West Joint Comput. Conf. (1960), 259-265 (Also in [BURK70, 277-296]).

HOLL76 Holland J.H., "Studies of the Spontaneous Emergence of Self-Replicating Systems Using Cellular Automata and Formal Grammars," in: A. Lindenmayer and G. Rozenberg (ed.), Automata, Languages, and Development, North-Holland Publ., (1976).

JACO63 Jacobs K., Lecture Notes on Ergodic Theory, Matematisk Inst., Aarhus Univ., (1963).

JUST71 Justin J., "Groups and Semigroups Having Linear Growth," Comptes Rendus de l'Academie des Sciences, Paris, 273A (1971).

KELL74 Keller R.M., "Toward A Theory of Universal Speed-Independent Modules," Trans.IEEE, C-23 (1974), 21-33.

KELL75 Keller R.M., "A Fundamental Theorem of Asynchronous Parallel Computation," in: T.Y.Feng (ed.) Parallel Processing, 102-112, Springer Verlag, Berlin, (1975).

KEYN69 Keynes A.B., Robertson J.B., "Generators for Topological Entropy and Expansiveness," Math. System Theory, 3 (1969), 51-59.

KLEE67 Kleene S.C., Mathematical Logic, John Wiley & Sons, Inc. (1967).

KOBA75 Kobayashi K., "Solutions of the Two-Dimensional Firing Squad Synchronization Problem Having Small Firing Time," Proc. Symp. Uniformly Structured Automata and Logic, 163-170, IEEE (1975).

KOLM65 Kolmogorov A.N., "Three Approaches to the Quantitative Definition of Information," Problemy Peredachi Informatsii 1 (1965), 3-11, (in Russian; Eng. Transl. in Int. J. Comput. Math., 2 (1968), 157-168.

KURO55 Kurosh A.G., Theory of Groups, Chelsea Publ.Co., New York, (1955).

LABU74 Labudde R.A., Greenspan D., "Discrete Mechanics--A General Treatment," J. Comput. Physics, 15 (1974), 134-167.

LAIN75 Laing R.A, "Artificial Molecular Machines: A Rapprochement between Kinematic and Tessellation Automata," Proc. Symp. Uniformly Structured Automata and Logic, 73-80, IEEE (1975).

LAND60 Landau L.D., Lipshitz E.M., Mechanics, Pergamon Press, (1960).

LAWL71 Lawler E.L.," The Complexity of Combinatorial Computations: A Survey," in: J. Fox (ed.), Computers and Automata, Polytechnic Inst. Brooklyn, NY, (1971), 305-311.

LIEN76 Lien Y.E., "A Note on Transition Systems," Inform. Sci.10 (1976), 347-362.

LIPS65 Lipschutz S., General Topology, Schaum's Outline Series, Mc Graw-Hill Co., (1965).

LIPT76 Lipton R.J., Miller R.E., Snyder L., "Synchronization and Computing Capabilities of Linear Asynchronous Structures," IBM Tech. Rep., No. RC-5857, (1976).

MACK63 Mackey G.W.,, Mathematical Foundations of Quantum Mechanics, W.A. Benjamin, Inc., (1963).

MARU76 Maruoka A., Kimura M., "Condition for Injectivity of Global Maps for Tessellation Automata," Inform, Contr., 32 (1976), 158-162, (Also in Proc. Symp. Uniformly Structured Automata and Logic, 40-42, IEEE, 1975).

MCNA64 McNaughton R., "Badly Timed Elements and Well Timed Nets," Tech. Rep. No. 65-02, Univ. Penn., Moore School Electrical Engin. (1964).

MEYE71 Meyer J.D., On the Limits of Linearity, Acad. Press, (1971).

MILN68 Milnor J., "Advanced Problem No. 5603," Amer. Math. Monthly, 75 (1968), 685-686.

MILN68 Milnor J., "Growth of Finitely Generated Solvable Groups," J. Differential Geometry, 2 (1968), 447-449.

MOOR62 Moore E.F., "Machine Models of Self-Reprocuction," Proc. Symp. Appl. Math. (Amer. Math. Soc.) 14 (1962), 17-33 (Also in [BURK70, 107-203].

MOOR64 Moore E.F., "The Firing Squad Synchronization Problem," in: E.F. Moore (ed.), Sequential Machines, Selected Papers, 213-214, Addison-Wesley, Reading, Mass., (1964).

MORO68 Morowitz H.J., Energy Flow in Biology--Biological Organization as a Problem in Thermal Physics, Acad. Press, (1968)

MOOR68 Moore F.R., Langdon G.C., "A Generalized Firing Squad Problem," Inform. Contr., 12 (1968), 212-220.

MOSE72 Moses J., "Toward A General Theory of Special Functions," Com. ACM, 15 (1972), 550-554.

MYLO71 Mylopoulos J.P., Pavlides T., "On the Topological Properties of Quantized Spaces. II. Connectivity and Order of Connectivity," J. ACM., 18 (1971), 247-254.

NGUY74 Nguyen H.B., Hamacher V.C., "Pattern Synchronization in Two-Dimensional Cellular Spaces," Inform. Contr., 26 (1974), 12-23.

OHAN76 Ohanian H.C., Gravitation and Spacetime, W.W. Norton, (1976)

PATT71 Patt Y.N., "Injections of Neighborhood Size Three and Four on the Set of Configurations from the Infinite One-Dimensional Tessellation Automata of Two-State Cells" (Unpublished Report cited in [AMOR72]), ECON-N1-P-1, Ft. Monmouth, NJ 07703, (1971).

