Von Neumann's Self-Reproducing Automata

ARTHUR W. BURKS

June 1969

JUN 24 1969

Computer and Communication Sciences Department

11.3

THE UNIVERSITY OF MICHIGAN

College of Ligerature, Science, and the Arts
Computer and Communication Sciences Department

Technical Report

VON NEUMANN'S SELF-REPRODUCING AUTOMATA

Arthur W. Burks

supported by:

ABSTRACT

John von Neumann's kinematic and cellular automaton systems are des-
cribed. A complete informal description of the cellular system is pre-
sented including an explanation of the realization of logical components,
the design of computer organs, the construction, destruction and move-
ment of organs by sequences of internally originated pulses, universal
computation and construction, and self-reproduction. Connections between
von Neumann's automaton research and his work on computer design are
brought out, and the significance of cellular arrays for biological research
discussed.

TABLE OF CONTENTS

LIST OF FIGURES

[This paper uses some of my figures and text from von Neumann's *Theory of Self-Reproducing Automata,* which I edited and completed. Some use is also made of figures from my "Cellular Automata," which appeared in Russian at pp. 100-111 of *Theory of Finite and Probabilistic Automata,* edited by M. A. Gavrilov, Moscow, Nauka, 1965.]

1. KINEMATIC SELF-REPRODUCTION

The late John von Neumann once pointed out that, in the past, science has dealt mainly with problems of energy, power, force and motion. He predicted that in the future science would be much more concerned with problems of control, programming, information processing, communication, organization, and systems. General purpose digital computers provide an excellent opportunity for studies of this kind, and von Neumann started a theory of automata based on them. He wished this theory to deal with the control, informational, and logical aspects of both man-made automata (such as digital and analog computers) and natural systems (such as cells, nervous systems, and brains). Von Neumann's conception of automata theory was very close to Wiener's conception of cybernetics, and each influenced the other. But von Neumann's automata theory placed more emphasis on logic and digital computers, while Wiener's cybernetics was oriented more around physiology and control engineering.

[Von Neumann, "The General and Logical Theory of Automata," "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," and review of Wiener's *Cybernetics*. See also my Introduction to *Theory of Self-Reproducing Automata,* and the Preface to Wiener's *Cybernetics*.]

One problem von Neumann posed and essentially solved was: What kind of logical organization is sufficient for an automaton to control itself in such a manner that it reproduces itself? He first formulated this question in terms of a kinematic automaton system, and later reformulated and solved it in terms of a cellular automaton system. We will explain the kinematic system briefly and then develop the cellular system sufficiently to see how self-reproduction is accomplished in it.

[For a rigorous and more complete discussion the reader is referred to von Neumann's *Theory of Self-Reproducing Automata.*]

Consider a digital computer or automaton which operates synchronously and which is composed entirely of switches ("and," "or," and "not") and delays (which delay pulses for one time unit). We will refer to these idealized elements as computing elements.

[Actually, von Neumann used idealized neurons as computing elements. These neurons combined the functions of switching and delay and were passive in that they produced no output unless they were stimulated. This type of neuron was used by von Neumann in working out the logical design of the first stored program electronic computer, the EDVAC. See *Theory of Self-Reproducing Automata,* pp. 9, 44, 99.]
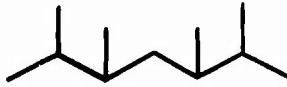
To represent an input capability and a dynamic output capability we will add five other kinds of primitive elements: a kinematic (muscle-like) element (e.g., an artificial hand), which can move elements around when signaled to do so by a computing element; a cutting element, which will disconnect two elements when signaled to do so by a computing element; a fusing (welding or soldering) element, which will connect two elements together when signaled to do so by a computing element; a rigid element (e.g., girder or bar), which will provide rigid or structural support to

assemblies of elements; and a sensing element, capable of recognizing each kind of element and communicating this information to a computing element. We shall call an automaton composed of these elements a *kinematic automaton.*
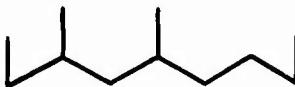
To model self-reproduction at even the most abstract logical level we need to place a kinematic automaton in an environment with which it can interact. This environment will be composed of the same stuff as the automaton is made of, namely, the various primitive elements just listed. Imagine an infinite body of liquid with infinitely many copies of each kind of part distributed in random fashion over its surface. There are switches, delays, kinematic elements, cutting elements, fusing elements, rigid elements, and sensing elements floating on the surface; imagine that they are moving back and forth in random motion, after the manner of molecules of a gas. On this arrangement, an indefinite supply of parts is available to any kinematic automaton which floats on the surface of the "lake." We will call this environment, together with any kinematic automata it contains, a *kinematic automaton system.*

It is well known how to construct a finite automaton out of switches and delays. A Turing machine consists of a finite automaton operating on an indefinitely expandable tape. We will indicate in a general way how to construct tape in the kinematic system, and how to augment a finite automaton so that it can read an arbitrary position on the tape, modify an arbitrary position on a tape, and extend (or contract) the tape.

The matrix or basis of the tape is a zig-zag arrangement of the rigid elements (girders). Each intersection holds a binary digit or bit of information: a "one" is represented by a protruding girder attached to the intersection, a "zero" by the absence of such a girder. For example,

stores 0110110 while

stores 1101001. The finite automaton can move itself relative to the tape by means of kinematic elements. It can change a "one" to a "zero" by separating a protruding girder from the tape intersection to which it is attached. It can change a "zero" to a "one", or extend the tape, by sensing a girder on the surface of the lake with a sensing element, picking up the girder and placing it in position with a kinematic element, and connecting the girder to the tape by means of a fusing element.

The primitive elements of von Neumann's kinematic system are at a higher level than atoms and molecules, so that this system is not suitable for modeling the chemical, physical, or biological aspects of self-reproduction. It was intended for modeling the control, organizational, programming, and logical aspects of self-reproduction. Let us view this process at the block diagram level.

One can see in a general way how construction takes place in the kinematic system. A finite kinematic automaton can be completely described by listing its parts and their connections, and this description $D(M)$ can be stored on the tape. A (finite) constructing automaton can then be designed which will interpret this description and carry out the construction. The parts (elements) needed are moving randomly on the surface of the lake, and hence will come in contact with the constructing automaton, which can sense the particular kinds of parts needed, pick them up, and assemble them according to the specification $D(M)$.

Self-reproduction in this scheme takes the following abstract form. Let M_c be a constructing machine. To begin with, machine M_c stores its own description $D(M_c)$ on its tape, and floats in an environment of an unlimited number of parts of each kind. The constructing machine M_c then interprets the description $D(M_c)$ and makes a copy of the machine described, namely M_c.

The constructing machine M_c next makes a tape for the new machine, copies its own tape contents {namely, $D(M_c)$} on this new tape, and attaches the tape to the newly constructed machine M_c. The process just described began with a machine M_c storing $D(M_c)$ on its tape, and ended with a second machine M_c storing $D(M_c)$ on its tape. Compare Figure 29. This is a kind of self-reproduction.

This is the general picture von Neumann had in mind. How can it be worked out in detail in a logically rigorous way? To do this we must completely specify the powers of each element and the rules of operation of the elements when they interact with one another in a machine and on the interface between a machine and the environment. Consider, as an example, a constructing automaton. This employs kinematic, cutting, and fusing elements to move and operate on other elements. How does the constructing automaton find the elements and loci for the kinematic, cutting, and fusing elements to operate on? It might do this via the kinematic, cutting,and fusing elements themselves, if these elements had sensing powers of their own. Alternatively, it might do this by means of sensing elements, provided it could coordinate the operation of a sensing element with a kinematic (or cutting or fusing) element.

It is a difficult problem to develop a complete, precise set of rules for the kinematic system which is at the same time simple and enlightening. Moreover, it is doubtful that anything of much value is contributed by the kinematic or motional capabilities of the system. On the one hand, these kinematic features of the model are too remote from chemistry, physics, and mechanics to be of much interest in their own right; while on the other hand, they are too remote from problems of organization, control, and logic to contribute to our understanding of these problems. Thus the motional

capability of the kinematic system is the source of complexities which, in the present context, are not worth their cost, and might better be eliminated.

2. CELLULAR AUTOMATA

The concept of a cellular automaton eliminates just these complexities.

[This concept was suggested to von Neumann by S. M. Ulam. See *Theory of Self-Reproducing Automata*, p. 94, where von Neumann gives his own reasons for shifting from the kinematic to the cellular model of self-reproduction. Though he does not mention it, he was probably also influenced by his interest in obtaining high speeds of computation by parallelism.]

A cellular system constitutes a basic framework or "space" in which automaton events can take place. Moreover, one can formulate precise and simple rules governing the operation of the system. In this section we will explain the general concept of a cellular system. In Sections 3-10 we will define the specific 29-state cellular system that von Neumann worked with, and show in a general, intuitive way how finite automata, Turing machines, and self-reproducing automata can be constructed in this system. In Section 11 we will compare von Neumann's kinematic and cellular models of self-reproduction, and in Section 12 we will make some general remarks about the investigation of cellular systems.

Cellular automata are the main theme of the essays collected in *Essays in Cellular Automata*. They are also called tessellation automata and iterative circuit computers.

The notion of a cellular automaton is built up in the following way. We begin with a *cellular space*, which consists of an infinite n-dimensional space together with a neighborhood relation defined on this space. The *neighborhood relation* gives, for each cell, a finite list of cells which are its neighbors. The time basis of the system is synchronous, with $t = 0$, 1, 2, 3, ...

A *cellular automaton system* (or "cellular system," for short) is

specified by giving a finite list of states for each cell, a distinguished state (called the "blank state"), and a rule which gives the state of a cell at time $t + 1$ as a function of its own state and the states of its immediate neighbors at time t. We will call the list of states for a cell together with the rule governing the state transition of the cell a *transition function*. The distinguished state corresponds to the blank state of a square of a computing machine tape, and it is required of a transition function that if a cell and its neighbors are in the blank state at time t, the cell is in the blank state at time $t + 1$.

Thus a cellular automaton system consists of a cellular space and a transition function defined over that space. A cellular automaton state is specified by a finite list of cells together with the cell state assigned to each, it being understood that all other cells are in the blank state. It follows inductively from the finitistic character of a transition function that each cellular automaton state is succeeded by a cellular automaton state. A *cellular automaton* consists of a cellular automaton system together with an assigned initial state (state at $t = 0$).

In most of the cellular systems we will discuss, the neighborhood relation and transition function are the same over the whole space. But the intent is to allow variations from cell to cell, provided that the neighborhood relation and transition function of any cell is calculable from its coordinates.

[One could also allow neighborhood relations and transition functions which change over time, as Holland does in his concept of path-building in iterative circuit computers.]

It follows that the state of a cellular automaton at time $t + 1$ is calculable from its state at time t.

The preceding definitions are actually for the deterministic case. A deterministic transition function yields a unique next state for each cell, and a *deterministic cellular automaton* has a single initial state, so that a deterministic cellular automaton has a unique history through time. An *indeterministic cellular automaton* may have more than one initial automaton state, and its transition function yields a set of possible next states for a cell. A *probabilistic cellular automaton* is an indeterministic cellular automaton with a probability distribution over the initial automaton states and with a probability distribution over the possible next states of each cell.

A transition function is equivalent to a finite automaton which is located in the cell and receives inputs from the neighboring cells via unit delays. The finite automaton located in a cell is deterministic, indeterministic, or probabilistic according to the nature of the transition function.

[See Burks and Wright, "Sequence Generators and Digital Computers,"
pp. 153, 193-197.]

3.   VON NEUMANN'S CELLULAR SYSTEM

We will discuss the general case again in Secion 12. Until then we
will restrict our attention to von Neumann's specific cellular automaton
system. It is based on a two-dimensional cellular space of square cells
in which each cell has as neighbors the four cells with which it shares
boundaries; see Figure 1. Each cell is capable of 29 different states;
see Figure 2. The transition function is deterministic, and is the same
for every cell of the space.

We proceed now to describe the transition function of von Neumann's
29-state cellular system. He called the blank state the "unexcitable
state" ($\underset{\sim}{U}$), because in his system a single pulse is not sufficient to
excite it. It is represented in our diagrams either by a blank or by
cross-hatching (as in Fig. 7). For expository purposes we will divide
the remaining 28-states into subsets, so we can treat the automaton occu-
pying each cell as a set of nine basic elements or circuits (as in Fig. 3),
together with control processes for creating any of these elements from
$\underset{\sim}{U}$ and for reducing any of these elements to $\underset{\sim}{U}$. There is one confluent
element ($\underset{\sim}{C}$), four ordinary transmission elements ($\rightarrow$, $\uparrow$, $\leftarrow$, $\downarrow$), and four
special transmission elements ($\Rightarrow$, $\Uparrow$, $\Leftarrow$, $\Downarrow$). Each transmission element
has a quiescent (passive, no-pulse, zero) state and an excited (active,
"pulse," one) state; the confluent element has a quiescent state and
three active states. Thus the nine basic elements require only 20
different states.

The control processes are of two kinds. The constructive process converts an unexcitable state into a passive confluent or transmission element, employing as intermediaries some transient (sensitized) states. The destructive process converts a confluent or transmission element into an unexcitable state. The complete set of 29-states and a summary of the transition rule is given in Figure 15.

4. CONFLUENT AND TRANSMISSION STATES

The confluent element $\underset{\sim}{C}$ has four states. It performs the functions of conjunction, double-delay, and wire-branching. The four states are the four on-off combinations of the two delay units; no states are allocated to specify input or output directions, since these are determined by the contents of the four bordering cells in the following way. A confluent element receives inputs from all ordinary transmission elements whose outputs are directed toward it. It transmits an output (after two units of delay) to all ordinary and all special transmission elements in contiguous cells provided that the outputs of these elements are not directed toward it. Cells *A1* and *B2* of Figure 3 contain confluent elements. They arc shown in Figure 3a as constructions of switches, and delays (whose initial outputs are zero) and in Figure 3b in abbreviated notation. The notation "$\underset{\sim}{C}$" does not indicate the specific state of the confluent element at any time. This is shown by subscripts when needed; for example, "$\underset{\sim}{C}_{01}$" indicates that the present output state is zero (0) and the next output state is one (1). The four confluent states are thus $\underset{\sim}{C}_{00}$, $\underset{\sim}{C}_{01}$, $\underset{\sim}{C}_{10}$, and $\underset{\sim}{C}_{11}$.

There are four ordinary transmission elements, $\rightarrow$, $\uparrow$, $\leftarrow$, $\downarrow$, one for each output direction. An ordinary transmission element performs the functions of disjunction and unit delay, and has two states. A right-directed ordinary transmission element is shown in cell *A2* of Figure 3, a left-directed ordinary transmission element is shown in cell *B1* . The number and direction of inputs to an ordinary transmission element are specified by the contents of the four neighboring cells. An ordinary transmission element receives disjunctively from all contiguous cells not in its output direction which contain either ordinary transmission elements

whose outputs are directed toward it or confluent elements. An ordinary transmission element in cell α transmits to the neighboring cell β in its output direction if either cell β contains a confluent element or if β contains an ordinary transmission element not directed toward α.

Each of these ordinary transmission elements has two states, a quiescent (passive, no-pulse, zero) state, and excited (active, pulse, one) state. When the state is to be indicated, the former is shown by a plain arrow, the latter by an arrow with a dot beside it.

There are four special transmission elements, ➡, ⬆, ⬅, ⬇, one for each output direction. Internally, special transmission elements operate in the same way as ordinary transmission elements, but their relations to other elements are different, especially with respect to the destruction process (Sec. 5). A special transmission element receives disjunctively from all contiguous cells (not in its output direction) which contain either confluent elements or special transmission elements whose outputs are directed toward it. A special transmission element transmits to the neighboring cell in its output direction only if this cell contains a special transmission element not directed toward it. In Figure 4, an ordinary transmission pulse into input α at time t will produce two special transmission pulses from output b, one at time $t + 5$ and the other at time $t + 9$.

Each special transmission element has two states, the quiescent and the excited. When these are to be indicated, the quiescent state is represented by a plain double-arrow, while the excited state is represented by a double arrow with a dot beside it. See Figure 15.

We have now explained how the confluent and transmission elements of the system operate, except for the construction and destruction processes. We will next show how these elements may be used to construct some organs.

5. SOME COMPUTER ORGANS

A useful method of operation in a cellular automaton is to convert a pulse into a binary sequence of pulses and spaces, transmit the sequence, and then decode it. This may be accomplished by means of the following two organs.

A *pulser* is an organ which converts a single pulse into a specific pulse-space sequence. The method of operation is clear from Figure 5. When supplied with a single input pulse at time t this pulser will produce the sequence 10101 from its output at times $t + 10$ through $t + 14$. The input pulse is sent along three paths of varying temporal lengths; these paths are merged disjunctively to produce the sequence 10101.

A *decoder* produces a single output pulse if the sequence it receives has pulses in certain specified positions. The method of operation is clear from Figure 6. This decoder produces an output pulse if and only if the input sequence is 10101 or a sequence which "covers" 10101 (i.e., which has a "1" in its first, third, and fifth positions). The technique is again that of diverging and merging paths, but here the paths are merged conjunctively (at confluent elements) rather than disjunctively, so that unless pulses appear at certain places in the input sequence, there will be no output pulse.

It is clear that there is an algorithm which, when given any finite binary sequence, will design both a pulser and a decoder for that sequence.

The decoder does not recognize only its defining sequence, but that sequence or any sequence which "covers" it. Thus the decoder of Figure 6 produces an output for any of the sequences 10101, 11101, 10111, and 11111. One can construct a recognizer which recognizes only its defining sequence;

see Section 5 and Figure 14 below. Alternatively, one can use sets of
sequences such that no sequence of the set covers any other one, so that
decoders are sufficient for recognizing sequences. This technique is used
in the coded channel, which is described next.

There is no wire-crossing primitive in von Neumann's system. Two
kinds of wire-crossing units may be synthesized. The first is a *coded
channel* which has any number of associated inputs and outputs. A single
pulse (one) fed into an input will later appear at the corresponding
output or outputs, provided that no other input is stimulated during the
operation of the coded channel. Thus the coded channel can cross any
(finite) number of wires or channels, provided that only one input is
used at a time. The coded channel cannot cross two continuous streams
of signals (ones and zeros) without interference. This can be done by
a "real time" crossing organ, which will be described in a moment.

The principle of a coded channel may be explained in connection with
Figure 7. There are inputs a_1, a_2, a_3, and outputs b_1, b_2, b_3; each input
a_i is associated with the corresponding output (or outputs) b_i. Thus a
pulse into input a_2 will eventually appear at both b_2 outputs (not simul-
taneously) and nowhere else. The coded channel is made up of seven pulsers
and seven decoding organs (all shown in reduced size), together with a
"main channel" running from the output of pulser $\underline{P}(10011)$ to the input of
decoder $\underline{D}(11001)$. A long arrow represents a sequence of ordinary trans-
mission states.

The coding for this particular coded channel is done with six sequences
of length five such that none of these sequences bitwise implies (is covered
by) any other. The sequences 11100, 11010, 11001, 10110, 10101, 10011 are
associated with a_1, a_2, a_3, b_1, b_2, b_3, in that order. The way the sequences

operate is best explained by means of an example. Suppose input a_2 is stimulated. This will cause pulser $P(11010)$ to inject its defining sequence 11010 into the main channel. This sequence will travel to the end of the main channel, but because it is distinct from every other sequence used it will affect only decoder $D(11010)$. $D(11010)$ will then send a pulse to pulser $P(10101)$, which will inject 10101 into the main channel. The sequence 10101 will travel to the end of the main channel, but because it differs from every other sequence used it will affect only the two decoders $D(10101)$, both of which will emit pulses from their outputs b_2.

The inputs and outputs of the coded channel may be positioned in any order. It is because of this that two sequences are associated with each input-output pair, the conversion from one sequence to its mate taking place at the top of Figure 7.

The inputs to the coded channel must be spaced sufficiently far apart in time to avoid corruption or cross talk. For suppose a_1 and a_2 were stimulated so their outputs 11100 and 11010 followed each other immediately in the main channel. The combined sequence 1110011010 contains the sequence 11001 which is assigned to input a_3, and hence which would operate $D(11001)$ and eventually cause an output at b_3.

The second kind of wire-crossing is the real-time crossing organ of Figure 8.

[This organ was designed by J. E. Gorman.]
Recall that a dot beside an arrow signifies that the transmission element is in the active state at some temporal reference point. Thus this organ contains five clocks, each producing an alternating sequence of zeros and ones; these sequences are used to gate the signals being crossed.

The crossing organ operates as follows. Consider first the behavior of the crossing organ when the inputs a_1 and a_2 are 000 ... The clocks send alternate zeros and ones into both inputs of each of the six confluent states $C3$, $C6$, $F3$, $F6$, $E1$, and $H5$. The phasing of the ones (i.e., pulses) as they enter these confluent cells is indicated in Figure 8a by dashed and dotted lines. A dashed line signifies a one (pulse) at every even time ($t \geq 4$) and a dotted line signifies a one (pulse) at every odd time ($t \geq 3$). It is clear from Figure 8a that the two sequences arriving at a confluent state are out of phase. The function of these clock sequences is to gate the sequences coming into a_1 and a_2 so that they can cross each other.

The sequence i_0, i_1, i_2, i_3, i_4, i_5, ... entering a_1 is split into two sequences by the confluent cell $A4$. The clocks insert ones into every even position of the upper sequence and into every odd position of the lower sequence. The odd bits -, i_1, -, i_3, -, i_5, ... are allowed by the gating pulses to pass along row 3 and out at b_1, while the even bits i_0, -, i_2, -, i_4, ... are allowed by the gating pulses to pass along row 6 and out at b_1. Similarly, the sequence j_0, j_1, j_2, j_3, j_4, j_5, ... entering a_2 is split, with the even bits j_0, -, j_2, -, j_4, -, ... traveling up column C and the odd bits -, j_1, -, j_3, -, j_5, ... traveling up column F. The phasing of the whole system is such that the sequence j_0, -, j_2, -, j_4, -, ... is interleaved with i_0, -, i_2, -, i_4, -, ... at cell $C6$ and with -, i_1, -, i_3, -, i_5, ... at cell $C3$. Likewise, the sequence -, j_1, -, j_3, -, j_5, ... is interleaved with i_0, -, i_2, -, i_4, -, ... at cell $F6$ and with -, i_1, -, i_3, -, i_5, ... at cell $F3$. For example, the sequences entering and leaving cell $C6$ are:

From the left: 0 0 1 0 1 i_0 1 i_2 1 i_4 ...

From below: 0 0 0 1 j_0 1 j_2 1 j_4 1 ...

Output: 0 0 0 0 0 0 j_0 i_0 j_2 i_2 j_4 i_4 ...

The sequences from cells $C3$ and $F3$ are combined in cell $E1$ to give the output

j_0, j_1, j_2, ... delayed 15 units of time. Similarly, the sequences from cells $F3$ and $F6$ are combined in cell $H5$ to give the output i_0, i_1, i_2, ... delayed 15 units of time. In this way information passes from a_1 to b_1 and from a_2 to b_2 without any cross interference.

This concludes our examples of organs synthesized from ordinary transmission elements and confluent elements alone. It should be noted that the operation of these elements depends in an essential way upon their context, i.e., upon the contents of neighboring cells. Whether a confluent or transmission element receives information from or transmits information to a neighboring cell depends upon what is in that cell. See, for example, Figure 5. The ordinary transmission element of cell $A2$ receives only from the ordinary transmission element of cell $A3$, and hence functions only as a unit delay. In contrast, the ordinary transmission element of cell $E1$ receives from both cell $D1$ and cell $E2$, and hence is a disjunction ("or") element as well as a unit delay. In Figure 6, the confluent element $C6$ operates as a wire-branching element, feeding $C5$ and $D6$, while the confluent element $E1$ operates as a conjunction ("and" element), producing an output (with two units of delay) only when it is stimulated by *both* $D1$ and $E2$.

6. CONSTRUCTION AND DESTRUCTION

Twenty-one states so far described (one unexcitable, four confluent, eight ordinary transmission, and eight special transmission) are not sufficient to generate all switching functions. The switching functions realized by these primitives are all "positive" in the sense that they carry zeros into zeros (quiescence into quiescence). Now, a negation element converts a passive input (quiescence, zero) into an active output (excited, one). Consequently, a negation element cannot be synthesized from the twenty-one states described so far.

