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

ON BACKWARDS-DETERMINISTIC, ERASABLE, AND GARDEN-OF-EDEN AUTOMATA

Arthur W. Burks

Technical Report No. 012520-4-T
September 1971

with assistance from:

Department of Health, Education, and Welfare
National Institutes of Health
Grant No. GM-12236
Bethesda, Maryland

and

National Science Foundation
Grant No. GJ-29989X
Washington, D.C.

ensm

UMR0869

On Backwards-deterministic, Erasable, and Garden-of-Eden Automata

Arthur W. Burks

On Backwards-deterministic, Erasable, and Garden-of-Eden Automata

1. Introduction

Moore [3] and Myhill [4] have shown some interesting relations between a certain type of erasability in a cellular automaton and the existence of Garden-of-Eden configurations. Their concepts of erasability and Garden-of-Eden configuration are local concepts, but there are also corresponding global or system concepts to which they are closely related, namely backwards-*in*determinism and the existence of Garden-of-Eden system states. Moreover, these global concepts apply to finite automata as well as to (infinite) cellular automata.

In the present chapter we will define various local and global concepts of these sorts and explore their interrelations somewhat. We will then make some conjectures about their relations to universal computability and universal constructibility in cellular spaces.

2. Finite automata

Our building blocks for both finite and infinite systems will be *finite automata* whose outputs are switching functions (projections) of the internal states (delay output states) and for which any internal state may be an initial state (the state at $t = 0$). A finite automaton may be defined as a state graph. Alternatively, it may be defined as a logical net such that every path from input to output goes through a delay, and with some designated set of allowable initial states; the set of internal states then consists of the given initial states and all states which can be reached from them.

A *finite system of automata* is any system of finite automata composed by means of the operations (a) juxtaposition (b) identification of input wires (c) series connections, in which an output wire drives an input wire, provided the connection introduces no cycle, and (d) cyclic connections. Cycles introduced by (d) can never be vicious, because all such cycles must pass through delays. We will call a finite system built without (d) a *finite series-parallel system.* It is clear that any finite system of finite automata is a finite automaton.

The following symbolism will be used. Let S, I, and Θ be the sets of internal (delay output), input, and output states respectively; lower case letters will be used for the states themselves. [s](0,t) will mean a sequence of internal states from t = 0 to t, [s](t') will mean an internal state at t + 1, and similarly for [i](t), [θ](0,t'), etc. A finite automaton is governed by a transition function [T: I x S → S) giving the next internal state as a function of the present internal and input states and an output function (L: S → Θ) giving the output state as a projection of the internal state. Clearly [s](0) and [i](0,t) determine [s](0,t') and hence [θ](0,t').

Finite automata as just defined are deterministic, the present i ε I and s ε S determining the *next* s' ε S and θ ε Θ. This is forwards-going determinism. Many deterministic systems of nature are also backwards-deterministic, for example, Newtonian machanics. This motivates the following definition.

A finite automaton is *backwards-deterministic* (BD) if and only if for each i ε I and s' ε S there is at most one s ε S such that T(i, s) = s'. Alternatively, an automaton is BD if and only if for each [i](0,t) and [s](t') there is at most one sequence [s](0,t) satisfying iterations of the transition function T.

Burks and Wang, [2] p. 286.

Since the output state θ is a projection of the internal state s, if an automaton is BD each [i](0,t) and [s](t') will determine a unique [θ](0,t'), and in the input-free case there will be a unique [θ](0,t'). A finite automaton is said to be backwards-indeterministic ($\overline{\text{BD}}$) if it is not backwards deterministic.

The simple examples of Figure 1 illustrate these concepts. The question mark in a delay indicates that the initial state (0 or 1) is unspecified. The cyclic binary counter of Figure 1a is BD, since its initial state is the same as its state at t just in case the number of ones received prior to t is even. The automaton of Figure 1b is $\overline{\text{BD}}$ since both $i = 1$, $s = 0$ and $i = 1$, $s = 1$ at t produce $s = 1$ at t'. The juxtapose of the BD Figure 1a and the $\overline{\text{BD}}$ Figure 1b is $\overline{\text{BD}}$.

We next establish a theorem concerning finite systems of automata. *Theorem I:* A finite series-parallel system of BD automata is BD. *Proof:* We show that a compound of BD automata by any of the rules (a) juxtaposition, (b) identification of inputs, or (c) series connection (without cycles) produces a BD automaton. This is obvious for the first two cases, so we prove it only for the third. See Figure 2a.

Since A_1 is BD,

(1) $[i_1](0,t)$, $[s_1](t')$ determines $[s_1](0,t')$.

Since A_2 is BD, in both the connected and unconnected net

(2) $[i_2,i_2^*](0,t)$, $[s_2](t')$ determines $[s_2](0,t')$.

In the connected net,

(3) $[i_1](0,t)$, $[s_1](t')$ determines $[i_2^*](0,t')$.

Hence clearly

(4) $[i_1,i_2](0,t)$, $[s_1](t')$ determines $[i_2,i_2^*](0,t')$.

Combining this with (2) gives

(5) $[i_1,i_2](0,t)$, $[s_1,s_2](t')$ determines $[s_2](0,t')$.

Combining this with (1) gives

(6) $[i_1,i_2](0,t)$, $[s_1,s_2](t')$ determines $[s_1,s_2](o,t')$.

Q.E.D.

Theorem I does not hold for finite systems of automata generally, that is, it does not hold if cycles are allowed in composition. This is shown in Figure 2b. The automata A and A' each operate as follows. Each keeps its internal state for all inputs except 11, which causes the internal state to reverse (change from 0 to 1 or 1 to 0). A and A' are clearly BD. But the cyclic compound of A and A' is $\overline{\text{BD}}$, for the input 11 to Figure 2b produces the state 00 whether the internal state is 00 or 11.

There are finite series-parallel and finite cyclic systems of $\overline{\text{BD}}$ automata which are themselves BD. A cyclic example is shown in Figure 3. The shift register of Figure 3a is $\overline{\text{BD}}$, for with both the data and shift inputs active (state one) at t, the internal (delay output) state at t' will be one regardless of the internal state at zero. Yet the cyclic shift register of Figure 3b is BD, as is clear from its state transition diagram. Note that the result of juxtaposing two copies of Figure 3a is $\overline{\text{BD}}$, but that when these copies are interconnected as in Figure 3b certain internal state histories previously possible (e.g., 00, 01, 11, 10, 00) are no longer possible.

