63-4

SOME MEMORY ASPECTS OF FINITE AUTOMATA

CHUNG LAUNG LIU

TECHNICAL REPORT 411

MAY 31, 1963

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
RESEARCH LABORATORY OF ELECTRONICS
CAMBRIDGE, MASSACHUSETTS

The Research Laboratory of Electronics is an interdepartmental laboratory in which faculty members and graduate students from numerous academic departments conduct research.

The research reported in this document was made possible in part by support extended the Massachusetts Institute of Technology, Research Laboratory of Electronics, jointly by the U.S. Army, the U.S. Navy (Office of Naval Research), and the U.S. Air Force (Office of Scientific Research) under Contract DA36-039-sc-78108, Department of the Army Task 3-99-25-001-08; and in part by Grant No. DA-SIG-36-039-61-G14.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

RESEARCH LABORATORY OF ELECTRONICS

Technical Report 411 May 31, 1963

SOME MEMORY ASPECTS OF FINITE AUTOMATA

Chung Laung Liu

This report is based on a thesis submitted to the Department of Electrical Engineering, M.I.T., August 20, 1962, in partial fulfillment of the requirements for the degree of Doctor of Science.

(Manuscript received February 27, 1963)

Abstract

The most important characteristic of a finite automaton is that it has a "memory." By this we mean that the behavior of an automaton is dependent upon its past history. In this report several special cases are studied in which the unique determination of the behavior of an automaton is possible, even when a portion of its past history is unknown.

TABLE OF CONTENTS

I. INTRODUCTION

The theory of finite automata was first developed by Huffman,[1] Kleene,[2] and Moore[3] in the years 1954-1956. Since then, a significant amount of effort has gone into further investigation and exploration in this area. This report is the result of a study of some special characteristics of a finite deterministic automaton.

A finite automaton is described by a quintuple $A = \{\Sigma, Z, S, M, N\}$. $\Sigma = \{\sigma_1, \sigma_2 \ldots\}$ is the finite set of all possible input symbols to the automaton; $Z = \{z_1, z_2 \ldots\}$ is the finite set of all possible output symbols from the automaton, and $S = \{s_1, s_2 \ldots\}$ is the finite set of the internal states of the automaton. M, which specifies the next internal state of the automaton, is a function whose domain is the Cartesian product $S \times \Sigma$ and whose range is S. N, which specifies the present output of the automaton, is a function whose domain is $S \times \Sigma$ and whose range is Z. The mappings of the functions M and N are usually exhibited as the "flow table." The definitions of both M and N can be extended recursively from the domain $S \times \Sigma$ to $S \times T$, where T is the set of all sequences of the input symbols from the set Σ. That is,

$$M(s, \sigma t) = M(M(s, \sigma), t)$$

$$N(s, \sigma t) = N(M(s, \sigma), t),$$

where $s \in S$, $\sigma \in \Sigma$, and $t \in T$.

For a deterministic finite automaton, both of the functions M and N are single-valued. When the present state of the automaton and the present input symbol to the automaton are known, the present output symbol and the next state of the automaton can be determined uniquely. Furthermore, if the initial state of the automaton and the input sequence to the automaton are known, the corresponding final state and the sequence of output symbols of the automaton also can be determined uniquely. On the other hand, in order to determine the final state and the sequence of output symbols of the automaton, it is necessary, in general, to have the initial state of the automaton and the input sequence to the automaton specified unambiguously. This, indeed, is the most important characteristic of an automaton, that an automaton has a "memory" and thus its behavior is dependent upon its past history. However, there are special conditions under which only a portion of the past history affects the behavior of the automaton; these are the conditions that we shall investigate here. In particular, we would like to know whether it is possible to determine the final behavior of an automaton if

(a) The initial state is unknown, and only the input sequence to the automaton is given.

(b) The initial state is unknown, and the input sequence and the corresponding output sequence are given.

(c) Both the initial state and the input sequence are unknown, and only the output sequence is given.

(d) The initial state is known, but only a portion of the input sequence is given.

(e) The initial state is known, but only portions of both the input sequence and the corresponding output sequence are given.

(f) Both the initial state and the input sequence are unknown, and only a portion of the output sequence is given.

We investigate this problem under the first three conditions in Sections III and IV; the rest, in Section V.

In a broader sense, the automata for all of these conditions can be categorized as "information lossy automata." Since portions of the past history are not needed in the specification of the final behavior of the automaton or, in other words, the final behavior of the automaton is independent of these portions of the past history, information about them is lost if only the final behavior of the automaton is known.

II. PRODUCT AUTOMATA

2.1 NOTATION

Let $A = \{\Sigma, Z, S, M, N\}$ and $B = \{\Sigma, Z', S', M', N'\}$ be two finite automata having the same set of input symbols. We define their product as $A \times B = \{\Sigma, Z \times Z', S \times S', M \times M', N \times N'\}$, where $Z \times Z'$ and $S \times S'$ are Cartesian products of sets, $M \times M'$ is a function that maps $(S \times S') \times \Sigma$ into $S \times S'$, namely

$$M \times M'((s, s'), \sigma) = (M(s, \sigma), M'(s', \sigma)),$$

and $N \times N'$ is a function that maps $(S \times S') \times \Sigma$ into $Z \times Z'$, namely,

$$N \times N'((s, s'), \sigma) = (N(s, \sigma), N'(s', \sigma)),$$

with $(s, s') \in (S, S')$, and $\sigma \in \Sigma$. Obviously, $A \times B$ is an automaton itself. Moreover, for automata A, B, C ... having the same set of input symbols, we can define $A \times B \times C \ldots$, which is also an automaton.

2.2 AUTOMATA A^p AND $A^{\{p\}}$

Let $A = \{\Sigma, Z, S, M, N\}$ be an automaton. We denote the product of p copies of A by $A^p = \{\Sigma, Z^p, S^p, M^p, N^p\}$ and call A^p the p^{th}-power automaton of A, where Z^p is the set of all ordered p-tuples of Z, S^p is the set of all ordered p-tuples of S, and

$$M^p((s_1, s_2 \ldots s_p), \sigma) = (M(s_1, \sigma), M(s_2, \sigma) \ldots M(s_p, \sigma))$$

$$N^p((s_1, s_2 \ldots s_p), \sigma) = (N(s_1, \sigma), N(s_2, \sigma) \ldots N(s_p, \sigma)).$$

We then can define $A^{\{p\}} = \{\Sigma, Z^{\{p\}}, S^{\{p\}}, M^{\{p\}}, N^{\{p\}}\}$ as the unordered p^{th}-power automaton of A. Here, $Z^{\{p\}}$ is a subset of the set of all the subsets of Z which contains at least one output symbol and, at most, p output symbols; $S^{\{p\}}$ is the set of all the subsets of S which contains at least one state and, at most, p states, and

$$M^{\{p\}}(\{s_1, s_2 \ldots s_p\}, \sigma) = \{M(s_1, \sigma), M(s_2, \sigma) \ldots M(s_p, \sigma)\}$$

$$N^{\{p\}}(\{s_1, s_2 \ldots s_p\}, \sigma) = \{N(s_1, \sigma), N(s_2, \sigma) \ldots N(s_p, \sigma)\},$$

where the elements in the braces are elements of a set and are, therefore, not necessarily distinct. Figure 1 is an example of the automata A^p and $A^{\{p\}}$.

2.3 AUTOMATON HOMOMORPHISM

When we are interested only in the structural characteristics of an automaton, we are not concerned with its outputs. Thus an automaton can be described by a triple $A = \{\Sigma, S, M\}$ with the set of the output symbols Z and the output function N disregarded. We have

S \ Σ	0	1
s_1	s_1,α	s_3,β
s_2	s_2,β	s_2,α
s_3	s_2,α	s_1,α

A

S^2 \ Σ	0	1
(s_1,s_1)	$(s_1,s_1),(\alpha,\alpha)$	$(s_3,s_3),(\beta,\beta)$
(s_1,s_2)	$(s_1,s_2),(\alpha,\beta)$	$(s_3,s_2),(\beta,\alpha)$
(s_1,s_3)	$(s_1,s_2),(\alpha,\alpha)$	$(s_3,s_1),(\beta,\alpha)$
(s_2,s_1)	$(s_2,s_1),(\beta,\alpha)$	$(s_2,s_3),(\alpha,\beta)$
(s_2,s_2)	$(s_2,s_2),(\beta,\beta)$	$(s_2,s_2),(\alpha,\alpha)$
(s_2,s_3)	$(s_2,s_2),(\beta,\alpha)$	$(s_2,s_1),(\alpha,\alpha)$
(s_3,s_1)	$(s_2,s_1),(\alpha,\alpha)$	$(s_1,s_3),(\alpha,\beta)$
(s_3,s_2)	$(s_2,s_2),(\alpha,\beta)$	$(s_1,s_2),(\alpha,\alpha)$
(s_3,s_3)	$(s_2,s_2),(\alpha,\alpha)$	$(s_1,s_1),(\alpha,\alpha)$

A^2

$S^{\{2\}}$ \ Σ	0	1
$\{s_1\}$	$\{s_1\}$, $\{\alpha\}$	$\{s_3\}$, $\{\beta\}$
$\{s_2\}$	$\{s_2\}$, $\{\beta\}$	$\{s_2\}$, $\{\alpha\}$
$\{s_3\}$	$\{s_2\}$, $\{\alpha\}$	$\{s_1\}$, $\{\alpha\}$
$\{s_1,s_2\}$	$\{s_1,s_2\}$, $\{\alpha,\beta\}$	$\{s_2,s_3\}$, $\{\alpha,\beta\}$
$\{s_1,s_3\}$	$\{s_1,s_3\}$, $\{\alpha\}$	$\{s_1,s_3\}$, $\{\alpha,\beta\}$
$\{s_2,s_3\}$	$\{s_2\}$, $\{\alpha,\beta\}$	$\{s_1,s_2\}$, $\{\alpha\}$

$A^{\{2\}}$

Figure 1

DEFINITION 1: f is an automaton homomorphism of A onto B if and only if (a) $A = \{\Sigma, S, M\}$ and $B = \{\Sigma, S', M'\}$ are automata having the same set of input symbols, (b) f is a mapping that maps S into S', that is, $f: S \rightarrow S'$; and (c) for all $s \in S$ and for all $\sigma \in \Sigma$, $f(M(s,\sigma)) = M'(f(s),\sigma)$.

THEOREM 1: An automaton $A = \{\Sigma, S, M\}$ is a homomorphic image of its p^{th}-power automaton A^p for any positive integer p.

PROOF 1: (a) $A = \{\Sigma, S, M\}$ and $A^p = \{\Sigma, S^p, M^p\}$ are automata having the same set of input symbols.

(b) Let f be a mapping for all ordered p-tuples of states which maps each ordered p-tuple into its first element, i.e., $f: (s_1, s_2, \ldots s_p) \rightarrow s_1$.

(c) For all $(s_1, s_2 \ldots s_p) \in S^p$ and for all $\sigma \in \Sigma$, $f(M^p(s^p,\sigma)) = f(M(s_1,\sigma), M(s_2,\sigma)\ldots M(s_p,\sigma)) = M(s_1,\sigma)$ and $M(f(s^p),\sigma) = M(f(s_1, s_2 \ldots s_p),\sigma) = M(s_1,\sigma)$.

Therefore, $f(M^p(s^p,\sigma)) = M(f(s^p),\sigma)$. Q.E.D.

THEOREM 2: Given an automaton $A = \{\Sigma, S, M\}$, $A^{\{p\}}$ is a homomorphic image of A^p for any positive integer p.

PROOF 2: (a) $A^{\{p\}} = \{\Sigma, S^{\{p\}}, M^{\{p\}}\}$ and $A^p = \{\Sigma, S^p, M^p\}$ are automata having the same set of input symbols.

(b) Let f be a mapping for all ordered p-tuples of states which maps each ordered p-tuple into the set that consists of all distinct elements of the p-tuple, i.e., f: $(s_1, s_2 \ldots s_p) \rightarrow \{s_1, s_2 \ldots s_p\}$.

(c) For all $(s_1, s_2 \ldots s_p) \in S^p$ and for all $\sigma \in \Sigma$, $f(M^p(s^p,\sigma)) = f(M(s_1,\sigma), M(s_2,\sigma)\ldots M(s_p,\sigma)) = \{M(s_1,\sigma), M(s_2,\sigma)\ldots M(s_p,\sigma)\}$, and $M^{\{p\}}(f(s^p),\sigma) = M^{\{p\}}(f(s_1, s_2 \ldots s_p),\sigma) = \{M(s_1,\sigma), M(s_2,\sigma)\ldots M(s_p,\sigma)\}$.

Therefore, $f(M^p(s^p,\sigma)) = M^{\{p\}}(f(s^p),\sigma)$. Q.E.D.

2.4 THE n^{th}-POWER AUTOMATON

For any automaton $A = \{\Sigma, S, M\}$ consisting of n internal states, let n copies of A be connected in "parallel." That is, they accept identical input sequences, but are not interrelated otherwise, as shown in Fig. 2. Considering the parallel-connected system as a

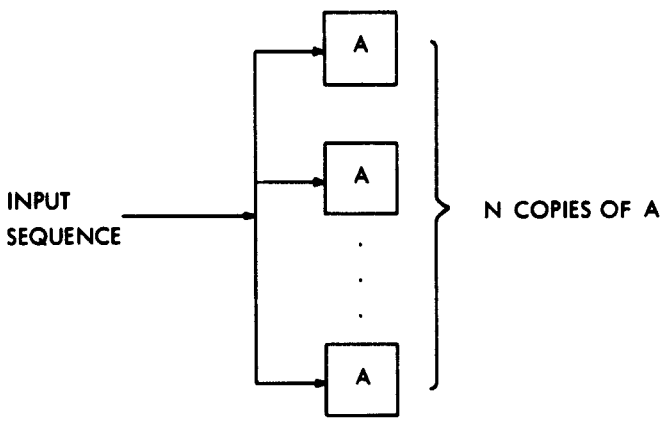


Figure 2

whole, we have an automaton whose input symbols are the set Σ and whose internal states are characterized by the total states of the n copies of A. This is exactly the A^n automaton.

When we are not concerned with the output from the automaton, an automaton is solely characterized by its ability to partition all input sequences into congruence classes. In algebraic language (see the appendix for full definitions), the set of all input sequences is a free monoid generated by the set of all input symbols. An automaton then defines a monoid homomorphism that maps the free monoid into the finite monoid of all congruence classes, as shown in Fig. 3. An automaton consisting of n internal states, in general, may partition the set of all input sequences into n^n congruence classes. If we designate the n internal states by $s_1, s_2, \ldots s_n$, for any input sequence t we have $M(s_1, t) = s_i$, $M(s_2, t) = s_j \ldots M(s_n, t) = s_r$, where the indices i, j ... r are among 1, 2 ... n. We can consider the input sequence t as one that defines a mapping that maps s_1 into s_i, s_2 into s_j ... and so on, or just a mapping that maps the ordered n-tuple $(s_1, s_2 \ldots s_n)$ into another ordered n-tuple $(s_i, s_j, \ldots s_r)$. The congruence relation $\equiv$ can then be defined as $t_1 \equiv t_2$ if and only if for all $s \in S$, $M(s, t_1) = M(s, t_2)$, or $t_1 \equiv t_2$ if and only if $M^n((s_1, s_2 \ldots s_n), t_1) = M^n((s_1, s_2 \ldots s_n), t_2)$. Since there are, in general, n^n ordered n-tuples that $(s_1, s_2 \ldots s_n)$ can be mapped into, there may be, in general, n^n congruence classes of input sequences. Thus it is clear that the A^n automaton provides a complete picture of the behavior of A. The initial state of A^n is always set to

$(s_1, s_2 \ldots s_n)$. At any subsequent instant, the state of A^n corresponds to the congruence class to which the input sequence belongs. Conventionally, when we specify the behavior of an automaton we have to specify the initial state of A. This specification is equivalent to choosing a particular copy of A among the n copies of A in A^n.