We could add a negation element to the list of primitives, but there are two reasons not to do so if it can be avoided. First, it would increase the number of states beyond twenty-nine. Second, when construction and self-reproduction take place in a cellular automaton it is desirable that the constructed automaton be completely passive, otherwise the automaton being constructed may start its own construction-destruction processes and thereby interfere with its construction. Moreover, we can obtain the effect of negation as a by-product of the destruction and construction processes which are needed anyhow.

There are two of these processes. The *construction process* (von Neumann called it the "direct process") transforms the unexcitable state into any of the nine passive forms $\underline{C}_{00}$, $\rightarrow$, $\uparrow$, $\leftarrow$, $\downarrow$, $\Rightarrow$, $\Uparrow$, $\Leftarrow$, using the eight "sensitive" or transient states $\underline{S}_\theta$, $\underline{S}_0$, $\underline{S}_1$, $\underline{S}_{00}$, $\underline{S}_{01}$, $\underline{S}_{10}$, $\underline{S}_{11}$, $\underline{S}_{000}$ as intermediaries. The *destruction process* (von Neumann called it the "reverse process") transforms both passive and active forms of the confluent, ordinary transmission, and special transmission states back into the unexcitable state in a single time step.

Ordinary and special transmission states operate the same way in the

construction process. Consider a cell α which is in the unexcitable state
and which has one or more immediate neighbors which contain transmission
elements directed toward it. The construction process is defined by the
tree of Figure 9, where a "1" means that at least one of the transmission
elements directed toward the cell α is active. A pulse is required to
start the process; after it is started the process continues until cell
α is in one of the passive states $\underline{C}$, →, ↑, ←, ↓, ⇒, ⇑, ⇐, ⇓. Consider,
for example, Figure 10, where the sequences 1101 and 1010 are supplied
to cells *A2* and *B3*. The disjointed sequence is 1111; it carries cell *B2*
through this sequence: $\underline{U}$, $\underline{S}_\theta$, $\underline{S}_1$, $\underline{S}_{11}$, $\underline{C}_{00}$.

The reason for having both special and ordinary transmission states
is that they play opposite roles in the destruction process. A pulse
from an ordinary transmission state into a cell containing a special
transmission state changes the latter to an unexcitable state in one
time step. Similarly, a pulse from a special transmission state into a
cell containing either an ordinary transmission state or a confluent
state transforms that cell into an unexcitable state. These destruction
pulses override any pulses used for communication and computation. In
Figure 11, a pulse into input a at time t will enter cell *D1* at time
$t + 4$ and cell *A3* at time $t + 5$ so these cells will contain unexcitable
elements at times $t + 5$ and $t + 6$ respectively.

Note that the destruction process is effective on both the active
and passive forms of the nine elements ($\underline{C}$,→ , ↑, ←, ↓, ⇒, ⇑, ⇐, ⇓).
The initial configuration of a cellular automaton may contain the active
as well as the passive forms of these elements; indeed, it may contain
any of the 29 states. On the other hand, the construction process leads
only to the passive forms of the nine elements ($\underline{C}$, →, ↑, ←, ↓, ⇒, ⇑, ⇐, ⇓),
that is, only to 9 different states.

Next we will illustrate how the destruction and construction processes may be used to synthesize negation. Figure 12 shows a *periodic pulser* which, when started, emits the sequence 10101 until it is turned off. It consists of a pulser which produces the desired sequence (10101) when it is stimulated by a start pulse, a repeater (cells $A4$, $B4$, $A5$, $B5$) which repeats the sequence until it is turned off, a pulser which produces a sequence 11111 when stimulated by a stop pulse, and a transducer (cells $A1$, $B1$, $B2$, $B3$) by means of which the repeater is turned off. The sequence 10101 enters repeater cell $A4$ and cycles around the cells $B4$, $B5$, $A5$, $A4$, and also out of cell $B4$ to the right, indefinitely. These four cells have a total delay of five units. The stop pulse produces a sequence 11111 which passes down channel $B1$-$B2$-$B3$ and enters cell $B4$. The first pulse kills $\underset{\sim}{C}$ to $\underset{\sim}{U}$ and the next four restore $\underset{\sim}{U}$ to $\underset{\sim}{C}_{00}$ via the sequence of sensitized states $\underset{\sim}{S}_{\theta}$, $\underset{\sim}{S}_{I}$, $\underset{\sim}{S}_{11}$, and $\underset{\sim}{S}_{111}$, the last being $\underset{\sim}{C}_{00}$. This erases the sequence 10101 in the repeater and leaves the repeater in its original state.

It is clear that there is an algorithm which, when given any finite sequence, will design a periodic pulser for that sequence. Larger output sequences will require longer killing sequences, and the length of the erasing sequence must be a multiple of five, so that the confluent cell ($B4$) feeding the output is left in its original state at the end of the stop operation.

A periodic pulser whose output is a repeated 1 is a binary storage element or "flip-flop". Since the minimal cell cycle consists of four cells and a delay of five, it must be designed to produce a sequence of length five or longer. Figure 13 shows a periodic pulser which produces the sequence 11111 until it is turned off, together with a gate (confluent element). When $\underset{\sim}{PP}$(11111) is on, it sends a constant stream of pulses

to the gate, and the signals recieved at the gate input are transmitted to the gate output. When the $\underset{\sim}{PP}$(11111) is off, the gate is closed.

The method used for erasing sequences in periodic pulsers can also be employed in recognizers. It will be recalled that a decoder responds not only to its defining sequence, but also to any sequence which "covers" its defining sequence. Thus the decoder $\underset{\sim}{D}$(10101) responds to any of the sequences 10101, 11101, 10111, and 11111. In contrast, a recognizer responds only to its defining sequence.

The underlying operating principle of a recognizer may be explained in connection with an example. Figure 14 shows a recognizer $\underset{\sim}{R}$(101001). Every input sequence is fed to a decoder $\underset{\sim}{D}$(101001), which produces an output pulse for any one of the eight sequences $1, x_1, 1, x_2, x_3, 1$, (where $x_1, x_2, x_3 = 0, 1$). Since we want a final output only for $x_1 = x_2 = x_3 = 0$, we need to detect when any of x_1, x_2, x_3 are one. This is done by a pulser $\underset{\sim}{P}$(1101), provided that the timing is correct. Hence if there is an output from $\underset{\sim}{D}$(101001) and none from $\underset{\sim}{P}$(1101), we want a final output from b; while if there is an output from $\underset{\sim}{D}$(101001) and also from $\underset{\sim}{P}$(1101), we do not want a final output from b. The arrangement of Figure 14 provides this. If there is an output from $\underset{\sim}{D}$(101001) and none from $\underset{\sim}{P}$(1101), a pulse leaves cell $G1$ and zig-zags down rows 1 and 2 to exit at b. But if there is an output from $\underset{\sim}{P}$(1101) in the right phase with respect to the output of $\underset{\sim}{D}$(101001), the confluent element in cell $I5$ will stimulate the pulser $\underset{\sim}{P}$(11111), whose purpose is to block the output at cell $T1$. The pulser $\underset{\sim}{P}$(1101) will respond to any input pulse, of course, but the delay-length of the path from input a through $\underset{\sim}{D}$(101001) to the confluent cell $I5$, relative to the delay-length of the path from input a through $\underset{\sim}{P}$(1101) to the same confluent cell $I5$, is such that $\underset{\sim}{P}$(11111) is

stimulated only when x_1, x_2, or x_3 is one. The output sequence 11111 from the pulser $\underline{P}$ (11111) cycles $T1$ from $\underline{C}$ through $\underline{U}$, $\underline{S}_\theta$, $\underline{S}_1$, $\underline{S}_{11}$, $\underline{S}_{111}$ and back to $\underline{C}_{00}$. Consider the two paths from the output of $\underline{D}(101001)$ to the final output b of the recognizer, one along rows 1 and 2, the other through $\underline{P}(11111)$. The timing along these paths is such that an output from $\underline{D}(101001)$ is blocked at cell $T1$ if and only if x_1 or x_2 or x_3 is 1. Hence the recognizer $\underline{R}(10101)$ produces a final output for the sequence 10101 but not for any sequence 1, x_1, 1, x_2, x_3, 1, where x_1 or x_2 or x_3 is 1.

We have now finished our description of the deterministic transition function governing von Neumann's 29-state cellular automaton system. We will next formulate and answer some specific questions concerning the logical powers of this system.

7. FINITE AUTOMATA

Finite automata are important in their own right, and constitute the bases of Turing machines, so we will first show how they can be embedded in von Neumann's cellular automaton system.

A finite automaton is a device or system $\langle I,\ \mathcal{D},\ 0,\ \tau;\ \lambda,\ d_0 \rangle$ which operates deterministically as follows. At each discrete moment of time (0, 1, 2, ...) it is in one of a finite set $\mathcal{D}$ of internal (delay) states, receives any one of a finite set I of input states, and transmits any one of a finite set 0 of output states. At time zero the automaton is in some distinguished internal state d_0. Its transition from state to state is governed by a function τ from inout state and internal state to internal state; τ is called its transition function. The automaton's output state is governed by a function λ from internal state and output state to output state; λ is called the "output function". An example is given in Figure 16a.

[This conception is used in Burks and Wright, "Theory of Logical Nets",
p. 1364, but without the various states being named. It is given in Moore,
"Gedanken-Experiments in Sequential Machines", p. 133 and Burks and Wang,
"The Logic of Automata", p. 203.]

To represent a finite automaton in von Neumann's cellular system,
one marks off a finite area of the cellular space, specifies the states
of the cells of this area so as to make a device which simulates the
automaton, and sends inputs into and takes outputs from the device across
the boundary of this area.

[More generally, our original definition of a cellular automaton system
(Sec. 2) can be extended to include finite cellular spaces with inputs
and outputs along the border.]

Finite automata embedded in this cellular system cannot in general operate
as fast as finite automata in the pure or idealized sense. This is so
because the finite automaton of each cell is of limited capability (having
only 29-states), and there is at least a unit delay across the boundary of each
cell. But finite cellular automata can simulate idealized finite automata.
In the terminology introduced later, for each idealized finite automaton
there is a finite cellular automaton which is computationally equivalent
to it, but not in general behaviorally equivalent to it (See Holland,
"Hierarchical Language"). In the following discussion we will assume
that the time-scale is slowed as much as is needed for the simulation.
For a fuller discussion of simulation speeds see especially Section 12
below.

In embedding a finite automaton in a cellular system there is a choice
as to how inputs and outputs are represented. We will assume given a finite
automaton, $< I, O, Y, \lambda, d_0 >$ with its input states coded as sequences,

each successive sequence starting at time $kt + m$, where k and m are integers

to be chosen on the basis of the time-delays of the simulating cellular automaton

The given finite automaton can be simulated in von Neumann's cellular

system in the manner of Figure 16. Each input state $i \in I$ is represented

by a recognizer $R(i)$ which produces a single pulse when the sequence

representing input state i is received as input. Each internal state

$d \in D$ is represented by a periodic pulser $PP(11111)$, which is turned

on to represent the given finite automaton being in state d; initially

the periodic pulser representing d_0 is turned on. Each output state o

represented by a pulser $P(o)$ which produces, when stimulated, a sequence

to represent o. The transition function and output function are rep-

resented by confluent elements functioning as gates ("and" elements),

to detect coincidences between periodic pulsers representing internal states

and pulses representing input states. The outputs of the coincidences

go to produce the desired output state, and, if a new internal state is

called for, to turn off the $PP(11111)$ representing the new input state.

All these organs are interconnected by means of a coded channel.

(The cellular automaton design described here is that of the decoded

normal form net, Burks and Wang, "The Logic of Automata", p. 281. A

variant of the simulation method of Figure 16 is described in von Neumann's

Theory of Self-Reproducing Automata, Sec. 5.1.3.)

We will explain the operation of this device for two cycles. Suppose

the system represents external state d when a sequence representing i

is received, followed (after a suitable delay), by an input representing

i_1. The sequence representing i_0 will be recognized by $R(i_0)$, which

will send a pulse into the coded channel at a_0. This will exit at both the

upper and lower b_0 exits, having an effect only in the first case. At

the upper exit it will pass through the conjunction ("and" switch) gated by the upper $\underset{\sim}{PP}$(11111), and will enter the coded channel at a_6, exiting a b_6 and stimulating $\underset{\sim}{P}(o_0)$ to produce the output sequence representing output state o_0. Thus internal state d_0 and input state i_1 produce output state o_0 and next internal state d_0.

After the simulation of this transition is finished, the sequence representing i_1 will be received and will be recognized by $\underset{\sim}{R}(i_1)$, which will send a pulse into the coded channel at a_1. This will exit at both the upper and lower b_1 exits, again having an effect only in the first case. Here it will pass through the "and" gate and enter the coded channel at a_4 (to exit at b_4 and start the internal state d_1) at a_3 (to exit at b_3 and stop the internal state d_0), and a_7 (to exit at b_7 to simulate $\underset{\sim}{P}(o_1)$ to produce the output sequence representing o_1). Thus the internal state d_0 and input state i_1 produce output state o_1 and the next internal state d_1.

In this cellular simulation of a finite automaton each successive input sequence starts at the time $kt + m$; the integer m of the formula $kt + m$ can be zero, but k must be chosen so that in the worst case the system can change its representation of the present internal state before the representation of the next internal state enters the system. The sequences representing output states will start at times $kt + n$, where n is an integer depending on the transit time of the simulating system.

The foregoing should suffice to show that any finite automaton can be simulated in von Neumann's cellular system, given a suitable representation and timing of input and output states. Actually, a considerably stronger result is inherent in the foregoing construction. For with the exception of the periodic pulser $\underset{\sim}{PP}$(11111) which represents internal state d_0,

all of the cells of the automaton of Figure 16 are initially quiescent.
Moreover, this automaton can be modified so that this $\underset{\sim}{PP}(11111)$ is initially
quiescent and is started by a pulse injected into the automaton from the
outside. This leads to the following definition.

An *initially quiescent automaton* is a finite area of the cellular
structure, every cell of which is in one of the ten quiescent states $\underset{\sim}{U}$,
→, ↑, ←, ↓, ⇒, ⇑, ⇐, ⇓, and $\underset{\sim}{C}$. It is easy to arrange for an initially
quiescent automaton to be started by a stimulus at its periphery, and we
will assume that this is done.

The method of construction illustrated in Figure 16 shows that the
following is the case. *For every finite automaton, there is an initially
quiescent cellular automaton which can simulate it.*

It should be noted that the crossing organ of Figure 8 is not initially
quiescent, since it contains five "clocks" (Fig. 8b) which must be operating
before it can operate. Moreover, one of these clocks is in the center, so
that the crossing organ cannot be started by a stimulus on its periphery.
In this respect the coded channel is superior to the real time crossing
organ.

(One can use a real time crossing organ in an in an initially quiescent
automaton in the following way. Construct the quiescent part of the crossing
organ, except for a path to the center clock. Then add pulsers to
complete the construction and start the organ. The starting stimulus
to the whole automaton must stimulate these pulsers, which then complete
and start the crossing organ. See von Neumann, *Theory of Self-Reproducing
Automata*, pp. 264-5.)

There is another method of simulating an arbitrary finite automaton,
which involves constructing only one finite automaton. This method is to

construct or embed a universal Turing machine in the cellular system.
Since a universal Turing machine can simulate any automaton, a fortiori,
it can simulate a finite automaton. Moreover, when it simulates a finite
automaton it need use only a finite amount of its tape, and hence is
also a finite automaton.

We will sketch the construction of a universal Turing machine
in Section 8. It turns out that the most efficient method of operating
the tape involves a constructing arm which can be used for general
construction purposes as well as part of a tape unit. We therefore
explain this constructing arm next.

8. CONSTRUCTING ARM

The constructing arm is used both for the process of construction
(see Fig. 17) and for operating an indefinite tape (see Fig. 18). We
will first explain how it is used to construct a finite, initially
quiescent automaton.

The general arrangement is shown in Figure 17. Appropriate sequences
of ordinary pulses are generated in the constructing device and fed into
the input s (through a confluent state $\underline{C}$) and the input o of the con-
structing arm. These signals cause the head of the constructing arm
to sweep backwards and downwards over the area $\alpha - \beta$, leaving each cell
of this area in the desired state.

The construction is carried out by the head of the construction
arm, shown in Figure 18a in normal form. Signals for construction
and destruction are sent alternately through the ordinary transmission
elements and the special transmission elements. The constructing arm
can be advanced either horizontally or vertically. It can also be

retracted either horizontally or vertically, leaving cells γ and δ

in any of the ten quiescent states U͟, C͟00 , →, ↑, ←, ↓, ⇒, ⇑, ⇐, ⇓.

The actual binary (pulse-no pulse) sequences needed for construction

and destruction are derivable from Figures 9 and 15. For purposes of

comprehension, however, it is best to replace these sequences by a

representation of their effects. We will explain this notation in

connection with the passage from Figure 19a to Figure 19b. The starting

configuration is given in Figure 19a. The following sequences are fed

into input s or input s' via special transmission or confluent states:

$$1110 \text{ changes cell } C1 \text{ from } \underline{U} \text{ to } ⇓$$

$$1101 \quad " \quad " \quad C2 \quad " \quad \underline{U} \quad " \quad ⇐$$

$$1 \quad " \quad " \quad B2 \quad " \quad ↑ \quad " \quad \underline{U}$$

$$10000 \quad " \quad " \quad B? \quad " \quad \underline{U} \quad " \quad →$$

The result is Figure 19b. Rather than writing this sequence as

"1110 1101 1 10000", however, we write it as " ⇓ U͟ ".

Figures 19 and 20 specify the sequences needed for advancing the

constructing arm by one cell. Figures 21 and 22 specify the sequences

needed for withdrawing the constructing arm and leaving two cells,

γ, δ in one of the ten quiescent states U͟, C͟00 , →, ↑, ←, ↓, ⇒, ⇑, ⇐, ⇓.

The actual binary sequence needed for γ and δ is obtained from Figures

9 and 15, as before.

Because construction proceeds from top to bottom and from right

to left, the lower left hand corner is completed last. It is therefore

convenient to stipulate that an initially quiescent automaton be started

by injecting a symbol into its left-most, lowest cell, from beneath. Figure

23 shows how this is accomplished.

This completes our discussion of the five operations of the con-

structing arm: horizontal advance, vertical advance, horizontal retreat

with γ-δ, vertical retreat with γ-δ, and injection of the starting
stimulus. These operations suffice for the construction of any
initially quiescent automaton in the first quadrant, assuming that the
required sequences are supplied to the input of the constructing arm.

A crude way in which these sequences can be obtained is by making
the constructing device consist of a row of special transmission states
feeding input s (Fig. 17) and a row of ordinary transmission states
feeding input 0. By choosing the proper sequence of quiescent and active
states in each case, one can obtain any desired finite, binary (pulse)
sequence. Since the automaton to be constructed is finite, the pulse
sequences needed for its construction are also finite. Consequently,
for each finite initially quiescent automaton A there is a finite non-
quiescent automaton B which will construct A.

Of more interest is a constructing automaton which is itself initially
quiescent. Such an automaton will be discussed in Section 10. But
since it contains the essentials of a universal Turing machine, we will
explain next how to embed a Turing machine in von Neumann's cellular
system.

9. UNIVERSAL COMPUTER

A Turing machine consists of a finite automaton operating on an
indefinitely extendable tape.

[Turing, "On Computable Numbers, with an Application to the
Entscheidungsproblem."]

In the usual arrangement, the tape and the finite automaton move
relatively to each other; usually the tape is viewed as moving back
and forth past the finite automaton, but occasionally the finite automaton
is viewed as moving back and forth along the tape. In both cases the

information on the tape moves relative to the finite automaton. In the cellular system both the tape and the finite automaton are "by nature" at rest, so it is best to connect them by a loop which can be extended or contracted.

[One could embody the usual arrangement by designing the finite automaton so that it constructs an indefinitely long shift register (adding new stages as needed) and shifts the information in the register back and forth. This is done in Burks, "The Logic of Fixed and Growing Automata" and "Computation, Behavior, and Structure in Fixed and Growing Automata".]

The arrangement is shown in Figure 24. The "tape" itself consists of an infinite linear array of cells leading off to the right; this is row 3 in the figure. "Zero" is represented by the unexcitable state (U) and "one" by an ordinary transmission state directed downward ($\downarrow$). In an initial cell assignment for the cellular system only a finite number of cells are in a state other than the unexcitable (U), so that initially (and hence also at any subsequent time) only a finite number of cells (squares) of the tape will register "one".

The following five tape operations are clearly sufficient for all computational purposes: reading, writing "one" and lengthening the reading loop by one cell, writing "zero" and lengthening, writing "one" and shortening, and writing "zero" and shortening.

Reading is accomplished by means of the reading loop, which consists of rows 1, column D(including the cell under scan) and row 4 of Figure 24. To read the contents of cell x_n, we inject the sequence 10101 into entry v. This sequence passes alon. row 1 and down into cell x_n. There are now two cases to consider. If cell x_n is in U (representing a "zero"), the sequence 1010 converts it into an ordinary transmission

state directed downward ($\downarrow$), and the final 1 of the sequence 10101

returns down row 4 and out exit w . If cell x_n is in $\downarrow$, the complete

sequence 10101 returns down row 4 and out exit w . Thus an output from

w of 1 indicates that $x_n = \underline{U}$, while an output of 10101 indicates that

$x_n = 1$. This readout procedure is destructive, since cell x_n is left in

state $\downarrow$ in either case (Fig. 24b). The old value of x_n is restored,

or a new value is inserted, during the process of lengthening or shortening

the reading loop.

Changing the length of the reading loop, and recording the new bit

in cell x_n, is accomplished by means of the constructing arm, which

consists of rows 1 and 2 of Figure 24. Note that row 1 is part of the

reading loop as well as part of the constructing arm. The constructing

arm used to modify the tape is the same as that used for construction

(see the previous section), except that the ordinary transmission

elements are located above (rather than below) the special transmission

elements. The routine for writing "one" in cell x_n and lengthening the

reading loop is given in Figure 25, where the notation of representing

a binary sequence is represented by its effects, as before. Note that

modifications in the bottom row of the unit are made through the cell

x_n before this cell is left in its final state; no other cell of the

tape is disturbed. Note also that since "one" is represented by $\downarrow$, the

process of writing "one" and lengthening (or shortening) is the same as

lengthening (or shortening). The other three cases (writing "zero"

and lengthening, writing "one" and shortening, and writing "zero" and

shortening) are handled similarly.

In the process described by Figure 25, binary sequences are fed

first into u , then into v , into u again, and finally into v again.

The two successive sequences into u can be combined into one long

sequence by inserting zeros during the moments when input v is being

used, and similarly for the alternate sequences into v. Consequently,

the operation of writing "one" and lengthening can be accomplished

by feeding two long sequences into u and v in parallel. These sequences

can be generated by two pulsers, one feeding input v and the other

feeding input u.

Let us now summarize the organs needed for our five tape operations.

Four pulsers feeding u and four pulsers feeding v are needed for the

four operations of writing "zero" (or"one") and lengthening (or shortening).

One pulser 10101 feeding v is needed for reading cell x_n. One recog-

nizer is needed to recognize the sequence 00100 (i.e.,$x_n = 0$) and

another to recognize the sequence 10101 (i.e., $x_n = 1$); both of these

recieve the output from w. Thus nine pulsers and two recognizers are

needed to operate the tape, its reading loop, and its construction arm.

This completes our brief explanation of how to embed an indefinitely

expandable tape in a cellular system, how to establish contact with

an arbitrary square (cell) of the tape, and how to read from and write

in this cell.

We now have the essential ingredients for embedding a Turing machine

in von Neumann's cellular system. In Section 7 we saw how to con-

struct an arbitrary finite automaton in this system, and we see now

how to construct and use an arbitrarily long tape. It remains to

combine these two entities.

The general arrangement is shown in Figure 26. The operation of

the tape, constructing arm (for lengthening-shortening and writing),

and reading loop have been explained. This operation is directed

by nine pulsers and two recognizers, which constitute part of the tape

control. To obtain the rest of the tape control, we must add organs
to control these pulsers on the basis of information received from
the recognizers according to the particular finite automaton operation
that is desired. Let $\mathcal{D}$ be the set of the internal states of the finite
automaton controlling the tape. Each internal state $d \in \mathcal{D}$ is represented
by a periodic pulser $\underset{\sim}{PP}(11111)$, as in Figure 16. At each stage one of
these periodic pulsers is on; the system starts operation when the
periodic pulser representing the initial state d_0 is turned on and the
pulser $\underset{\sim}{P}(10101)$ is stimulated to inaugurate the reading cycle. The
outputs of the two recognizers representing $x_n = 0$ and $x_n = 1$ are gated
via confluent cells against the outputs of the internal state periodic
pulsers to determine the next operation. According to the specific
transition function τ of the finite automaton being simulated, a next
state $d \in \mathcal{D}$ is selected, one of the four operations lengthen-shorten
and write "zero"-write "one" is selected, and after a suitable delay
the pulser $\underset{\sim}{P}(10101)$ for reading the next tape square is stimulated.
This whole sequence is iterated indefinitely.