We next make some observations about closed (input-free) finite automata. The state graph of a closed finite automaton consists either of "pure cycles" (in which a finite state sequence is indefinitely repeated) or "prefixed cycles" (in which one finite state sequence is followed by another which is then repeated indefinitely), or both. See Figure 4. An open finite automaton can be closed by means of a switch whereby a given input state i is repeated indefinitely (iii...).

Theorem II: The state graph of a closed finite automaton consists of pure cycles if and only if that automaton is BD. A finite automaton is BD if and only if each of its closures with a constant input is a pure cycle. If a finite automaton is BD, every internal state belongs to a cycle.

Proof: The first part is obvious, and the last part is a simple corollary of the second part, so it will suffice to prove the second part. In the forward direction, note that if an automaton is BD, its closure with a constant input is also BD. In the backwards direction, assume that automaton A has three internal states s_1, s_2, s such that

$T(i,s_1) = s$

$T(i,s_2) = s.$

The closure of A with the constant input iii... will have the transition s_1,s and s_2,s and hence will have a prefixed cycle. Q.E.D.

It is the theme of this paper that BD automata are less powerful and less interesting than $\overline{\text{BD}}$ automata. This claim is substantiated in the finite case by the following result: No BD finite automaton is behaviorally equivalent to Figure 1b. Consider the closure of Figure 1b with the constant input 111...; this net produces the output 0111... . By Theorem II the constant input closure of a BD net is a pure cycle. But 0111... can't be the projection of any pure cycle, for the internal state that produces output zero will be repeated indefinitely.

Roughly speaking, a BD automaton can never "forget" its state, for the internal state at t can be recovered from the input history and the present internal state. This property of BD-ism is negatively correlated with the following property. An *internal state is a Garden-of-Eden (GOE)* state if and only if it can occur only at time zero. A *finite automaton is GOE* if and only if it has a GOE state. The concept of a GOE state is a global concept, to be contrasted with the local concept of a GOE configuration introduced in connection with cellular automata below.

The next theorem is a corollary of Theorem II. *Corollary III:* If a finite automaton is BD, then it is $\overline{\text{GOE}}$. A *closed* finite automaton is BD if and only if it is $\overline{\text{GOE}}$.

3. Uniform cellular automata

We consider next finite and infinite arrays of finite automata which are regular in the following respects. Each array is a regular geometric arrangement of copies of the same finite automaton. Without any real loss of generality we can assume these arrays to be arranged in discrete Cartesian n-dimensional space, each point being viewed as a cell. There is defined a neighborhood relation N(c) which gives in sequence the m neighbors that are directly connected to a cell c. Each automaton has m times as many inputs as outputs so that in an infinite cellular automaton there will be no free inputs or outputs. It is customary to view a cell as belonging to its own neighborhood. Hence the local transition function maps from the state of N(c) at t to the state of c at t'. Every infinite, closed system and every finite system satisfying these conditions is a *cellular automaton.*

A finite cellular automaton is a finite automaton, so the concepts of internal, input, and output states, transition and output functions, backwards-determinism (BD), and Garden-of-Eden (GOE) states and automata apply to it without further ado. An infinite cellular automaton is, in an appropriate sense, closed, and does not have any outputs, so all these concepts except those involving inputs and outputs can be made to apply to it with an appropriate definition of internal state. The *internal state* of an infinite cellular automaton is an array of its cell states which includes a representation of the origin and the coordinate system. Note that the result of shifting (translating) or rotating the internal state of an infinite cellular automaton in space is generally a different state. In many situations we will want to distinguish an infinite cellular automaton *system* and its states from its *cells* and their states and from its *finite areas* (subsystems) of cells and their states.

Closed Turing machines and von Neumann's cellular automata[5] are special cases of infinite cellular automata. In both cases there is a particular stable cell state called the "blank" state, and internal states with infinitely many non-blank cell states are prohibited. The notion of an algorithm is usually stated in terms of a blank state, since an algorithm must be finitely expressible.

von Neumann's notion of construction in a cellular automaton further presupposes a set of stable cell states which includes the blank state, construction being defined with respect to these stable states rather than with respect to the set of all cell states. Indeed, if one of our conjectures is true (see Sec. 6 below), universal construction is not possible in an infinite cellular automaton if it is defined with respect to the set of all cell states.

We proceed now to define these notions more precisely. Let $S\{N(c), t\}$, $S\{c, t'\}$, etc. describe the state of the region indicated at the time indicated. A set σ of cell states is *stable* (quiescent) when: if every cell state of $S\{N(c), t\}$ belongs to σ, then $S\{c\ t\} = S\{c, t'\}$. A cell state is stable if its unit set is stable.

A cellular automaton with no set of stable states will be called an *unstable cellular automaton.* Note that an unstable cellular automaton may have stable configurations. An example is afforded by any two-way infinite linear cellular automaton whose cell transition function satisfies these conditions:

$T(00,0) = 1$	$T(11,1) = 0$
$T(00,1) = 1$	$T(11,0) = 0.$

The sets $\{0\}$, $\{1\}$ and hence $\{0,1\}$, are all unstable, but the infinite internal state ...01010101... is stable (is self-repeating). I am doubtful that unstable cellular automata are of much interest, so we will consider mainly (but not exclusively) stable cellular automata.

A *stable cellular automaton* has, by definition, at least one stable state. If there is only one, it is called the *blank state*. If there is more than one, we will assume that one has been designated as the blank state. In all our examples, zero (0) will be used for the blank state.

Note that the blank state plays the role of free space in traditional physics: nothing happens there. On the continuous creation theory things happen in free space, but this requires a probabilistic model anyhow. Since computability and constructability are defined with respect to sets of stable states, these notions presuppose a universe which is stable in certain fundamental respects.

The internal (system) states of an infinite, stable cellular automaton are of two kinds. A state is *algorithmic* or *non-algorithmic* according to whether finitely many cells are non-blank. Note that the successor of an algorithmic state is always algorithmic, but that a non-algorithmic state may be succeeded by either a non-algorithmic state or an algorithmic state; for an example of the latter see Example $4\bar{a}$ of the Appendix A. An infinite stable cellular automaton has two variants: an *algorithmic* variant, in which the initial internal state (and hence all states) is algorithmic; and a *non-algorithmic* variant, in which any state may be an initial internal state.