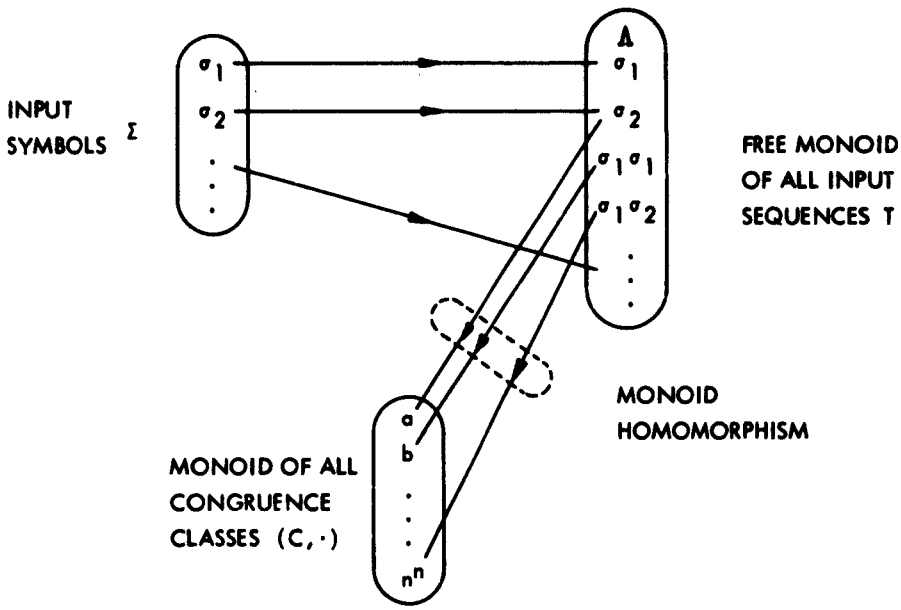


Figure 3

2.5 TRANSITION MATRICES

Since every input sequence t maps the ordered n-tuple $(s_1, s_2 \ldots s_n)$ into another ordered n-tuple, we can consider the mapping as a linear transformation that transforms an n-dimensional vector into another n-dimensional vector. Following Seshu, Miller, and Metze,[4] we represent the initial state of A^n by the column matrix

$$\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix}$$

and the linear transformation by an $n \times n$ matrix $[T]$, called the transition matrix. We then have

$$[T]\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} = \begin{bmatrix} s_i \\ s_j \\ \vdots \\ s_r \end{bmatrix},$$

corresponding to $M^n((s_1, s_2 \ldots s_n), t) = (s_i, s_j \ldots s_r)$. For every input symbol σ_a there is a corresponding transition matrix T_a. For any input sequence $t = \sigma_a\sigma_b\sigma_c \ldots$ the corresponding transition matrix is $T = T_aT_bT_c \ldots$. As we can see, this matrix representation follows exactly the model of the n^{th}-power automaton.

Some of the properties of the transition matrix are stated here for future reference.

(a) Every row contains exactly one nonzero entry, 1. All of the other entries are zeros.

(b) Property (a) is invariant under multiplication of transition matrices.

(c) An identity transformation corresponds to a unit matrix.

(d) The number of columns for which all entries are zero will not decrease under postmultiplication of transition matrices.

Property (d) is not as obvious as the other properties, and it can be shown as follows. Because of property (a), property (d) can be restated: If r of the n nonzero entries in the transition matrix are in columns containing more than one nonzero entry, then the number r will not decrease under postmultiplication of transition matrices. Let $T_aT_b = T$, and let a_{ij}, b_{ij}, and c_{ij} denote the entries in T_a, T_b, and T. Suppose that the p^{th} column of T_a contains u nonzero entries, 1, i.e., $a_{i_1p} = a_{i_2p} = \ldots a_{i_up} = 1$. Since the p^{th} row of T_b contains exactly one nonzero entry 1, suppose that $b_{pk} = 1$. From matrix multiplication,

$$C_{ik} = \sum_{h=1}^{n} a_{ih}b_{hk} = a_{ip}b_{pk} = a_{ip};$$

therefore, $C_{i_1k} = C_{i_2k} = \ldots C_{i_uk} = 1$. Thus property (d) follows.

III. MEMORY SPAN OF AUTOMATA

3.1 MEMORY SPAN WITH RESPECT TO THE INPUT

An automaton is said to have a "memory" because its behavior depends on its past history. Except for the degenerate case of automata consisting of only one state, this statement is generally true. However, the behavior of some automata depends on remote history, while the behavior of others depends only on recent history. In other words, in the former case we say that the automata have a longer memory span, while in the latter case, a shorter one. We shall define these precisely.

DEFINITION 2: An automaton $A = \{\Sigma, S, M\}$ has a finite memory span m if and only if (a) for all s_i, $s_j \in S$ and all $t \in T$ with $L(t) \geq m + 1$ ($L(t)$ means the length of t), $M(s_i, t) = M(s_j, t)$, and (b) there exist s_i, $s_j \in S$ and $t \in T$ with $L(t) = m$, which are such that $M(s_i, t) \neq M(s_j, t)$.

Figure 4 is an example of an automaton having a finite memory span 2.

S \ Σ	0	1
s_1	s_1	s_2
s_2	s_3	s_2
s_3	s_1	s_4
s_4	s_3	s_2

Figure 4

S \ Σ	0	1
s_1	s_1	s_2
s_2	s_2	s_1

Figure 5

DEFINITION 3: An automaton $A = \{\Sigma, S, M\}$ has an infinite memory span if and only if for any given integer N, there exist s_i, $s_j \in S$ and $t \in T$ with $L(t) \geq N$, which are such that $M(s_i, t) \neq M(s_j, t)$.

Figure 5 is an example of an automaton having an infinite memory span.

a. Bounds on the Memory Span

It is not true that an automaton with a given number of states might have a finite memory span equal to any positive integer. We shall obtain an upper bound and a lower bound on the values of memory span.

DEFINITION 4: In the square automaton (second-power automaton) A^2 and the

unordered square automaton $A^{\{2\}}$, the states (s_i, s_j) in A^2 and $\{s_i, s_j\}$ in $A^{\{2\}}$, where $s_i \neq s_j$, are called "compound" states; the states (s_i, s_i) in A^2 and $\{s_i\}$ in $A^{\{2\}}$ are called "simple" states.

Consider A^2 and $A^{\{2\}}$ corresponding to two copies of A running in parallel. A compound state signifies that the two copies of A are in different states, while a simple state signifies that the two copies of A are in the same state.

DEFINITION 5: There is a loop around a state s_i in $A = \{\Sigma, S, M\}$ if there exists a nonempty input sequence t, that is, $L(t) \geq 1$, which is such that $M(s_i, t) = s_i$.

LEMMA 1: If there is a loop around a compound state $\{s_i, s_j\}$ in $A^{\{2\}}$, there is a loop around (s_i, s_j) and a loop around (s_j, s_i) in A^2.

PROOF OF LEMMA 1: There exists a nonempty input sequence t that is such that $M^{\{2\}}(\{s_i, s_j\}, t) = \{s_i, s_j\}$, that is, $\{M(s_i, t), M(s_j, t)\} = \{s_i, s_j\}$. Since $s_i \neq s_j$ and the function M is single-valued, we have either

CASE I. $M(s_i, t) = s_i$ and $M(s_j, t) = s_j$, or

CASE II. $M(s_i, t) = s_j$ and $M(s_j, t) = s_i$.

For Case I, $M^2((s_i, s_j), t) = (M(s_i, t), M(s_j, t)) = (s_i, s_j)$.

For Case II, $M^2((s_i, s_j), tt) = (M(M(s_i, t), t), M(M(s_j, t), t)) = (M(s_j, t), M(s_i, t)) = (s_i, s_j)$.

Q.E.D.

DEFINITION 6: In an automaton $A = \{\Sigma, S, M\}$, a set of states $s_i, s_j \ldots \in S$ is "structurally indistinguishable," if for all $\sigma \in \Sigma$, $M(s_i, \sigma) = M(s_j, \sigma) = \ldots$.

Obviously, any state is structurally indistinguishable from itself. In this report, we shall abbreviate the term "structurally indistinguishable" to "indistinguishable."

DEFINITION 7: In an automaton $A = \{\Sigma, S, M\}$, a (homomorphic) mapping c, called "combining," is defined as: $c(s_i) = c(s_j) = \ldots = s_i$ for the states of an indistinguishable set $s_i, s_j \ldots$. In other words, c maps a set of indistinguishable states into an arbitrarily chosen representative state in that set.

DEFINITION 8: Given an automaton $A = \{\Sigma, S, M\}$, we define the combined automaton $A_{c1} = \{\Sigma, S_{c1}, M_{c1}\}$, where (a) $c(s) \in S_{c1}$ for all $s \in S$, and (b) $M_{c1}(s_i, \sigma) = c(M(c^{-1}(s_i), \sigma))$ for all $\sigma \in \Sigma$ and all $s_i \in S_{c1}$. Here, $c^{-1}(s_i)$ means the inverse image of s_i under c. Although $c^{-1}(s_i)$ is not unique, $M(c^{-1}(s_i), \sigma)$ is unique (by the definition of indistinguishable states).

In general, there may be indistinguishable sets in A_{c1} which can be combined further. We adopt the notation A_{c2}, A_{c3} ... for successively combined automata.

LEMMA 2: If there is a loop around a compound state $\{s_i, s_j\}$ in $A_{c1}^{\{2\}}$, then there exists a loop around one of the compound states $\{c^{-1}(s_i), c^{-1}(s_j)\}$ in $A^{\{2\}}$. (Since $c^{-1}(s_i)$ and $c^{-1}(s_j)$ are not unique, $\{c^{-1}(s_i), c^{-1}(s_j)\}$ denotes all possible pairs of states from $c^{-1}(s_i)$ and $c^{-1}(s_j)$.)

PROOF OF LEMMA 2: Let $\{s_{i1}, s_{i2} \ldots s_{iu}\}$ and $\{s_{j1}, s_{j2} \ldots s_{jv}\}$ denote the sets of

indistinguishable states in A which are mapped into s_i and s_j in A_{c1}, respectively. There exists a nonempty input sequence t so that $M_{c1}^{\{2\}}(\{s_i, s_j\}, t) = \{M_{c1}(s_i, t), M_{c1}(s_j, t)\} = \{s_i, s_j\}$. Correspondingly, $M^{\{2\}}(\{s_{i1}, s_{j1}\}, t) = \{s_{ip}, s_{jq}\}$, where $s_{ip} \in \{s_{i1}, s_{i2} \cdots s_{iu}\}$ and $s_{jq} \in \{s_{j1}, s_{j2} \cdots s_{jv}\}$, since $c(s_{ip}) = s_i$, $c(s_{jq}) = s_j$. Because $\{s_{i1}, s_{i2} \cdots s_{iu}\}$ and $\{s_{j1}, s_{j2} \cdots s_{jv}\}$ are sets of indistinguishable states and t is nonempty, it follows that

$$M^{\{2\}}(\{s_{i1}, s_{j1}\}, t) = M^{\{2\}}(\{s_{i1}, s_{j2}\}, t) = \cdots M^{\{2\}}(\{s_{i1}, s_{jv}\}, t) = \cdots$$

$$= M^{\{2\}}(\{s_{iu}, s_{jv}\}, t) = \{s_{ip}, s_{jq}\}.$$

Among all of these u · v equalities, there must be one that reads

$$M^{\{2\}}(\{s_{ip}, s_{jq}\}, t) = \{s_{ip}, s_{jq}\}. \qquad \text{Q.E.D.}$$

Lemma 2 can be extended to: If there is a loop around a compound state in A_{cr}, where A_{cr} is any one of the series of automata A_{c1}, A_{c2}, A_{c3} ..., then there is a loop around a corresponding compound state in A.

LEMMA 3: If $A = \{\Sigma, S, M\}$ contains two or more internal states, and if for all s_i, $s_j \in S$, there exists $\sigma \in \Sigma$ that is such that $M(s_i, \sigma) \neq M(s_j, \sigma)$, then there is a loop around some compound state in $A^{\{2\}}$.

PROOF OF LEMMA 3: Starting with any compound state $\{s_a, s_b\}$ in $A^{\{2\}}$, we find that there exists $\sigma_1 \in \Sigma$ that is such that $M^{\{2\}}(\{s_a, s_b\}, \sigma_1) = \{M(s_a, \sigma_1), M(s_b, \sigma_1)\}$, with $M(s_a, \sigma_1) \neq M(s_b, \sigma_1)$. Also, there exists $\sigma_2 \in \Sigma$ that is such that $M^{\{2\}}(\{M(s_a, \sigma_1), M(s_b, \sigma_1)\}, \sigma_2) = \{M(s_a, \sigma_1\sigma_2), M(s_b, \sigma_1\sigma_2)\}$, with $M(s_a, \sigma_1\sigma_2) \neq M(s_b, \sigma_1\sigma_2)$.

We can repeat this argument for as many input symbols as we wish and still end with a compound state. For an automaton A containing n internal states there are $n(n-1)/2$ compound states in its unordered square automaton $A^{\{2\}}$. If we repeat this argument for more than $n(n-1)/2$ steps, at least one compound state is visited more than once. Thus there is a loop around that compound state. Q.E.D.

We are now able to prove the following theorem.

THEOREM 3: An automaton consisting of n internal states has a memory span either equal to infinity or equal to n-2 or less.

PROOF 3: Suppose that an automaton $A = \{\Sigma, S, M\}$ has a finite memory span larger than n-2; that is, there exist states s_i, s_j and an input sequence t of length n-1 which are such that $M(s_i, t) \neq M(s_j, t)$. We claim that the automaton then must have an infinite memory span. We have to show that there is a loop around a compound state in A^2. However, from Lemma 1, we have only to show that there is a loop around a compound state in $A^{\{2\}}$. The portion of the transition diagram of $A^{\{2\}}$ that corresponds to the input sequence $t = \sigma_1\sigma_2 \cdots \sigma_{n-1}$ is shown in Fig. 6. If within the n-1 steps of transition

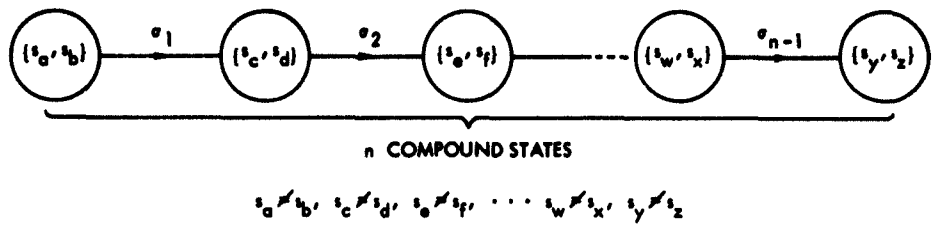


Figure 6

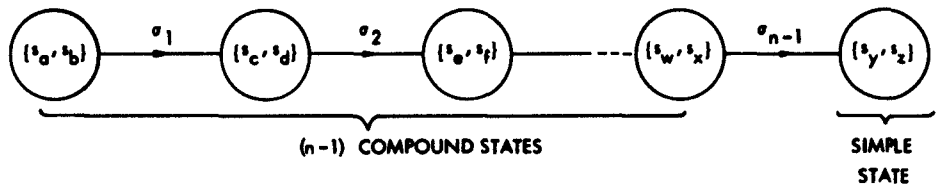


Figure 7

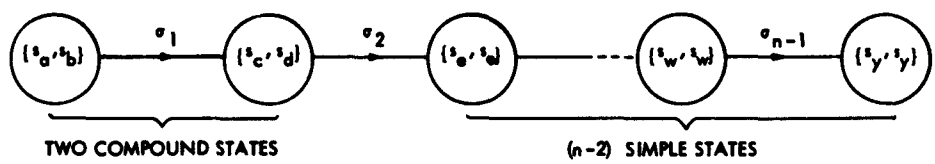


Figure 8

any compound state is visited more than once, there is certainly a loop around that state; if not, we repeatedly combine all sets of indistinguishable states to form A_{cr} (the r^{th}-combined automaton) either until no further combination is possible or until s_y and s_z become two indistinguishable states in A_{cr}. (Notice that the n compound states in Fig. 6 are still compound states in $A_{cr}^{\{2\}}$ because none of the n pairs of states would become indistinguishable before s_y and s_z are combined.) We can now examine two possibilities: (a) No further combination is possible in A_{cr}. Because A_{cr} contains at least two states, s_y and s_z, by Lemma 3 there is a loop around a compound state in $A_{cr}^{\{2\}}$. (b) The pairs s_y and s_z are indistinguishable in A_{cr}. Combine s_y and s_z to form $A_{c(r+1)}$. The portion of the transition diagram of $A_{c(r+1)}^{\{2\}}$ that corresponds to the input sequence t is shown in Fig. 7. If s_w and s_x are distinguishable, we return to the argument of possibility (a). If s_w and s_x are indistinguishable, we combine them to form $A_{c(r+2)}$ and repeat the argument above. The procedure of combining

indistinguishable states can go on, at most, to $A_{c(r+n-2)}$, since the portion of the transition diagram of $A^{\{2\}}_{c(r+n-2)}$ corresponding to the input sequence t is that which is shown in Fig. 8. Since after (n-2) steps of combining indistinguishable states, $A_{c(r+n-2)}$ would have only two states, $\{s_a, s_b\} = \{s_c, s_d\}$ in Fig. 8. There is still an input symbol σ_1, however, that will lead a compound state to a compound state. Therefore, there is a loop around a compound state in $A^{\{2\}}_{c(r+n-2)}$.