The foregoing sketch indicates in a general way how, given any
Turing machine M , we can embed in the 29-state cellular structure
an initially quiescent automaton which performs the same computations
as M. Since this is so for any Turing machine, it is a fortiori so for
a universal Turing machine (universal computer). The full design for
a universal Turing machine is given in Thatcher's "Universality in
the von Neumann Cellular Model".

(An earlier design is given in von Neumann's *Theory of Self-*
Reproducing Automata, Part II, Chapters 3-5.)

Hence we conclude: *For any given Turing machine M (and hence for a universal Turing machine M_u), there is an initially quiescent automaton in von Neumann's 29-state cellular automaton system which will perform the same computation as M (or M_u).* As von Neumann expressed this result, his cellular space is "logically universal".

The next step is to extend this result to obtain a corresponding result for universal construction.

10. UNIVERSAL CONSTRUCTOR AND SELF-REPRODUCTION

Let us return for a moment to von Neumann's kinematic self-re-producing automaton. In the kinematic system we start with an environment containing an unlimited supply of computirg, kinematic, cutting, fusing, rigid, and sensing elements. We introduce into this environment a constructing automaton which is itself composed of these parts and which contains an explicit plan of some desired automaton. The constructing automaton proceeds to collect parts and construct the desired automaton from these parts according to its explicit plan.

It is instructive to view this whole process in terms of the states of the system. The original environment is more or less uniform, homogeneous, and unorganized. We introduce a constructing automaton containing the plan of a secondary automaton; this amounts to putting a finite area of the space (i.e., the space occupied by the constructing automaton and its plan) into a certain "state". The constructing automaton then organizes another area or region of the space according to the plan of the secondary automaton; in other words, it puts a certain secondary area into the state specified by the plan. If the constructing automaton contains its own plan, the area of the secondary automaton

is, at the end of the process, in the same state as the area of the
constructing automaton was at the beginning of the process. This is
self-reproduction in the kinematic model.

The corrseponding situation on the cellular model is this. We
begin with an infinite, uniform cellular system of 29-state automata,
each automaton being in the unexcitable state U. We then organize by
fiat a finite area of this space so that the area constitutes a con-
structing automaton which contains the plan of a secondary automaton; that
is, we put a certain area into a certain state. The constructing automaton
then sends out a constructing arm which organizes another area according
to the plan of the secondary automaton; that is, it puts a certain
secondary area into the state specified by its plan. If the constructing
automaton contains its own plan, the state of the secondary area at the
end of the process is the same as the initial state of the area occupied
by the constructing automaton. This is self-reproduction in the cellular
system.

A universal constructor which is initially quiescent is shown in
Figure 27. It is composed of two units, each consisting of a finite
automaton and an indefinitely expandable part. The first unit is a
tape unit, which can store information on, and read it from, an
indefinitely extendable linear array of cells, or tape. The tape unit
itself consists of two parts: a finite tape control; and an indefinitely
long tape, together with its reading loop and constructing arm. The tape
unit is a kind of Turing machine; see Fig. 25. The second unit of
the universal constructor is a constructing unit, which can carry out
the construction of any quiescent automaton whose description is on
the tape. The constructing unit also consists of two parts: a finite
construction control, and an indefinitaly long constructing arm.

A description of the automaton to be constructed is stored in explicit form on the tape of the universal constructor. Under direction of the construction control, the tape control reads the description from the tape and transmits it to the construction control. Using this description, the construction control sends signals into the constructing arm to bring about the construction.

The automaton to be constructed is to be initially quiescent, which means that it can be explicitly described by giving the desired quiescent state of each cell (U, $\rightarrow$, $\uparrow$, $\leftarrow$, $\downarrow$, $\Rightarrow$, $\Uparrow$, $\Leftarrow$, $\Downarrow$, or C_{00}). The limitation to in $\cdots$ ally quiescent automata should be noted. There are finite configuration (areas of cells, each of which is in one of the 29-states) which are not initially quiescent and are nevertheless constructible. For example, one could easily construct a loop of active ordinary transmission elements, each of which feeds its successor. But there are also finite configurations (not initially quiescent) which are not constructible. A simple example is a sensitized state S_θ surrounded by a band of cells in state C_{00}. Since confluent states do not construct, this configuration can exist only at time zero and hence is not constructible. Another example is S_{000} surrounded by (eight) cells in state C_{00}. This can exist only at times 0, 1, 2, or 3, and is not constructible.

Moore and Myhill establish an interesting result related to constructibility in those cellular structures in which information requires at least one time unit to pass from a cell to any of its eight bordering cells. Von Neumann's 29-state cellular structure is clearly one of these structures. Moore speaks of a finite area of cells in some state (i.e., with an assignment of a state to each cell) as a configuration, and calls a configuration which can exist only initially (i.e., at time zero), a "Garden-of-Eden" configuration. Since construction takes one or more time steps, every

Garden-of-Eden configuration is non-constructible. The converse is not the case, as our example of S_{000} surrounded by cells in state C_{00} shows. Moore and Myhill establish a necessary and sufficient condition for existence of Garden-of-Eden configurations in cellular structures of the type they consider. The condition is essentially that the cellular structure be *non-backwards deterministic*, that is, that a given state of an area can be derived from two or more preceeding states.

[The concept of backwards determinism is defined in Burks and Wang, *The Logic of Automata*, Sec. 3.3.]

Actually, one must modify this concept to take account of the influence of the environment on the edges of an area; Moore uses the term "erasable". Von Neumann's 29-state cellular structure is clearly not backwards deterministic, for the configuration consisting of $\overrightarrow{\Leftarrow}$ surrounded by a wide band of U's leads to all U's (except at the edges), as does also a configuration consisting of all U's.

Before leaving this topic we note that the restriction to initially quiescent automata is no restriction as far as computation (for either finite automata or Turing machines) is concerned. The fact that not all configurations can be constructed in von Neumann's cellular structure is analogous to the fact that not all finite automaton behaviors are realizable in the cellular structure, because of the limited capability of each cell together with the delay between cells.

We return now to the task of designing a universal constructor. The following conventions may be adopted without any essential loss of generality. The constructed automaton is to occupy a rectangular area of width α and height β, where β is an even integer. The desired cell states are designated $\lambda_{00}, \lambda_{01}, \cdots \lambda_{\alpha-1, \beta-1}$, and are to be arranged on the tape in reverse order from which they are to be used. The lower left-hand corner of the constructed

automaton is to be located at point x_0, y_0. Moreover, the constructed automaton may be started by injecting a stimulus into cell x_0, y_0 from below (see Figure 23). The tape contents are then as follows: a period, the origin point x_0, y_0; the dimensions α, β; an enumeration of the states of the cells λ_{00}, λ_{10}, λ_{01}, λ_{11}, ..., $\lambda_{\alpha-1, \beta-1}$; and a terminal asterisk.

The construction control must execute the following program or algorithm to construct the secondary automaton specified by the tape contents.

(1) The construction control orders the value of x_0 read and sends pulse sequences into the constructing arm for $x_0 + 2$ horizontal advances. It then orders the values of y_0 and β read and sends pulse sequences into the constructing arm for $y_0 + 3$ vertical advances. The head of the constructing arm is now in the upper right-hand corner of the construction area so that construction may begin.

(2) The construction control then iterates the following operation $\beta/2$ times:

    (a) A horizontal advance of the constructing arm $\alpha-2$ times

    (b) A horizontal retreat with construction of γ and δ, for $\alpha-2$ times, with the specification of γ and δ coming from the values of the λ's stored on the tape.

    (c) A vertical retreat with construction of γ and δ, for two times, with the specification of γ and δ coming from the appropriate λ's stored on the tape.

This completes the construction of the area $\alpha-\beta$.

(3) The construction control next orders a single horizontal retreat and then injects the starting signal into the left-most, lowest cell of the constructed automaton (see Fig. 23). Following this the constructing control executes y_0 vertical retreats and then $x_0 + 2$ horizontal

retreats to return the constructing arm to its initial position;

in each of these retreats both γ and δ are to be blank ($\underset{\sim}{U}$).

This completes our sketch of the algorithm or program for the construction of an arbitrary finite initially quiescent automaton. We have not gone into details on such matters as how the numbers x_0, y_0, α, and β are represented on the tape and how the construction control keeps track of what it is doing. While there are many such small problems, there are standard methods in the computer art for solving them.

The construction control can be designed by translating this algorithm into a finite automaton, and then designing a cellular automaton to simulate the resultant finite automaton, using the method given in Section 6 (cf. Fig. 16).

Since a standard Turing machnie consists of a finite automaton attached to a tape, it is desirable to allow any automaton which is to be constructed to have these two parts, and so we will provide for the following three cases.

(1) The case already discussed, to wit: The quiescent automaton to be constructed is a finite rectangular machine M. Its description D (M) is placed on the tape of the universal constructor preceded by a period and followed by an asterisk, as in Figure 27. The universal constructor reads D (M) and makes M, and then starts M by injecting a starting stimulus into it at a standard position.

(2) Let the description $D(M)$ of the desired quiescent automaton M be placed on the tape of the universal constructor, followed by a diamond, followed by the contents $T(M)$ of the tape to be attached to M, followed by a concluding square. The universal constructor will execute the description D (M) and thereby construct machine M, as before. Seeing the diamond, it will proceed to construct a tape with contents $T(M)$ and attach it to

M at some standard place; see Figure 28. More specifically, the tape will contain a period, followed by $T(M)$, followed by a concluding square. After the tape has been constructed, the constructing arm will return to the constructed automaton M and give it a starting signal. Note that the tape and its contents constitute a quiescent automaton of rectangular shape, just as M does, so that essentially the same technique can be used to construct a tape with contents $T(M)$ as was used to construct M. The tape is a special case in that it is only one cell wide, and contains only the two states $\underline{U}$ and $\downarrow$.

(3) As a special case of the preceding we will allow the description $D(M)$ to be copied into the tape of M. For this operation, the following is placed on the tape of the universal constructor: a period, the description $D(M)$, and a concluding square. The universal constructor executes $D(M)$ and makes M, then makes a tape with the contents "period- $D(M)$ - square" by copying $D(M)$, and finally starts machine M.

Let the universal constructor be designated "M_c". We then have the following result concerning the construction universality of our cellular system. *There can be embedded in von Neumann's 29-state cellular automaton system a universal constructor M_c with this ability: for each initially quiescent automaton M with tape contents $T(M)$ there is a coded description $D(M)$ of M such that, when $D(M)$ and $T(M)$ are placed on the tape attached to M_c, M_c will construct M, will attach to M a tape with contents $T'M)$, and will activate M.*

Self-reproduction now follows as a special case. See Figure 29. Put the following on the tape of the universal constructor M_c: a period, $D(M_c)$, a square. The unviersal constructor executes $D(M_c)$ and thereby constructs M_c, it next copies $D(M_c)$ onto the tape of the newly constructed M_c, and finally it activates the new M_c. We began with M_c and tape $D(M_c)$.

After the constructor is finished the cellular structure contains a second copy of M_c and tape $p(M_c)$. This is automaton self-reproduction. Hence an automaton which reproduces itself can be embedded in von Neumann's 29-state cellular automaton system. Iterated construction and iterated self-reproduction can be achieved by obvious modifications of the universal constructor.

This result is obviously substantial, but to express its real force we must formulate it in such a way that it cannot be trivialized. Consider, for example, a two state cellular system whose transition function takes a cell into state "one" when any of its neighbors is in state "one". Define an automaton to be any area, even a single cell. A cell in state "one" then "reproduces itself" trivially in its neighboring cells. Clearly, what is needed is a requirement that the self-reproducing automaton have some minimal complexity. This requirement can be formulated in a number of ways. We will do it by requiring that the self-reproducing automaton also be a Turing machine.

Since a universal Turing machine or computer can be embedded in von Neumann's cellular system (Fig. 26), such a machine can easily be combined with the universal constructor. When the description of this combined machine is placed on its own tape, followed by a square, it will produce a copy of itself. Hence, *there can be embedded in von Neumann's 29-state cellular automaton system a universal constructor-computer which is self-reproducing.*

We conclude that von Neumann's cellular automaton system is computation-universal, construction-universal, and self-reproductive. In this system, self-reproduction is a special case of construction, and construction and computation are similar activities.

11. KINEMATIC AND CELLULAR SELF-REPRODUCTION

A brief comparison of von Neumann's two kinds of self-reproducing automata may increase our understanding of them. Both are automata, being composed of computer or computer-like primitives. Both exist in environments made of the same elements as themselves, differing from their environment only in tha they are "organized", while the environment is not.

The most basic and important difference between the kinematic and the cellular automata systems is in the treatment of motion. The primitive elements of the kinematic system are capable of motion, while each finite automaton of the cellular system is fixed to its cell and cannot move.

[Myhill, in *"The Abstract Theory of Self-Reproduction"*, discusses a system which combines features of both.]
The processes which are achieved in the kinematic system by means of motion are achieved in the cellular system by transmitting information from cell to cell so as to realize the desired change of state. For example, in the kinematic system a finite automaton is constructed by collecting the needed parts, moving them into position, and soldering them together, while in the cellular system a finite automaton is constructed by sending out a communication channel (the constructing arm) and changing the state of a cellular area by means of signals sent through this channel. The difference is brought out by observing that in the kinematic system, self-reproduction results from (is a special case of) organized motion, whereas in the cellular system, motion is a special case of self-reproduction. To move an object in cellular space we "merely" make a copy of that object in the new location and then destroy the original.

As far as studies of logic and automata theory are concerned, the cellular system is superior to the kinematic system just because it does not include motion as a basic operation. Motion is not a proper object of study for logic, and as we noted in discussing von Neumann's passage from the kinematic to the cellular system, the motional aspects of the kinematic system complicate it from the point of view of logical analysis. Viewed abstractly, the processes of computation, self-reproduction, and action are changes in the state of a system. In both the kinematic and the cellular system we are only simulating these processes. Hence, the ability to represent states and changes in them is the important thing, not the type of entity (kinematic element versus cell) which possesses the state. Since the detailed analysis of self-reproduction is facilitated by the absence of motion from the cellular system, it is superior to the kinematic system for von Neumann's purposes.

One minor difference between the two systems should be noted. The kinematic system has a random or probabilistic feature, while the cellular system does not. But randomness does not play an essential role in the kinematic system. It is used only as a device for making all the parts accessible to a constructing automaton, and alternate schemes are possible. For example, the parts could be arranged systematically and the constructing automaton could move to the parts it needs.

[Von Neumann did recognize the importance of probabilistic automata and felt that an adequate theory of automata should cover them as well as deterministic automata. He discussed the question "How can reliable systems be constructed from unreliable components?" at length (see Burks, *"Toward a Theory of Atuomata Based on More Realistic Primitive Elements"*), and he hoped ultimately to study self-reproduction and evolution in a probabilistic cellular system (*Theory of Self-Reproducing Automata*, p. 99).]

The remaining difference between the two automaton systems concerns the number of primitive elements in each. The cellular system has a single primitive, the finite automaton located in each cell, whereas the kinematic system has nine kinds of primitive elements, three elements which switch ("and", "or", and "not"), and elements which delay, move (kinematic element), cut, join (fusing element), sense, and provide structure (rigid element). This difference is not fundamental, however, because conceptually one could combine all of these into a single element capable of all functions.

The various types of primitives of the kinematic system do correspond to functions to be performed in the system, and it is instructive to see how these functions are performed in the cellular system. From this point of view the nine primitives fall into four distinct classes.

First, the variable structures made possible by the rigid elements (girders) of the kinematic system are replaced by the underlying fixed structure of the whole cellular system.

Second, the switch and delay functions performed by the computing elements of the kinematic system are performed by the 29-state automata located in the cells of the cellular system; each of these automata could itself be synthesized from these elements. The context of a cell (that is, the states of its immediate neighbors) is essential in determining what switching and delay functions a 29-state automaton is performing. Within appropriate contexts, the following comparisons are valid. The confluent states C_{00}, C_{01}, C_{10}, C_{11} perform the functions of conjunction, double-delay, and branching. The ordinary transmission states $\rightarrow$, $\cdot\rightarrow$, $\uparrow$, $\cdot\uparrow$, $\leftarrow$, $\cdot\leftarrow$, $\downarrow$, $\cdot\downarrow$ perform the functions of transmission, disjunction, and unit delay; these states can recieve from and transmit to the comfluent states. The special transmission states $\Rightarrow$, $\cdot\Rightarrow$, $\Uparrow$, $\cdot\Uparrow$, $\Leftarrow$, $\cdot\Leftarrow$, $\Downarrow$, $\cdot\Downarrow$ also perform the functions of transmission, disjunction and delay, but while they can recieve from

confluent states they cannot transmit to them.

Von Neumann's kinematic and cellular computer elements have this in common: they are passive in the sense that they produce no output until stimulated. Consequently, negation can only be realized in the kinematic system if there is a source of pulses (a "clock") available, and negation is realized in the cellular system by means of the destructive ("killing") process. In other words, negation is not a primitive in either system and can only be synthesized by rather indirect means. Von Neumann wanted his computing primitives to have this characteristic so that the constructed automaton would remain quiescent during the construction process. He rec- ognized that a partly constructed automaton which was active could inter- fere with, and even prevent the completion of, the construction process. He thought there were interesting analogies here with the semantic paradoxes of mathmatical logic.

[See *Theory of Self-Reproducing Automata*, pp. 121-126.]

Third, let us consider how sensing is achieved in the cellular system. The content of a tape cell is sensed during the process of reading the tape (Sec. xy). What is actually sensed is whether the cell is in state $\underline{U}$ or in state $\downarrow$; this difference is, of course, sufficient for purposes of storing information in the cell. The method is to send a sequence 10101 around the reading loop and through the cell to be read (Fig. 24), and to note whether the output of the reading loop is 1 (in which case $x_n = \underline{U}$) or 10101 (in which case $x_n = \downarrow$). Not every state of a cell can be detected or sensed in the cellular system, but this makes no real difference to the computational or constructive powers of the system.

Fourth and last, let us consider how the functions performed by the kinematic, fusing, and cutting elements of the kinematic automaton system

are achieved in the cellular automaton system. In the kinematic system
these elements are used to construct automata from primitive elements,
modify automata (as when the contents of a tape are changed), and destroy
(dis-assemble) automata. These functions are simulated in the cellular
system by changing the states of the cells in a given area. This is
accomplished by sending out a constructing arm, sweeping the area with it,
and "writing" the desired state in each cell. Both the constructive process
(changing $\underline{U}$ into a passive confluent or passive transmission state) and the
destructive process (changing a confluent or transmission state back to $\underline{U}$)
play essential roles in this process.

12. HEURISTICS OF CELLULAR SPACES

So far we have considered only one particular cellular automaton
system, which von Neumann defined for the purpose of investigating some
organizational and programming aspects of self-reproduction (Secs. 3-10).
But the concept of a cellular automaton is very broad, allowing many basic
variations from von Neumann's system. We can change the geometry of the
space and the neighborhood relation; we can consider indeterministic and
probabilisitc transition functions as well as deterministic ones; and we can
even consider non-homogeneous neighborhood relations and transition functions.

The concept of a cellular automaton system is thus a very rich one.
Its essential features are a quantized time and space, a finite number of
possible states for each point of space-time, and a computable local trans-
ition function or law (not necessarily deterministic or uniform over space)
governing the operation of the system through time. Many natural systems
can be fruitfully approximated and simulated in a cellular framework. The
chief theoretical restrictions are two in number. First, the natural system
to be studied must be governed by a local law so that it can be represented

by a finite transition function. This precludes the use of cellular
automata for the simulation of Newtonian gravitational systems, for example,
because the law of gravitation includes action-at-a-distance where there is
no upper limit on the effective distance over which gravity acts. Second,
the behavior of the system to be simulated cannot depend on essential dis-
continuities, that is, discontinuities which cannot be approximated in a
discrete framework. However, there may be no such natural systems.

The most binding restriction on the use of cellular automata for the
study of natural processes is a practical one. In any interesting case
some or all of the following numerical parameters are large: the number of
possible states of a neighborhood, the number of different transition func-
tions to be investigated, and the number and size of the cellular automata
(i.e., histories) to be considered. The practical import of these numbers
depends on the extent to which sets of cases can be handled as a group, and
where simulation is involved, on one's abilities and techniques for sel-
ecting or generating the significant cases.

Cellular automata systems have been successfully used for the study
of natural systems. It has long been the practice to solve partial dif-
ferential equations for a vibrating membrane, or heat flow, or diffusion
processes, by handling them in a discrete grid, and since the laws in these
cases are local in character this amounts to using a cellular space, the
transition function being expresssd by the difference equation version
of the differential equations. Non-homogeneous cellular automata have been
used in the simulation ᴐf neural nets and of information processing by heart
tissue, and I think cellular automata systems will be very useful for
simulating evolutionary systems.

[Rochester, et. al., "Tests on a Cell Assembly Theory of the Action of the Brain."]

[Swain and Flanigan, "Computer Simulation of A-V Nodal Conduction."]

[Rosenberg, *Simulation of Genetic Populations with Biochemical Properties*, Sec. 3.3.]

Many questions about cellular automata systems can be answered by analytical methods. Von Neumann's results about the computation universality and construction univer ality of his 29-state cellular system were established in this way, as were the results of Moore and Myhill concerning the existence of configurations which can exist only initially (Moore, Machine Models of Self-Reproduction"; Myhill, "The Converse of Moore's Garden-of-Eden Theorem") . But many interesting questions about cellular systems cannot be answered analytically. This is especially so for cellular systems which are adequate for modeling natural systems. Here the sheer complexity of the situation and/or the indefinite nature of the problem demands simulation.

Simulation can be carried out in different ways and for different purposes. One type of problem is this. Given a cellular automaton system, and hence a transition function, to find particular cases (i.e., cellular automata) which have certain properties. Examples are found in the work of Ulam and his collaborators (eg. Ulam, "Some Mathematical Problems Concerned with Patterns of Growth of Figures"). It is generally necessary to run many cases before finding an interesting one, and sometimes even before acquiring sufficient experience to select initial states wisely. Such computer investigations are best carried out with the investigator in intimate contact with the machine, so that he can observe the effects of his decisions, and can terminate unpromising runs before they waste valuable machine time.

An even more difficult problem is that of searching for a transition
function which defines a cellular automaton system having certain properties.
One may seek a cellular system which simulates some given natural system;
for example, an evolutionary system, the fibrillation of the heart, individual
behavior, or group behavior. Von Neumann sought a cellular automaton system
with "not too many states" which was computationally and constructionally
universal. The problem of finding a transition function which generates
automata with certain properties is a very important and basic problem, so
we will discuss its general nature and then consider a promising method
for solving it by means of a man-machine complex.

With respect to the number of automata under consideration, the
problem of finding a transition function which satisfies certain conditions
is one level higher than the problem of finding a cellular automaton of a given
system which satisfies a certain condition. The latter problem concerns
the class of cellular automata defined by a single cellular automaton system,
while the former problem concerns a class of cellular automata systems.
Suppose we choose a cellular space, that is a geometry and neighborhood
relation. Each transition function defines a cellular automaton system
based on this space, and consisting of a set of cellular automata. Hence
when one considers all possible transition functions definable on a given
cellular space, he is considering not merely a set of cellular automata,
but a set of sets of cellular automata.

The problem of finding a transition function which generates certain
behaviors is analogous to the inductive problem of finding a law which
accounts for observed phenomena, just as a cellular automaton system is
analogous to (and therefore can be used to simulate) a natural system.
A look at this analogy will help us understand how cellular automata can
be used to simulate natural systems. Consider first some natural system

in sapce-time, e.g., the solar system. The laws of a natural system define
a set of possible universes, namely, all those universes which result from
applying these laws to some state of the system at some arbitrary time.

[These universes are causally possible universes, as distinguished,
for example, from logically possible universes. The logic of statements
about causally possible and causally necessary universes is treated in
my "The Logic of Causal Propositions."]

Consider second a cellular automaton system. Its underlying cellular space
corresponds to the space-time basis of a natural system. Its transition
function corresponds to the law (or set of laws) of a natural system. Similarly,
its transition function defines a set of possible universes, namely, all
the cellular automata which result from applying the transition function
to an initial state of the system.