In von Neumann's cellular automaton system the set of ten states $\{\underset{\sim}{U}, C_{00}, \rightarrow, \uparrow, \leftarrow, \downarrow, \Rightarrow, \Uparrow, \Leftarrow, \Downarrow\}$ is a stable set which plays an essential role in universal construction. It is, moreover, the largest stable set of the system. von Neumann called these states "quiescent" because of his biological point of view; from the point of view of physical systems, they are "stable." The undecidable state $\underset{\sim}{U}$ is designated as the blank state of a von Neumann cellular system. Actually von Neumann's transition function is defined so that, intuitively speaking, $\underset{\sim}{U}$ is the most stable of all the

29 states, but it is difficult to define this kind of stability formally. This system is $\overline{\text{BD}}$, and every finite subsystem of it is $\overline{\text{BD}}$. The system is GOE.

The set of tape characters of a Turing machine is a stable set, with zero or a blank usually designated on the blank state. All input-free Turing machines can be embedded in von Neumann cellular automata. Moreover, a Turing machine with a single tape (and no inputs) can be *defined* as a one-dimensional von Neumann automaton. Imagine that the finite control automaton moves along the tape, instead of vice-versa. Let T be the tape alphabet and F the set of finite automaton states. Let Λ mean that the finite automaton is not at a tape square. The set of cell states is then $T \times (F \cup \{\Lambda\})$ constitutes a stable set. Note that exactly one cell is non-stable at any moment of time; this is the cell consisting of the tape square under scan and the finite automaton. Each different Turing machine has a different transition function.

4. Local concepts of erasability, Garden-of-Eden, and constructibility

Backwards-*in*determinism ($\overline{BD}$) consists in two internal states having the same successor. (If the automaton has inputs this must be so for some fixed input state.) When two states merge into one, certain information is lost, so backwards-*in*determinism is a kind of "erasability." We will next introduce two other closely related but distinct senses of erasability.

Backwards-*in*determinism is local as well as global, applying to every finite area of an infinite cellular automaton, including individual cells, as well as to the whole automaton. The second kind of erasability is a local variant of backwards-*in*determinism, in which two configurations agreeing on their borders have the same successor. This is Moore's concept of erasability.

Moore[3] applied this only to algorithmic cellular automata, but Myhill [4] applied it to all infinite cellular automata. Both defined this concept for the particular neighborhood relation in which a cell touches each of its neighbors at at least a point, whereas we define it for any (finite) neighborhood relation.

We will call it "configuration-erasability," to distinguish it from our third kind of erasability, to be called "cell-erasability."

A finite *area* is any finite, designated set of cells. Two areas are *similar* if one can be converted into the other by a shift in each dimension. A *configuration* is a state of an area, in the sense of an assignment of a cell state to each cell of the area, the same assignment for similar areas being called the same configuration, except that when we count the number of subconfigurations included in a large configuration we do not count overlapping subconfigurations. Since all cells have the same transition function, what happens in any configuration happens in any similar configuration, i.e., any configuration obtained by shifting each cell a specified amount in each dimension. If s is an internal state and A a (finite) area, $\{s|A\}$ will be the configuration of A in state s. This notation applies to finite as well as infinite cellular automata, but in the former case we will assume that A is such that the neighborhood of the neighborhood of A, i.e., $N^2(A)$, exists.

Let s_1 and s_2 be two internal states of a cellular automaton and A a finite area. The pair of configurations $\{s_1|A\}$ and $\{s_2|A\}$ *erases* if and only if the following three conditions are satisfied:

(1) These configurations differ, i.e., $\{s_1|A\} \neq \{s_2|A\}$

(2) They agree on $N^2(A) - A$, i.e., $\{s_1|N^2(A) - A\} = \{s_2|N^2(A) - A\}$

(3) Their successors agree on N(A), i.e., $\{T(s_1)|N(A)\} = \{T(s_2)|N(A)\}$.

A pair of internal states s_1 and s_2 *erases* if and only if there is an area A such that the pair of configurations $\{s_1|A\}$ and $\{s_2|A\}$ erase. A cellular automaton is *configuration-erasable* if and only if it has a pair of internal states which erase.

Let $\overline{A}$ represent all of a cellular automaton other than the cells in A. Myhill defines a concept of "indistinguishability" in terms of all environments, i.e., in terms of all states of $\overline{A}$. His concept can be proved equivalent to Moore's by taking account of the delay in each cell.

The following theorem helps explain the concept of configuration-erasability.

Theorem IV: If an automaton is configuration-erasable then it has an area (subsystem) which is $\overline{BD}$. If an automaton is configuration-erasable, then it is $\overline{BD}$.

Proof: The $\overline{BD}$ area referred to in the first part is simply $N^2(A)$. Note that the converse of the first part reads: if every finite area is BD, the automaton is *not* configuration-erasable. The proof of the second part involves an interesting and useful heuristic principle, which we will call the "insulation principle." The proof proceeds by converting a confluence or merging of two local (area) states into a confluence or merging of two internal (system) states, taking advantage of the fact that the border $N^2(A) - A$ acts as an insulator, preventing anything in the area $\overline{N^2(A)}$ from influencing anything in the area A in one time step. Specifically, we are given the confluence

$$\left.\begin{array}{ll} \{s_1 | N^2(A)\} & \text{into} \\ \{s_2 | N^2(A)\} & \text{into} \end{array}\right\} \quad \{T(s_1) | N(A)\} \qquad [= \{T(s_2) | N(A)\}]$$

where $\{s_1 | N^2(A) - A\} = \{s_2 | N^2(A) - A\}$. From this we generate the internal state confluence

$$\left.\begin{array}{ll} s_1' & \text{into} \\ s_2' & \text{into} \end{array}\right\} \quad T(s_1') \qquad [= T(s_2')]$$

by the following definitions

$$s_1' = s_1$$
$$s_2' = \{s_2 | N^2(A)\} \text{ and } \{s_1 | \overline{N^2(A)}\}.$$

Since s_1 and s_2 agree on the border $N^2(A) - A$, $T(s_1') = T(s_2')$. QED

Note that the converse of the last part of Theorem IV fails, i.e., there are automata which are $\overline{\text{BD}}$ but *not* configuration-erasable. A finite shift register is an example; see the discussion under Example 5 of the Appendix. Indeed, this fact motivates the definition of configuration-erasability, as the example makes clear.

We proceed now to discuss the third kind of erasability, cell-erasability. Hao Wang [6] showed that there is a universal Turing machine in which erasure is not allowed on the tape. Specifically, on a two-state tape-square (cell) system, once a tape-square is changed from 0 to 1 it can never be changed again. The general technique of his proof was to use alternate squares for markers. When the machine needed to change a message on the tape, it copied that message into a blank region, putting a one in each marker square to show that the old message was "dead."