By Lemmas 1 and 2, we then prove Theorem 3. Q.E.D.

The proof of this theorem brings up an interesting point. For an automaton $A = \{\Sigma, S, M\}$ consisting of n states, even when the function M is only partially specified (that is, the flow table is partially specified), as long as the specified values of M ensure the existence of an input sequence of length n-1 which will bring any two different states into different states, the automaton has an infinite memory span no matter how the rest of the values of M are specified. As an example, no matter which values are assigned for the bars in the flow table in Fig. 9, the automaton will have an infinite memory span.

S \ Σ	0	1
s_1	s_2	-
s_2	s_3	s_1
s_3	s_2	s_4
s_4	-	-

$M(s_1, 001) = s_4$

$M(s_2, 001) = s_1$

Figure 9

THEOREM 4: If a strongly connected automaton[3] consisting of n states has a finite memory span m, then $\alpha^{m+1} \geq n$, where α is the number of possible input symbols.

PROOF 4: There are α^{m+1} different input sequences of length m+1. Starting from any arbitrary state s_i, the automaton can enter, at most, α^{m+1} different final states after m+1 steps of transition. Suppose that $n > \alpha^{m+1}$. There then must be a state (or states), say s_k, that is not included in these α^{m+1} final states. However, there exist an input sequence t of length m+1 and an initial state s_j which are such that $M(s_j, t) = s_k$. Then $M(s_j, t) \neq M(s_i, t)$ is a contradiction of the statement that the memory span equals m. Q.E.D.

b. Canonical Form of an Automaton Having a Finite Memory Span

Let us define a series of relations R_o, R_1, R_2 ... R_{k-1}, R_k ... as follows:

R_o: $s_i R_o s_j$ if and only if $M(s_i, \Lambda) = M(s_j, \Lambda)$, where Λ denotes the empty input sequence. (Since we assume that all states in the automaton are distinct, only $s_i R_o s_i$ is true.)

R_1: $s_i R_1 s_j$ if and only if $M(s_i, \sigma) R_o M(s_j, \sigma)$ for any input symbol σ.

R_k: $s_i R_k s_j$ if and only if $M(s_i, \sigma) R_{k-1} M(s_j, \sigma)$ for any input symbol σ. In other words, $s_i R_k s_j$ if and only if $M(s_i, t) = M(s_j, t)$ for any input sequence t of length k.

It is obvious that all of the R's are equivalence relations, and thus they define a series of partitions P_o, P_1 ... P_{k-1}, P_k ... on the set of internal states.

We claim that P_{k-1} is a refinement of P_k, that is, for any pair of states s_i and s_j, if $s_i R_{k-1} s_j$, then $s_i R_k s_j$. $s_i R_{k-1} s_j$ means that for any input sequence t of length k-1, $M(s_i, t) = M(s_j, t)$. Thus $M(s_i, t\sigma) = M(s_j, t\sigma)$ for any input symbol σ. Therefore, $s_i R_k s_j$. The procedure for forming these successive partitions will terminate whenever P_{j-1} is identical with P_j. Therefore, this procedure will terminate, at most, at P_{n-1} because P_{n-1} will consist of only one block. (n is the number of internal states.) Thus we have the following theorems.

THEOREM 5: An automaton will have an infinite memory span if the procedure for forming successive partitions P_o, P_1 ... terminates at a partition P_j that consists of more than one block.

PROOF 5: If the partition terminates at P_j, then for any two states s_i and s_j that are not in the same block of P_j, there is an input symbol σ_1 that is such that $M(s_i, \sigma_1)$ and $M(s_j, \sigma_1)$ are not in the same block in P_j. (Otherwise, $P_{j+1} \neq P_j$.) Repeating this argument for the states $M(s_i, \sigma_1)$ and $M(s_j, \sigma_1)$, we have an input symbol σ_2 that is such that $M(s_i, \sigma_1\sigma_2)$ and $M(s_j, \sigma_1\sigma_2)$ are not in the same block in P_j. Repeating again, we can have an input sequence t of any arbitrary length so that $M(s_i, t)$ and $M(s_j, t)$ are not in the same block. Therefore, $M(s_i, t) \neq M(s_j, t)$ because P_j is a partition (blocks in P_j are disjoint). Q.E.D.

THEOREM 6: An automaton will have a finite memory span j-1 if the series of partitions P_o, P_1 ... terminates at a P_j that consists of only one block.

PROOF 6: $s_i R_j s_j$ means that for any input sequence of length j, $M(s_i, t) = M(s_j, t)$. Since P_j consists of only one block, this equality holds true for any pair of states. On the other hand, since P_{j-1} consists of two or more blocks, there exist a pair of states s_i and s_j and an input sequence t of length j-1 which are such that $M(s_i, t) \neq M(s_j, t)$. Q.E.D.

Theorem 3 can be proved alternatively as follows. Since the series of partitions P_o, P_1 ... will terminate, at most, at P_{n-1}, then $M(s_i, t) = M(s_j, t)$ for any input sequence t of length n-1 and any pair of states s_i and s_j.

S \ Σ	0	1
s_1	s_1	s_2
s_2	s_3	s_2
s_3	s_1	s_4
s_4	s_3	s_2

A

S \ Σ	0	1
s_1	s_1	s_2
s_2	s_3	s_2
s_3	s_1	s_2

A_{c1}

S \ Σ	0	1
s_1	s_1	s_2
s_2	s_1	s_2

A_{c2}

S \ Σ	0	1
s_1	s_1	s_1

A_{c3}

Figure 10

S \ I	0	1
s_1	s_3	s_4
s_2	s_3	s_2
s_3	s_1	s_4
s_4	s_3	s_2

B

S \ I	0	1
s_1	s_3	s_2
s_2	s_3	s_2
s_3	s_1	s_2

B_{c1}

S \ I	0	1
s_1	s_3	s_1
s_3	s_1	s_1

B_{c2}

Figure 11

This series of partitions P_0, P_1 ... corresponds exactly to the step-by-step combinations of indistinguishable states. A set of states that are in the same block in P_k is indistinguishable after k steps of combination. Therefore, the canonical form of an automaton having a finite memory span is one that can be reduced to an automaton consisting on only one internal state after repeated combinations of indistinguishable states. (See Fig. 10 for an automaton with a finite memory span and Fig. 11 for one with an infinite memory span.)

c. Essential Part of $A^{\{n\}}$

Suppose that we do not know the initial state of an automaton but do know the input sequence to the automaton and would like to determine the present state of the automaton. In general, with such limited knowledge, it is impossible to determine the present state. If, however, the automaton has a finite memory span, say equal to m, the present state can be determined when (m+1) or more of the most recent input symbols are known. From the definition of an automaton having a finite memory span m, the automaton will terminate at the same state for a given input sequence of length that is equal to or larger than (m+1) no matter what the initial state is. In other words, although there is initially complete ambiguity about the state of the automaton (the automaton can be in any initial state), this ambiguity will eventually be removed after m+1 steps of transition. If we look at the unordered n^{th}-power automaton $A^{\{n\}}$, we can see the manner in which the ambiguity is narrowed down step by step. From the initial set of states $\{s_1, s_2 \ldots s_n\}$ (any state could be the initial state), a transition will lead to another set of states $\{s_i, s_j \ldots s_r\}$ which are the possible states that the automaton will be in after that transition. As we can see, the number of states in the set $\{s_i, s_j \ldots s_r\}$ indicates the amount of ambiguity that exists, as far as the state that the automaton may be in is concerned. From property (d) of the transition matrices (see Sec. II) we know that this ambiguity will never increase. Further transitions may further decrease the ambiguity. If no ambiguity exists after any (m+1) steps of transition, the automaton is said to have a finite memory span. In order to determine the memory span of an automaton it is not necessary to examine the complete $A^{\{n\}}$.

We can look at only a portion of $A^{\{n\}}$ which consists of all states that can be reached from $\{s_1, s_2 \ldots s_n\}$. This part of $A^{\{n\}}$ carries all of the information that we need, and is called the essential part of $A^{\{n\}}$. (See Fig. 12.) In the example in Fig. 12 after, at most, three transitions, no ambiguity about the state of the automaton will exist. The diagram of the essential part of $A^{\{n\}}$ can be illustrated in a more explicit fashion, called the tree of the essential part of $A^{\{n\}}$; the tree of the essential part of $A^{\{n\}}$ of Fig. 12 is illustrated in Fig. 13. Notice that we stop extending any branch of the tree once ambiguity no

S \ Σ	0	1
s_1	s_1	s_2
s_2	s_3	s_2
s_3	s_1	s_4
s_4	s_3	s_2

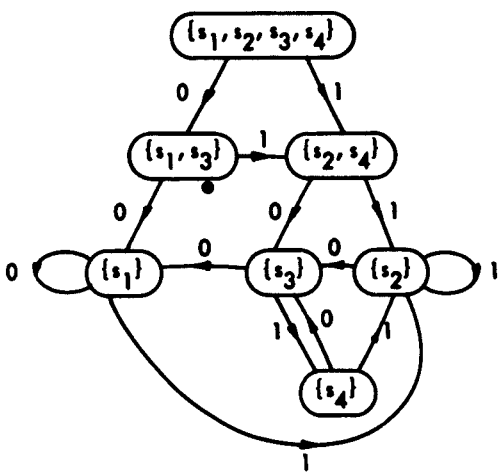


Figure 12

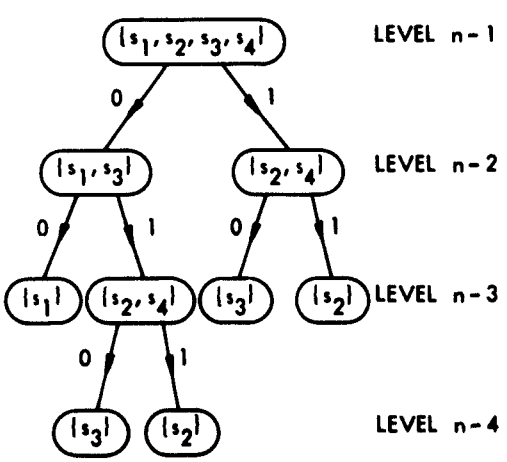


Figure 13

S \ Σ	0	1
s_1	s_3	s_4
s_2	s_3	s_2
s_3	s_1	s_4
s_4	s_3	s_2

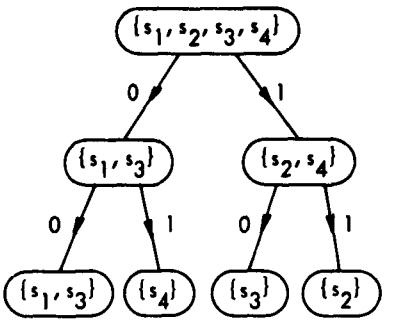


Figure 14

longer exists along the branch.

Therefore the tree of the essential part of $A^{\{n\}}$ can be employed to check the memory span of an automaton. As indicated above, after n-1 steps of transition (n is the number of internal states of the automaton), no ambiguity will exist for an automaton having a finite memory span. On the other hand, a certain ambiguity after (n-1) steps of transition or a loop around any set containing more than one state indicates that the automaton has an infinite memory span. The automaton in Fig. 14 has an infinite memory span because of the loop around $\{s_1, s_3\}$.

d. A Series of Partitions Related to the Tree of Essential Part of $A^{\{n\}}$

We define a series of relations T_{n-1}, T_{n-2}, ... T_{n-k+1}, T_{n-k} ... as follows:

T_{n-1}: $s_i \; T_{n-1} \; s_j$ for any pair of states s_i and s_j.

T_{n-2}: If $s_i \; T_{n-1} \; s_j$, then $M(s_i, \sigma) \; T_{n-2} \; M(s_j, \sigma)$ for any $\sigma \in \Sigma$.

(If there exists a state s_t that is such that no state s_i and input will give $M(s_i, \sigma) = s_t$, we define $s_t \; T_{n-2} \; s_t$. The same definition is applied in the successive relations.)

⋮

T_{n-k}: If $s_i \; T_{n-k+1} \; s_j$, then $M(s_i, \sigma) \; T_{n-k} \; M(s_j, \sigma)$ for any $\sigma \in \Sigma$.

Moreover, all of the T's are defined as transitive relations. In other words, if $s_i \; T_{n-k} \; s_j$ and $s_j \; T_{n-k} \; s_m$, we then have $s_i \; T_{n-k} \; s_m$ by definition. We can show that the T's are equivalence relations:

(a) Reflexivity: Proved by induction. T_{n-1} is reflexive because by assumption $s_i \; T_{n-1} \; s_j$ for any s_i, s_j. Assume that T_{n-k+1} is reflexive. For any s_i, if there exist s_k and σ that are such that $M(s_k, \sigma) = s_i$, then $s_i \; T_{n-k} \; s_i$ because $s_k \; T_{n-k+1} \; s_k$; if there does not exist s_k and σ that are such that $M(s_k, \sigma) = s_i$, then $s_i \; T_{n-k} \; s_i$ by definition.

(b) Transitivity: By assumption.

(c) Symmetry: Proved by induction. T_{n-1} is obviously symmetric. Assume that T_{n-k+1} is symmetric. Let $s_i \; T_{n-k} \; s_j$.

CASE I: There exist s_q, s_m, and input symbol σ that are such that $M(s_q, \sigma) = s_i$ and $M(s_m, \sigma) = s_j$, where $s_q \; T_{n-k+1} \; s_m$. By assumption, $s_q \; T_{n-k+1} \; s_m$ gives $s_m \; T_{n-k+1} \; s_q$, and thus $M(s_m, \sigma) \; T_{n-k} \; M(s_q, \sigma)$. Therefore, $s_j \; T_{n-k} \; s_i$.

CASE II: There exists a chain relation $s_i \; T_{n-k} \; s_a \; T_{n-k} \; s_b \; T_{n-k} \; s_c \cdots T_{n-k} \; s_j$ and exist s_q, s_t, s_u, s_v, s_w, s_x, $s_y \cdots$ and σ_1, σ_2, σ_3, $\sigma_4 \cdots$ that are such that $s_q \; T_{n-k+1} \; s_t$, $s_u \; T_{n-k+1} \; s_v$, $s_w \; T_{n-k+1} \; s_x$, $s_y \; T_{n-k+1} \; s_m$ and

$$M(s_q, \sigma_1) = s_i \qquad M(s_t, \sigma_1) = s_a$$

$$M(s_u, \sigma_2) = s_a \qquad M(s_v, \sigma_2) = s_b$$

$$M(s_w, \sigma_3) = s_b \qquad M(s_x, \sigma_3) = s_c$$

$$M(s_y, \sigma_4) = s_c \qquad M(s_m, \sigma_4) = s_j,$$

or pictorially as shown in Fig. 15. As proved in Case I, we have $s_a \; T_{n-k} \; s_i$, $s_b \; T_{n-k} \; s_a$, $s_c \; T_{n-k} \; s_b$, $s_j \; T_{n-k} \; s_c$. Thus, $s_j \; T_{n-k} \; s_i$. Therefore, the relations T_{n-1}, T_{n-2}, $\cdots$

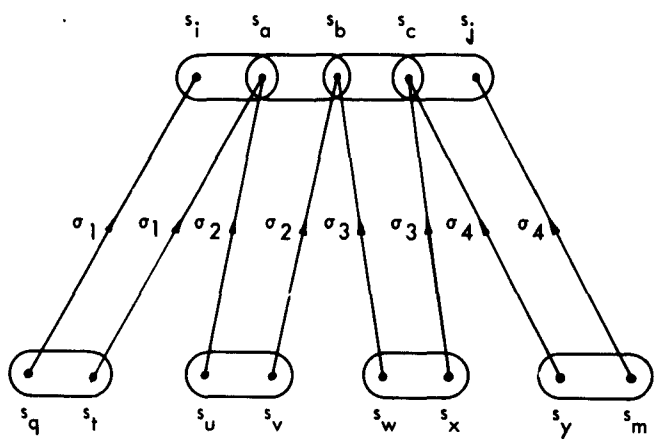


Figure 15

$T_{n-k} \cdots$ define a series of partitions Q_{n-1}, Q_{n-2}, $\cdots$ $Q_{n-k} \cdots$.