[If the transition function is deterministic, each initial state defines
a single cellular automaton. If the transition function is indeterministic,
each initial state defines a set (possibly infinite) of cellular automata.
If the transition function is probabilistic, there is a succession of
poobability distributions over the succession of finite histories of
cellular automata resulting from an initial state.]

A cellular automaton is thus a possible world, and a cellular automaton
system is a set of possible worlds.

We can now see more clearly the analogy between the automata theorists'
problem of starting with a given type of cellular automaton behavior (e.g.,
self-reproduction, evolution) and ending with a cellular automaton system
in which this behavior occurs, and the natural scientists' problem of
beginning with certain observed phenomenon and ending with a law which
accounts for that phenomenon. Both begin with some phenomenon or behavior
and end with a law or transition function such that a system governed by

that law or transition function displays that phenomenon or behavior. The scientist may proceed by considering different possible laws, inferring consequences from each, and checking the consequences against the given observed phenomenon. Analogously, the automaton theorist may proceed by considering different transition functions and ascertaining what automaton behaviors each generates. This may be done analytically (as von Neumann did in his design of a self-reproducing automaton) or computationally (by simulating cellular automata which have these transition functions). The analogy between the natural scientists' procedure for discovering and confirming laws to explain observed phenomena and the automata theorists' procedure for discovering transition functions and showing that they produce certain behavior becomes an identity when cellular automata are used to model or simulate natural systems.

[In my forthcoming book, *Cause, Chance, and Reason,* I will give an analysis of inductive inference in terms of a probabilistic choice, from among alternative possible laws, of a law to account for observed phenomena.]

This concludes our general remarks about the problem of finding a transition function satisfying certain behavioral conditions. We will explain next an interactive man-machine method for solving certain problems of this sort which has already been used successfully and which is a promising tool for this type of basic investigation. The general principle is this. The investigator partly defines a transition function for his cellular space. He then specifies an initial automaton state and has the computer generate a finite fragment of the resultant cellular automaton in an attempt to produce one of the desired phenomena. He repeats this step until it succeeds or until he decides it is not promising. In the latter case he tries an alternate partial definition. If the step succeeds, he augments the partial definition further in an attempt to produce other

desired phenomena, and repeats the step.

The best way to explain this heuristic method in detail is to describe a case in which it was successful.

The problem was that of improving on von Neumann's cellular automaton results with respect to the number of states needed for computation and construction universality, so let us first formulate von Neumann's problem in a general way. After considering various kinds of cellular frameworks, von Neumann decided on a cellular space consisting of a two-dimensional infinite array of square cells (a "checkerboard") and a "neighborhood" composed of a cell and the four cells with which it shares boundaries.

[*Theory of Self-Reproducing Automata*, Part II, Ch. 1 and Sec. 2.1] His problem was then to define a transition function (and thus a cellular automaton system) satisfying the following four conditions.

(1) The system is homogeneous in the sense that the neighborhood relation and the transition function are the same for every cell. This restriction prevents one from building a specific automaton design into the cellular space by varying the transition function from cell to cell.

(2) A universal computing (Turing) machine can be embedded in the cellular system. This machine will occupy a finite number of cells initially, but will have the capability of extending itself indefinitely so as to have unlimited storage capacity.

(3) The system is construction-universal in the following sense. Any finite number of designated cells in arbitrarily assigned states constitutes an (initially) finite machine. The construction-universality requirement is that there exist a class M of finite automata which contains a universal Turing machine M_u and also a machine M_c with this ability: for any machine M belonging to M, when M_c is supplied with a

description of M and placed in an environment of blank states, M_c will construct M somewhere in this environment. The machine M_c is a universal constructing machine. In Section 10 we took M to be the class of initially quiescent automata.

(4) The number of states in each cell is "relatively small". The problem just stated is clearly not a formal one, but depends on the interpretation of such concepts as "embedding" and "relatively small". This is so even though a solution to the problem can be stated formally, as we stated von Neumann's own solution to the problem (Secs. 9, 10). The reason for the fourth requirement, that the number of states be relatively small, is that without this requirement the problem is easily solved. For we could place a general-purpose computer (or the finite part of a universal Turing machine) in each cell, and arrange that each such computer could communicate with its four neighbors. It is easy to see that any infinite row (or equivalent) of cells constitutes a universal computing machine, with one cell serving as the finite automaton part and the cells to its left and right serving as an infinite tape. Moreover, universal construction and self-reproduction would not be hard to achieve in such a cellular automaton system.

Von Neumann approached and solved this problem analytically. The primitive elements he devised were based on those he used in the logical design of the EDVAC, the first stored program electronic computer.

[See *Theory of Self-Reproducing Automata*, pp. 9, 44, 99, 157. Essentially the same primitives were used in his kinematic automata; See Sec. 1 above.] We will briefly describe these primitives, and then show how von Neumann very cleverly adapted them to his cellular space so as to achieve a construction-universal and computation-universal cellular automaton system with a relatively small number (29) of states.

The fundamental units of the design of the EDVAC were switch-delay elements with one to three excitatory inputs, possibly one or two inhibitory inputs, and a threshold number 1, 2, or 3. An element emits a stimulus (pulse) at time $t + 1$ if and only if two conditions are satisfied at time t: no inhibitory input is stimulated, and the number of excitatory inputs stimulated is at least as great as the threshold number. No stimulus is emitted at time zero, but there is in addition a "clock" element which emits a pulse at every moment of time.

To adapt this set of primitives to the two-dimensional cellular framework, it was necessary to provide for wire-branching and wire-crossing, either directly or indirectly. In his cellular systen von Neumann accomplished wire-branching directly, by making this one of the time functions of the confluent element ($\underline{C}$). In contrast, he accomplished wire-crossing indirectly, by synthesizing a coded channel from the primitives he did adopt (Fig. 7). Note that all of the EDVAC primitives are initially quiescent, except for the clock. To synthesize a clock in the cellular system one establishes a cellular path (like the repeater in Fig. 12a), and stipulates that the elements of the path be active initially.

The adaptation of the EDVAC logical design primitives to a cellular framework, described so far, is fairly straightforward. But to make a cell capable of becoming each of these primitives would require a very large number of states, since each primitive itself has many possible input and output directions. This large number of states would then require a large number of intermediate states for construction (von Neumann's sensitized states $\underline{S}_\theta$, $\underline{S}0$, $\underline{S}_1$ ...). Further states would be required for destruction. What is really ingenious is the way von Neumann was able to realize all the EDVAC functions with a transition function having only

29 states. His main method for accomplishing this reduction was to exploit the context of a cell. The data needed to specify the input and output directions of an element are minimized by specifying only the output direction within the cell (and this only for transmission states). The input directions for an element in a cell are determined by the states of the neighboring cells. In the end only twelve states are required for conjunction, disjunction, and wire-branching. In addition, von Neumann made good use of the duality of ordinary and special transmission states to realize a destructive capability, as well as negation, with only eight additional states.

The transition function for a cell is a finite table, giving for each state of the cell snd its neighbors the state of the next cell. If the different local behaviors to be realized have some degree of logical independence, the function may be defined piecemeal, by partially defining it (i.e., defining it for certain cases) to get one behavior, further defining it to get another behavior, etc., etc. In this way a sequence of stronger partial transition functions is defined, allowing more and more types of behavior. Now suppose at a certain stage a number of desired behaviors are possible but a further desired behavior is excluded. One then goes back in the sequence and attempts to modify the partial definitions so as to preserve the achieved desired behaviors and also obtain the further desired behavior.

Note that a partially defined transition function is actually an indeterministic transition function, allowing many possible succeeding states for the unspecified cases. Hence the procedure just described is one of defining stronger and stronger indeterministic transition functions, until one finally obtains a transition function, either deterministic or indeterministic, which achieves the desired behavior.

This is the procedure von Neumann actually followed in defining his transition function (see Fig. 15).

[*Theory of Self-Reproducing Automata,* Part II, Ch. 2.]

He first defined the ordinary transmission states, out of which one can build disjunctive switches and signal transmission paths which can go straight or turn corners. He next defined the confluent states; from these and the ordinary transmission states one can construct conjunctive switches and wire-branching devices. At this stage in his synthesis the confluent states had a single unit of delay. He then added the special transmission states to achieve the destructive process: active special transmission states kill both ordinary transmission and confluent states back to U, and active ordinary transmission states kill special transmission states back to U. He decided to realize the constructive process by means of a sequence of stimuli sent into a cell in state U from a transmission state; this required "sensitized" states to store the initial segments of the sequence (see Fig. 9). Now the natural way to generate a sequence of stimuli is to take a single stimulus, send it along alternative paths with different delays, and then merge these paths (as in the pulser of Fig. 5). But this could not be done with the transition function as it was defined. For consider the delays along two different paths from one cell to another, where the paths are composed of transmission and confluent states. The difference in cell length of these paths must be a multiple of two, and since both transmission and confluent states had (at this point of the definition) a unit delay, the time delays along these two paths must differ by a multiple of two. Hence one could not produce, by this natural technique and in the system as defined, an arbitrary finite pulse sequence $i_1 \, i_2 \, \ldots \, i_N$ (each $i_n = 0, 1.$) To remedy this defect, von Neumann went back and modified his transition function by giving confluent states a

double delay.

[*Theory of Self-Reproducing Automata*, pp. 146-148 and Fig. 8]

In this way von Neumann defined a transition function with a relatively small set of states (29) which was nevertheless sufficient to achieve a rather powerful set of elementary operations or behaviors: the logical operations of "or" and of "and", transmission, wire-branching, arbitrary relative delays down alternate paths, construction, and destruction. Using these operations, he proceeded to synthesize various organs (pulser, decoder, coded channel, periodic pulser, constructing arm head) and higher-level functions (sensing, negation). With these he synthesized still higher level organs (control unit, constructing arm, tape, constructing unit), and finally a universal computer and a universal constructor.

In the last few pages we have formulated von Neumann's problem as he might have viewed it at the start, and indicated his general method of solving it. Using this method, von Neumann found a solution with 29 states per cell. The problem arises: can this number be reduced appreciably?

This problem was taken by Edgar Codd as his doctoral problem, and he solved it very successfully.

[Edgar F. Codd, *Propagation, Computation, and Construction in Two-Dimensional Cellular Spaces*. Ph.D. Thesis in Communication Sciences, University of Michigan, 1965. This has been published as a book by Academic Press (1968) under the title *Cellular Automata*.]
He was able to devise an eight-state cellular system which is both computational-universal and construction-universal. Moreover, this system is stronger than von Neumann's in two additional respects. The transition-function has a stronger kind of rotation-symmetry than does von Neumann's. Also, there are two states such that any finite configuration (area) of

these two states can be read and erased as well as written (constructed).
This is possible because that state of a cell which is in one of these two
states can be sensed by means of an "echo signal", which reflects (bounds
back) from the cell, the nature of the reflected signal depending on the
state of the cell. In contrast, to sense whether a cell in von Neumann's
system is in state $\underset{\textbf{M}}{\text{U}}$ or state $\downarrow$, we send a signal *through* the cell (see
Fig. 24). While it would be very hard to prove, it seems doubtful that
there are two states of von Neumann's system such that any finite area of
these two states could be read or sensed from within the system.

Codd's cellular automaton system is primarily of interest to us here
because it was generated by what we have called "the interactive man-machine
method."

[He also used this method to search, unsuccessfully, for universal
systems with eight neighbors per cell but only two or three states.]
As von Neumann did, Codd chose as sub-goals certain elementary behavioral
functions, which he thought he could later synthesize into organs, larger
units, and finally universal computers and constructors. He then proceeded
to define his transmission function piecemeal so as to obtain these
behaviors, retreating when a partial definition turned out to have undes-
irable consequences, and either modifying the definition as it had been
specified at an earlier stage or seeking alternative behavior (sub-goals)
to realize the final goal. But whereas von Neumann proceeded analytically,
using only his own reasoning and testing a few cases by hand, Codd used
a computing machine to assist him.

The procedure was as follows. The user first gives the computer the
neighborhood relation, the number of states per cell, and the symmetry
condition to be imposed on the final transition function. The user then
gives the computer a partial transition function by means of which he hopes

to generate a certain kind of behavior; the computer extends the partial

transition function in accordance with the symmetry condition, or lets the

user know that this extension is impossible so he can modify his definition.

Next, the user gives the computer an initial state he hopes will lead to a

certain phenomenon. The computer calculates the behavior of the resultant

cellular automaton time-step by time-step, printing out the result at each

time-step, stopping at each time-step to allow the user to make a choice,

and stopping when it comes on a condition not covered by the partial

transition function. When the latter happens, the user may further specify

the transition function. At any time-step the user can go back to earlier

time-steps and change a partial definition, or go back to the beginning

and start with a different initial state and/or partial definition. The

user makes these choices on the basis of the simulated behavior he observes.

There are obvious advantages to such simulations. The computer can

make routine calculations rapidly as well as accurately, it can check

constraints automatically, and it can assume responsibility for storing and

arranging large amounts of data in a way that makes vital information

immediately available to the user when he needs it. These advantages lead

in turn to deeper advantages. For example, a human generally finds it

difficult to trace out the successive states of an automaton when the

transition function is basically different from those functions he is

familiar with. A machine has no such difficulties, since it calculates

by brute force and not in terms of certain intuitions derived from ex-

perience. Hence by using a machine one can quickly explore the consequences

of an unfamiliar rule.

In the process of defining his 29-state transition function, von Neumann

started with the usual computer primitives (switches and delays), and adapted

them to the cellular context so he could realize not only the functions
essential to computing per se, but also the functions needed to carry out
computation (and construction as well) in a cellular space. The most basic
of these latter functions are: sensing the state of a cell, as in reading
a tape (Fig. 24); changing the state of a cell -- this includes construction
and destruction processes, as well as logical and transmission processes;
and operating a signal transmission path from a source -- this includes
extending, retracting, and carrying out operations at the terminus of the
path (see the constructing arm, Figs. 18-23). With a machine to test each
possible transition function by simulation, Codd could and did aim directly
at achieving the basic functions needed for computation and construction
in a cellular space. Thus whereas von Neumann took disjunction and
conjunction as primitives and synthesized negation, Codd was able to
synthesize all switching functions from operations which, in a cellular
framework, are more fundamental. After conducting his investiagtion with the
assistance of a computer, he achieved an eight-state universal cellular
system which has a number of novel and interesting features, and his success
was due in part to the assistance he recieved from the computer. The most
striking case of this assistance was the discovery of the echo phenomenon.
The technique of reading (sensing) the state of a cell by reflecting an
echo signal from that cell was not designed or planned. Rather, the phenomenon
of an echo signal was accidentally generated in the course of a simulation
which had a different objective, and then recognized by the experimenter
on the basis of a possible sensing technique.

A brief summary of the eight states and their chief functions will
bring out some of the novel features of Codd's cellular automaton system.

[The system he defined is indeterministic, since he only specified
the transition function up to the point where universal computation and

universal construction are possible.]

Four of the states (0, 1, 2, 3) mainly perform structural functions, while four (4, 5, 6, 7) are used for signalling. State zero (0) is the blank state. State one (1) is used to mark out a bi-directional transmission path through the cellular space. Such a path is ready to conduct signals when it is insulated or "sheathed" by neighbors which are in the sheathing state (2). State three (3) is used to perform gating functions. Two of the signal states (4, 5) are used for information transmission and processing. The direction of a signal down a transmission line is determined by means of a blank state (0); for example, "04" moves to the right, "40" moves to the left. The signal state six (6) is used to cause a sheath to be formed on the two sides of a transmission line of ones; compare the myelin sheath surrounding a neuron. The signal state seven (7) is used to activate an automaton by setting up certain gates and starting the automaton. Thus the construction of an automaton takes place in three stages. First, the basic structure is laid down in an area by changing certain cells from state zero to state one. Second, the transmission paths are sheathed by the sheathing signal. Finally, the constructed automaton is activated by the activating signal.

We said earlier that cellular automaton systems are not only of interest in their own right, but provide a framework for investigating many natural systems, including the heart, nerve nets, and evolutionary systems. It therefore seemed worthwhile to develop an interactive man-machine system especially suited to this purpose, and some of us are now doing this.

[This is a project of the Logic of Computers Group, Department of Computer and Communication Sciences, University of Michigan. It is under

the direction of John Holland, Ronald Brender, and myself. The work is
supported by the National Institutes of Health and the Advanced Research
Projects Agency of the Department of Defense.]

We have two small computing machines, one connected to a disc file, another
to an oscilloscope unit with a light pen, and both interconnected via an
interface. The first machine stores the information concerning the cellular auto-
maton under investigation: the cellular geometry, the neighborhood relation,
the transition function as so far defined, the initial state under con-
struction, and the history of the automaton as so far computed. This history
will be stored on the disc file. The second machine displays either the
transition function or a selected portion of the history, as requested by
the investigator. This machine also receives the investigator's input
to the process via light pen or typewriter. The investigator may insert
a new initial state or modify the transition function. He may ask to see
a different part of the history or to see the present state in more detail,
or he may request the machine to return to an earlier point in history
and compute the subsequent history with a modified transition function.

An essential part of this interactive man-machine combine is a
a software system that permits the investigator to easily control and direct
the computation, so he can devote most of his intellectual energies to
guiding the investigation: making decisions, looking for interesting
phenomena, and thinking up suggestions and hypotheses. This software
system will consist of data structures, programs, and languages especially
designed for convenient man-machine communication during the investigative
process. When completed, the software system will allow easy and sepatate
specification of the cell geometry, the neighborhood relation, the trans-
ition function, and initial states, and will provide ample options for
control.

Our aim is to optimize the division of labor between man and machine for this particular type of research, by arranging for each to do those tasks he is best suited for. If we have a reasonable measure of success, the resultant interactive man-machine complex should be a very powerful instrument for the investigation of cellular automata systems.
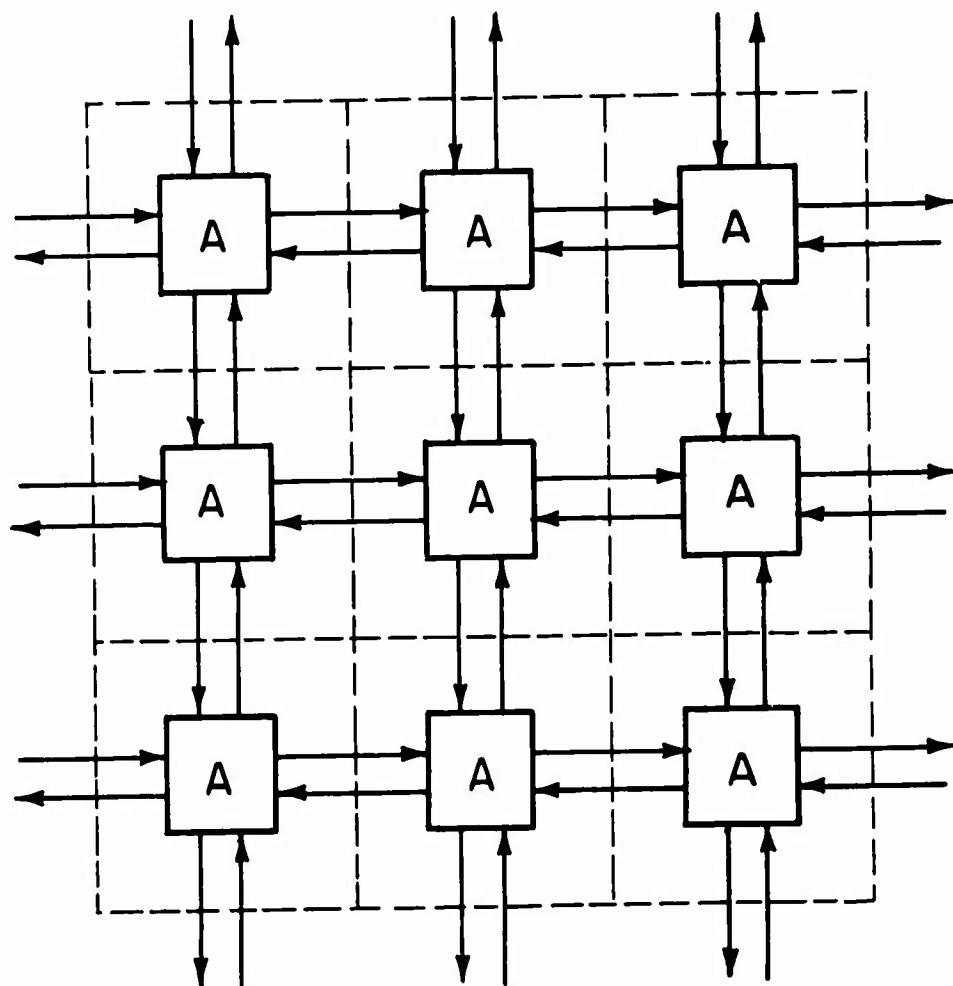
Fig.1   CELLULAR   SPACE
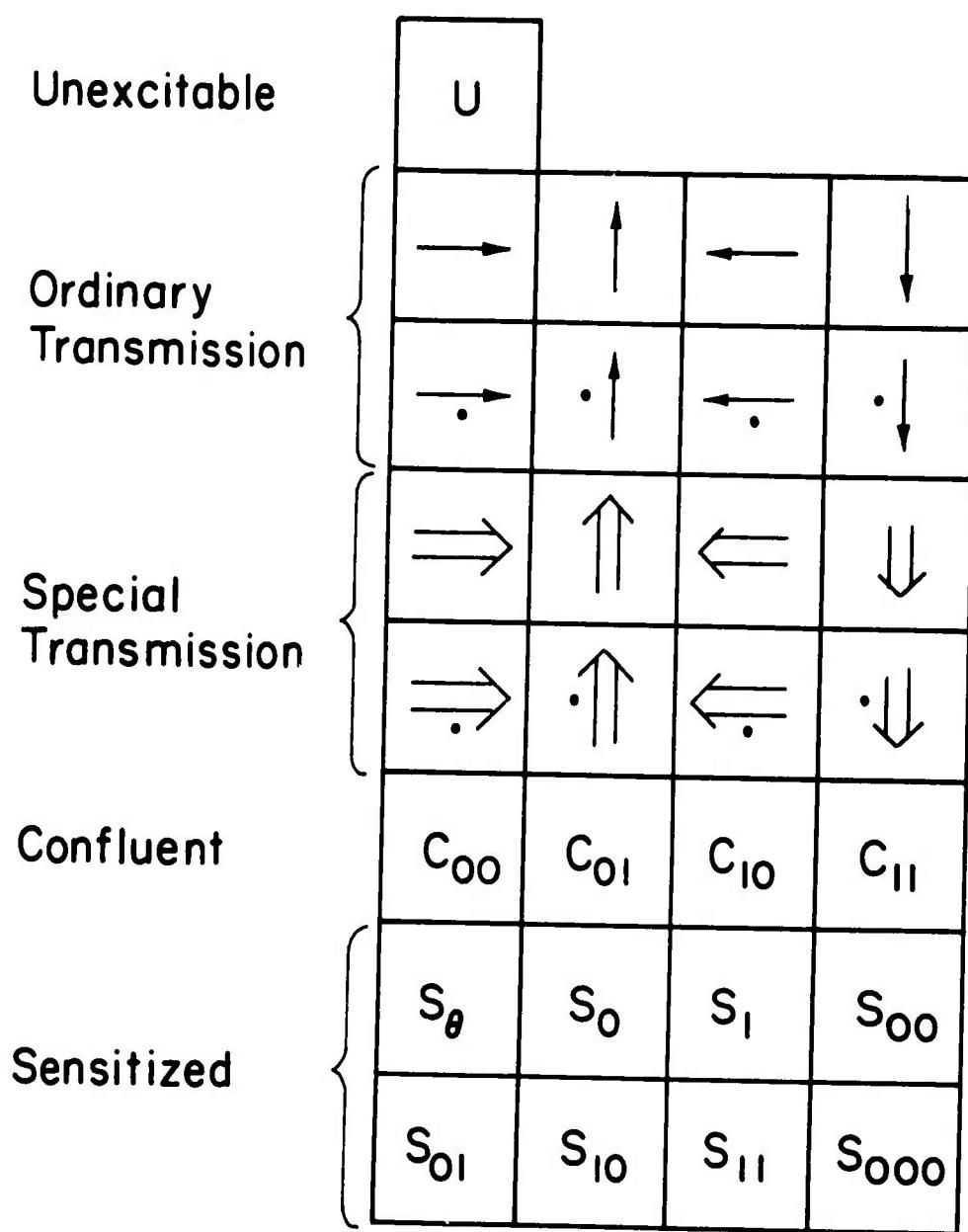
Fig. 2 VON NEUMANN'S 29 STATES
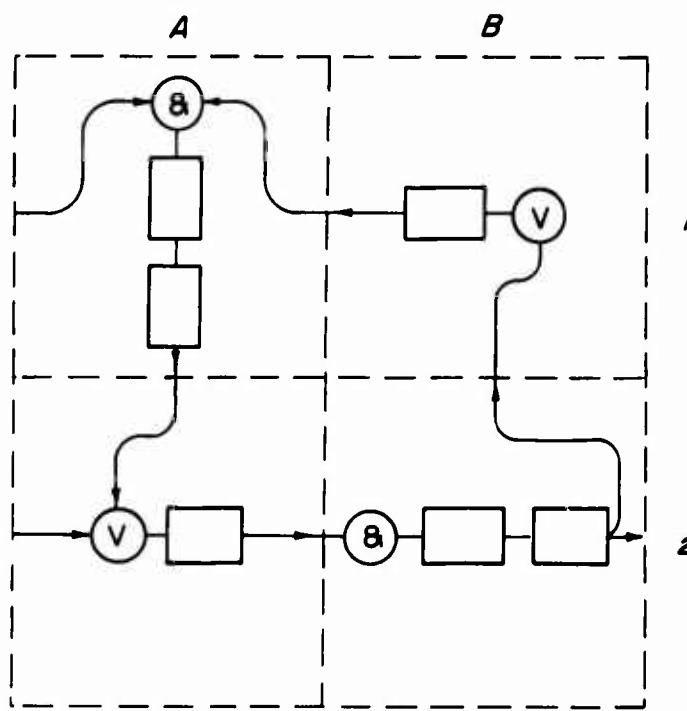
Fig. 3 (a) SWITCH AND DELAY CIRCUITS
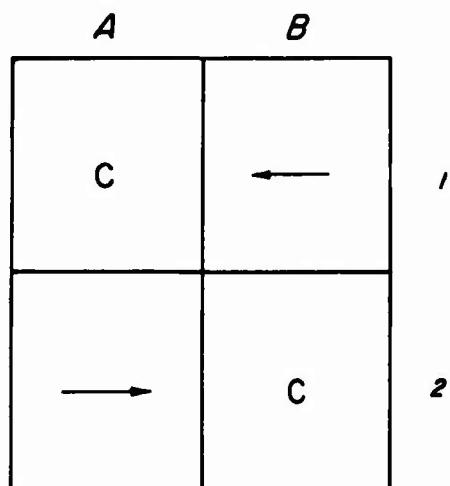
Fig.3(b) CORRESPONDING CONFLUENT AND
ORDINARY TRANSMISSION ELEMENTS