Wang placed no restrictions on the finite control automaton for this universal machine, which is probably $\overline{\text{BD}}$. It is of interest to generalize the notion of cell-erasability and consider whether a cellular automaton lacking this property could be computation-universal. See the conjectures in Section 6 below.

Consider stable automata only. Let 0 be the blank state, and consider the cell transition function. An automaton is *cell-erasable* if and only if for every ordering of the cell states with 0 as the initial element the state of a cell may both increase and decrease. Thus a blank tape that can only be marked, never erased, is not cell-erasable *as far as the tape alone is concerned*.

Some examples will clarify the concept. An automaton with two states (0,1) per cell in which 1 is not allowed to change is *not* cell-erasable. The one-dimensional infinite automaton in which a cell becomes 1 if either immediate neighbor becomes 1 is not cell-erasable and is $\overline{\text{BD}}$, since 0010100 and 0011100 both produce 0111110. Example 2a of the Appendix is not cell-erasable but is BD. Is there a non-cell-erasable BD automaton with a symmetrical transition function? It is obvious that a BD cell is erasable.

There are two further local concepts to be defined. A configuration is *locally GOE* if it occurs in an internal state only at time zero. An automaton is locally GOE if it has a GOE configuration. These concepts are the local versions of our concepts of a GOE internal state and a GOE automaton.

To define the negatively related notion of a constructible configuration we consider stable automata only. Consider a particular cellular automaton *A* and some configuration C of area A. Configuration C is *constructible in* A if there is an internal state s and a time t such that

$C = \{T^t(s) | A\}$ and $t > 0$,

where $T^1(s) = T(s)$ and $T^{n+1}(s) = T\{T^n(s)\}$.

As defined this concept applies to finite and non-algorithmic automata, but it is of interest chiefly with respect to algorithmic automata. A GOE configuration is clearly non-constructible, but not conversely. We will argue later that the previous definition of constructibility constitutes only a minimum condition (Sec. 6).

5. Some relations among these global and local concepts

When introducing the local concepts of BD, GOE, configuration-erasability, cell-erasability, and constructibility we mentioned a few relations. These concepts, together with the global concepts of BD and GOE, constitute a set of which we can ask in general: Which concepts are connected, and for what types of cellular automata, unstable, algorithmic, and non-algorithmic? We will answer part of this question in the present section.

The state graph of a BD closed automaton is composed of subgraphs of the three types shown in Figure 5: pure cycle, one-way infinite, and two-way infinite. Of course these subgraphs also occur in $\overline{\text{BD}}$ closed automata. All three do in fact occur in BD algorithmic automata. Since the all blank state is self-repeating, a unit cycle occurs in every BD algorithmic automaton.

A one-dimensional automaton which merely shifts left (or right) is BD, though its cells are not. See Figure 3 and Example 5 of the Appendix. It is $\overline{\text{GOE}}$. If it is algorithmic, its state graph consists of a single unit cycle and many two-way infinite subgraphs. If it is non-algorithmic it also contains pure cycles of every integral length. Is there a symmetrical example of this? See Example 3 of the Appendix.

Example 2 of the Appendix is a BD algorithmic automaton whose state graph consists of a unit cycle and one-way infinite subgraphs. It is GOE. Its non-algorithmic extension is $\overline{\text{BD}}$. This suggests the question: is there a non-algorithmic BD automaton with one-way infinite subgraphs?

More generally, one can investigate the conditions under which the three subgraphs of Figure 5 occur in BD automata, treating cycles of different length as different, and raising the question separately for algorithmic and non-algorithmic automata, and for symmetrical and non-symmetrical automata.

The infinite shift register of Example 5 shows that global BD-ism does not entail local BD-ism (i.e., that every finite area is BD). What of the converse: If every finite area is BD, or every sufficiently large finite area is BD, is the automaton BD? The following theorem provides a partial answer to this question.

Theorem V: If every finite area of an algorithmic automaton is BD, so is the automaton.

Proof: Assume a BD automaton *A* is algorithmic and use RAA. There are two system states s_1 and s_2 with the same direct descendant s_3. Since these states are algorithmic we can find a finite area A' of *A* which encompasses the transformations s_1 to s_3 and s_2 to s_3 in the following states. A' has the transitions s_1' to s_3' and s_2' to s_3' under blank inputs, where each s_i' agrees with s_i $(i = 1,2,3)$ on all non-blank cells. Then the area A' is $\overline{\text{BD}}$. QED

Does this theorem hold for non-algorithmic automata?

We note that the simple relations which hold between BD and GOE finite automata do not hold for cellular automata. A BD automaton may or may not be GOE. We will see later that von Neumann's system is $\overline{\text{BD}}$ and GOE. Is there a cellular automaton which is $\overline{\text{BD}}$ and $\overline{\text{GOE}}$?

There are a number of relations between global and local concepts which hold for algorithmic automata because these always have finite areas whose inputs and border cells are blank. Theorem V above is an example, as is the next theorem.

Theorem VI: For *algorithmic* automata (A) An automaton is $\overline{\text{BD}}$ if and only if it is configuration-erasable (B) An automaton is locally GOE if and only if it is GOE.

It is a problem to investigate how many of these relations hold for non-algorithmic automata. The following theorem only partly answers this. In proving part of it we use the insulation principle of Section 4.

Theorem VII: (A) If an automaton is locally GOE, it is GOE. (B) If an automaton is configuration-erasable, it is $\overline{\text{BD}}$. (C) An algorithmic automaton is configuration-erasable if and only if its non-algorithmic variant is.

Proof: For (A), note that if a configuration is GOE, every state including it is. The proof of (B) is: Let s_1 and s_2 be an erasable pair of system states and define

$$s_1' = s_1 \text{ on } A$$
$$= s_2 \text{ on } \bar{A}.$$

Then s_1' and s_2 are different on A but have the same successor, since the successors of s_1 and s_2 agree on N(A). (C) is proved by the insulation principle.

If a non-algorithmic automaton is GOE, then is it locally GOE? Example $4\bar{a}$ refutes: If a (non-algorithmic) cellular automaton is $\overline{BD}$ then it is configuration-erasable.

Moore and Myhill have proved some theorems for the particular neighborhood relation defined by: the neighborhood of a cell consists of itself and all cells having at least one point in common with it.

Moore's Theorem: If an automaton is configuration-erasable, then it is locally GOE.