We claim that Q_{n-k} is a refinement of Q_{n-k+1}. This can be proved by induction. Q_{n-2} is obviously a refinement of Q_{n-1}. Assume that Q_{n-k+1} is a refinement of Q_{n-k+2}. Let $s_i \; T_{n-k} \; s_j$, and we have two cases that are identical with the last two cases.

CASE I: There exist s_q, s_m, and input symbol σ that are such that $M(s_q, \sigma) = s_i$

and $M(s_m, \sigma) = s_j$, where $s_q \; T_{n-k+1} \; s_m$. By assumption, $s_q \; T_{n-k+2} \; s_m$ and thus $M(s_q, \sigma) \; T_{n-k+1} \; M(s_m, \sigma)$. Therefore, $s_i \; T_{n-k+1} \; s_j$.

CASE II: There exists a chain relation as shown in Fig. 15. By assumption, $s_q \; T_{n-k+2} \; s_t$, $s_u \; T_{n-k+2} \; s_v$, $s_w \; T_{n-k+2} \; s_x$, $s_y \; T_{n-k+2} \; s_m$. Thus, $s_i \; T_{n-k+1} \; s_a$, $s_a \; T_{n-k+1} \; s_b$, $s_b \; T_{n-k+1} \; s_c$, $s_c \; T_{n-k+1} \; s_j$. Therefore, $s_i \; T_{n-k+1} \; s_j$.

For any s_i that is in a block consisting of more than one state in Q_{n-k}, there exist a state s_j in the same block as s_i, a pair of states s_q and s_m, and an input sequence t of length k-1, which are such that $M(s_q, t) = s_i$ and $M(s_m, t) = s_j$. This statement is not quite obvious and can be shown as follows. The partition Q_{n-k} is closely related to sets of states at level n-k of the tree of the essential part of $A^{\{n\}}$. As a matter of fact, from the tree of the essential part of $A^{\{n\}}$, if we put the unions of all sets of states that are chain-connected into blocks and put all states that are not reachable at level n-k into separate individual blocks, we shall have Q_{n-k}. This fact can be proved by induction. It is certainly true for Q_{n-1} and Q_{n-2} by the definition of Q_{n-1} and Q_{n-2}. Suppose that this is true for Q_{n-k+1}. Let $\{I_1\}$, $\{I_2\}$, $\{I_3\}$, $\{J_1\}$, $\{J_2\}$, $\{J_3\}$ ... $\{K_1\}$, $\{K_2\}$, $\{K_3\}$ ... denote sets of states at level n-k+1 in the tree of the essential part of $A^{\{n\}}$, where the I's, as well as the J's and K's, are chain-connected. The sets of states at level n-k can be denoted by $M^{\{n\}}(\{I_1\}, \sigma_1)$, $M^{\{n\}}(\{I_2\}, \sigma_1)$, $M^{\{n\}}(\{I_3\}, \sigma_1)$, $M^{\{n\}}(\{I_1\}, \sigma_2)$ .... On the other hand, Q_{n-k+1} consists of blocks $\{\{I_1\} \cup \{I_2\} \cup \{I_3\}\}$, $\{\{J_1\} \cup \{J_2\} \cup \{J_3\}\}$, ... $\{V_1\}$, $\{V_2\}$, where $\{V_1\}$ and $\{V_2\}$ are states that are not reachable at level n-k+1. Then Q_{n-k} would consist of $M^{\{n\}}(\{\{I_1\} \cup \{I_2\} \cup \{I_3\}\}, \sigma_1)$, $M^{\{n\}}(\{\{I_1\} \cup \{I_2\} \cup \{I_3\}\}, \sigma_2)$, $M^{\{n\}}(\{\{J_1\} \cup \{J_2\} \cup \{J_3\}\}, \sigma_1)$, $M^{\{n\}}(\{\{J_1\} \cup \{J_2\} \cup \{J_3\}\}, \sigma_2)$ ... $M^{\{n\}}(\{V_1\}, \sigma_1)$, $M^{\{n\}}(\{V_1\}, \sigma_2)$, with the chain-connected sets combined into blocks and also would consist of states that are not reachable at level n-k. Since $\{I_1\}$, $\{I_2\}$, $\{I_3\}$ are chain-connected, $M^{\{n\}}(\{I_1\}, \sigma_1)$, $M^{\{n\}}(\{I_2\}, \sigma_1)$, $M^{\{n\}}(\{I_3\}, \sigma_1)$ are chain-connected. Therefore, $M^{\{n\}}(\{I_1\}, \sigma_1) \cup M^{\{n\}}(\{I_2\}, \sigma_1) \cup M^{\{n\}}(\{I_3\}, \sigma_1) = M^{\{n\}}(\{\{I_1\} \cup \{I_2\} \cup \{I_3\}\}, \sigma_1)$. Since V_1 consists of only a single state, $M^{\{n\}}(\{V_1\}, \sigma_1)$ either is a state that is not reachable at level n-k or is included in some other set. This completes the proof that Q_{n-k} is formed by the unions of connecting sets plus individual blocks for states that are not reachable at level n-k. Therefore, if s_i is in a block consisting of more than one state in Q_{n-k}, there must be a set of states consisting of s_i and s_j at level n-k of the tree of the essential part of $A^{\{n\}}$.

Theorems 7 and 8 will then follow immediately.

THEOREM 7: An automaton will have an infinite memory span if the procedure for forming successive partitions Q_{n-1}, Q_{n-2} ... terminates at Q_{n-j}, a partition that consists of fewer than n blocks (that is, there is a block consisting of more than one state).

PROOF 7: The termination of the series of partitions means that if we continue to form partitions for successive steps, they will all be identical with Q_{n-j}. This result

implies that for all input sequences of any length there is at least one subset consisting of more than one state at that level of the tree of the essential part of $A^{\{n\}}$. Q.E.D.

THEOREM 8: An automaton will have a finite memory span j - 2 if the series of successive partitions terminates at Q_{n-j}, a partition that consists of n blocks.

PROOF 8: The proof of this theorem is obvious.

Moreover, Theorem 3 can be proved immediately by observing that the series of Q's are refinements of the previous Q's.

Another interesting point comes up from the examination of this series of partitions. We can tell that an automaton has an infinite memory span, even when the function M is

S \ Σ	0	1
s_1	s_2	-
s_2	-	s_4
s_3	s_3	-
s_4	s_6	s_1
s_5	-	s_3
s_6	s_5	-

$$Q_{n-1} = \{\overline{s_1, s_2, s_3, s_4, s_5, s_6}\}$$

$$Q_{n-2} = \{\overline{s_1, s_2, s_3, s_4, s_5, s_6}\}$$

Figure 16

partially specified, if the specified values of M ensure that the series of partitions Q_{n-1}, Q_{n-2} ... terminates at one consisting of less than n blocks. (See Fig. 16.) Since $Q_{n-1} = Q_{n-2}$ in Fig. 16, the automaton will have an infinite memory span regardless of the values assigned to the barred entries in the flow table.

e. Partial Information from Input Sequences

For an automaton having a finite memory span m, any input sequence of length (m+1) will furnish complete information about the present state of the automaton. On the other hand, for an automaton having an infinite memory span, there always exists at least one input sequence of any arbitrary length which does not supply enough information to enable the unique specification of the state of the automaton. Although the complete information about the state of the automaton will not be furnished by some input sequences, there may be some partial information available. In other words, although the input sequences cannot completely remove the ambiguity about the present state of the automaton when the initial state is unknown, they can at least narrow down the ambiguity to a certain

extent, as can be seen from the tree of the essential part of $A^{\{n\}}$. For example, let us look at the automaton in Fig. 14. If the input sequence is a string of zeros, then we never can tell exactly whether the automaton is in s_1 or s_3, regardless of the number of zeros that we may observe. Knowing that the input is a "0" (a "1"), however, enables us to determine that the state of the automaton is either s_1 or s_3 (either s_2 or s_4) and reduces the ambiguity from four possible states to two possible states. Therefore, if we know the previous input symbols (0 or 1), we only need the extra information to distinguish s_1 from s_3 or s_2 from s_4 in order to specify uniquely the present state of the automaton. For this purpose, we can extend the tree of the essential part of $A^{\{n\}}$ to a certain level and determine the ambiguity, corresponding to each input sequence, about the state of the automaton at that level. As we have indicated, whenever there is a loop around any set containing more than one state, further extension of the tree will not reduce the ambiguity within that set of states.

f. New Model of an Automaton

As indicated in Section I, a finite automaton is characterized by a quintuple $A = \{\Sigma, Z, S, M, N\}$. The functions M and N can be written more explicitly as

$$M(s^t, \sigma^t) = s^{t+1}$$

$$N(s^t, \sigma^t) = s^t,$$

where the superscript t is the index of successive time periods. From our discussion, we see that the past input sequence plus some extra information will specify uniquely the state of the automaton. This extra information is needed whenever the automaton does not have a finite memory span. Suppose that k of the latest input symbols to the automaton are known. The set of states corresponding to this input sequence at level $n - k - 1$ in the tree of the essential part of $A^{\{n\}}$ will be the set of possible present states of the automaton. For all possible input sequences of length k, the set that contains the largest number of states at level $n - k - 1$ displays the largest possible ambiguity about the present state of the automaton that k of the latest input symbols are not capable of removing. We now define a new variable, called the "substate" of the automaton, which will assume the values $r_1, r_2, r_3 \ldots$. When k of the latest input symbols are known, the number of substates will be equal to the number of states in the largest set at level $n - k - 1$ in the tree of the essential part of $A^{\{n\}}$. We shall then assign different substates to denote different states in the same set, so that knowing the past input sequence and the substate will be sufficient to determine the state of the automaton. We may then write

$$s^t = H(\sigma^{t-1}, \sigma^{t-2}, \ldots \sigma^{t-k}, r^t),$$

where H is the functional relation between s^t and σ^{t-1}, $\sigma^{t-2} \ldots r^t$. Since $M(s^t, \sigma^t) = s^{t+1}$, we have

$$M(H(\sigma^{t-1}, \sigma^{t-2} \ldots \sigma^{t-k}, r^t), \sigma^t) = H(\sigma^t, \sigma^{t-1} \ldots \sigma^{t-k+1}, r^{t+1}).$$

In general, changing the value of any argument of the function M changes the value of s^{t+1}, which, in turn, would change the value of r^{t+1}. Therefore, we can express explicitly

$$r^{t+1} = F(\sigma^t, \sigma^{t-1}, \ldots \sigma^{t-k+1}, \sigma^{t-k}, r^t).$$

We propose now that an automaton can be described by a quintuple $A = \{\Sigma, Z, R, F, G\}$. $\Sigma = \{\sigma_1, \sigma_2 \ldots\}$ and $Z = \{z_1, z_2 \ldots\}$ are the finite set of input symbols and output symbols; $R = \{r_1, r_2 \ldots\}$ is the set of substates; F is a function that maps all ordered (k+2)-tuples $(\sigma^t, \sigma^{t-1} \ldots \sigma^{t-k}, r^t)$ into R, as expressed above; G is a function that maps all of these ordered (k+2)-tuples into Z, namely

$$z^t = G(\sigma^t, \sigma^{t-1} \ldots \sigma^{t-k+1}, \sigma^{t-k}, r^t).$$

Therefore, given an automaton represented in the conventional way $A = \{\Sigma, Z, S, M, N\}$, we can have another equivalent representation $A = \{\Sigma, Z, R, F, G\}$, where the set R and

S \ Σ	0	1
s_1	$s_1, 0$	$s_5, 1$
s_2	$s_2, 1$	$s_5, 1$
s_3	$s_1, 1$	$s_3, 1$
s_4	$s_2, 0$	$s_6, 1$
s_5	$s_1, 1$	$s_3, 0$
s_6	$s_4, 0$	$s_3, 0$

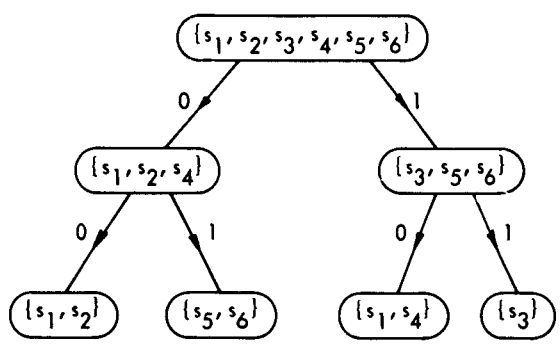


Figure 17

the functions F and G are different for different values of k, the number of known past input symbols. The function H defined above will then indicate the relations between

R^t \ $\Sigma^{t-1}\Sigma^t$	00	01	10	11
r_1	$r_1, 0$	$r_2, 1$	$r_1, 1$	$r_1, 1$
r_2	$r_2, 1$	$r_2, 1$	$r_1, 1$	$r_1, 0$
r_3	$r_2, 0$	$r_3, 1$	$r_3, 0$	$r_1, 0$

Figure 18

the two representations. For the automaton illustrated in Fig. 17 we construct the function H for k = 1 and k = 2, as shown in Figs. 18 and 19, respectively.

For k = 1, we define

$$H(0, r_1) = s_1 \qquad H(0, r_2) = s_2 \qquad H(0, r_3) = s_4$$

$$H(1, r_1) = s_3 \qquad H(1, r_2) = s_5 \qquad H(1, r_3) = s_6$$

and obtain the table illustrated in Fig. 18. For example,

$$M(H(0, r_1), 0) = M(s_1, 0) = s_1 = H(0, r_1)$$

$$N(H(0, r_1), 0) = N(s_1, 0) = 0$$

$$M(H(0, r_1), 1) = M(s_1, 1) = s_5 = H(1, r_2)$$

$$N(H(0, r_1), 1) = N(s_1, 1) = 1, \text{ etc.}$$

For k = 2, we define

$$H(00, r_1) = s_1 \qquad H(00, r_2) = s_2$$

$$H(01, r_1) = s_5 \qquad H(01, r_2) = s_6$$

$$H(10, r_1) = s_1 \qquad H(10, r_2) = s_4$$

$$H(11, r_1) = s_3 \qquad H(11, r_2) = s_3$$

and obtain the table illustrated in Fig. 19. Notice that H can have the same value for different arguments, for example, $H(00, r_1) = s_1$, $H(10, r_1) = s_1$, because s_1 has appeared

R^t \ $\Sigma^{t-2}\Sigma^{t-1}\Sigma^t$	000	001	010	011	100	101	110	111
r_1	$r_1, 0$	$r_1, 1$	$r_1, 1$	r_1 or $r_2, 0$	$r_1, 0$	$r_1, 1$	$r_1, 1$	r_1 or $r_2, 1$
r_2	$r_2, 1$	$r_1, 1$	$r_2, 0$	r_1 or $r_2, 0$	$r_2, 0$	$r_2, 1$	$r_1, 1$	r_1 or $r_2, 1$

Figure 19

in more than one set of states at level n-3 of the tree. Another point that is worth noticing is the fact that because the set s_3 consists of only one state, both substates r_1 and r_2 are assigned to s_3. This is the reason that there are optional entries in the flow table shown in Fig. 19. Suppose that we define H as

$H(00, r_1) = s_1$ $\quad$ $H(00, r_2) = s_2$

$H(01, r_1) = s_5$ $\quad$ $H(01, r_2) = s_6$

$H(10, r_1) = s_4$ $\quad$ $H(10, r_2) = s_1$

$H(11, r_1) = s_3$ $\quad$ $H(11, r_2) = s_3$

Some of the entries in the flow table will be as shown in Fig. 20. Notice that all four entries in Fig. 20 correspond to a transition to s_1. Since $H(00, r_1) = H(10, r_2) = s_1$, whether the next substate should be r_1 or r_2 depends on the past input symbols. From

R^t \ $\Sigma^{t-2}\Sigma^{t-1}\Sigma^t$	000	001	010	011	100	101	110	111
r_1	$r_1, 0$		$r_2, 1$				$r_2, 1$	
r_2							$r_2, 1$	

Figure 20

$H(\sigma^{t-1}, \sigma^t, r^t) = s^{t+1}$, we see that since σ^{t-1} and σ^t are given, we would choose r^t so that the function H will yield the correct s^{t+1}.

g. Transient States

DEFINITION 9: In an automaton $A = \{\Sigma, S, M\}$, a state $s_i \in S$ is a transient state if (a) there exists an input symbol σ that is such that for every input sequence t, $M(M(s_i, \sigma), t) \neq s_i$, or (b) there exists an input sequence t that is such that $M(s_i, t)$ is a transient state.