PILG72 Pilgrim P.C., "A Cellular Space Approach to Physics," in: B.P. Zeigler (ed.), "Cellular Space Models for Particle Physics, Biological Development and Highway Traffic Control: Some Initial Explorations," 1-14, Tech. Rep. No. 133, Logic of Computers Group, Comp. Comm. Sci. Dept., Univ. Michigan.

POIN02 Poincaré H., Science and Hypothesis, Flammarion, Paris (1902), (in French).

POIN80 Poincaré H., "On the Curves Defined by a Differential Equation," Comptes Rendus de l'Academie des Sciences, Paris 90 (1880), 673-675 (in French).

REIF65 Reif F., Fundamentals of Statistical and Thermal Physics, McGraw-Hill Co., (1965).

REYC74 Rey C.A., Vaucher J., "Self-Synchronized Asynchronous Sequential Machines," Trans. IEEE C-23 (1974).

RICH72 Richardson D., "Tessellations with Local Transformations," J. Comput. Systems Sci., 6 (1972), 373-388.

ROBI77 Robinson A.L., "Critical Phenomena: Experiments Show Theory on Right Track," Science 196 (1977).

ROMA76 Romani E., "Cellular Automata Synchronization," Inform. Sci. 10 (1976), 299-318.

ROSE66 Rosenstiehl P., "The Existence of Finite Automata Capable of Synchronization Even Though Arbitrarily Connected in an Arbitrarily Large Network," Int. Computat. Centre. Bull., 5 (196 (1966), 245-261. (in French).

ROSE73 Rosen B.K., "Tree-Manipulating Systems and Church-Rosser Theorems," J. ACM, 20 (1973), 160-167.

ROSS76 Rosset S., "A Property of Groups of Nonexponential Growth," Proc. Amer. Math. Soc., 54 (1976), 24-26.

RYAN72 Ryan J.P., "The Shift and Commutativity," Math. System Theory, 6 (1972), 82-85.

SATO72 Sato T., Honda N., "Period-Preservability and Poisson Stability of Parallel Maps of Tessellation Automata," Proc. Symp. Uniformly Structured Automata and Logic, 53-61, IEEE (1975).

SCOT77 Scott D.S., "Logic and Programming Languages," Comm. ACM, 20 (1977), 634-640.

SEAR73 Sears M., "Topological Models for Generators," Math. System Theory, 7 (1973), 32-38.

SHIN74 Shinahr I., "Two- and Three-Dimensional Firing-Squad Synchronization Problems," Inform. Contr., 24.(1974), 163-180.

SIBI75 Sibirsky K.S., Introduction to Topological Dynamics, Noordhoff Intern. Publ., Leyden, The Netherlands, (1975).

SMIT68 Smith A.R.(III), "Simple Computation Universal Cellular Spaces and Self-Reproduction," IEEE Symp. Switch Automata Theory, 9 (1968), 269-277.

SMIT69 Smith A.R.(III), "Cellular Automata Theory," Tech. Rep. No. 2 (1969), Stanford Electr. Lab., Stanford Univ.

STUR69 Sturman J., "Asynchronous Operations of an Iteratively Structured General-Purpose Digital Computer," Trans. IEEE, 17 (1969), 10-17.

TOFF72 Toffoli T., "On the Large-Scale Implementation of Cellular Spaces by Means of Integrated-Circuit Arrays," Consiglio Nazionale delle Ricerche, IAC, Rome, Italy (1972).

TOFF75 Toffoli T., " A Technique for Constructing Nontrivial Injective Parallel Maps for Tessellation Spaces of an Arbitrary Number of Dimensions," (unpublished notes, 1975).

TOFF** Toffoli T., (personal notes).

ULAM52 Ulam S. "Random Processes and Transformations," Proc.Int. Congr. Math., 11 (1952), 264-275, AMS, Providence.

USAL75 International Symposium on Uniformly Structured Automata and Logic, Conference Proceedings, Catalog No. 75CH1052-OC, IEEE, New York, NY (1975).

VAID60 Vaidyanathaswamy R., Set Topology, Chelsea Publish. (1960).

VITE66 Viterbi A.J., Principles of Coherent Communication, McGraw-Hill (1966).

VONN66 von Neumann J., Theory of Self-Reproducing Automata (ed. and completed by A.W. Burks), Univ. Illinois Press (1966).

WAKS66 Waksman A., "An Optimum Solution to the Firing Squad Synchronization Problem," Inform. Contr., 9 (1966), 66-78.

WHIT73 White A.T., Graphs, Groups and Surfaces, North-Holland Publ. Co., (1973).

WILL70 Willard S., General Topology, Addison-Wesley (1970).

WILL75 Willson S.J., "On the Ergodic Theory of Cellular Automata," Math. System Theory, 9 (1975), 132-141.

WOLF68 Wolf J.A., "Growth of Finitely Generated Solvable Groups and Curvature of Riemannian Manifolds," J. Differential Geometry, 2 (1968), 421-446.

YAMA69 Yamada H., Amoroso S., "Tessellation Automata," Inform. Contr., 14 (1969), 299-317.

YAMA70 Yamada H.,Amoroso S., "Completeness Problem for Pattern Generation in Tessellation Automata," J. Comput. Systems Sci. 4 (1970), 137-176.

YAMA71 Yamada H., Amoroso S., "Structural and Behavioral Equivalences of Tessellation Automata," Inform. Contr., 18 (1971), 1-31.

ZASS49 Zassenhaus H., The Theory of Groups, Chelsea Publ. (1949).

ZUSE67 Zuse K., "Computing Spaces," Elektronische Datenverarbeitung, 8 (1967), 336-344, (in German).