Myhill's Theorem: If a non-algorithmic automaton is locally GOE, then it is configuration-erasable.

Amoroso and Cooper [1] have shown that there is an algorithmic automaton which is locally GOE but not configuration-erasable; Example 4a of the Appendix is an example.

I conjecture that the Moore and Myhill theorems hold for arbitrary neighborhood relations. I actually made this conjecture in June, 1962, and discussed it with John Holland and John Myhill. The intuitive reasoning underlying this conjecture is as follows. A key idea in the proofs of these two theorems is the following. Suppose there is an erasable pair of configurations of given area. Then in this area two configurations merge into one. But then 2^n of these configurations (where n is the dimensionality), placed in a larger area and protected from one another by insulating borders, will merge into one large configuration. For a sufficiently large area it can be shown that there is so much merging that at least one configuration has no predecessor and hence is a GOE configuration. As a corollary of this, one would expect that as N grows in size the smallest GOE configuration would tend to grow in size.

Another way of viewing the matter is in terms of how many different configurations can be produced in a given area, starting from a given configuration, by controlling the inputs on the border. There is a limited amount of information that can be sent across the border. Moreover, as the number of cells in a contiguous area increases, the number of input states increases more slowly than the number of possible configurations.

Since our cellular automata have the same transition function for every cell, one would expect further connections between local and global concepts. There are clearly many relations to be worked out.

6. Some problems concerning computability and constructibility in cellular automata

Both computability and constructibility are normally defined for stable automata. An algorithm must be finitely expressible, and this requirement is usually formulated by saying that only a finite number of squares of a tape may be non-blank. Construction takes place in "empty space," which in a cellular automaton is represented by a blank state.

In what follows we will sometimes refer to *the* stable set of states of an automaton. For any given cellular automaton there may be alternatives here, but we will require that the stable set always be maximal; that is, if any other cell state is added, the resultant set of states will be unstable. By definition the stable set includes the blank state.

We noted earlier that a Turing machine can be conceived as a one-dimensional algorithmic linear automaton with the restriction that at each moment only one cell is in an unstable state. We will call a one-dimensional linear algorithmic automaton operating under the restriction that only one cell is unstable at any moment a *one-dimensional cellular Turing machine*. Given an appropriate initial state, and with suitable conventions for representing problems and answers, it *realizes* a Turing machine of the usual kind. This leads to the following set of problems.

Problem set VIII: Can a universal Turing machine be realized by a one-dimensional cellular Turing machine which (1) is BD, (2) has a BD cell, (3) has a BD finite area, (4) is not cell-erasable, (5) is not GOE? Note that by Theorem IV, condition (5) is equivalent to: (6) is not locally GOE; also, condition (1) is equivalent to: (7) is not configuration-erasable.

Since a Turing machine may sometimes be *embedded* in a cellular automaton of two or more dimensions, similar problems arise for these. Here we establish the convention that a one-way infinite row of cells is designated as the *tape*, on which problem and answer are stored. Note that with this convention a non-algorithmic cellular automaton may not be computation universal, despite the fact that all computable answers may be stored in it initially. For it may lack the power to retrieve the correct answer for the stated problem or the power to move it to the tape.

Problem set IX: Can a universal Turing machine be embedded in an (a) algorithmic or (b) non-algorithmic cellular automaton which (1) is BD, (2) has a BD cell, (3) has a BD finite area, (4) is not cell-erasable, (5) is not GOE, (6) is not locally GOE, or (7) is not configuration-erasable. Conditions (5) and (6) are equivalent by Theorem IV and also conditions (1) and (7) for the algorithmic case. By Theorem V the answer to (7) is the same for both the algorithmic and the non-algorithmic cases.

My own conjecture is that the answers to most of the problems in both sets VIII and IX are negative. This opinion is based partly on the intuitive feeling that erasability in each of these senses is a necessary condition for universal computability, partly on the fact that in the finite case BD automata are more limited in power than $\overline{\text{BD}}$ automata, and partly in the constructibility considerations to be discussed next.

How should "constructible configuration" and "universal constructibility" be defined? The definition at the end of Section 4 is open to the objection that everything is constructible in a shift-right (or shift-left) cellular automaton. For this reason it should be regarded as only a minimal condition on constructibility.

What other conditions should be imposed? We will discuss these first in connection with constructibility per se, and next with respect to universal constructibility. With respect to the first we might require that several (or an arbitrary number) of copies be constructible, as Moore did. We might also require that the constructed configuration remain in place forever, once it is constructed. Another possibility is to require in a space of dimension two or greater that the constructor be linear even when the constructed configuration is not.

We turn next to reasonable requirements for universal constructibility. von Neumann had two. First, that there be a single universal constructing configuration which can construct any configuration composed of stable states, and second that there be a universal Turing machine which is a configuration of stable states. With these requirements, negative solutions to problem set IX would imply negative to solutions to the corresponding problems for universal construction. More than that, I'm inclined to believe that no constructions in any interesting or reasonable sense can take place in cellular automata which are non-erasable in any of the seven senses of problem set IX.

We comment on a definition of universal constructibility in terms of all finite configurations. If a cellular automaton has GOE configurations, it is not universal in this strong sense. If a cellular automaton is not locally GOE, then I conjecture that there is no interesting sense of constructible such that every configuration is constructible.

Actually, there is no more reason to expect complete constructibility of configurations from within a system, than to expect complete predictability, or definability of truth, within a system. It might seem that a GOE automaton has limitations, e.g., that it cannot construct the GOE state. But universality is defined with respect to computability and structure, rather than with respect to behavior and real-time considerations.

It should be noted that when universal constructibility is defined as requiring the construction of all configurations of stable cell states, with the obvious requirement that there be at least two, then an automaton which merely shifts to the right is not universal. This would also be so for the following automaton: It behaves as von Neumann's does for even moments of time, and at odd moments of time the contents of every cell is shifted to the right.

Since self-reproduction follows on universal constructibility, we also conjecture that no interesting form of self-reproduction can take place in a cellular system which is non-erasable in any of the seven senses of problem set IX.

If my conjectures about the weakness of non-erasable cellular systems are true, they seem to mark a contrast between discrete and continuous deterministic systems. The classical example of determinism is Newtonian mechanics, which is BD. I doubt that Newtonian mechanics can be simulated in a BD cellular system, in any strong sense of simulation. I don't know of any $\overline{\text{BD}}$ closed deterministic systems in nature which are of interest.