DEFINITION 10: In an automaton $A = \{\Sigma, S, M\}$, a state $s_i \in S$ is a transient state of finite duration d if (a) s_i is a transient state, (b) for any state s_j and for any input sequence t with $L(t) > d$, $M(s_j, t) \neq s_i$, and (c) there exist a state s_j and an input sequence t, $L(t) = d$, which are such that $M(s_j, t) = s_i$.

DEFINITION 11: In an automaton $A = \{\Sigma, S, M\}$, a state $s_i \in S$ is a transient state of infinite duration if (a) s_i is a transient state, and (b) for any positive integer N, there exist a state s_j and an input sequence t, $L(t) \geq N$, which are such that $M(s_j, t) = s_i$.

DEFINITION 12: In an automaton $A = \{\Sigma, S, M\}$, a state $s_i \in S$ is a persistent state if it is not a transient state.

For example, for the automaton shown in Fig. 21, s_1 is a transient state of duration 0; s_2, a transient state of duration 1; s_3, a transient state of infinite duration, and s_4,

S \ Σ	0	1
s_1	s_2	s_5
s_2	s_4	s_3
s_3	s_3	s_4
s_4	s_5	s_5
s_5	s_5	s_6
s_6	s_4	s_5

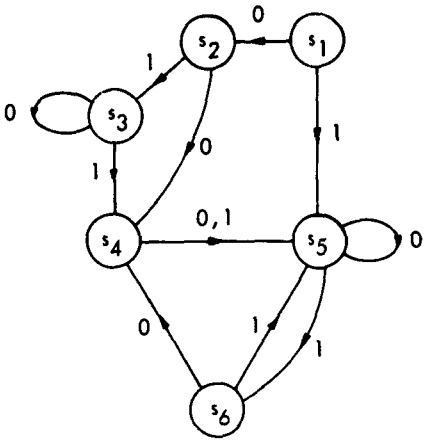


Figure 21

S \ Σ	0	1
s_1	$s_1, 0$	$s_4, 1$
s_2	$s_1, 1$	$s_1, 1$
s_3	$s_1, 1$	$s_4, 0$
s_4	$s_3, 0$	$s_3, 1$

A

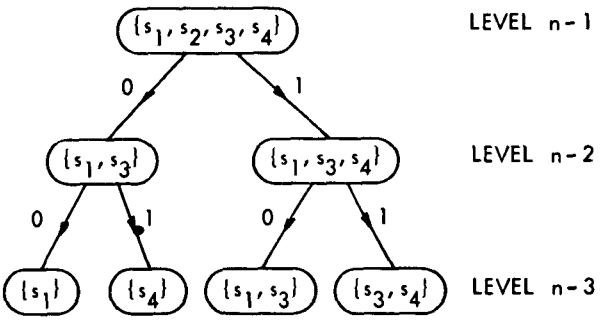


Figure 22

s_5, and s_6 are persistent states.

When an automaton consists of transient states of finite duration, extra steps are needed if we want to represent the automaton by the new model discussed above. In this new representation, suppose that k of the latest input symbols are known; since any transient state of finite duration less than k will not appear in any subset of states at level n - k - 1 of the tree of the essential part of $A^{\{n\}}$, no substate will be assigned to that state and the representation is not complete. Figure 22 is an example illustrating a procedure that will complete the assignment. Suppose that k = 2. Since s_2 is a transient state of duration 0, s_2 does not appear in any subset of states at level n-3 of the tree of the essential part of $A^{\{n\}}$. In order to have a substate assignment for s_2 as well, we augment the automaton by adding two more transient states as in Fig. 23 so that s_2 becomes a transient state of duration 2. Other than this requirement, these two transient

S \ Σ	0	1
s_1	$s_1, 0$	$s_4, 1$
s_2	$s_1, 1$	$s_1, 1$
s_3	$s_1, 1$	$s_4, 0$
s_4	$s_3, 0$	$s_3, 1$
s_5	$s_6, 0$	$s_1, 0$
s_6	$s_2, 0$	$s_1, 0$

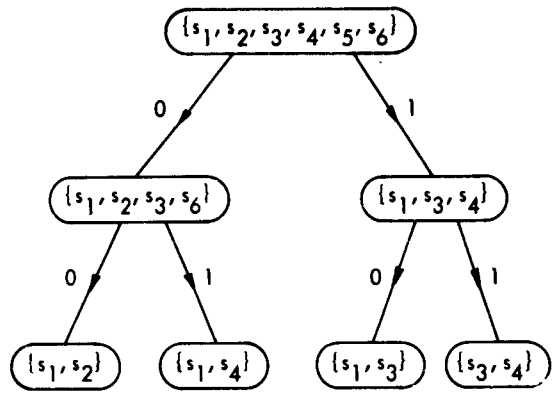


Figure 23

states can be added arbitrarily. If we look at the tree of the essential part of $A^{\{n\}}$, however, we discover that for k = 2 there will be two substates. If the transitions from the two added transient states are such that s_2 is put into the set s_1 or s_4 at level n-3, there still will be only two substates for the augmented automaton. Here, we choose to put s_2 into the set s_1. Therefore, for the transient states s_5 and s_6, $M(s_5, 00) = M(s_6, 0) = s_2$. The other entries of the transitions of s_5 and s_6 can be any of s_1, s_2, s_3, s_4. A simple rule to follow is to choose entries that already appear in the same column. Here, we arbitrarily choose $M(s_5, 1) = M(s_6, 1) = s_1$. The output symbols of the transition from s_5 and s_6 are completely arbitrary. As long as the automaton in Fig. 23 does not have s_5 or s_6 as the initial state, the augmented automaton is equivalent to the original automaton.

For the duration of a transient state we have the following theorem.

THEOREM 9: In an automaton consisting of n states, any transient state of finite duration would have a duration equal to or less than n-2.

PROOF 9: We claim that any automaton has at least one persistent state. Suppose that there is an automaton whose states are all transient states. Pick any arbitrary state s_i. Since s_i is a transient state, by definition we have

CASE I: There exists an input symbol σ_x, with $M(s_i, \sigma_x) = s_k$, that is such that no input sequence will lead s_k to s_i again.

CASE II: There exist an input sequence $t = \sigma_1\sigma_2\sigma_3 \ldots$, with $M(s_i, t) = s_j$, and an input symbol σ_x, with $M(s_j, \sigma_x) = s_k$, which are such that no input sequence will lead s_k to s_j again. No input sequence will lead s_k to any of s_i, $M(s_i, \sigma_1)$, $M(s_i, \sigma_1\sigma_2) \ldots$ either; otherwise it would be possible to lead s_k to s_j, which is a contradiction.

In both cases, because s_k is again a transient state, this argument can be repeated. Under the conclusion that once a state has been visited, it will never be visited again, after a finite number of repetitions of the above argument, there will be a state s_u and

an input symbol σ_y which are such that $M(s_u, \sigma_y)$ can go nowhere. This is obviously impossible.

Suppose that there is a transient state s_j and an input sequence $t = \sigma_1\sigma_2\sigma_3 \cdots \sigma_{n-1}$ which are such that $M(s_j, t) = s_i$. The states s_j, $M(s_j, \sigma_1)$, $M(s_j, \sigma_1\sigma_2)$, $\cdots$ s_i, are all transient states. Since there are, at most, n-1 transient states, at least one of them is visited more than once. Therefore, for any positive integer N there exists an input sequence t' with $L(t') \geq N$, which is such that $M(s_j, t') = s_i$. s_i is then a transient state of infinite duration. Q.E.D.

h. Minimum Feedback-Loop Realization of an Automaton

Figure 24 illustrates the general physical model of an automaton. Notice that feedback loops are needed to carry the information about the present state of the automaton.

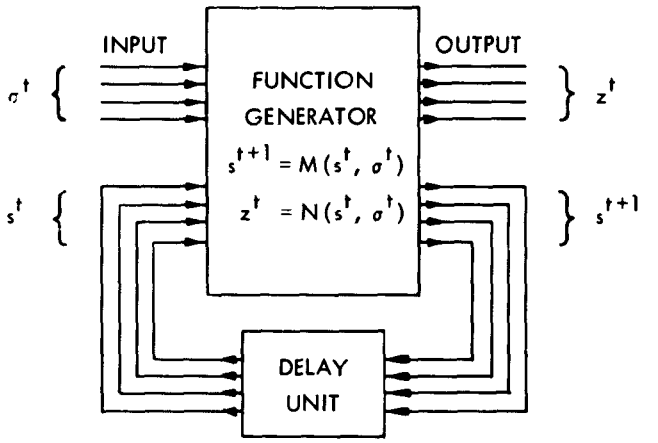


Figure 24

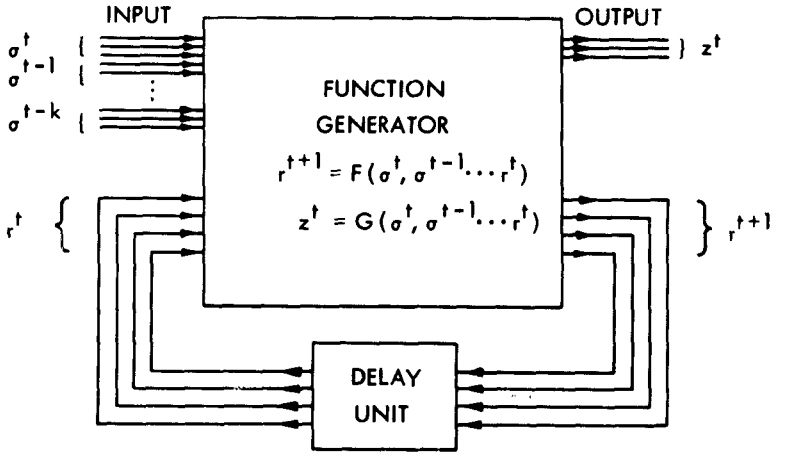


Figure 25

Thus the number of feedback loops depends on the number of states. Suppose that every feedback loop carries one bit of information (0 or 1); then the number of feedback loops is equal to $\lceil \log_2 (\text{total number of states}) \rceil$. (This means the smallest integer that is greater than or equal to $\log_2$ (total number of states).)

Suppose that storage devices are available to store the past input symbols. We can have a realization based on the new model of the automaton with the number of feedback loops equal to $\lceil \log_2 (\text{total number of substates}) \rceil$, as shown in Fig. 25. If we extend the tree of the essential part of $A^{\{n\}}$ to such a level that no further reduction of the number of substates is possible, the realization will be a minimum feedback-loop realization. Thus, the realization of an automaton having a finite memory span m can be realized with no feedback loops but requires m storage devices to store the past input symbols.

3.2 MEMORY SPAN WITH RESPECT TO THE INPUT-OUTPUT SEQUENCE

Thus far, we have discussed the memory span of an automaton with only the input sequences known. This will be called the memory span with respect to the input. We now want to investigate the memory span with both the input sequence and its corresponding output sequence available. For an automaton $A = \{\Sigma, Z, S, M, N\}$, we define a function R that is such that for all $s \in S$, and for all $(\sigma, z) \in \Sigma \times Z$

$$R(s, (\sigma, z)) = \begin{cases} M(s, \sigma) & \text{if } N(s, \sigma) = z \\ \text{undefined} & \text{if } N(s, \sigma) \neq z. \end{cases}$$

Since the flow table of the automaton explicitly exhibits the functional relations of M and N, we can generate a table of the function R from the flow table, as illustrated in Fig. 26, in which the bar entries are undefined. We then can consider the table of the function R as the flow table of an automaton $A_r = \{\Sigma \times Z, S, R\}$. The input symbols to A_r are ordered pairs (σ, z); some of the transitions of A_r are impossible, and the output

S \ Σ	0	1
s_1	$s_3, 0$	$s_2, 0$
s_2	$s_3, 1$	$s_5, 1$
s_3	$s_5, 1$	$s_4, 0$
s_4	$s_1, 0$	$s_4, 0$
s_5	$s_3, 0$	$s_2, 1$

A

S \ Σ × Z	0,0	0,1	1,0	1,1
s_1	s_3	-	s_2	-
s_2	-	s_3	-	s_5
s_3	-	s_5	s_4	-
s_4	s_1	-	s_4	-
s_5	s_3	-	-	s_2

A_r

Figure 26

symbols are not of interest. If we want to form the product automata of A_r, we shall write a state like $(s_i, -, s_j, -, -)$ as (s_i, s_j) by simply omitting all bar entries, and shall disregard a state with all undefined entries like $(-, -, -, -, -)$.

An input-output sequence w is a sequence of ordered pairs $(\sigma, z) \in \Sigma \times Z$. The length of an input-output sequence is the number of ordered pairs in the sequence. Thus we can extend the definition of R as follows

$$R(s, (\sigma, z)w) = \begin{cases} R(R(s, (\sigma, z)), w) & \text{if } R(s, (\sigma, z)) \text{ is defined} \\ \text{undefined} & \text{if } R(s, (\sigma, z)) \text{ is undefined.} \end{cases}$$

DEFINITION 13: $R(s_i, w_1)$ and $R(s_j, w_2)$ are compatible and will be denoted by $R(s_i, w_1) * R(s_j, w_2)$, if (a) both $R(s_i, w_1)$ and $R(s_j, w_2)$ are defined and have the same final state, or (b) either one or both of $R(s_i, w_1)$ and $R(s_j, w_2)$ are undefined.

It follows that $R(s_i, w_1)$ and $R(s_j, w_2)$ are incompatible if both $R(s_i, w_1)$ and $R(s_j, w_2)$ are defined and are different.

DEFINITION 14: An automaton $A = \{\Sigma, Z, S, M, N\}$ has a finite memory span m with respect to the input-output sequence if and only if (a) $R(s_i, w) * R(s_j, w)$ for all s_i, s_j in S and for any w with $L(w) \geq m + 1$, and (b) there exist $s_i, s_j \in S$, and w with $L(w) = m$ which are such that $R(s_i, w)$ and $R(s_j, w)$ are incompatible.

DEFINITION 15: An automaton $A = \{\Sigma, Z, S, M, N\}$ has an infinite memory span with respect to the input-output sequence if and only if for any given integer N there exist s_i, $s_j \in S$ and w with $L(w) \geq N$ which are such that $R(s_i, w)$ and $R(s_j, w)$ are incompatible.

a. Test for the Memory Span with Respect to the Input-Output Sequence

DEFINITION 16: Two states s_i and s_j have a unique final state with respect to the input-output sequence if $R(s_i, w) * R(s_j, w)$ for any input-output sequence w with $L(w) \geq K$, where K is a specific constant for the automaton.

From Definition 16, it follows that two states s_i and s_j will not have a unique final state with respect to the input-output sequence if, for any integer N, there exists an input-output sequence w, with $L(w) \geq N$, which is such that $R(s_i, w)$ and $R(s_j, w)$ are incompatible.

THEOREM 10: An automaton will have a finite memory span with respect to the input-output sequence if and only if every pair of its internal states has a unique final state with respect to the input-output sequence.

PROOF 10: First, we prove the necessity. Suppose that there are two states s_i and s_j that do not have a unique final state with respect to the input-output sequence. That is, there exists $(\sigma, z) \in \Sigma \times Z$ that is such that $R(s_i, (\sigma, z))$ and $R(s_j, (\sigma, z))$ are defined and do not have a unique final state with respect to the input-output sequence. By repeating this argument on $R(s_i, (\sigma, z))$ and $R(s_j, (\sigma, z))$, and so on, it can be shown that for any given integer N there exists w with $L(w) \geq N$ that is such that $R(s_i, w)$ and $R(s_j, w)$ are incompatible. The automaton will then have an infinite memory span with

respect to the input-output sequence.

Second, we prove the sufficiency. We claim that if two states s_i and s_j have a unique final state with respect to the input-output sequence, then $R(s_i, w) * R(s_j, w)$ for any w with $L(w) \geq n(n-1)/2$. This fact can be seen by forming $A_r^{\{2\}}$. If the automaton starts from $\{s_i, s_j\}$ and terminates at any compound state after $n(n-1)/2$ steps of transition, then there must be a loop around a certain compound state, since $A_r^{\{2\}}$ consists of only $n(n-1)/2$ compound states. This is a contradiction to the assumption that s_i and s_j have a unique final state. Therefore, for any w with $L(w) \geq n(n-1)/2$, each of $R(s_i, w)$, $R(s_2, w)$, ..., $R(s_n, w)$ will either be undefined or equal to the same final state. Q.E.D.