SWITCH AND DELAY REALIZATION OF
CONFLUENT AND ORDINARY TRANS-
MISSION ELEMENTS
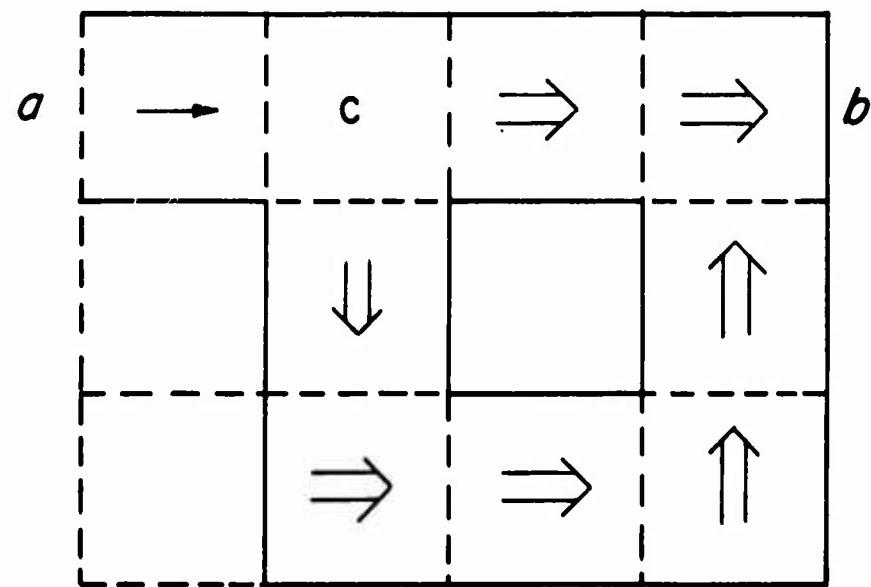
Fig.4 OPERATION OF SPECIAL
TRANSMISSION STATES
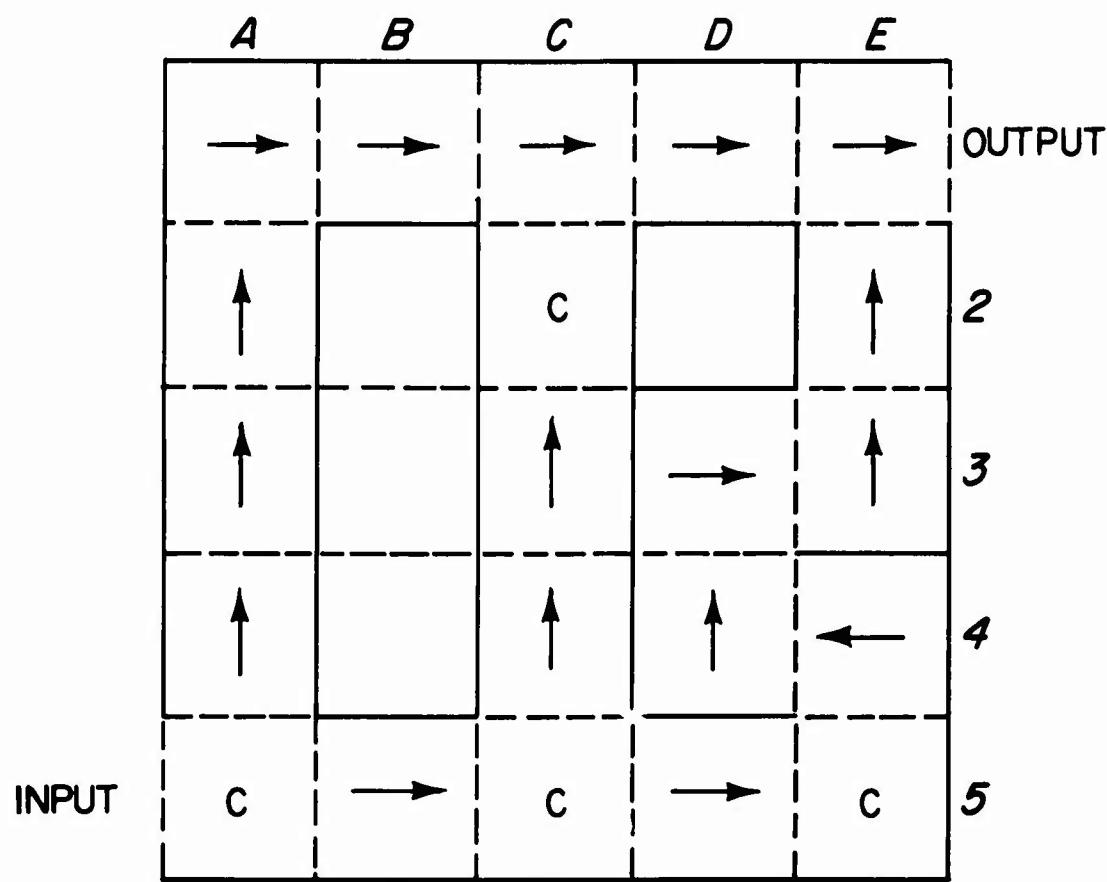
Fig. 5   PULSER **P** (IOIOI)
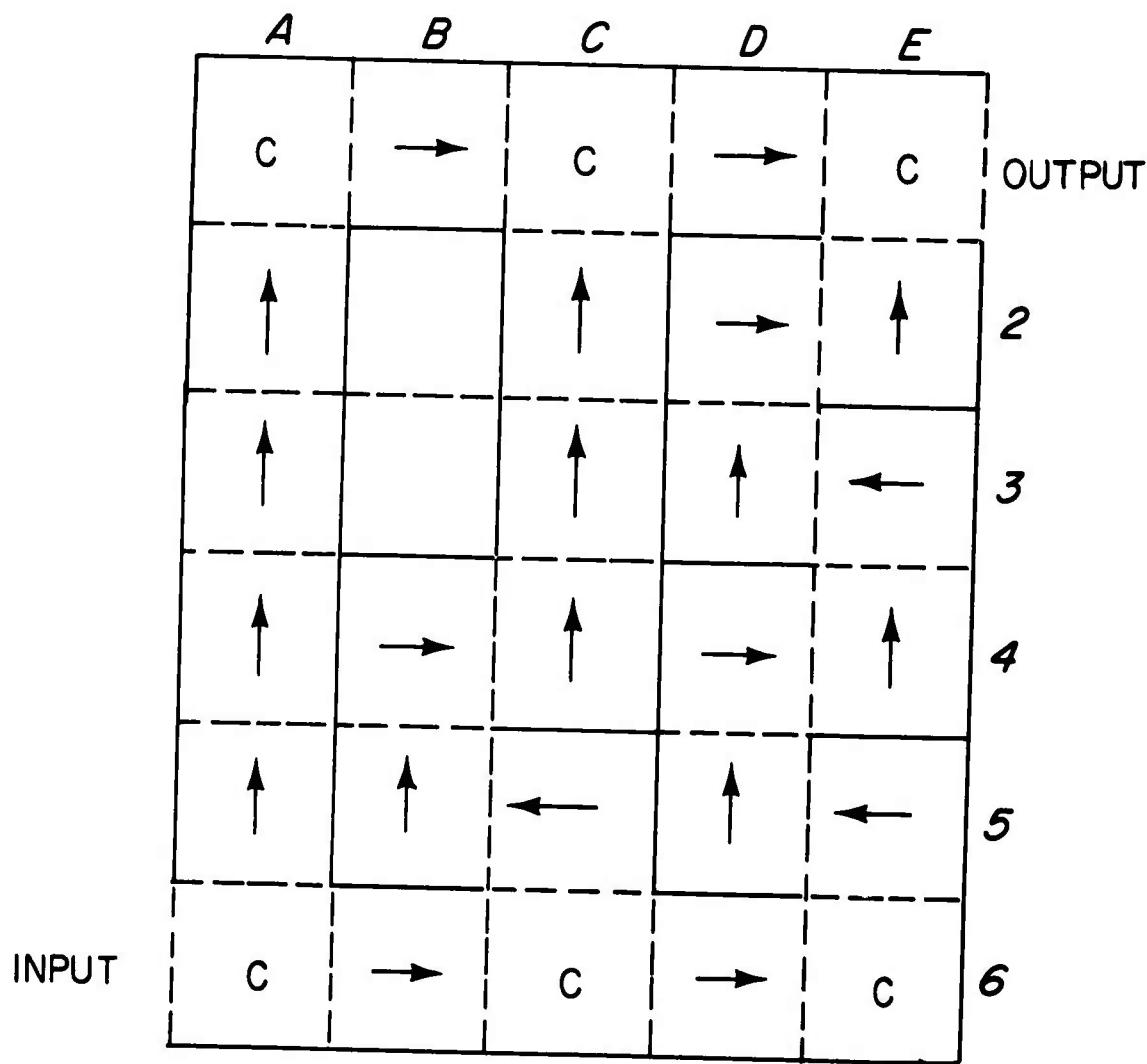
Fig. 6   DECODER **D** (IOIOI)
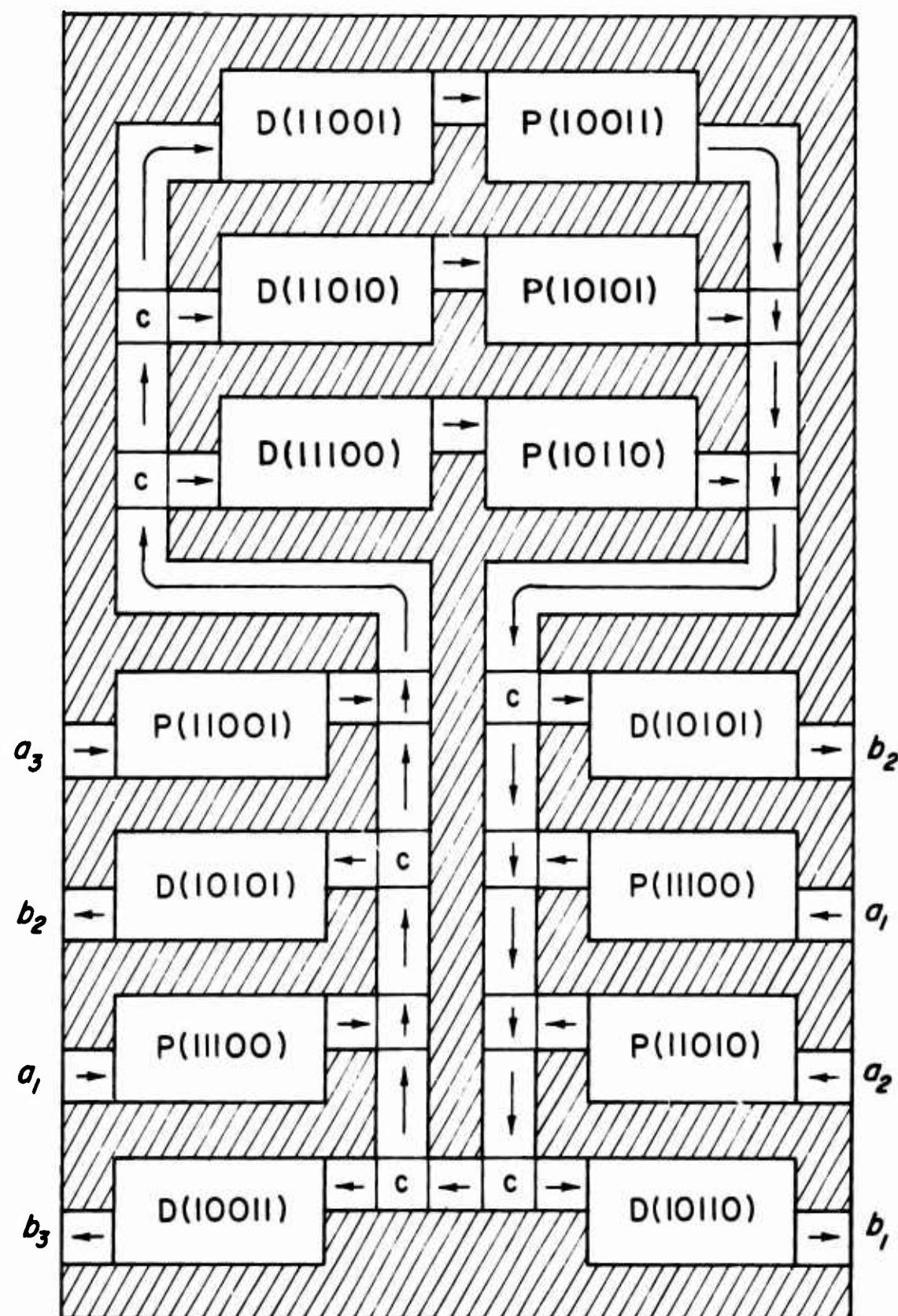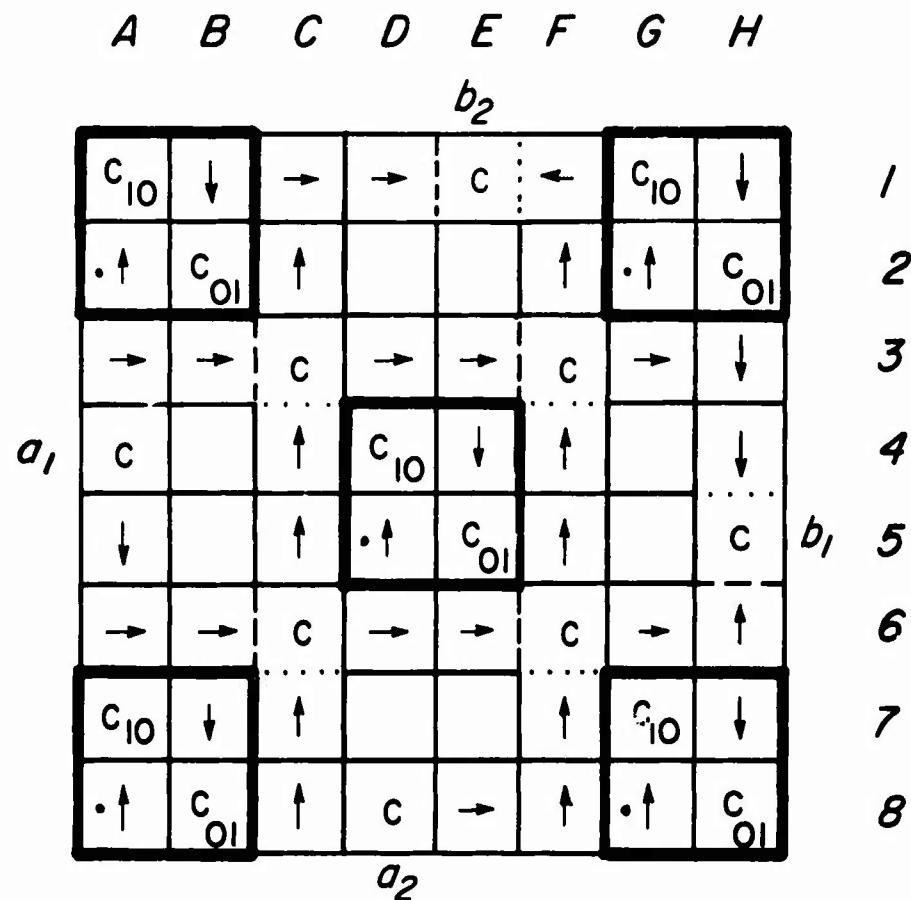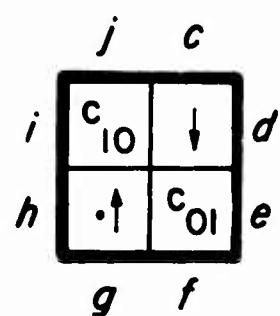
Fig.7 A CODED CHANNEL

(a) CROSSING ORGAN

Output from i and j : 101010
Output from e and f : 010101

(b) INITIAL STATE OF CLOCK

Fig.8 CROSSING ORGAN
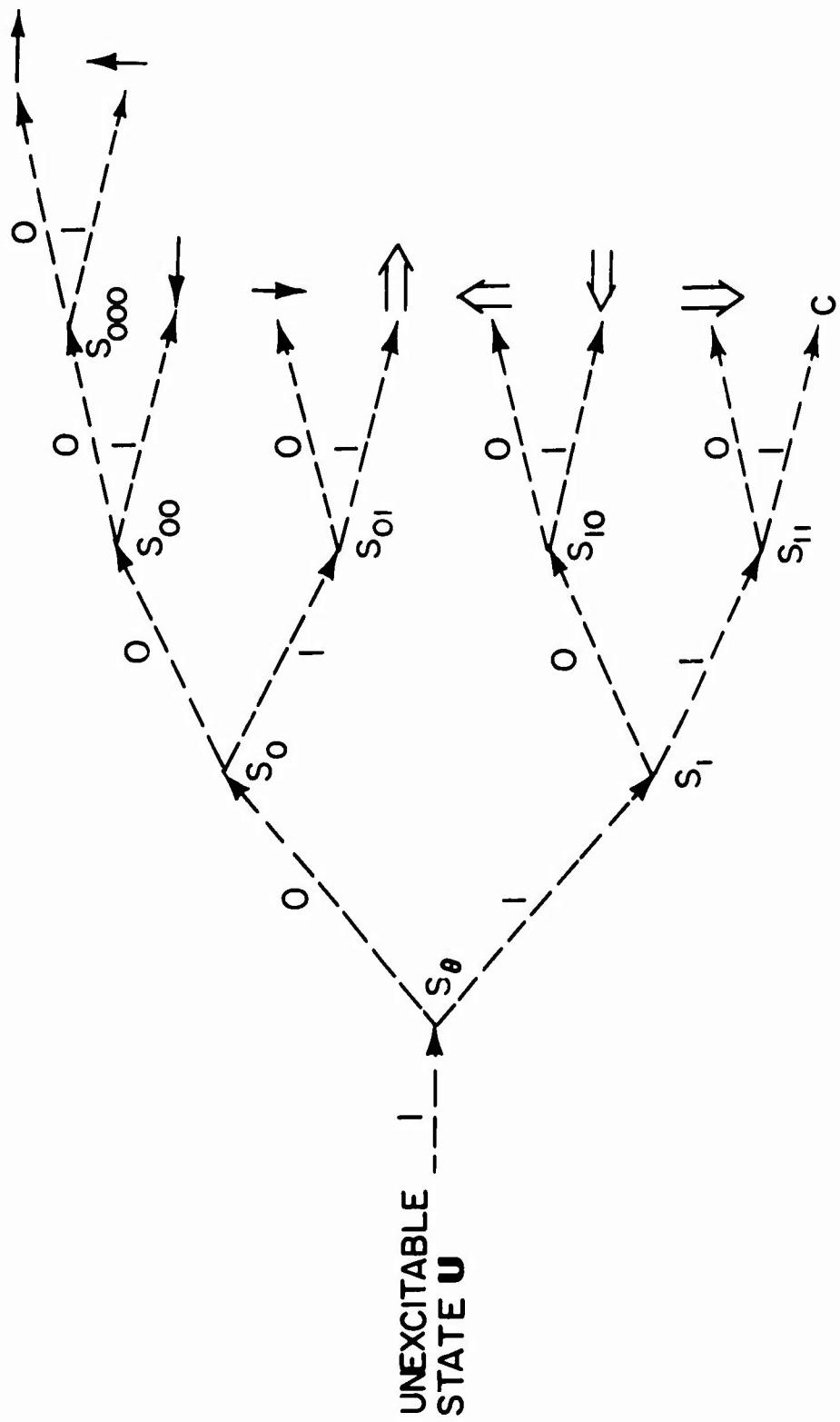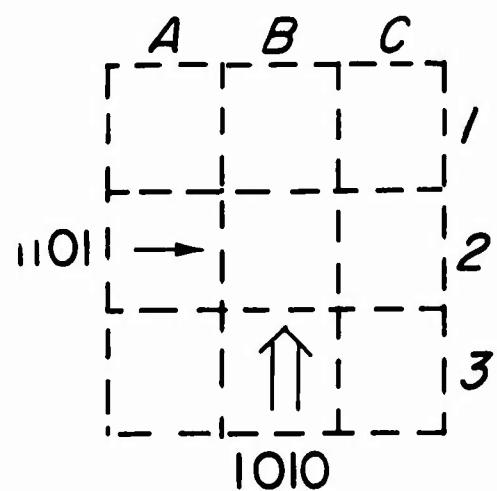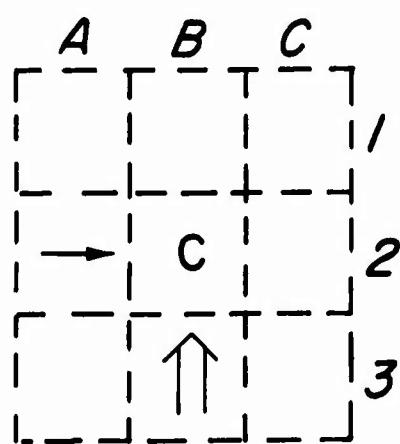
Fig. 9 CONSTRUCTION PROCESS