Arthur W. Burks
8/12/70

References

[1] Amoroso, S. and G. Cooper. "The Garden-of-Eden Theorem for Finite Configurations" *Proceedings of the American Mathematical Society 44* (1970) 189-197.

[2] Burks, A.W. and H. Wang. "The Logic of Automata" *J. Association for Computing Machinery 4* (1957) 193-218, 279-297.

[3] Moore, Edward F. "Machine Models of Self-Reproduction", *Proceedings of Symposia in Applied Mathematics 14*, Providence, Rhode Island, American Mathematical Society, 1962. 17-33. Also pp. 187-203 of *Essays on Cellular Automata* (edited by A.W. Burks), University of Illinois Press, Urbana, 1970.

[4] Myhill, John. "The Converse of Moore's Garden-of-Eden Theorem" *Proceedings of the American Mathematical Society 14* (August, 1963) 685-686. Also pp. 204-205 of *Essays on Cellular Automata* (edited by A.W. Burks), University of Illinois Press, Urbana, 1970.

[5] von Neumann, J. *Theory of Self-Reproducing Automata* (edited and completed by A.W. Burks), University of Illinois Press, Urbana, 1966.

[6] Wang, Hao, "A Variant to Turing's Theory of Computing Machines" *J. Association for Computing Machinery 4,1.* (January, 1957), 63-92.

Appendix

Examples of one-dimensional infinite cellular automata

Example 1a

Algorithmic automaton with 0 as a blank state defined by

t	000	010	001	011	100	110	101	111
t'	0	1	1	0	1	0	1	0

The cell transition function is symmetrical and BD. Two adjacent cells are not BD, since the following transitions occur

```
0000      0110
 00        00
```

An enumeration of cases shows that every pattern consisting of ...000 followed by a finite number of repetitions of 10 followed by 000... (e.g., ...0001000.., ...00010101000...) is GOE. Moreover, since both 001 and 100 produce 1, any pattern with 1's will produce a pattern with 1's further out. Hence the state graph consists of a one unit cycle (the all blank state is self-perpetuating) and denumerably many one-way infinite graphs (of the form $s_1 s_2 s_3 s_4 \ldots$). Hence this automaton is BD.

Example $1\bar{a}$

This is the non-algorithmic version of example 1a. The three non-algorithmic states ...1110000..., ...0000111..., and ...1110111... produce ...0001000..., so this automaton is $\overline{\text{BD}}$.

An enumeration of cases shows that any pattern containing 10101 is GOE, hence this automaton is GOE also.

Example 2a

Consider the algorithmic automaton defined by the cell transition function

t	000	010	001	011	100	110	101	111
t'	0	1	1	1	0	1	0	1

This cell transition function is $\overline{BD}$, but it causes 1's to "grow" to the left to within one zero of a 1 on the left. The automaton is therefore BD, with two kinds of subgraphs: a unit cycle (the all blank state is stable) and one-way infinite subgraphs. It is therefore GOE.

Example $2\bar{a}$

The non-algorithmic variant of example 2a is $\overline{BD}$ but it is GOE.

Example 3

The following, partial, cell-transition-function is both symmetrical and BD, and can be extended to a complete cell-transition-function which is symmetrical and BD.

t	001	100	021	120	012	210	002	200
t'	1	1	0	0	2	2	0	0

t	000	020	121	212	222	122	221
t'	0	2	2	1	2	0	0

Any automaton with this partial transition function has a two-way infinite subgraph, for the sequence 0210 travels to the right, 0120 to the left, and when they meet at 2 they are reflected and travel out, thus

```
02100200120
00210201200
00021212000
00001210000
00012221000
00120202100
01200200210
```

Is the automaton so-far defined BD? What of its extensions to complete transition functions?

Example 4a

This is the example Amoroso and Cooper use to show that the converse of Myhill's theorem fails. I.e., the algorithmic version of this automaton has GOE configurations but is not configuration-erasable.

The cellular transition function is $\overline{BD}$ and is asymmetrical.

t	000	010	001	011	100	110	101	111
t'	0	1	1	0	1	1	0	0

For algorithmic internal states, the global transition function is 1-1 and properly into, so there are GOE internal states, and hence GOE configurations. Consequently this automaton is BD, with a unit cycle and one-way infinite subgraphs. Are there are two-way infinite subgraphs?

Example 4a

The non-algorithmic version of this is $\overline{BD}$, as the following transitions show.

```
...11110111...      ...00001111...
...00010000...      ...00010000...
```

Note that these transitions do not involve any erasable configurations.

Since an algorithmic automaton is configuration-erasable if and only if its non-algorithmic variant is, this example is a counter example to: If a cellular automaton is $\overline{BD}$ then it is configuration-erasable.

Is this automaton GOE? If it is, we have a GOE automaton which is not locally GOE.

Example 5

A two-way infinite shift register, in which everything shifts to the right one cell per time step, is BD, while every finite area is $\overline{BD}$. It is cell-erasable, but not configuration-erasable.

The following transitions motivate the definition of configuration-erasability:

Cell	1	2	3	4	5	6	7	8	9	10	11	12
t	0	0	1	1	0	1	1	0	1	1	0	0
t+1		0	0	1	1	0	1	1	0	1	1	
t'	0	0	1	1	0	1	1	0	1	0	0	0
t'+1		0	0	1	1	0	1	1	0	1	0	

The cellular transition function is $\overline{BD}$ and is asymmetrical.

t	000	010	001	011	100	110	101	111
t'	0	1	1	0	1	1	0	0

For algorithmic internal states, the global transition function is 1-1 and properly into, so there are GOE internal states, and hence GOE configurations. Consequently this automaton is BD, with a unit cycle and one-way infinite subgraphs. Are there are two-way infinite subgraphs?

Example 4a

The non-algorithmic version of this is $\overline{BD}$, as the following transitions show.

```
...11110111...     ...00001111...
...00010000...     ...00010000...
```

Note that these transitions do not involve any erasable configurations.

Since an algorithmic automaton is configuration-erasable if and only if its non-algorithmic variant is, this example is a counter example to: If a cellular automaton is $\overline{BD}$ then it is configuration-erasable.

Is this automaton GOE? If it is, we have a GOE automaton which is not locally GOE.