COROLLARY 1: If an automaton has a finite memory span m with respect to the input-output sequence, then $m \leq [n(n-1)/2] - 1$.

From Theorem 10, we can test whether or not an automaton has a finite memory span with respect to the input-output sequence by checking whether or not all pairs of states have a unique final state with respect to the input-output sequence. Notice that the compatibility relation of the function R is not transitive. That is, if $R(s_i, w) * R(s_j, w)$ and $R(s_j, w) * R(s_k, w)$, it is not necessary that $R(s_i, w) * R(s_k, w)$ because $R(s_j, w)$ may be undefined. A systematic procedure to carry out this test is illustrated in Fig. 27. First, we generate the table of the function R, and then form a "checking table" like that shown in Fig. 28a. A cross is entered if a state s_i (in the row) and a state s_j (in the column) have a unique final state with respect to the input-output sequence. The diagonal entries will be crossed off at the beginning, as shown in Fig. 28a, because $R(s_i, (\sigma, z)) * R(s_i, (\sigma, z))$ for all $(\sigma, z) \in \Sigma \times Z$. We then go through all pairs of states and examine them one by one. For example, we start with s_1, s_2. Because $R(s_1, (0, 0)) = s_4$ and $R(s_2, (0, 0)) = s_1$ and s_4 and s_1 still have not been shown to have a unique final state with respect to the input-output sequence (no cross mark in the s_1, s_4 entry), s_1 and s_2 might not have a unique final state with respect to the input-output sequence.

S \ Σ × Z	0,0	0,1	1,0	1,1
s_1	s_4	-	-	s_3
s_2	s_1	-	s_5	-
s_3	-	s_3	s_5	-
s_4	-	s_3	-	s_3
s_5	s_2	-	-	s_2

A_r

S \ Σ	0	1
s_1	$s_4, 0$	$s_3, 1$
s_2	$s_1, 0$	$s_5, 0$
s_3	$s_3, 1$	$s_5, 0$
s_4	$s_3, 1$	$s_3, 1$
s_5	$s_2, 0$	$s_2, 1$

A

Figure 27

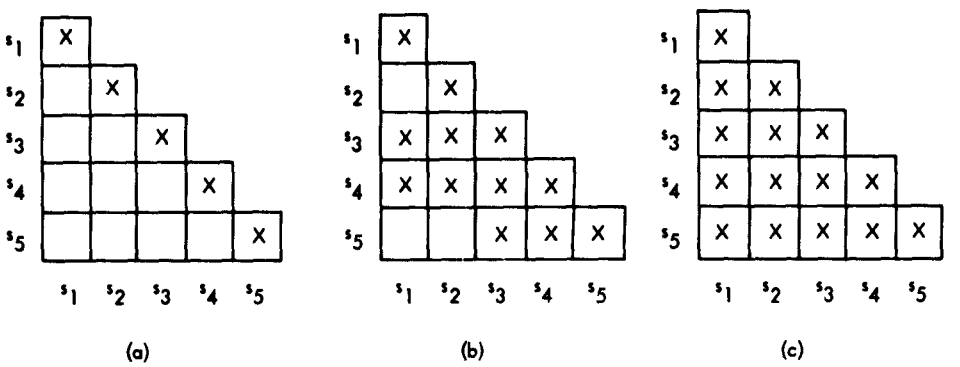


Figure 28

Examination of the pair s_1 and s_3 shows that the entry corresponding to s_1, s_3 in the table should be crossed off. So should the entries corresponding to s_1, s_4; s_2, s_3; s_2, s_4 . . . . After the first pass, we have the entries shown in Fig. 28b. We then start to test again those pairs whose corresponding entries in the table are still unchecked. The pair s_1 and s_2 now should be checked off as having a unique final state with respect to the input-output sequence because s_1, s_4 has been checked already. So should the pairs s_1, s_5 and s_2, s_5. We finally have the table in Fig. 28c. Since all entries in the checking table are checked, we know that the automaton has a finite memory span with respect to the input-output sequence; otherwise, the automaton would have an infinite memory span with respect to the input-output sequence.

b. Bounds on the Memory Span

Corollary 1 places an upper bound on the values of the memory span with respect to the input-output sequence. Theorem 11 provides a lower bound.

THEOREM 11: If a strongly connected automaton consisting of n states has a finite memory span m with respect to the input-output sequence, then $(\alpha\beta)^{m+1} \geq n$, where α is the number of input symbols and β is the number of output symbols.

PROOF 11: Since the automaton has a finite memory span m with respect to the input-output sequence, for all s in S and any input-output sequence w of length m+1, R(s, w) would either be undefined or would have the same final state. There are $(\alpha\beta)^{m+1}$ different input-output sequences of length m+1. Therefore, at most, $(\alpha\beta)^{m+1}$ states could be the possible final state after m+1 steps of transition from all n possible initial states. Suppose that $(\alpha\beta)^{m+1} < n$. There must be at least one state, say s_k, that is not included in these $(\alpha\beta)^{m+1}$ final states. Because the automaton is assumed to be strongly connected, however, there exist an input sequence t of length m+1 and an initial state s_j which are such that $M(s_j, t) = s_k$. These imply that there exists w that is such that $R(s_j, w) = s_k$, a contradiction to our assumption.

Q.E.D.

c. Procedures to Determine the Value of the Memory Span

After an automaton is found to have a finite memory span with respect to the input-output sequence, the value of the memory span can be found by constructing the tree of the essential part of $A_r^{\{n\}}$. For example, the memory span with respect to the input-output sequence of the automaton in Fig. 27 can be found to be 2. The tree of the essential part of $A_r^{\{n\}}$ is shown in Fig. 29. The memory span can also be found by slightly

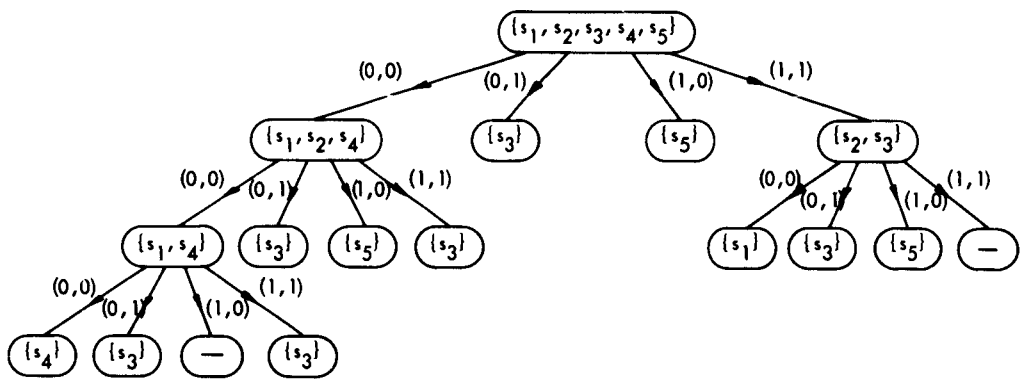


Figure 29

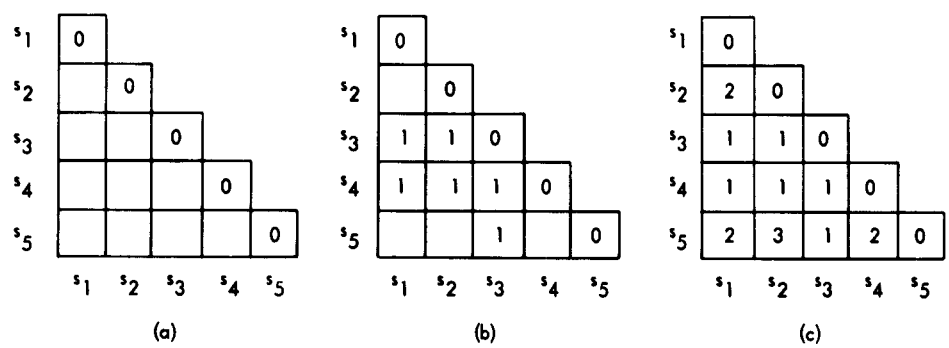


Figure 30

modifying the construction of the checking table. Instead of checking off pairs of states having a unique final state with a cross, we shall enter the numerals 1, 2 ... to indicate that they are checked off in the first, second, or subsequent pass. (Zeros indicate entries checked off by definition.) For example, the checking table in Fig. 28 is reconstructed and shown in Fig. 30. In Fig. 30a, 0's are entered in the diagonal boxes. In Fig. 30b pairs of states are checked off in the first pass. Figure 30c shows the final

form of the checking table. Since the highest count is 3 in the table, the memory span is 3 - 1 = 2. (The memory span always equals the highest count minus one.) Notice that the box corresponding to s_4, s_5 is crossed off in the second pass instead of the first because the pair s_4 and s_5 cannot be crossed off until the pair s_2 and s_3 is crossed off. However, the pair s_2 and s_3 is crossed off in the first pass. A slight reflection on the basic idea of the construction of the checking table will indicate that this procedure always works.

d. Extension of the Discussions of the Input to the Input-Output Sequence

As we can realize immediately, complete or partial information about the present state of an automaton can be obtained from the past input and output sequences. The discussions on the ambiguity of the present state, the tree of the essential part of $A^{\{n\}}$, and a new model for the automaton in the case of the memory span with respect to the input can be extended to the case of the memory span with respect to the input-output sequence and will not be repeated here.

3.3 MEMORY SPAN WITH RESPECT TO THE OUTPUT

It seems natural to ask about the memory span of an automaton with respect to its output sequences. Although some of the ideas developed thus far can be extended, there are some peculiar differences that must be taken into account. For an automaton $A = \{\Sigma, Z, S, M, N\}$, we define a function Q. For all $s \in S$ and all $z \in Z$,

$$Q(s, z) = \begin{cases} M(s, \sigma), & \text{for all } \sigma \text{ that are such that } N(s, \sigma) = z \\ \text{undefined}, & \text{if there is no } \sigma \text{ that is such that } N(s, \sigma) = z. \end{cases}$$

Q(s, z) may be multivalued because when Q(s, z) is defined there may exist more than one σ that is such that $N(s, \sigma) = z$. Let $y = z_1 z_2 z_3 \ldots$ denote a sequence of output symbols. The definition of the function Q can be extended to

$$Q(s, zy) = \begin{cases} Q(Q(s, z), y), & \text{if } Q(s, z) \text{ is defined and is single-valued} \\ Q(Q(s, z), y) = \{Q(M(s, \sigma_1), y), Q(M(s, \sigma_2), y) \ldots\}, & \\ \quad \text{if } Q(s, z) \text{ is defined and is multivalued, i.e., } Q(s, z) = \{M(s, \sigma_1), M(s, \sigma_2) \ldots\} & \\ \text{undefined}, & \text{if } Q(s, z) \text{ is undefined.} \end{cases}$$

DEFINITION 17: $Q(s_i, y_1)$ and $Q(s_j, y_2)$ are compatible and will be denoted by $Q(s_i, y_1) * Q(s_j, y_2)$, if (a) both $Q(s_i, y_1)$ and $Q(s_j, y_2)$ are single-valued and equal to the same next state or (b) either $Q(s_i, y_1)$ or $Q(s_j, y_2)$ is undefined and the other is single-valued or (c) both $Q(s_i, y_1)$ and $Q(s_j, y_2)$ are undefined.
Notice that $Q(s_i, y_1) * Q(s_i, y_1)$ is not necessarily true because $Q(s_i, y_1)$ may be multivalued, and by Definition 17 all multiple values are automatically incompatible.

As an example, for the table of the function Q shown in Fig. 31, we have

$$Q(s_1, 1) * Q(s_2, 1)$$

$$Q(s_1, 1) * Q(s_4, 1)$$

and the fact that

$$\left.\begin{array}{l} Q(s_1, 1) \text{ and } Q(s_3, 1) \\ Q(s_1, 0) \text{ and } Q(s_2, 0) \end{array}\right\} \text{are incompatible.}$$

If the function Q of an automaton is always defined and is single-valued, it can be considered the function M of some other automaton, and all of our previous discussions of the memory span with respect to the input can be applied. If the function Q is always single-valued when defined, it can be considered the function R of some other automaton, and all of our previous discussions of the memory span with respect to the input-output sequence can be applied. For an example, see Fig. 32.

In general, the function Q might be multivalued. We shall now study this general case.

DEFINITION 18: For an output sequence y, a set of states s_i, s_j, s_k ... has a unique intermediate state if $Q(s_u, y_1) * Q(s_v, y_1)$ for all s_u and s_v in the set and a prefix portion of y, say y_1 ($y=y_1y_2$).

DEFINITION 19: An automaton $A = \{\Sigma, Z, S, M, N\}$ has a finite memory span m with respect to the output if and only if (a) for any output sequence whose length is equal to or larger than m+1, the set of all states s_1, s_2, ... s_n has a unique intermediate state, and (b) there exist a pair of states s_i and s_j and an output sequence whose length is equal to m, for which s_i and s_j do not have a unique intermediate state.

The reason that we have to suggest such a definition, which is somewhat different from that for the memory spans with respect to the input and with respect to the input-output sequence, is the fact that the function Q might be multivalued. Thus $Q(s_i, y_1) * Q(s_2, y_1)$ does not guarantee that $Q(s_i, y_1z) * Q(s_2, y_1z)$. We can say, however, that for an automaton having a finite memory span m with respect to the output, within m+1 steps of transition corresponding to any output sequence y, there must be at least one time period during which the automaton is unambiguously in a certain state, regardless of the possible initial state.

DEFINITION 20: An automaton $A = \{\Sigma, Z, S, M, N\}$ has an infinite memory span with respect to the output if and only if for any given integer N, there exist a pair of states s_i and s_j and an output sequence whose length is equal to or larger than N, for which s_i and s_j do not have a unique intermediate state.

a. Test for the Memory Span with Respect to the Output

DEFINITION 21: Two states s_i and s_j have a unique next state with respect to the output if for all $z \in Z$ (a) $Q(s_i, z) * Q(s_j, z)$, or (b) both $Q(s_i, z)$ and $Q(s_j, z)$ are

S \ Σ	0	1
s_1	$s_3, 1$	$s_5, 0$
s_2	$s_1, 0$	$s_4, 0$
s_3	$s_3, 1$	$s_2, 1$
s_4	$s_3, 1$	$s_3, 1$
s_5	$s_2, 1$	$s_3, 1$

A

S \ Z	0	1
s_1	s_5	s_3
s_2	s_1, s_4	-
s_3	-	s_2, s_3
s_4	-	s_3
s_5	-	s_2, s_3

A_q

Figure 31

S \ Σ	0	1
s_1	$s_1, 0$	$s_2, 1$
s_2	$s_3, 1$	$s_4, 0$
s_3	$s_4, 0$	$s_3, 1$
s_4	$s_2, 1$	$s_1, 0$

A

S \ Z	0	1
s_1	s_1	s_2
s_2	s_4	s_3
s_3	s_4	s_3
s_4	s_1	s_2

A_q

(a)

S \ Σ	0	1
s_1	$s_3, 0$	$s_3, 0$
s_2	$s_4, 1$	$s_1, 0$
s_3	$s_3, 1$	$s_2, 0$
s_4	$s_4, 1$	$s_4, 1$

A

S \ Z	0	1
s_1	s_3	-
s_2	s_1	s_4
s_3	s_2	s_3
s_4	-	s_4

A_q

(b)

Figure 32

single-valued and have a unique next state with respect to the output , or (c) when either or both of $Q(s_i, z)$ and $Q(s_j, z)$ are multivalued, any pair of the possible next states will have a unique next state with respect to the output. (For example, if $Q(s_i, z) = s_k, s_m$ and $Q(s_j, z)$ is undefined, then s_k and s_m should have a unique next state with respect to the output. If $Q(s_i, z) = s_k, s_m$, $Q(s_j, z) = s_p, s_q$, then any pair of s_k, s_m; s_k, s_p; s_k, s_q; s_m, s_p; s_m, s_q; s_p, s_q should have a unique next state with respect to the output.)

DEFINITION 22: A state s_i has a unique next state with respect to the output, if, for all $z \in Z$, (a) $Q(s_i, z)$ is undefined or is single-valued, or (b) when $Q(s_i, z)$ is defined and is multivalued, any pair of the possible next states will have a unique next state with respect to the output.