(a) BEGINNING

(b) END

Fig.10 EXAMPLE OF THE
        CONSTRUCTION PROCESS
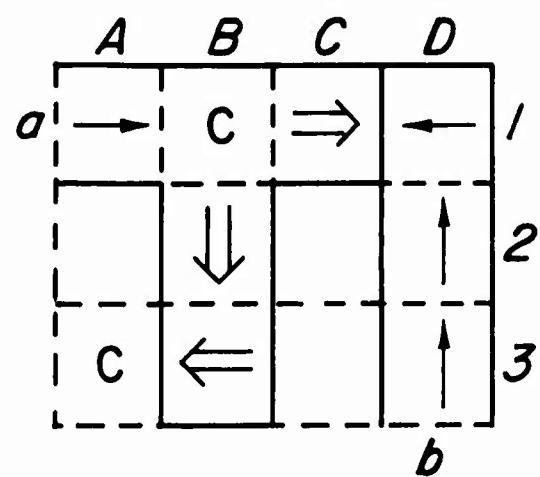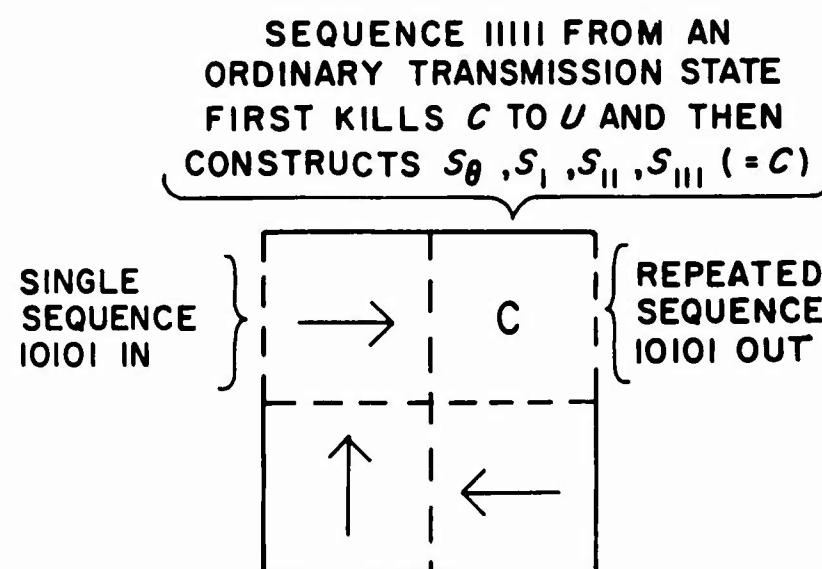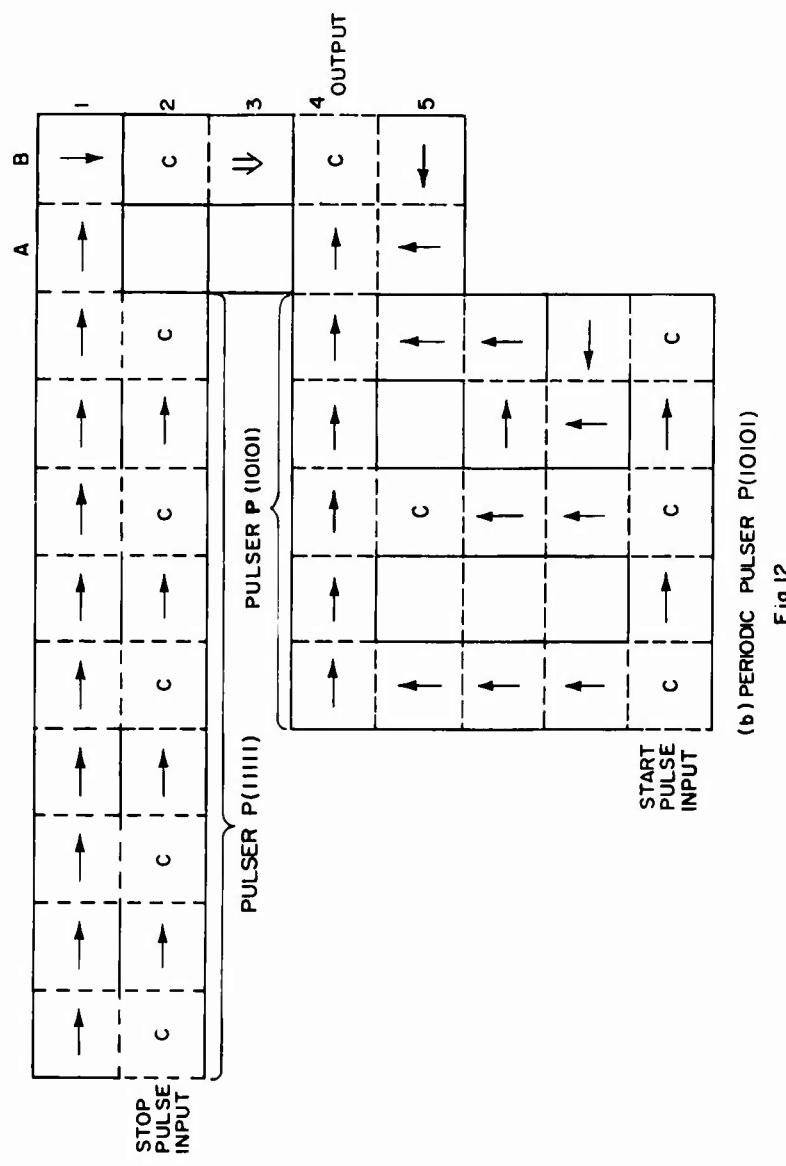
Fig.II EXAMPLE OF THE
      DESTRUCTION PROCESS

SEQUENCE IIIII FROM AN
ORDINARY TRANSMISSION STATE
FIRST KILLS C TO U AND THEN
CONSTRUCTS S_θ , S_1 , S_{11} , S_{111} $(= C)$

SINGLE
SEQUENCE
IOIOI IN

C

REPEATED
SEQUENCE
IOIOI OUT

(a) REPEATER FOR PERIODIC PULSER

Fig.12 PERIODIC PULSER **PP**(IOIOI)

(b) PERIODIC PULSER P(IOIOI)

Fig.12

79
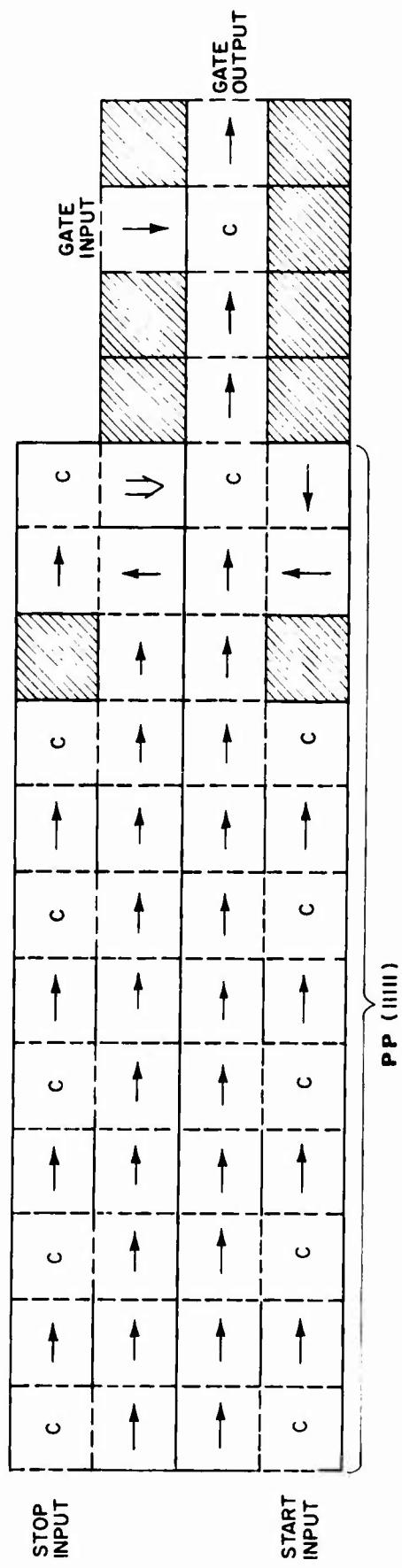
Fig. 13 FLOP-FLOP AND GATE
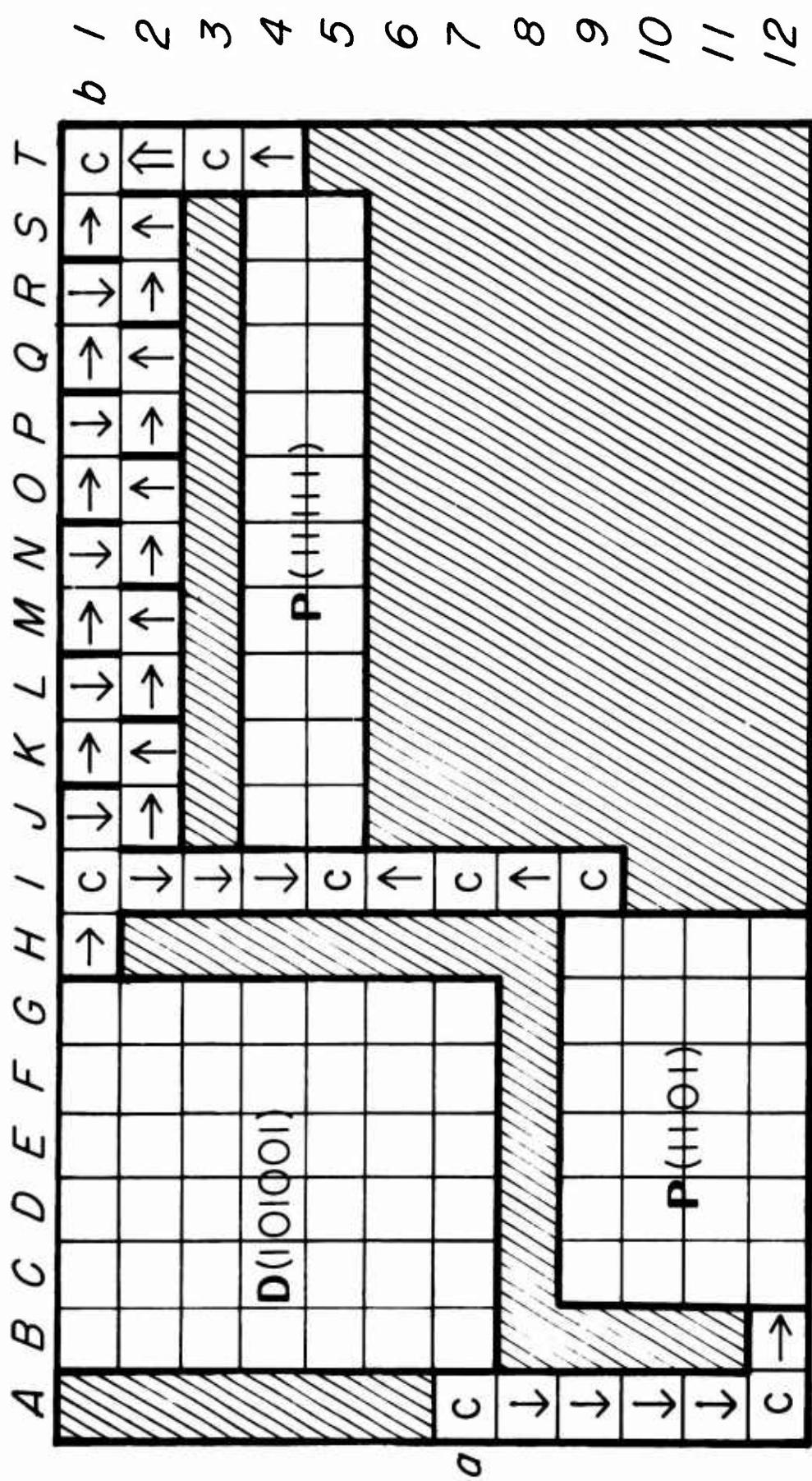
RECOGNIZER **R**(101001)

Fig. 14

Class	Name	List of states — Passive	List of states — Active	Number	Summary of transition function
Ordinary	Unexcitable	U or blank		1	Construction process changes U into sequence of sensitized states and then into passive forms of confluent and transmission states. Destruction process changes any confluent or transmission state into U in one time step.
	Confluent	C_{00}	C_{01}, C_{10}, C_{11}	4	Receives conjunctively from any ordinary transmission state directed toward it; emits with double delay to all transmission states not directed toward it. In $C_{\alpha\beta}$, α is the present output, β is the next output. Killed to U in one time step by any active special transmission state directed toward it; killing dominates reception.
	Transmission	↑, ←, ↓, →	↑•, •←, ↓•, →•	8	Receives disjunctively from ordinary transmission states directed toward it and from confluent states; emits in output direction with single delay (a) to every ordinary transmission state not directed toward it and to confluent states (b) to U or sensitized states for construction process (c) to kill any special transmission state by destruction process. Killed to U in one time step by any active special transmission state directed toward it; killing dominates reception.
	Transmission	=>, <=, <=, =>	=>•, •<=, <=•, •=>	8	Receives disjunctively from special transmission states directed toward it and from confluent states; emits in output direction with single delay (a) to every special transmission not directed toward it (b) to U or sensitized states for construction process (c) to kill any ordinary transmission state or confluent state by destruction process. Killed to U in one time step by any active ordinary transmission state directed toward it; killing dominates reception.
Special	Sensitized	S_{θ}; S_0, S_1; S_{00}, S_{01}, S_{10}, S_{11}; S_{000}		8	These are intermediary states in the construction process. Any active transmission state directed to U changes it to S_{θ}. Thereafter, S_{α} is followed by the cell (a) $S_{\alpha 1}$ if some active transmission state is directed toward the cell (b) $S_{\alpha 0}$ otherwise, until the destruction process terminates in one of the states $S_{0000}(\rightarrow)$, $S_{0001}(\uparrow)$, $S_{001}(\leftarrow)$, $S_{010}(\downarrow)$, $S_{011}(\Rightarrow)$, $S_{100}(\Uparrow)$, $S_{101}(\Leftarrow)$, $S_{110}(\Downarrow)$, $S_{111}(C_{00})$.

Von Neumann's transition function

Figure 15

PRESENT INTERNAL STATE	INPUT STATE	NEXT INTERNAL STATE	OUTPUT STATE
d_0	i_0	d_0	o_0
d_0	i_1	d_1	o_1
d_1	i_0	d_1	o_2
d_1	i_1	d_0	o_0

(a) Transition and output table for a
particular finite automaton

Fig. 16
SCHEMATIC DIAGRAM OF FINITE CELLULAR AUTOMATON
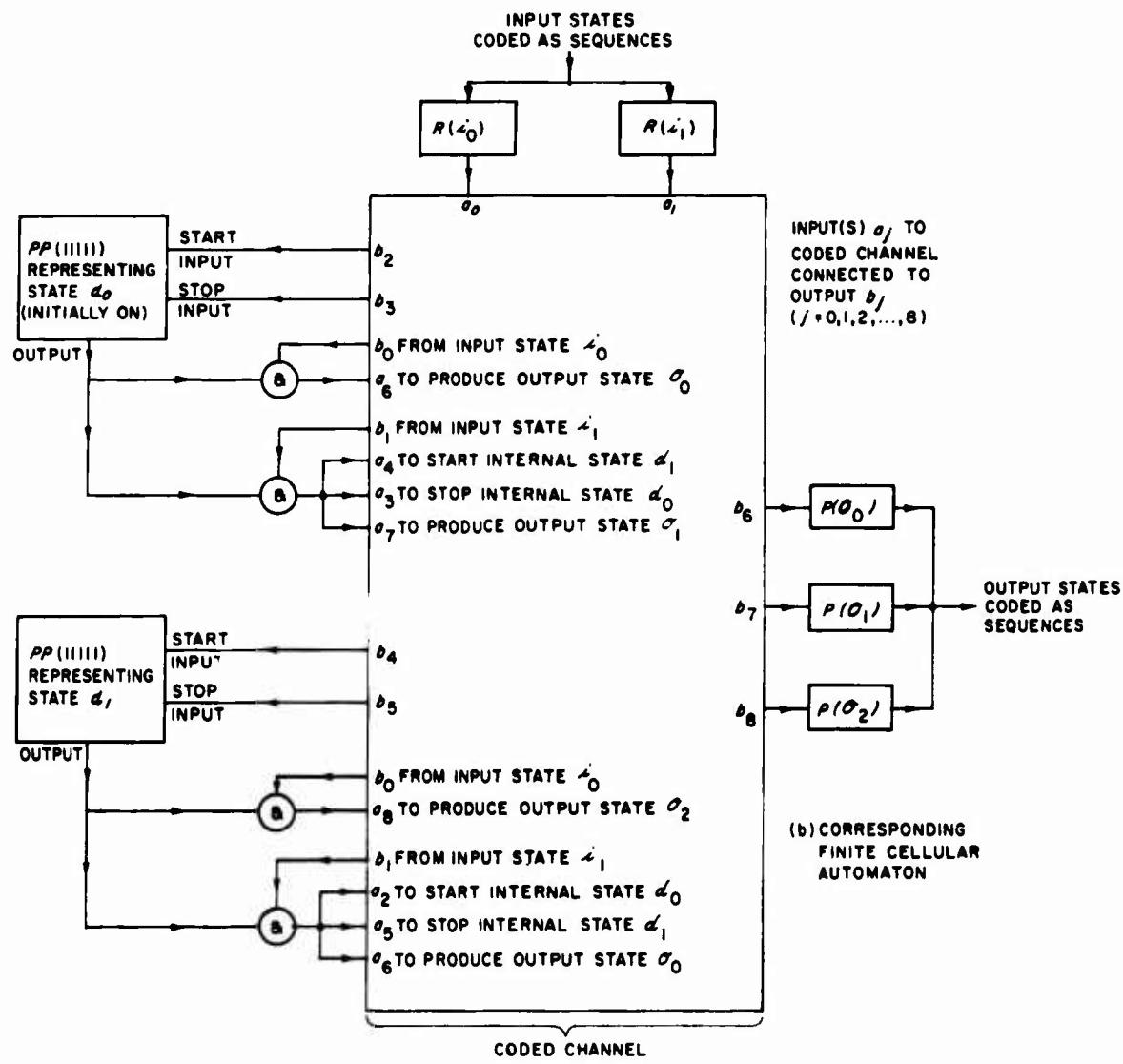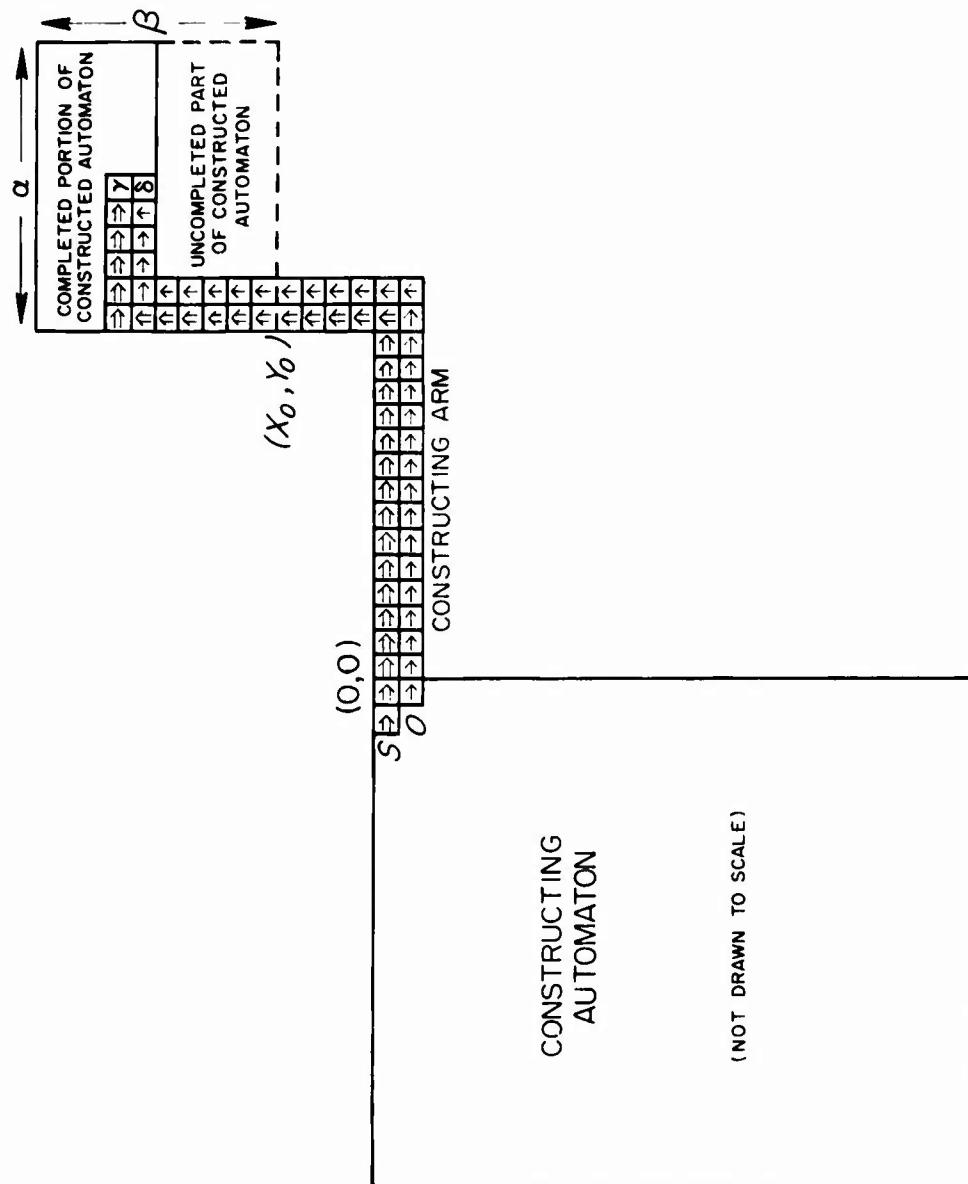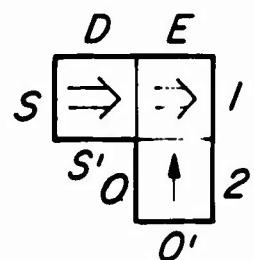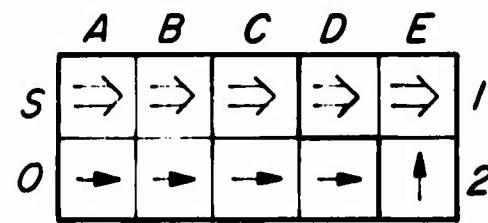
Fig. 16
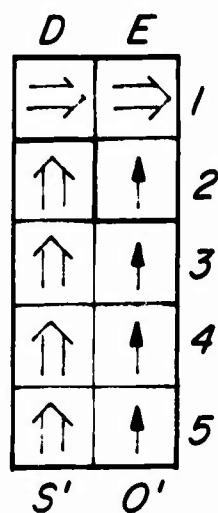
Operation of the Constructing Arm

Fig 17

(a) Head of Constructing Arm (b) Head Fed from Left
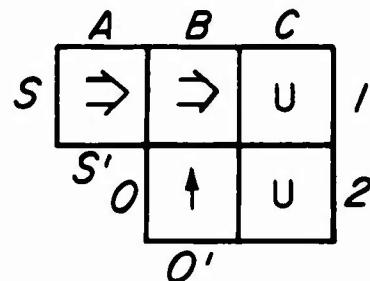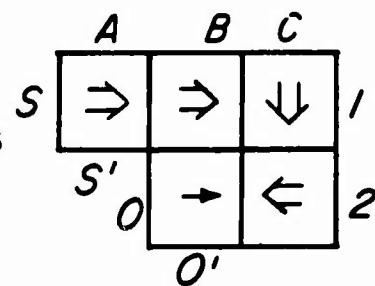
(c) Head Fed from Below

Fig. 18 CONSTRUCTING ARM

(a) Starting with

(b) $\Downarrow \Leftarrow U \rightarrow$ into S or S' produces

(c) $U \uparrow U \Rightarrow$ into O or O' produces
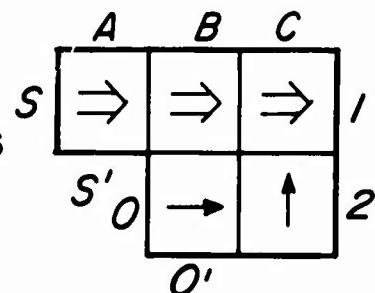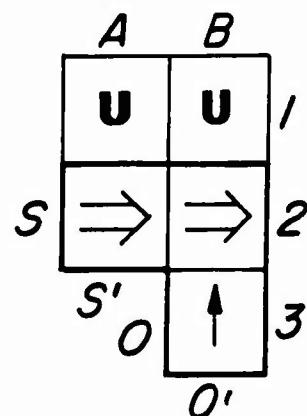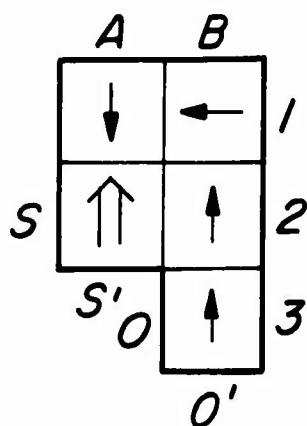
Fig. 19 HORIZONTAL ADVANCE OF CONSTRUCTING ARM

Starting with

Starting arrangement diagram:

Grid with columns A, B and rows 1, 2, 3.
- Row 1: A = **U**, B = **U**
- Row 2 (S): A = ⇒, B = ⇒
- Row 3 (S', O): B = ↑ (O')

U ↑ ←— ↓⇑**U** into O or O' produces

Grid with columns A, B and rows 1, 2, 3.
- Row 1: A = ↓, B = ←
- Row 2 (S): A = ⇑, B = ↑
- Row 3 (S', O): B = ↑ (O')

U ⇒ **U** ⇒ into S or S' produces

Grid with columns A, B and rows 1, 2, 3.
- Row 1: A = ⇒, B = ⇒
- Row 2 (S): A = ⇑, B = ↑
- Row 3 (S', O): B = ↑ (O')

Fig. 20 VERTICAL ADVANCE OF CONSTRUCTING ARM
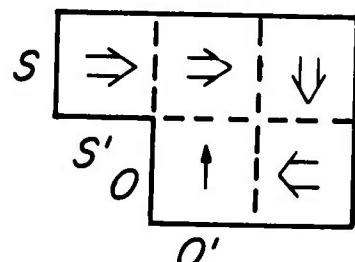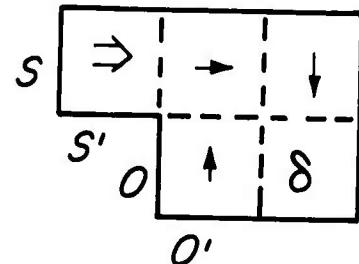
Starting with

U ⇓ Into *O* or *O'* Produces

U ⇐ U ↑ Into *S* or *S'* Produces

U → U ↓ U δ Into *O* or *O'* Produces

U ⇒ U γ Into *S* or *S'* Produces
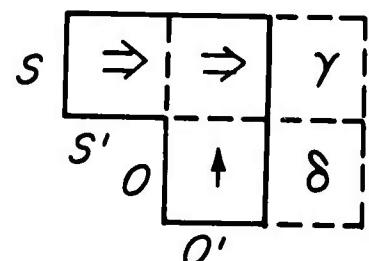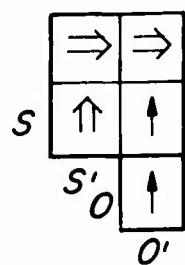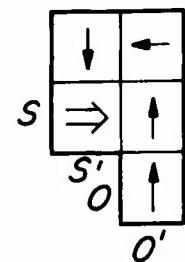
Fig.2I HORIZONTAL RETREAT OF CONSTRUCTING ARM
WITH CONSTRUCTION OF γ AND δ
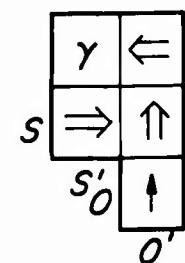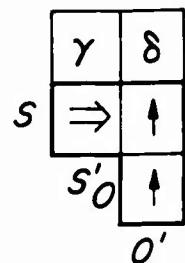
Starting with

U ← U ↓ U ⇒ into O or O' produces

U ⇑ U ⇐ U γ into S or S' produces

U ↕ U δ into O or O' produces

U ⇒ into S or S' produces
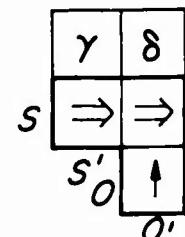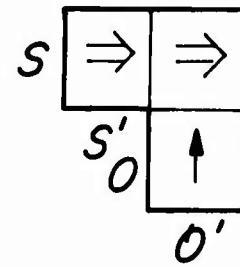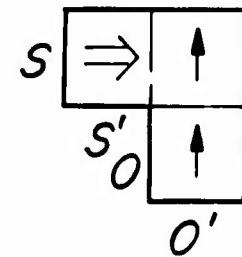
Fig. 22 VERTICAL RETREAT OF CONSTRUCTION OF γ AND δ
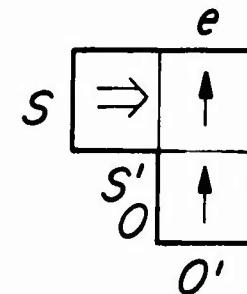
Starting with

U ↑ into *O* or *O'* produces

A pulse into *O* or *O'* exits at *e* as the stimulus to start the constructed automaton

U ⟹ into *S* or *S'* produces
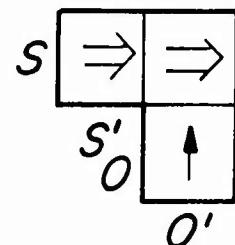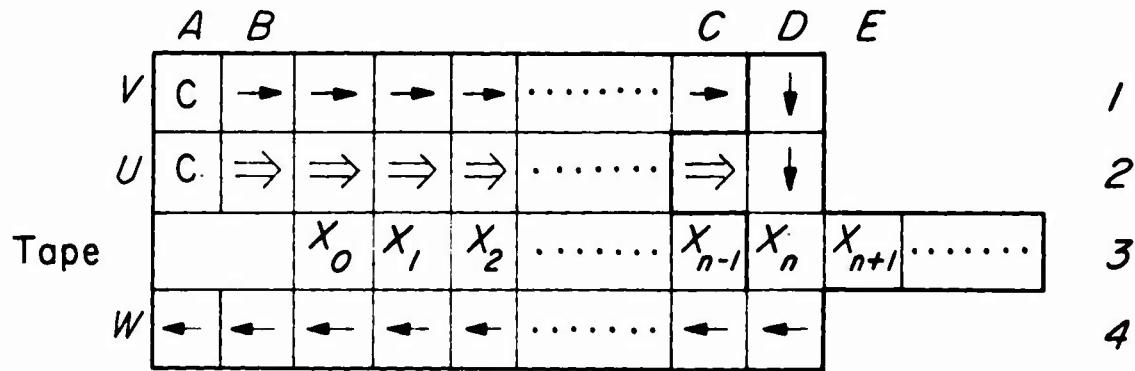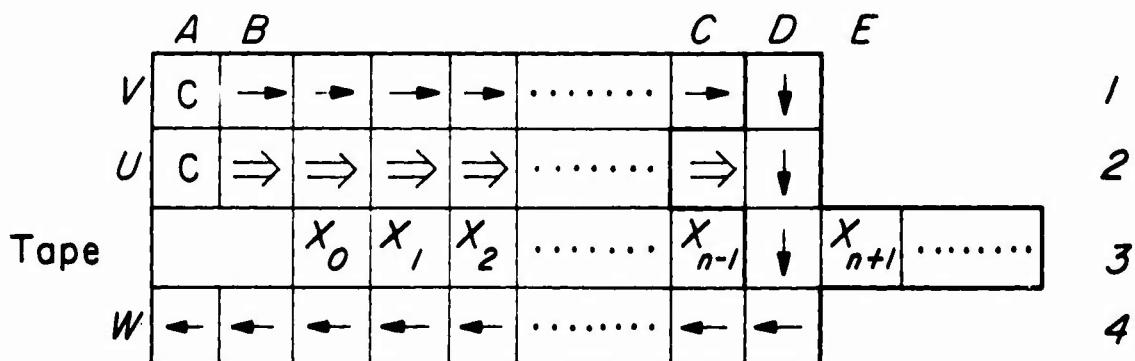
Fig. 23 INJECTION OF STARTING STIMULUS
          INTO THE CONSTRUCTED AUTOMATON

(a) Before Reading

  X_n is in **U** for "zero"
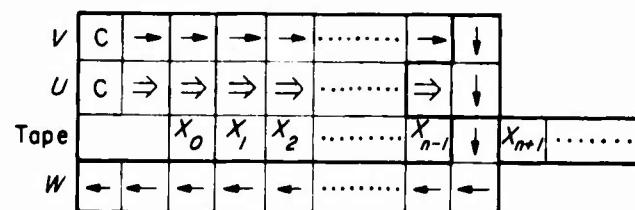
  X_n is in ↓ for "one"
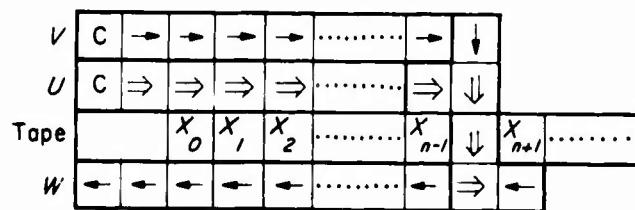
(b) After Reading

Fig. 24 TAPE, READING LOOP, AND CONSTRUCTING ARM
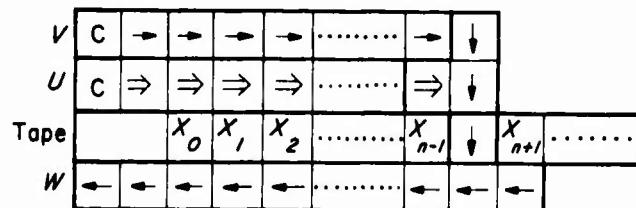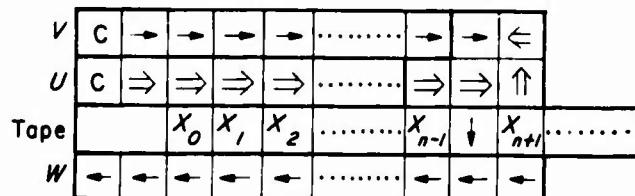
Fig.25 WRITING "ONE" IN CELL X_n AND LENGTHENING
THE READING LOOP

TAPE

CONSTRUCTING ARM AND
UPPER HALF OF READING LOOP

LOWER HALF OF READING LOOP

TAPE CONTROL
(NOT DRAWN TO SCALE)

UNIVERSAL COMPUTER
Fig. 26
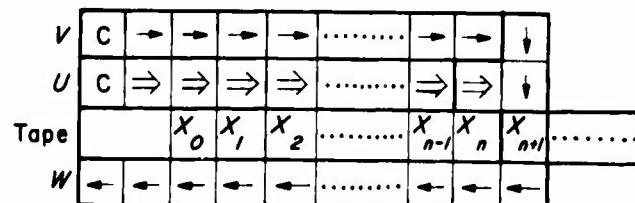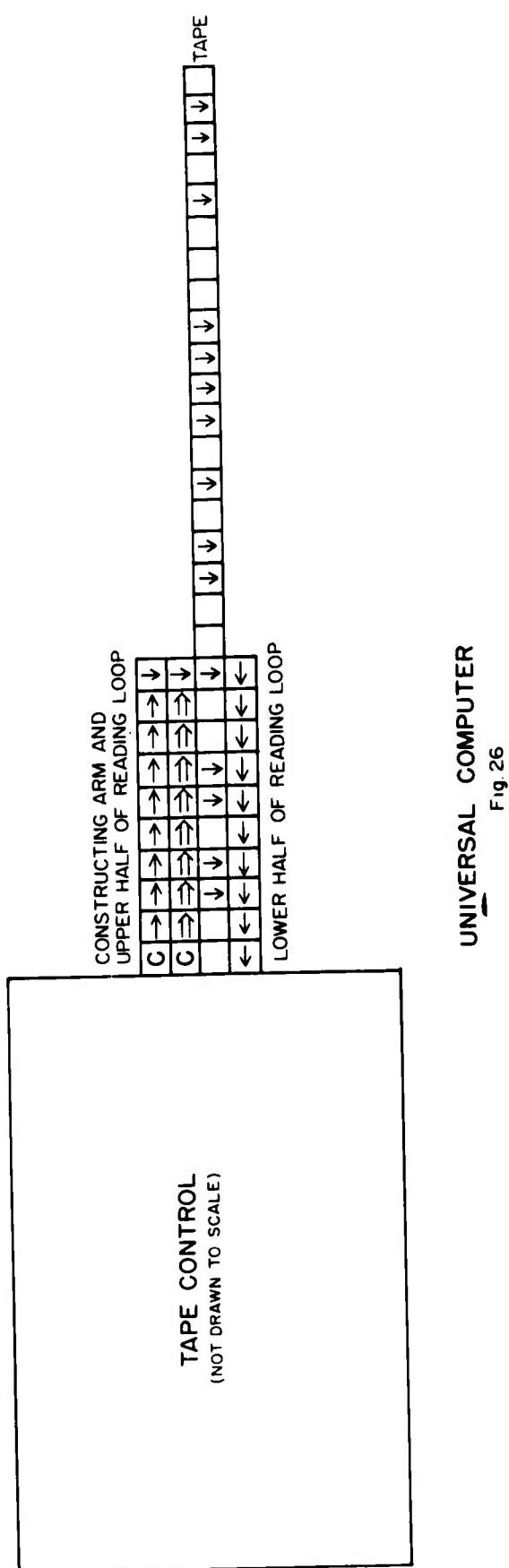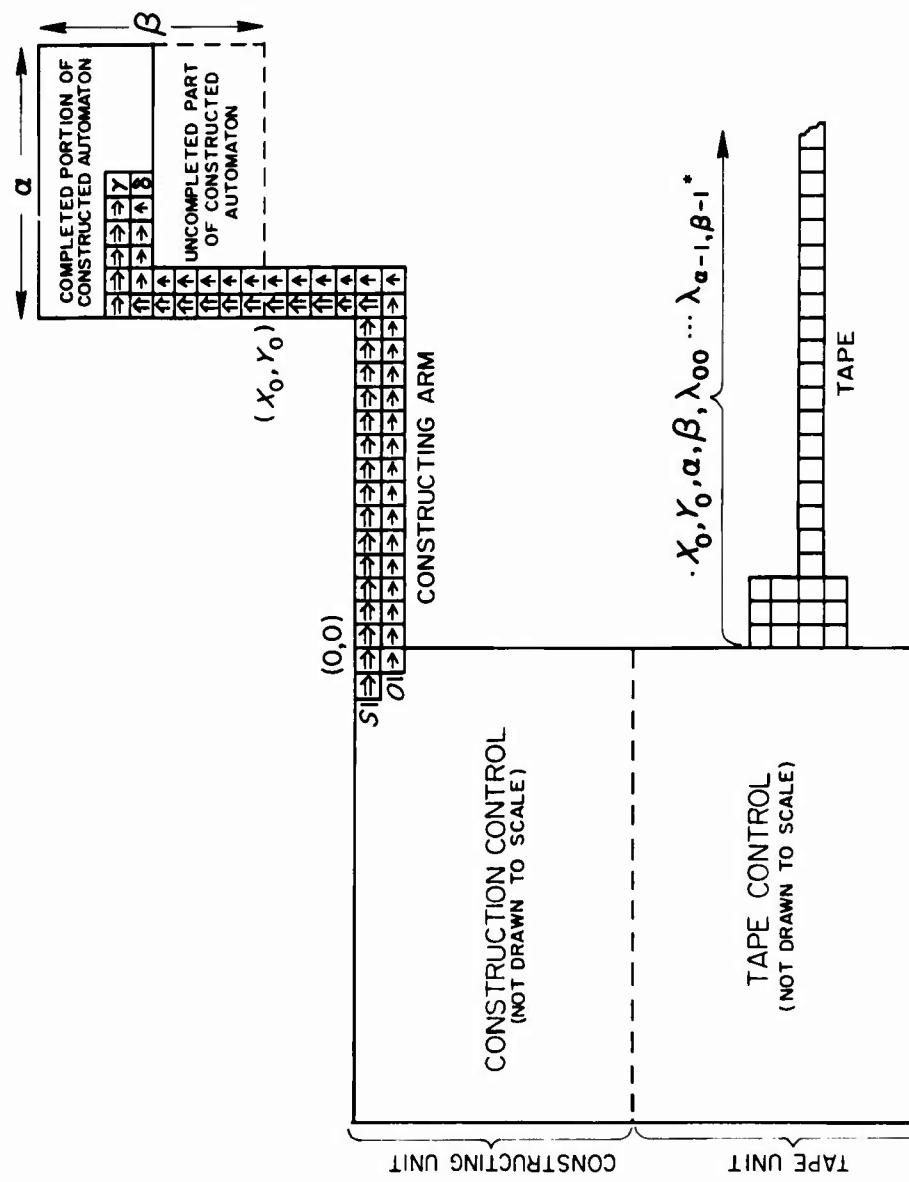
UNIVERSAL CONSTRUCTOR
Fig.27
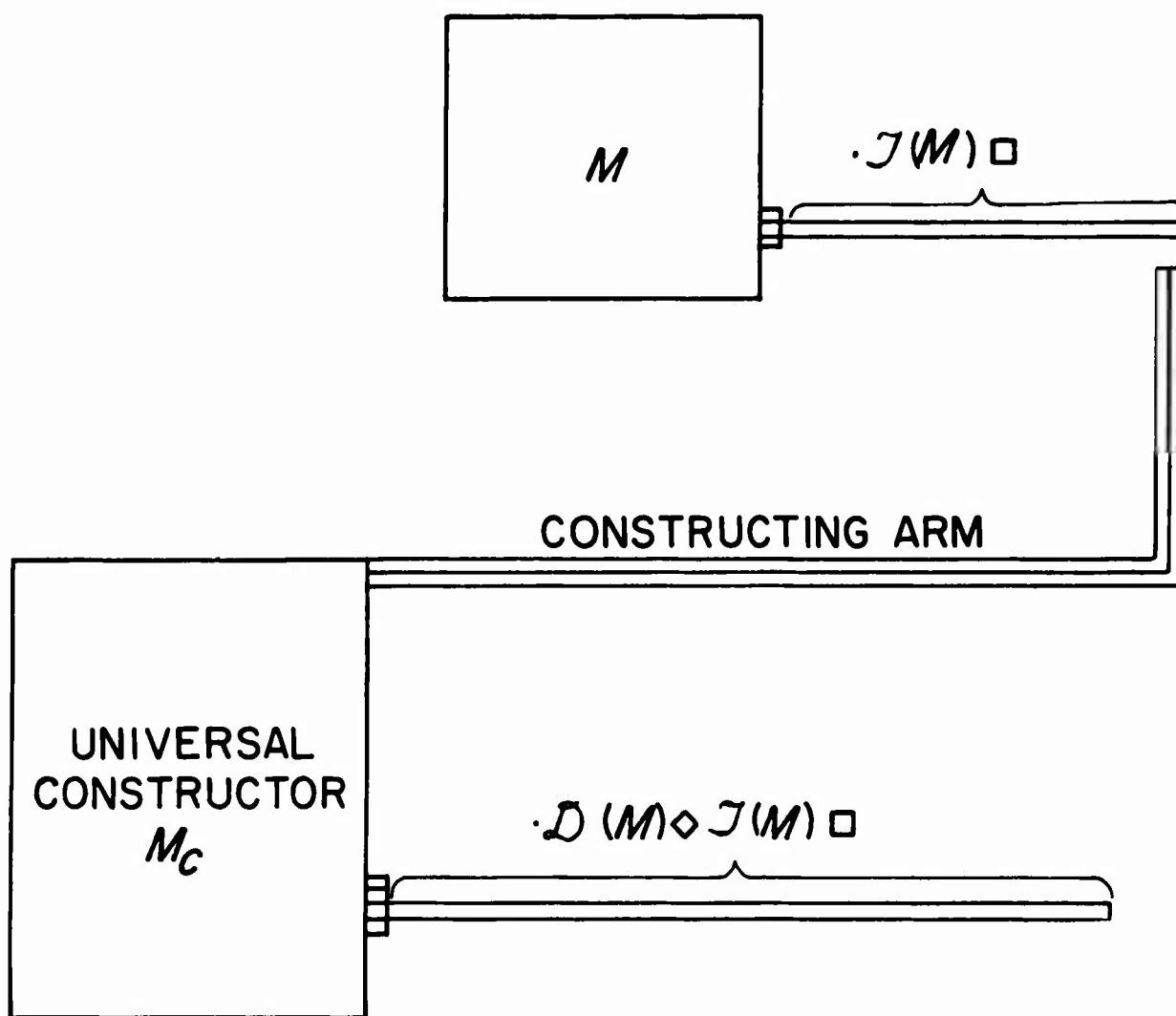
Fig. 28 MODIFIED UNIVERSAL CONSTRUCTOR
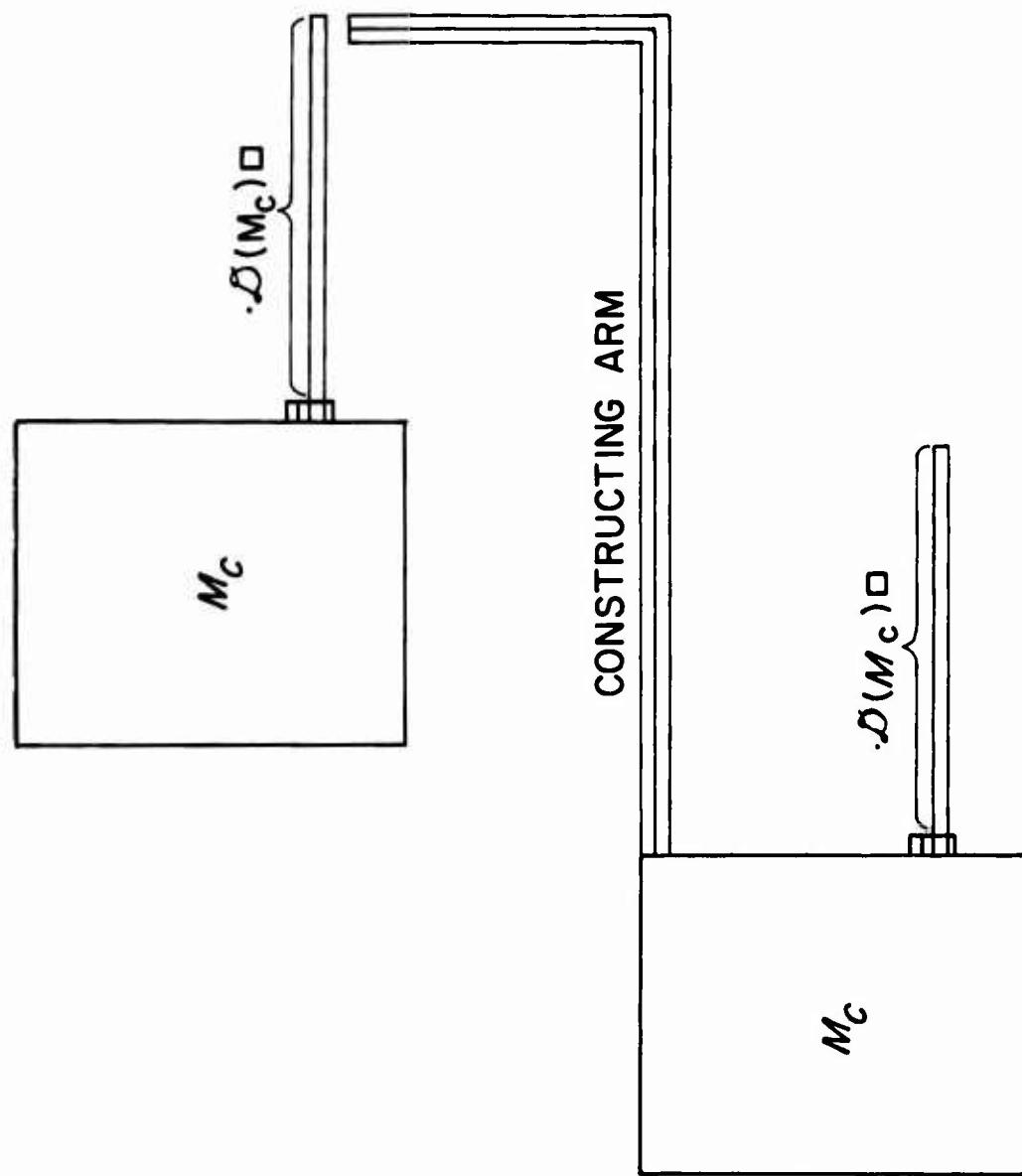
Fig. 29 AUTOMATA SELF-REPRODUCTION

BIBLIOGRAPHY

Arbib, Michael A., "Simple Self-Reproducing Universal Automata." *Information and Control 9* (1966) 177-189.

Bagley, J. D., *The Behavior of Adaptive Systems which Employ Genetic and Correlation Algorithms,* Ph.D. Thesis, University of Michigan, 1967.

Balzer, Robert M., *Studies Concerning Minimal Time Solutions to the Firing Squad Synchronization Problem,* Ph.D. Thesis, Carnegie Institute of Technology, 1966.

Burks, Arthur W., "Automata Models of Self-Reproduction." *Information Processing* (May, 1966) 121-123. Information Processing Society of Japan.

——————— , "Cellular Automata." Pp. 100-111 of *Theory of Finite and Probabilistic Automata,* edited by M. A. Gavrilov. Moscow: Nauka, 1965. (Russian).