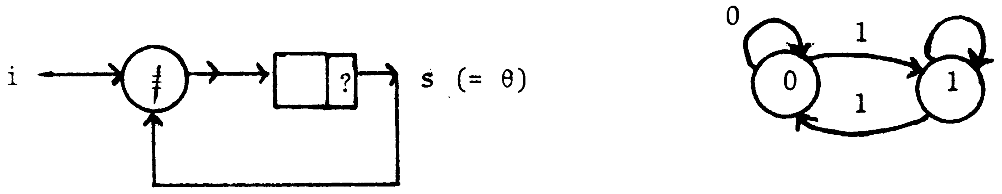
Example 5

A two-way infinite shift register, in which everything shifts to the right one cell per time step, is BD, while every finite area is $\overline{BD}$. It is cell-erasable, but not configuration-erasable.

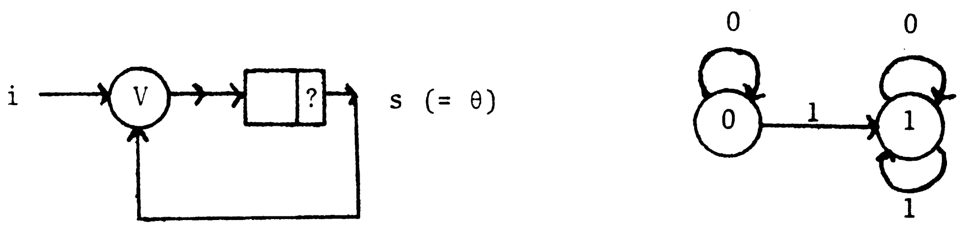
The following transitions motivate the definition of configuration-erasability:

Cell	1	2	3	4	5	6	7	8	9	10	11	12
t	0	0	1	1	0	1	1	0	1	1	0	0
t+1		0	0	1	1	0	1	1	0	1	1	
t'	0	0	1	1	0	1	1	0	1	0	0	0
t'+1		0	0	1	1	0	1	1	0	1	0	

The information in cells 3-10 at t is coded so that the ends are identifiable; 11 is a period, and 01 and 10 represent 0 and 1 respectively. This information is not lost after a unit shift, since a shift-register is not configuration-erasable. But a finite area is $\overline{BD}$, as may be seen by comparing the upper and the lower transition.



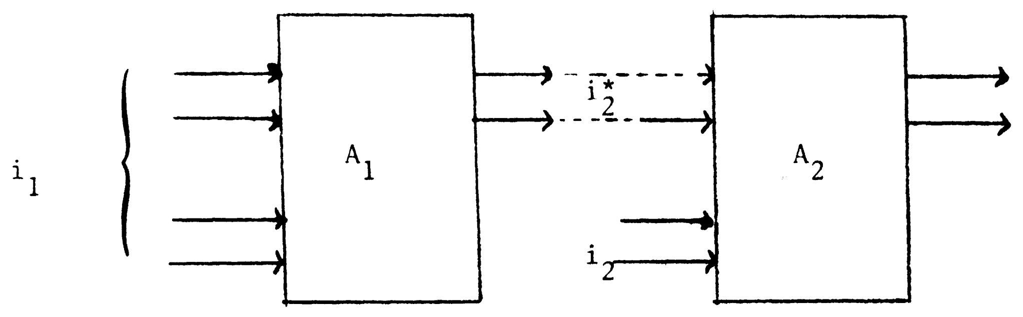
(a) Binary counter (BD)



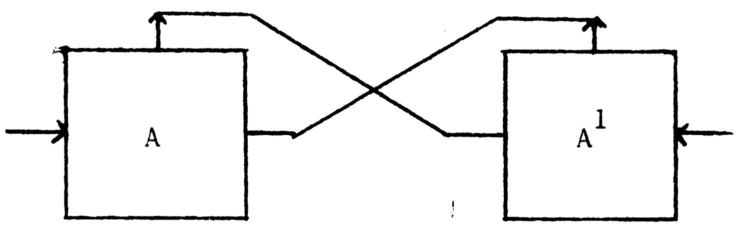
(b) A $\overline{\text{BD}}$ automata

Examples of BD and $\overline{\text{BD}}$ automata

Figure 1



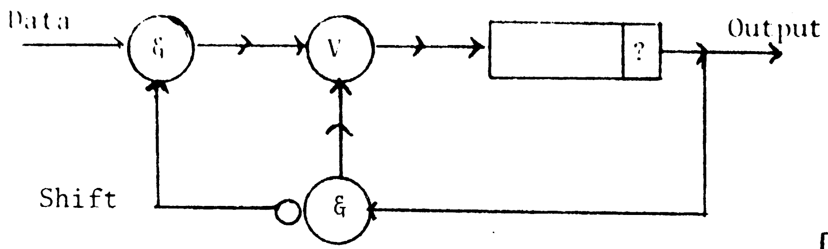
(a) If A_1 and A_2 are BD, their series connection is BD also



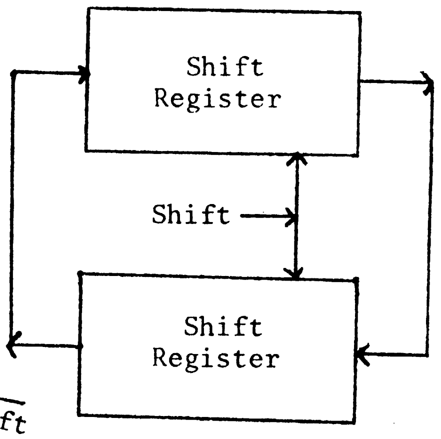
(b) A and A^1 are BD, but their cyclic connection is $\overline{BD}$