Our test is based on a theorem that, in turn, is based on the following lemmas.

LEMMA 4: For a set of states $s_i, s_j, s_k, s_m \ldots$ and for an output sequence y, if all pairs of states in this set have a unique intermediate state, that is, if $Q(s_i, y_1) * Q(s_j, y_1)$, $Q(s_i, y_2) * Q(s_k, y_2) \ldots$, $Q(s_j, y_3) * Q(s_k, y_3) \ldots$, where $y = y_1y'_1 = y_2y'_2 = y_3y'_3 = \ldots$, then $Q(s_u, y_x) * Q(s_v, y_x)$ for all s_u and s_v in the set, where y_x is the longest sequence among $y_1, y_2, y_3 \ldots$.

PROOF OF LEMMA 4: Suppose that the pair s_k and s_m corresponds to the sequence y_x that is such that $Q(s_k, y_x) * Q(s_m, y_x)$. If both $Q(s_k, y_x)$ and $Q(s_m, y_x)$ are undefined, we can pick the pair s'_k and s'_m corresponding to the next longest sequence y' and use the argument that follows. Assume that at least one of $Q(s_k, y_x)$ and $Q(s_m, y_x)$ is defined and is equal to s_w. For any state s_i, we have $Q(s_i, y_a) * Q(s_k, y_a)$ and $Q(s_i, y_b) * Q(s_m, y_b)$, where y_a and y_b are prefix portions of y. Either one or both of $Q(s_k, y_a)$ and $Q(s_m, y_b)$ must be defined; otherwise, $Q(s_k, y_x)$ and $Q(s_m, y_x)$ will be undefined. Assume that $Q(s_k, y_a) = s_u$. (The case that $Q(s_k, y_a)$ is undefined while $Q(s_m, y_b)$ is defined or the case that both $Q(s_k, y_a)$ and $Q(s_m, y_b)$ are defined can be proved in a similar manner.) If $Q(s_i, y_a)$ is undefined, we immediately have $Q(s_i, y_x) * Q(s_k, y_x)$ and $Q(s_i, y_x) * Q(s_m, y_x)$ because y_a is a prefix portion of y_x. If $Q(s_i, y_a) = s_u$, and since we can write $y_x = y_a y'_a$, then $Q(s_k, y_x) = Q(s_k, y_a y'_a) = Q(s_u, y'_a) = Q(s_m, y_x) = s_w$. Therefore $Q(s_i, y_x)$ is either undefined or equal to s_w, and we have $Q(s_i, y_x) * Q(s_k, y_x)$ and $Q(s_i, y_x) * Q(s_m, y_x)$. Similar argument can be applied to all states in the set. Q.E.D.

LEMMA 5: In an automaton consisting of n states, if any pair of states has a unique next state with respect to the output, then the set of all n states will have a unique intermediate state for any output sequence of a length that is equal to or larger than $n(n-1)/2$.

PROOF OF LEMMA 5: By Lemma 4, we have only to prove that any pair of states s_i and s_j will have a unique intermediate state for any output sequence of a length that is equal to or larger than $n(n-1)/2$. Let $\{Q(s_i, z) \cup Q(s_j, z)\}$ denote the set of all possible values of $Q(s_i, z)$ and $Q(s_j, z)$. By the definition of two states having a unique next state, we see that if $\{Q(s_i, z) \cup Q(s_j, z)\}$ consists of only one state, the lemma holds. If, however, $\{Q(s_i, z) \cup Q(s_j, z)\}$ consists of two or more states, say $s_u, s_v, s_w, s_x \ldots$, we can group $s_u, s_v, s_w, s_x \ldots$ into all possible pairs, say $\{s_u, s_v\}, \{s_u, s_w\} \ldots \{s_v, s_w\} \ldots$.

S \ Σ	0	1
s_1	$s_2, 0$	$s_3, 0$
s_2	$s_4, 0$	$s_5, 1$
s_3	$s_6, 1$	$s_4, 0$
s_4	$s_6, 1$	$s_6, 1$
s_5	$s_2, 0$	$s_2, 0$
s_6	$s_1, 1$	$s_1, 1$

A

S \ Z	0	1
s_1	s_2, s_3	-
s_2	s_4	s_5
s_3	s_4	s_6
s_4	-	s_6
s_5	s_2	-
s_6	-	s_1

A_q

Figure 33

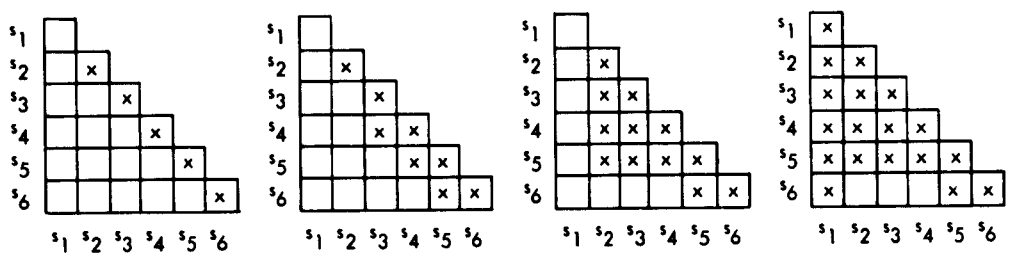


Figure 34

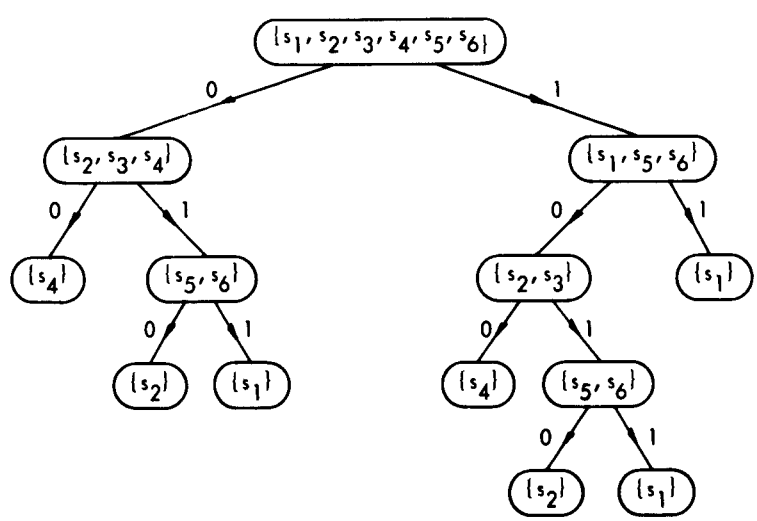


Figure 35

These pairs of states must also have a unique next state, and we can repeat the above-given argument to examine the value of $\{Q(s_u, z) \cup Q(s_v, z)\}$, $\{Q(s_u, z) \cup Q(s_w, z)\} \ldots \{Q(s_v, z) \cup Q(s_w, z)\} \ldots$. Since there are $n(n-1)/2$ pairs of states, s_i and s_j should terminate at a unique intermediate state after, at most, $n(n-1)/2$ steps of transition. Otherwise, s_i and s_j will not have a unique next state with respect to the output, in contradiction to the assumption. Q.E.D.

THEOREM 12: An automaton will have a finite memory span with respect to the output if and only if all pairs of states have a unique next state with respect to the output.

PROOF 12: Theorem 12 follows immediately from Lemmas 4 and 5.

COROLLARY 2: If an automaton has a finite memory span m with respect to the output, $m \leq [n(n-1)/2] - 1$.

From Theorem 12, we have a means for checking the memory span of an automaton with respect to the output. The procedure is very similar to the one employed to check the memory span with respect to the input-output sequence. As an example, the test procedure, applied to the automaton shown in Fig. 33, is shown in Fig. 34. From the functions M and N of A we form the table of the function Q. We check each individual state to see whether or not it has a unique next state. States s_2, s_3, s_4, s_5, s_6 do, and we do not know whether s_1 does or not at this moment because $Q(s_1, 0) = s_2, s_3$, as shown in Fig. 34a. Then we check all pairs of states to determine whether or not they have a unique next state. We have the table shown in Fig. 34b after the first pass and that of Fig. 34c after the second pass. Because the pair s_2 and s_3 has been checked, s_1 now can be checked as having a unique next state with respect to the output. The pairs (s_1, s_2), $(s_1, s_3) \ldots (s_1, s_6)$ can then be checked as shown in Fig. 34d. Another pass will fill up all of the entries in the checking table, and we know that the automaton has a finite memory span with respect to the output.

We can form the tree of the essential part of $A_q^{\{n\}}$, as shown in Fig. 35. We see that the automaton has a finite memory span 3 with respect to the output. Notice that, in forming the tree, we stop at any branch if a set containing a single state is encountered along that branch. Also, notice that our definition of finite memory span with respect to the output (Definition 18) guarantees that there is a unique intermediate state for any output sequence of length 4, but not necessarily a unique final state. For example, an output sequence 0110 will not give a unique final state (both s_2 and s_3 are possible).

s_1	3					
s_2	3	0				
s_3	3	2	0			
s_4	3	2	1	0		
s_5	3	3	3	1	0	
s_6	3	4	4	4	1	0
	s_1	s_2	s_3	s_4	s_5	s_6

Figure 36

The value of the memory span with respect to the output can also be found by entering numerals instead of crosses in the checking table. This procedure is identical with that shown for finding the value of the memory span with respect to the

input-output sequence. (See Fig. 30.) The checking table of the automaton in Fig. 33, which has a memory span of 3 with respect to the output, is shown in Fig. 36.

3.4 EXTENSION TO INPUT AND OUTPUT SEQUENCES OF UNEQUAL LENGTH

We shall investigate the determination of the present state of an automaton with p of the past input symbols and q of the past output symbols known, for $p \neq q$ (for $p = q$, see the memory span with respect to the input-output sequence). We find that for a given automaton, p and q do not have fixed values. The techniques developed thus far can be employed to find the possible p-q values.

We use the automaton of Fig. 37 as an example and form the checking table for A_r as shown in Fig. 38a. To find the possible p-q values with p larger than q, we form the tree of the essential part of $A^{\{n\}}$, as shown in Fig. 38b. For $p = q$, we examine the set of states at level n-1 in the tree of the essential part of $A^{\{n\}}$. The value of q will be equal to the largest number in the checking table which corresponds to pairs of states that are in this set. In Fig. 38, $q = 4$ because of the pair s_1 and s_2. (We can consider any unchecked box in the checking table as having an entry that is equal to infinity.) For $p = q + 1$, we examine all sets of states at level n-2. Because of the pair of states s_1 and s_4 in the subset $\{s_1, s_3, s_4\}$, $q = 3$, that is, the present state of the automaton can be determined uniquely when four of the latest input symbols and three of the latest output symbols are known. For $p = q + 2$, we examine all of the sets of states at level n-3. It turns out that $q = 2$. Because the subset $\{s_3, s_4\}$ will occur in all subsequent levels of the tree, the value of q will be 2 at all subsequent levels of the tree.

To find the possible q-p values with q larger than p, we shall go through the same procedure as above, except that we shall form the tree of the essential part of $A_q^{\{n\}}$. The tree is shown in Fig. 39. For $q = p$, $p = 3$. For $q = p + 1$, $p = 3$ because of the subset $\{s_1, s_2, s_3\}$. For $q = p + 2$, $p = 2$ because of the subsets $\{s_2, s_3\}$ and $\{s_1, s_3\}$.

In general, we shall never extend the tree of the essential part of $A^{\{n\}}$ or of $A_q^{\{n\}}$

S \ Σ	0	1
s_1	$s_2, 0$	$s_4, 1$
s_2	$s_3, 1$	$s_1, 1$
s_3	$s_2, 0$	$s_3, 0$
s_4	$s_3, 0$	$s_3, 1$

A

S \ Σ × Z	0,0	0,1	1,0	1,1
s_1	s_2	-	-	s_4
s_2	-	s_3	-	s_1
s_3	s_2	-	s_3	-
s_4	s_3	-	-	s_3

A_r

S \ Z	0	1
s_1	s_2	s_4
s_2	-	s_1, s_3
s_3	s_2, s_3	-
s_4	s_3	s_3

A_q

Figure 37

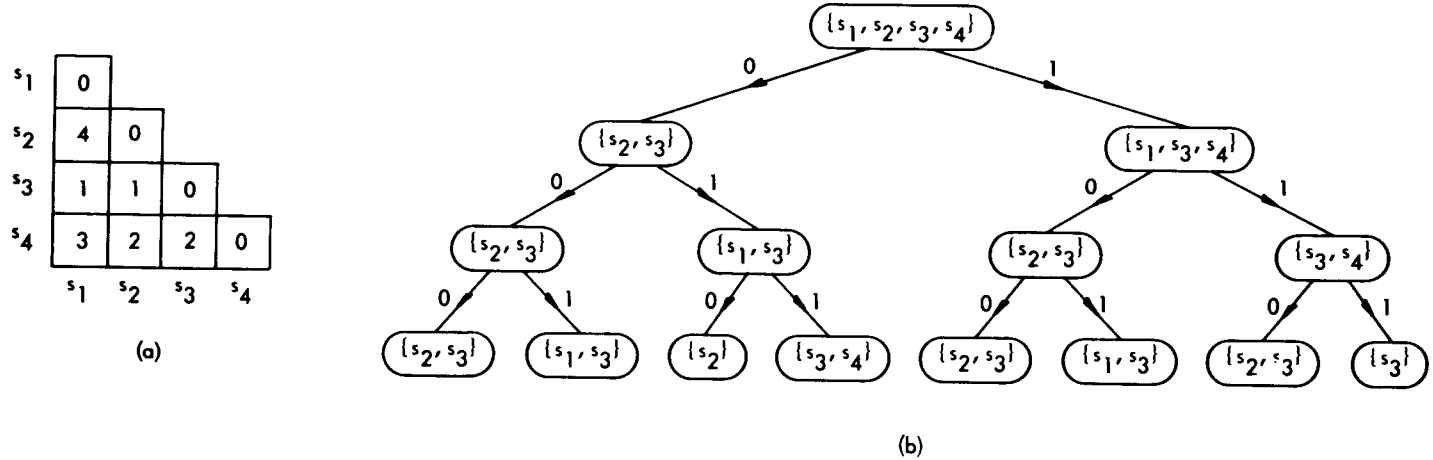


Figure 38

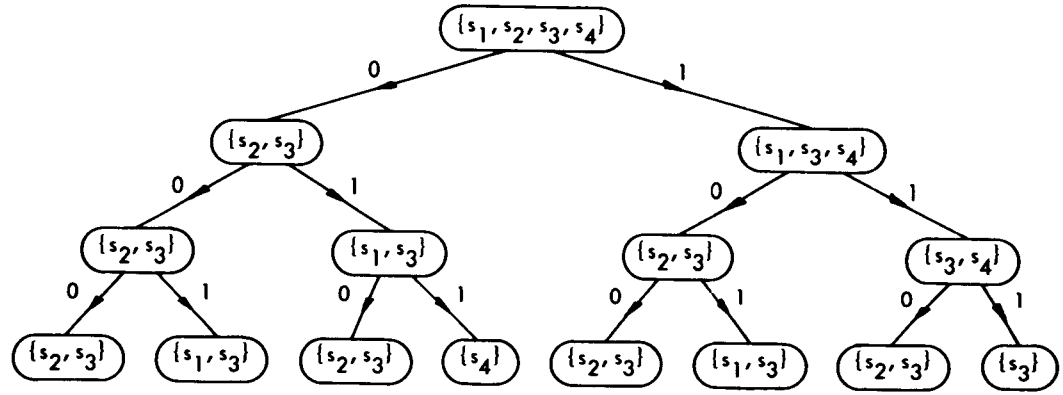


Figure 39

beyond n(n-1)/2 levels, where n is the number of states of the automaton. Because a branch of length n(n-1)/2 will certainly include all possible pairs of states in the n(n-1)/2 subsets of states along that branch, further extension of that branch will never decrease the value of q (for p > q) or p (for q > p).

a. Relation between the Value of p - q and q (or q - p and p)