——————— , "Computation, Behavior and Structure in Fixed and Growing Automata." *Behavioral Science 6* (1961) 5-22. Original version and discussion in *Self-Organizing Systems* (edited by Marshall Yovits and Scott Cameron). New York: Pergamon Press, 1960, 282-311, 312-314.

——————— , "Electronic Computing Circuits." *Proceedings of the Institute of Radio Engineers* 35 (August, 1947) 756-767.

——————— , "The Logic of Fixed and Growing Automata." *Proceedings of an International Symposium on the Theory of Switching, 2-5 April 1957.* Cambridge: Harvard University Press, 1959, Part I, 147-188.

——————— , "Programming and the Theory of Automata." Pp. 100-117 of *Computer Programming and Formal Systems.* Edited by P. Braffort and D. Hirschberg. Amsterdam: North-Holland Publishing Company, 1963. Also to appear in *Essays on Cellular Automata.*

——————— , "Super Electronic Computing Machine." *Electronics Industries* 5 (July, 1946) 62-67, 96.

Burks, Arthur W., "Toward a Theory of Automata Based on More Realistic Primitive Elements." Pp. 379-385 of *Information Processing 1962, Proceedings of IFIP Congress 62.* Edited by C. M. Popplewell. Amsterdam: North-Holland Publishing Company, 1963. Also to appear in *Essays on Cellular Automata.*

Burks, Arthur W. and Hao Wang, "The Logic of Automata." *Journal of the Association for Computing Machinery* 4 (1957) 193-218, 279-297.

Burks, Arthur W. and J.B. Wright, "Sequence Generators and Digital Computers." Pp. 139-199 of *Recursive Function Theory* (Proceedings of Symposia in Pure Mathematics, Vol. V). Providence, Rhode Island: American Mathematical Society, 1962.

Abstract and discussion in *Proceedings of the International Conference on Information Processing, UNESCO, Paris 15-20 June 1959*, pp. 425. Paris: UNESCO, 1960.

—————, "Theory of Logical Nets." *Proceedings of the Institute of Radio Engineers* 41 (October, 1953), 1357-1365. Reprinted in *Sequential Machines -- Selected Papers* 193-212. Edited by E.F. Moore. Reading, Mass.: Addison-Wesley, 1964.

Church, Alonzo, "Application of Recursive Arithmetic to the Problem of Circuit Synthesis." In *Summer Institute for Symbolic Logic Summaries*, Institute for Defense Analysis (1957).

Church, Alonzo, "Application of Recursive Arithmetic to the Theory of Computers and Automata." Notes from summer conference course in *Advanced Theory of the Logical Design of Digital Computers*, The University of Michigan (1958).

Codd, Edgar F., "Propagation, Computation, and Construction in Two-Dimensional Cellular Spaces." 152 pp. Ph.D. Dissertation, University of Michigan, 1965. Also published as *Cellular Automata*, 1968. (Academic Press).

Comfort, W.T., "Highly Parallel Machines." Pp. 126-155 of *Proceedings of a Workshop on Computer Organization*. Edited by Barnum and Knapp. New York: Spartan Books, 1963.

—————, "A Modified Holland Machine," *Proceedings of the 1963 Fall Joint Computer Conference*, Institute of Electrical and Electronic Engineers, 1963.

Davis, M., *Computability and Unsolvability*. McGraw-Hill, 1958.

Elgot, Calvin C., "Decision problems of finite automata design and related arithmetics." *Transactions of the American Mathematical Society* 98 (1961) 21-51.

Fisher, R.A., *The Genetical Theory of Natural Selection*. Dover, 1958.

Flanigan, Larry K., and Henry H. Swain, "Computer Simulation of A-V Nodal Conduction." *The University of Michigan Medical Center Journal* 33 (1967) 234-241.

Friedberg, R.M., "A Learning Machine: Part I." *IBM Journal of Research and Development* 2 (1958) 2-13.

Friedman, Joyce, "Some Results in Church's Restricted Recursive Arithmetic." *Journal of Symbolic Logic* 22 (December, 1957) 337-342.

—————, "A Decision Procedure for Computations of Finite Automata." *Journal of the Association for Computing Machinery* 9 (1962) 315-323.

Garner, H.L., and J.S. Squire, "Iterative Circuit Computers." Pp. 156-181 of *Proceedings of a Workshop on Computer Oragnization*. Edited by Barnum and Knapp. New York: Spartan Books, 1963.

Gerlernter, H., et. al. "Empirical Explorations of the Geometry-Theory
Proving Machine." Pp. 143-149 of *Proceedings of the Western Joint
Computer Conference, San Francisco,* May, 1960.

Gödel, Kurt, "Über formal unentscheidbare Sätze der Principia Mathematica
and verwandter Systeme I." *Monatshefte für Mathematik und Physik*
38 (1931) 173-198. (English translation by Elliott Mendelson in
The Undecidable, edited by Martin Davis, pp. 4-38. Hewlett, New York:
Raven Press, 1965.)

Gödel, Kurt, "On Undecidable Propositions of Formal Mathematical Systems."
Mimeographed notes on lectures delivered at the Institute for Advanced
Study, Princeton, New Jersey. February-May, 1934. 31 pages. (Reprinted
as pp. 39-74 of *The Undecidable,* above.)

Goldstine, H.H., and Adele Goldstine, "The Electronic Numerical Integrator
and Computer (ENIAC)." *Mathematical Tables and Other Aids to Computation*
2 (July, 1946) 97-110 and frontispiece.

Guttman, B.S., "A Resolution of Rosen's Paradox for Self-Reproducing Automata."
Bulletin of Mathematical Biophysics 28 (1966) 191-194.

Hennie, Frederick C., "Analysis of bilateral iterative networks." *Trans-
actions of the Institute of Radio Engineers Professional Group on
Circuit Theory* CT-6 (1959) 35-45.

Herdan, R., "A Logical Model for Growth and Differentiation in Multi-Celled
Organisms." *Bulletin of Mathematical Biophysics* 27 (1965) 379-393.

Holladay, J.C. and Ulam, S.M., "On Some Combinational Problems in Patterns
of Growth, I." *Notices of the American Mathematical Society 7* (1960) 234.

Holland, J.H., "Concerning Efficient Adaptive Systems." Pp. 215-230 of
Self-Organizing Systems-- 1962. Edited by M.C. Yovits, G.T. Jacobi,
and G.D. Goldstein. Washington,D.C.: Spartan Books, 1962.

——————, "Iterative Circuit Computers." Pp. 259-265 of *Proceedings of
the 1960 Western Joint Computer Conference,* Institute of Radio
Engineers, 1960. Also in *Essays on Cellular Automata.*

——————, "Outline for a Logical Theory of Adaptive Systems." *Journal
of the Association for Computing Machinery* 9 (July, 1962) 297-314.
Also in *Essays on Cellular Automata.*

——————, "A Universal Computer Capable of Executing an Arbitrary Number
of Sub-Programs Simultaneously." Pp. 108-113 of the *Proceedings of
the 1959 Eastern Joint Computer Conference,* Institute of Radio Engineers,
1959. Also in *Essays on Cellular Automata.*

——————, "Information Processing in Adaptive Systems." Pp. 330-338 of
Information Processing in the Nervous System, Vol. III of the
Proceedings of the International Union of Physiological Sciences, XXII
International Conference, Leiden, 1962.

——————, "Universal Embedding Spaces for Automata." Pp. 223-243 of *Cybernetics of the Nervous System, Progress in Brain Research,* Vol. 17. Edited by Norbert Wiener and J.P. Schade. New York: Elsevier, 1965.

——————, "Iterative Circuit Computers: Characterization and Resume of Advantages and Disadvantages." Pp. 171-178 of *Microelectronics and Large Systems.* Edited by Mathis, Wiley, and Spandorfer. New York: Spartan Books, 1965. Also in *Essays on Cellular Automata.*

Holland, J.H., "Universal Spaces: A Basis for Studies of Adaption." Pp. 218-230 of *Automata Theory.* Edited by E.R. Caianiello. New York: Academic Press, 1965.

Jacobson, Homer, "On Models of Reproduction." *American Scientist* 46 (1958) 255-284.

Kemeny, John G., "Man Viewed as a Machine." *Scientific American* 192 (1955) 58-67.

Kister, Stein, Ulam, Walden and Wells, "Experiments in Chess." *Journal of the Association for Computing Machinery* 4 (April, 1957).

Kleene, S.C., "Representation of Events in Nerve Nets and Finite Automata." Pp. 3-41 of *Automata Studies,* edited by C.E. Shannon and J. McCarthy. Princeton: Princeton University Press, 1956. (This appeared originally as Rand Corporation Memorandum RM-704, 101 pp., dated December 15, 1951.)

Lee, C.Y., "A Turing Machine Which Prints Its Own Code Script." Pp. 155-164 of *Proceedings of the Symposium on Mathematical Theory of Automata, New York, April, 1962.* Brooklyn, New York: Polytechnic Press, 1963.

Lehmer, D.H., E. Lehmer, W.H. Mills, and J.L. Selfridge, "Machine Proof of a Theorem on Cubic Residues." *Mathematics of Computation* 16 (October, 1962) 407-415.

Lofgren, Lars, "Automata of High Complexity and Methods of Increasing Their Reliability by Redundancy." *Information and Control* 1 (1958) 127-147.

——————, "Self-Repair as a Computability Concept in the Theory of Automata." Pp. 205-222 of *Proceedings of the Symposium on Mathematical Theory of Automata, New York, April, 1962.* Microwave Research Institute Symposium Series, Vol. XII, edited by Jerome Fox. Brooklyn, New York: Polytechnic Press of the Polytechnic Institute of Brooklyn, 1963.

Menzel, M.T., P.R. Stein, and S.M. Ulam, *Quadratic Transformations,* Part I. Report LA-2305, Los Alamos Scientific Laboratory, 1959. 158 Pages. Office of Technical Services, U.S. Department of Commerce, Washington, D.C.

Mesarovic, Mihajlo D., ed., *Views on General Systems Theory.* Proceedings of the Second Systems Symposium at Case Institute of Technology. New York: John Wiley, 1964.

Minsky, M., "Steps Toward Artificial Intelligence." *Proceedings of the Institute of Radio Engineers* 46 (1961) 8-30.

Moore, Edward F., "Artificial Living Plants." *Scientific American* 195 (1956) 118-126.

——————, "The Firing Squad Synchronization Problem." Pp. 213-214 of *Sequential Machines*. Edited by E.F. Moore. Reading, Mass.: Addison-Wesley Publishing Company, 1964.

——————, "Machine Models of Self-Reproduction." Pp. 17-33 of *Mathematical Problems in the Biological Sciences*. Proceedings of Symposia in Applied Mathematics, Vol. XIV. Providence, Rhode Island: American Mathematical Society, 1962. Also in *Essays on Cellular Automata*.

——————, "Machine Models of Self-Reproduction." Abstract 560-52, *Notices of the American Mathematical Society* 6 (1959) 622-623.

——————, "Gedanken-Experiments on Sequential Machines." Pp. 129-153 of *Automata Studies*, edited by C.E. Shannon and J. McCarthy. Princeton: Princeton University Press, 1956.

——————, "Mathematics in the Biological Sciences." *Scientific American* 211 (September, 1964) 148-164.

Mukhopadhyay, A., "Representation of Events in the von Neumann Cellular Model." Technical Report No. 28, July, 1967. Computer Group, Tata Institute of Fundamental Research, Bombay, India. Also in *Journal of the Association for Computing Machinery* 15 (1968) 693-705.

Myhill, John, "The Abstract Theory of Self-Reproduction." Pp. 106-118 of *Views on General Systems Theory*, Proceedings of the Second Systems Symposium at Case Institute of Technology. Edited by M.D. Mesarovic. New York: John Wiley, 1964. Also to appear in *Essays on Cellular Automata*.

——————, "The Converse of Moore's Garden-of-Eden Theorem." *Proceedings of the American Mathematical Society* 14 (August, 1963) 685-686. Also to appear in *Essays in Cellular Automata*.

McCulloch, W.S., and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity." *Bulletin of Mathematical Biophysics* 5 (1943) 115-133.

McNaughton, R., "The Theory of Automata, a Survey." Pp. 379-421 of *Advances in Computers*, Vol. 2. Edited by F.L. Alt. New York: Academic Press, 1961.

McNaughton, Robert, "On Nets Made up of Badly Timed Elements, Part I; Slow But Perfectly Timed Elements." Mimeographed, 30 pages. Philadelphia: Moore School of Electrical Engineering, University of Pennsylvania, 1961.

Newell, A., "On Programming a Highly Parallel Machine to be an Intelligent Technician," *Proceedings of the Western Joint Computer Conference*, Institute of Radio Engineers, 1960, pp. 267-282.

Newell, A., J.C. Shaw and H.A. Simon, "Empirical Explorations of the Logic Theory Machine: A Case Study in Heuristics." Report P-951, Rand Corporation (1957).

—————, "A Variety of Intelligent Learning in a General Problem Solver." In *Self-Organizing Systems*. Edited by Marshall Yovits and Scott Cameron. New York: Pergamon Press, 1960.

Oparin, A.I., *The Origin of Life on Earth*, third edition. New York: Academic Press, 1957.

Pattee, Howard H., "On the Origin of Macromolecular Sequences." *Biophysical Journal* (November, 1961) 683-710.

Penrose, L.S., "Self-Reproducing Machines." *Scientific American* 200 (1959) 105-114.

Pontecorvo, G., "Self-Reproduction and All That." Pp. 1-5 of *Symposium of the Society for Experimental Biology* (Great Britain), Vol. 12. Cambridge, England: Cambridge University Press, 1958.

Post, E.L., "Formal Reductions of the General Combinatoral Decision Problem." *American Journal of Mathematics* 65 (1943) 197-215.

—————, "Finite Combinatoral Processes -- Formulation I." *Journal of Symbolic Logic* 1 (1936) 103-104.

Rabin, Michael O., "Probabilistic Automata," *Information and Control* 6 (September, 1963) 230-245. Reprinted in *Sequential Machines*, pp. 98-114. Edited by E.F. Moore. Reading, Mass.: Addison-Wesley, 1964.

Rabin, M.O. and D. Scott, "Finite Automata and Their Decision Problems." *IBM Journal of Research and Development* 3 (1959) 114-125.

Rado, T., "On Non-Computable Functions." *Bell System Technical Journal* 41 (1962) 877-884.

Rochester, N., L.H. Haibt, John Holland, and W.L. Duda, "Tests on a Cell Assembly Theory of the Action of the Brain." *Institute of Radio Engineers Transactions on Information Theory* (September, 1956) 80-93.

Rosen, Robert, "On a Logical Paradox Implicit in the Notion of a Self-Reproducing Automaton." *Bulletin of Mathematical Biophysics* 28 (1966) 149-151.

Rosenberg, Richard, *Simulation of Genetic Populations with Biochemical Properties*. Ph.D. Thesis, University of Michigan, 1967.

Samuel, A.L., "Some Studies in Machine Learning Using the Game of Checkers." *IBM Journal of Research and Development* 3 (1959) 211-229. "Some Studies in Machine Learning Using the Game of Checkers II -- Recent Progress." *Ibid.* 11 (1967) 601-617.

Schrandt, R.G. and S.M. Ulam, "On Patterns of Growth of Figures in Two Dimensions." *Notices of the American Mathematical Society* 7 (1960)642.

—————, "On Recursively Defined Geometrical Objects and Patterns of Growth." To appear in *Essays on Cellular Automata*.

Selfridge, O.G., "Pandemonium, a Paradigm for Learning." In *Mechanization of Thought Processes*. National Physical Laboratory Symposium Number 10. London: Her Majesty's Stationary Office, 1959.

Shannon, Claude E., "Computers and Automata." *Proceedings of the Institute of Radio Engineers* 41 (1953) 10.

—————, "A Mathematical Theory of Communication." *The Bell System Technical Journal* 27 (1948) 379-423, 623-656.

—————, "Programming a Computer for Playing Chess." *Philosophical Magazine* 41 (March, 1950).

—————, "Von Neumann's Contributions to Automata Theory." *Bulletin of the American Mathematical Society* 64 (No. 3, Part 2, 1958) 123-129.

Squire, J.S. and S.M. Palais, "Programming and Design Considerations of a Highly Parallel Computer." *Proceedings of the Spring Joint Computer Conference*, Institute of Electrical and Electronic Engineers, 1964.

Stahl, Walter R., "Algorithmically Unsolvable Problems for a Cell Automaton." *Journal of Theoretical Biology* 8 (1965) 371-394.

—————, "A Model of Self-Reproduction Based on String Processing Finite Automata." Pp. 43-72 of *Natural Automata and Useful Simulations;* Proceedings of a Symposium on Fundamental Biological Models, Stanford University, 1965. Edited by H.H. Pattee, E.A. Edelsack, Louis Fein, and A.B. Callahan. Washington, D.C.: Spartan Books, 1966.

—————, "Self-Reproducing Automata." *Perspectives in Biology and Medicine* 8 (1965) 373-393.

Stahl, Walter, Robert W. Coffin and Harry E. Goheen, "Simulation of Biological Cells by Systems Composed of String-Processing Finite Automata." Pp. 89-102 of *Proceedings of the Spring Joint Computer Conference*, 1964.

Stein, P.R. and S.M. Ulam, "Non-Linear Transformation Studies on Electronic Computers." *Rozprany Matematyczne XXXIX*, Warsaw, 1964. Also to appear in *Essays on Cellular Automata*.

Thatcher, James, "The Construction of a Self-Describing Turing Machine." Pp. 165-171 of *Proceedings of the Symposium on Mathematical Theory of Automata, New York, April, 1962*. Brooklyn, New York: Polytechnic Press, 1963.

Turing, A.M., "The Chemical Basis of Morphogenesis." *Phil. Trans. Roy. Soc. of London Mathematical Society*, Series B, Biological Sciences, Vol. 237, August, 1952, pp. 37-72.

—————, "On Computable Numbers, With an Application to the Entscheidungsproblem." *Proceedings of the London Mathematical Society*, Series 2, 42 (1936-37) 230-265. "A Correction", *Ibid*, 43 (1937) 544-546. (Reprinted in *The Undecidable*, edited by Martin Davis, pp. 115-154, Hewlett, New York: Raven Press, 1965.)

Turing, A.M., "Computing Machinery and Intelligence." *Mind* 59 (1950) 433-460.

Ulam, S.M., *A Collection of Mathematical Problems*. New York: Interscience Publishers, Inc., 1960.

—————, "Electronic Computers and Scientific Research." Pp. 95-108 of *The Age of Electronics*. Edited by C.F.J. Overhage. New York: McGraw-Hill, 1962.

—————, "Random Processes and Transformation." *Proceedings of the International Congress of Mathematicians*, 1950, Vol. II, pp. 264-275. Providence, Rhode Island: American Mathematical Society, 1952.

—————, "On Some Mathematical Problems Connected With Patterns of Growth of Figures." Pp. 215-224 of *Mathematical Problems in the Biological Sciences*. Proceedings of Symposia in Applied MAthematics, Vol. 14, Providence, Rhode Island: American Mathematical Society, 1962. Also to appear in *Essays on Cellular Automata*.

—————, "On Some New Possibilities in the Organization and Use of Computing Machines." IBM Research Report 68, 1957, IBM Corporation, Yorktowm Heights, New York.

Von Neumann, John, *Collected Works*. Six volumes. Edited by A.H. Taub. New York: MacMillan, 1961-1963. Vol. V is entitled *Design of Computers, Theory of Automata and Numerical Analysis*. References to these volumes will be made as follows: "*Collected Works* 5.288-328" refers to pp. 288-328 of Vol. V.

Von Neumann, John, *The Computer and the Brain*. With a preface by Klara von Neumann, pp. v-x. New Haven: Yale University Press, 1958.

—————, Review of Norbert Wiener's *Cybernetics, or Control and Communication in the Animal and the Machine*. *Physics Today* 2 (1949) 33-34.

—————, "The General and Logical Theory of Automata." Pp. 1-41 of *Cerebral Mechanisms in Behavior - The Hixon Symposium*, edited by L.A. Jeffress. New York: John Wiley, 1951. *Collected Works* 5.288-328.

—————, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components." Pp. 43-98 of *Automata Studies*, edited by C.E. Shannon and J. McCarthy. Princeton: Princeton University Press, 1956. *Collected Works* 5.329-378.

—————, *Theory of Self-Reproducing Automata*. Edited and completed by Arthur W. Burks. Urbana: University of Illinois Press, 1966.

Errata:
      pp. 159, par. 2 - the times should be t+8, t+9, t+10;
      pp. 171, line 21 - "19d" should be "19e";
      pp. 174, line 4 - replace "n" by "$5p$";
      pp. 179, line 18 - "pulser" should be "decoder".

Von Neumann, John and O. Morgenstern, *Theory of Games and Economic Behavior*. Princeton: Princeton University Press, 1947.

Wagner, Eric G., "An Approach to Modular Computers,1: Spider Automata and Embedded Automata." IBM Research Report 1107, January 28, 1964, Yorktown Heights, New York.

Wagner, Eric G., "Modular Computers, II: Graph Theory and the Interconnection of Modules." IBM Research Report RC 1414, June 4, 1965, Yorktown Heights, New York.

Waksman, Abraham, "An Optimum Solution to the Firing Squad Synchronization Problem." *Information and Control 9* (1966) 67-78.

Wang, Hao, "Universal Turing Machines: An Exercise in Coding." *Z. Math. Logik Grundlagen Math.* 3 (1957) 69-80.

——————, "Circuit Synthesis by Solving Sequential Boolean Equations." *Z. Math. Logik Grundlagen Math.* 5 (1959) 291-322.

——————, "A Variant to Turing's Theory of Computing Machines." *Journal of the Association of Computing Machines* 4 (1957) 63-92.

Watanabe, Shigeru, "5-Symbol 8-State ar.¹ 5-Symbol 6-State Universal Turing Machines." *Journal of the Association for Computing Machinery* 8 (October, 1961) 476-483.

Wiener, Norbert, *Cybernetics, or Control and Communication in the Animal and the Machine*. New York: John Wiley & Sons, Inc., 1948.

Wigner, E.P., "The Probability of the Existence of a Self-Reproducing Unit." Pp. 231-248 of *The Logic of Personal Knowledge* (Essays presented to Michael Polanyi). Glencoe, Illinois: The Free Press, 1961.

Wright, S., "Physiological Genetics, Ecology of Populations, and Natural Selection." Pp. 429-475 of *The Evolution of Life, Vol. 1*. Edited by Sol Tax. Chicago: University of Chicago Press, 1960.

Yamada, H., "Real-time Computation and Recursive Functions Not Real-time Computable." *Transactions of the Institute of Radio Engineers* EC-11 (December, 1962) 753-760.

DOCUMENT CONTROL DATA · R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Logic of Computers Group The University of Michigan Ann Arbor, Michigan	Unclassified
	2b. GROUP

3. REPORT TITLE

VON NEUMANN'S SELF-REPRODUCING AUTOMATA

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Technical Report

5. AUTHOR(S) (Last name, first name, initial)

Arthur W. Burks

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1969	110	132

8a. CONTRACT OR GRANT NO. DA-31-124-ARO-D-483	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	08226-11-T
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. AVAILABILITY/LIMITATION NOTICES

Distribution of This Document is Unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	U. S. Army Office (Durham) Durham, North Carolina

13. ABSTRACT

John von Neumann's kinematic and cellular automaton systems are described. A complete informal description of the cellular system is presented including an explanation of the realization of logical components, the design of computer organs, the construction, destruction and movement of organs by sequences of internally originated pulses, universal computation and construction, and self-reproduction. Connections between von Neumann's automaton research and his work on computer design are brought out, and the significance of cellular arrays for biological research discussed.

DD FORM 1473 (1 JAN 64)

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
cellular automata						
von Neumann						
self-reproduction						
biological models						
universal computation and construction						

INSTRUCTIONS

1. **ORIGINATING ACTIVITY:** Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization (*corporate author*) issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. **REPORT TITLE:** Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. **DESCRIPTIVE NOTES:** If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. **AUTHOR(S):** Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. **REPORT DATE:** Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers (*either by the originator or by the sponsor*), also enter this number(s).

10. **AVAILABILITY/LIMITATION NOTICES:** Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through

_____ ."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through

_____ ."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through

_____ ."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. **SUPPLEMENTARY NOTES:** Use for additional explanatory notes.

12. **SPONSORING MILITARY ACTIVITY:** Enter the name of the departmental project office or laboratory sponsoring (*paying for*) the research and development. Include address.

13. **ABSTRACT:** Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS), (S), (C),* or *(U)*

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. **KEY WORDS:** Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.