Finite systems of finite automata

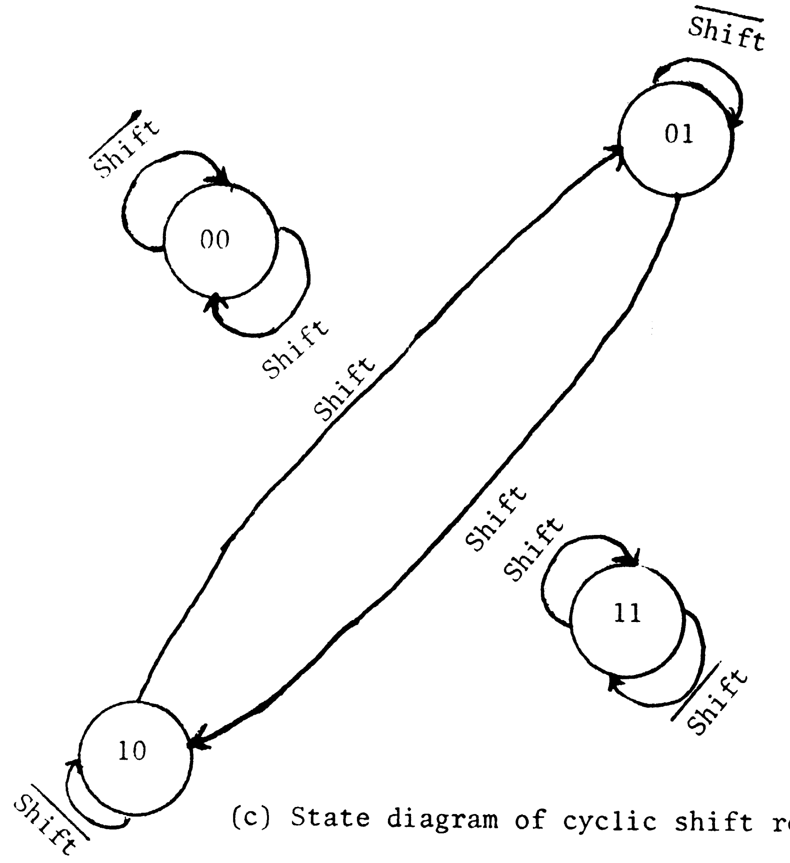
Figure 2



(a) Shift register ($\overline{BD}$)



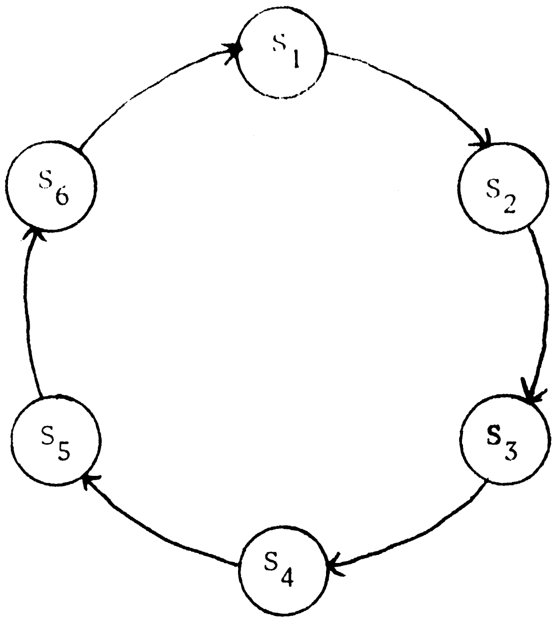
(b) Cyclic shift register


$\overline{\text{Shift}}$
01
$\overline{\text{Shift}}$
00
Shift
Shift
Shift
Shift
11
$\overline{\text{Shift}}$
10
$\overline{\text{Shift}}$

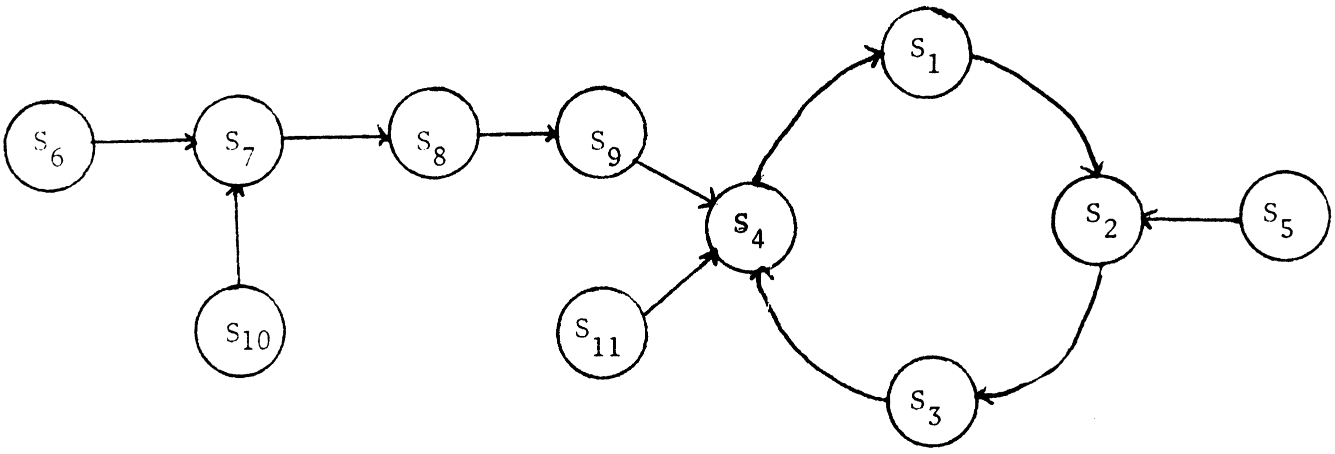
(c) State diagram of cyclic shift register(b)

A finite system of $\overline{BD}$ automata which is BD

Figure 3



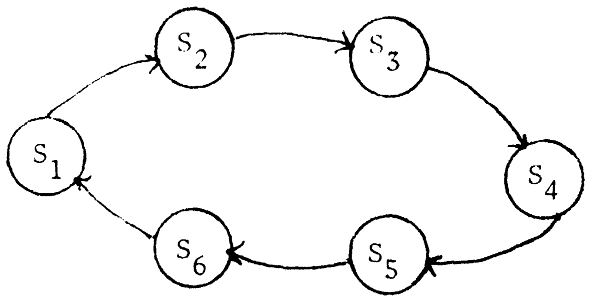
(a) Pure cycle



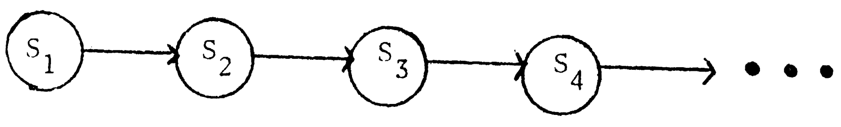
(b) Cycle with prefixes

Components of state graphs
of closed finite automata

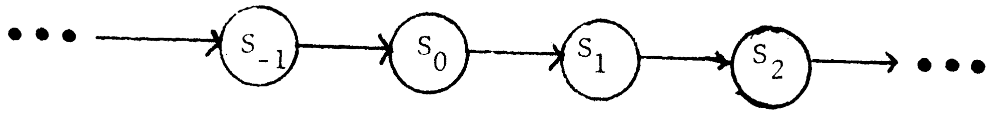
Figure 4



(a) Pure-cycle



(b) One-way infinite



(c) Two-way infinite

Subgraphs of state diagrams
for closed BD automata

Figure 5