We want to show that the value of q is nonincreasing as we increase the value of p - q by extending the branches of the tree of the essential part of $A^{\{n\}}$ for p > q. The same result is true for q > p, where p is nonincreasing for increasing value of q - p. We prove this by induction. Figure 40 shows the top levels of the tree of the essential part of $A^{\{n\}}$. S is the set of all of the states of the automaton. {a}, {b} {c}, {d}, {e}, and {f} are subsets of states. Of course, {a} and {b} are subsets of {S}. Since $M^{\{n\}}(\{S\}, 0)$ = {a},and $M^{\{n\}}(\{a\}, 0)$ = {c}, and since {a} is a subset of {S}, {c} is a subset of {a}. Similarly {d} is a subset of {b}, {e} is a subset of {a} and {f} is a subset of {b}. For levels n-k, n - k - 1, and n - k - 2 as shown in Fig. 41, suppose that every subset of states in level

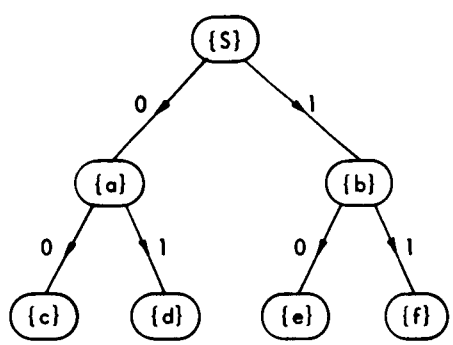


Figure 40

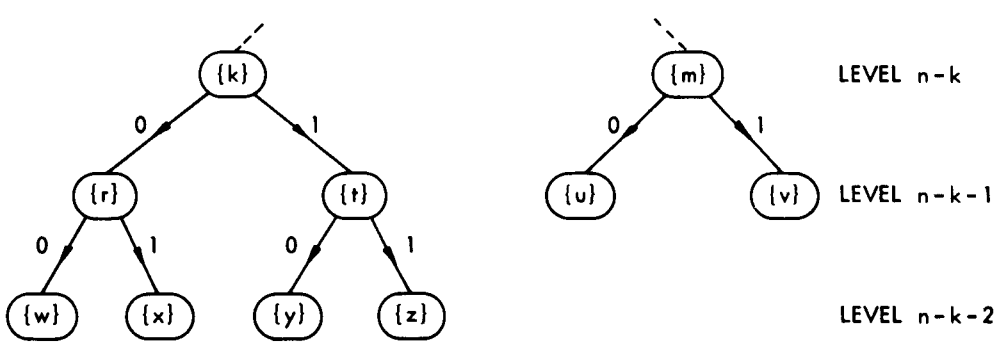


Figure 41

n - k - 1 is a subset of some subset of states in level n-k. For any subset of states in level n - k - 2, say $\{z\}$, $M^{\{n\}}(\{t\}, 1) = \{z\}$. By assumption, $\{t\}$ is a subset of a subset of states in level n-k, say $\{m\}$. Since $M^{\{n\}}(\{m\}, 1) = \{v\}$, then $\{z\}$ is a subset of $\{v\}$. The result is the fact that all subsets of states in level n - k - 2 of the tree of the essential part of $A^{\{n\}}$ are subsets of subsets of states in level n - k - 1. This immediately leads to the conclusion that the value of q corresponding to subsets of states in level n - k - 2 will not be larger than that corresponding to subsets of states in level n - k - 1.

IV. SYNCHRONIZATION OF AUTOMATA

4.1 INTRODUCTION

In an automaton $A = \{\Sigma, S, M\}$, a set of states $s_i, s_j, s_k \ldots$ is said to be synchronizable with a state s_u if there exists an input sequence t of finite length which is such that $M(s_i, t) = M(s_j, t) = M(s_k, t) = \ldots = s_u$. The input sequence t is called the synchronizing sequence that synchronizes this set of states with s_u or, briefly, the synchronizing sequence with s_u. More generally, a set of states $s_i, s_j, s_k \ldots$ is said to be synchronizable if there exists an input sequence t of finite length which is such that $M(s_i, t) = M(s_j, t) = M(s_k, t) = \ldots$. The input sequence t is called the synchronizing sequence that synchronizes this set of states or, briefly, the synchronizing sequence.

THEOREM 13: In a strongly connected automaton, a set of synchronizable states is synchronizable with any specific state s_u.

PROOF 13: Let $s_i, s_j, s_k \ldots$ be the set of states which is synchronizable by the input sequence t. It follows that $M(s_i, t) = M(s_j, t) = M(s_k, t) = \ldots = s_v$, where s_v is some state of the automaton. Since the automaton is strongly connected, there exists an input sequence t' that is such that $M(s_v, t') = s_u$. Input sequence tt' is, then, the synchronizing sequence that synchronizes this set of states with s_u. Q. E. D.

Furthermore, we say that an automaton is synchronizable (with a state s_u) if the set of all states is synchronizable (with s_u). We have the following theorem.

THEOREM 14: An automaton cannot be synchronized with a transient state.

PROOF 14: No input sequence will lead a persistent state to a transient state. By Theorem 9, however, any automaton has at least one persistent state. Q. E. D.

4.2 RELATION TO THE A^P OR $A^{\{p\}}$ AUTOMATON

To investigate the synchronizability of a set of p states in an automaton A, we investigate the A^P or the $A^{\{p\}}$ automaton. It is rather obvious that a set of p states $s_i, s_j, s_k \ldots$ is synchronizable with a state s_u if and only if in A^P there exists an input sequence t that is such that $M^P((s_i, s_j, s_k \ldots), t) = (s_u, s_u, s_u \ldots)$, or in $A^{\{p\}}$, $M^{\{p\}}(\{s_i, s_j, s_k \ldots\}, t) = \{s_u\}$. Thus we have a way to check the synchronizability of a set of states and to find the corresponding synchronizing sequence. As an example, for the automaton shown in Fig. 42, 010 is a synchronizing sequence that will synchronize s_1 and s_3 with s_2; but there is no synchronizing sequence that will synchronize s_1 and s_3 with s_1 or with s_3.

We have the following procedure to check systematically the synchronizability of a set of p states and, at the same time, to find the shortest possible synchronizing sequence:

(a) Form the flow table of the automaton $A^{\{p\}}$.

(b) Put the state $\{s_i, s_j \ldots\}$ corresponding to the set of states to be synchronized at the top of a tree. Cross off all appearances of the state $\{s_i, s_j \ldots\}$ in the flow table.

(c) Extend the tree by listing all of the states that still have not been crossed off in

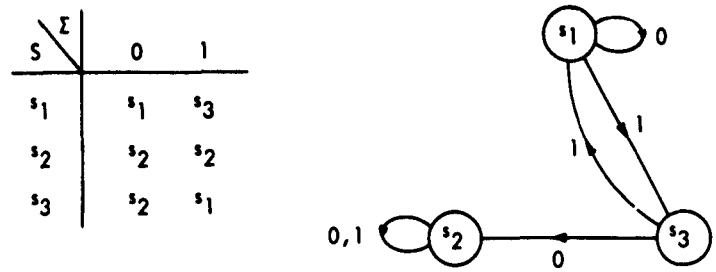


Figure 42

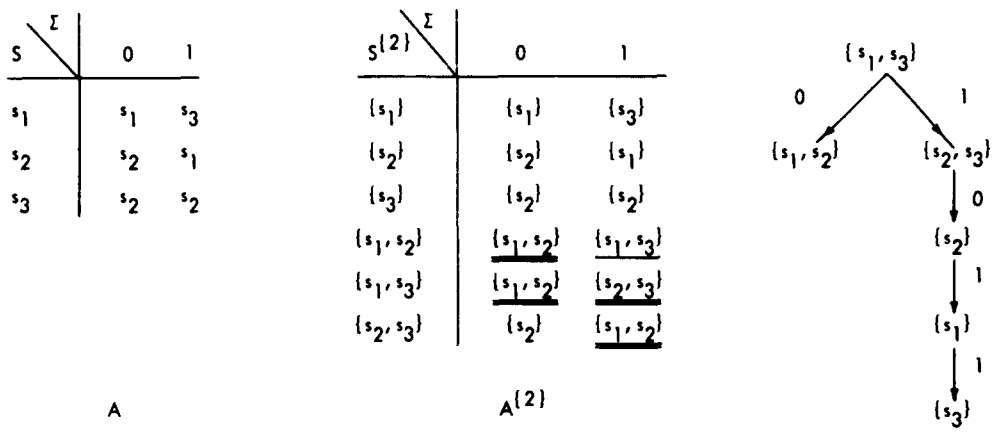


Figure 43

the flow table and that can be reached from states already in the tree through one step of transition. Then cross off all appearances of these states in the flow table.

(d) Repeat step (c), either until a specific simple state $\{s_u\}$ (or any simple state $\{s_v\}$ in the case of synchronizing a set of states) is reached or until no further repetition is possible (that is, until all of the states have been crossed off which can be reached from states already in the tree through one step of transition).

Figure 43 illustrates this test procedure. In order to check the synchronizability of s_1 and s_3, we form the flow table of $A^{(2)}$ and grow a tree, starting from $\{s_1, s_3\}$. All appearances of $\{s_1, s_3\}$ in the flow table are crossed off first (single underline in Fig. 43). Since one step of transition from $\{s_1, s_3\}$ will lead to $\{s_1, s_2\}$ and $\{s_2, s_3\}$, their appearances in the flow table are crossed off accordingly (double underline in Fig. 43). Repeating the procedure, we have the tree shown in Fig. 43. Notice the termination of the left branch at $\{s_1, s_2\}$ because all states that can be reached through one step of transition from $\{s_1, s_2\}$ have been crossed off in the flow table. Therefore, the shortest synchronizing sequence that synchronizes s_1 and s_3 in Fig. 43 is 10, while

the shortest synchronizing sequence that synchronizes s_1 and s_3 with s_3 is 1011.

4.3 A SIMPLER WAY TO CHECK THE SYNCHRONIZABILITY OF AN AUTOMATON

Although the synchronizability of sets of states can always be checked by the method described above, the fact that the number of states in $A^{\{p\}}$ increases very rapidly with increasing number of states in A makes the procedure laborious when A contains a large number of states. The following theorem leads to a simpler procedure to check the synchronizability of an automaton.

THEOREM 15: An automaton is synchronizable if and only if every pair of its states is synchronizable.

PROOF 15: First, we prove the necessity. Suppose that there is a pair of states s_i and s_j that are not synchronizable, that is, for every input sequence t, $M(s_i, t) \neq M(s_j, t)$. Obviously, this is a contradiction to the fact that there is an input sequence t' that is such that $M(s_1, t') = M(s_2, t') = \ldots M(s_i, t') = \ldots M(s_j, t') = \ldots M(s_u, t')$.

Second, we prove the sufficiency. By assumption,

$$\exists t_1: \quad M(s_1, t_1) = M(s_2, t_1)$$

$$\exists t_2: \quad M(M(s_1, t_1), t_2) = M(M(s_3, t_1), t_2)$$

$$M(s_1, t_1t_2) = M(s_2, t_1t_2) = M(s_3, t_1t_2)$$

$$\exists t_3: \quad M(M(s_1, t_1t_2), t_3) = M(M(s_4, t_1t_2), t_3)$$

$$M(s_1, t_1t_2t_3) = M(s_2, t_1t_2t_3) = M(s_3, t_1t_2t_3) = M(s_4, t_1t_2t_3)$$

$$\vdots$$

$$\exists t_{n-1}: \quad M(M(s_1, t_1t_2t_3\ldots t_{n-2}), t_{n-1}) = M(M(s_n, t_1t_2t_3\ldots t_{n-2}), t_{n-1})$$

$$M(s_1, t_1t_2t_3\ldots t_{n-2}t_{n-1}) = M(s_2, t_1t_2t_3\ldots t_{n-2}t_{n-1}) = \ldots$$

$$M(s_n, t_1t_2t_3\ldots t_{n-2}t_{n-1}).$$ Q.E.D.

COROLLARY 3: A sufficient condition for a set of p states in an automaton to be synchronizable is that every pair of states of the automaton is synchronizable.

On the basis of Theorem 15, we can employ the following procedures to check the synchronizability of an automaton.

(a) Form the flow table of the $A^{\{2\}}$ automaton.

(b) List all simple states and cross them off in the column $S^{\{2\}}$ of the flow table.

(c) Add to the list all compound states that have not yet been crossed off in the flow table and can reach some of the states in the list in one step of transition, and cross off these compound states.

S \ Σ	0	1
s_1	s_3	s_2
s_2	s_4	s_1
s_3	s_5	s_2
s_4	s_6	s_1
s_5	s_5	s_3
s_6	s_6	s_4

A

$S^{\{2\}}$ \ Σ	0	1
$\{s_1\}$	$\{s_3\}$	$\{s_2\}$
$\{s_2\}$	$\{s_4\}$	$\{s_1\}$
$\{s_3\}$	$\{s_5\}$	$\{s_2\}$
$\{s_4\}$	$\{s_6\}$	$\{s_1\}$
$\{s_5\}$	$\{s_5\}$	$\{s_3\}$
$\{s_6\}$	$\{s_6\}$	$\{s_4\}$
$\{s_1, s_2\}$	$\{s_3, s_4\}$	$\{s_1, s_2\}$
$\{s_1, s_3\}$	$\{s_3, s_5\}$	$\{s_2\}$
$\{s_1, s_4\}$	$\{s_3, s_6\}$	$\{s_1, s_2\}$
$\{s_1, s_5\}$	$\{s_3, s_5\}$	$\{s_2, s_3\}$
$\{s_1, s_6\}$	$\{s_3, s_6\}$	$\{s_2, s_4\}$
$\{s_2, s_3\}$	$\{s_4, s_5\}$	$\{s_1, s_2\}$
$\{s_2, s_4\}$	$\{s_4, s_6\}$	$\{s_1\}$
$\{s_2, s_5\}$	$\{s_4, s_5\}$	$\{s_1, s_3\}$
$\{s_2, s_6\}$	$\{s_4, s_6\}$	$\{s_1, s_4\}$
$\{s_3, s_4\}$	$\{s_5, s_6\}$	$\{s_1, s_2\}$
$\{s_3, s_5\}$	$\{s_5\}$	$\{s_2, s_3\}$
$\{s_3, s_6\}$	$\{s_5, s_6\}$	$\{s_2, s_4\}$
$\{s_4, s_5\}$	$\{s_5, s_6\}$	$\{s_1, s_3\}$
$\{s_4, s_6\}$	$\{s_6\}$	$\{s_1, s_4\}$
$\{s_5, s_6\}$	$\{s_5, s_6\}$	$\{s_3, s_4\}$

$A^{\{2\}}$

$\{s_1\}$ $\{s_2\}$ $\{s_3\}$ $\{s_4\}$ $\{s_5\}$ $\{s_6\}$

$\{s_1, s_3\}$ $\{s_2, s_4\}$ $\{s_3, s_5\}$ $\{s_4, s_6\}$

$\{s_2, s_5\}$ $\{s_4, s_5\}$ $\{s_1, s_6\}$ $\{s_3, s_6\}$

$\{s_1, s_5\}$ $\{s_2, s_6\}$

$\{s_2, s_3\}$ $\{s_1, s_4\}$

Figure 44

(d) Repeat step (c) until no further repetition is possible. If all of the states of $A^{\{2\}}$ are crossed off, A is synchronizable; otherwise, not.

Figure 44 illustrates this test procedure. States in the first row of the list are underlined once in the column $S^{\{2\}}$; those in the second row of the list are underlined twice. The list terminates before all states in $A^{\{2\}}$ are crossed off,

and the automaton A is thus not synchronizable.

4.4 ABSOLUTELY SYNCHRONIZABLE AUTOMATA

In an automaton $A = \{\Sigma, S, M\}$, a set of states $s_i, s_j, s_k \cdots$ is said to be absolutely synchronizable with state s_u if, for any input sequence t whose length is equal to or larger than a constant m, $M(s_i, t) = M(s_j, t) = M(s_k, t) = \ldots = s_u$. The set is said to be absolutely synchronizable if $M(s_i, t) = M(s_j, t) = M(s_k, t) = \ldots$. Similarly, an automaton is said to be absolutely synchronizable (with a state s_u) if the set of all states is absolutely synchronizable (with s_u). Obviously, an automaton having a finite memory span with respect to the input is absolutely synchronizable and vice versa.

4.5 BOUNDS ON THE LENGTH OF THE SYNCHRONIZING SEQUENCE

THEOREM 16: If a set of p states $s_i, s_j, s_k \cdots$ in an automaton A is synchronizable with a state s_u, an upper bound on the length of the shortest synchronizing sequence with s_u is $\sum_{j=1}^{p} C_j^n - 1$, where n is the total number of states in A.

PROOF 16: The automaton $A^{\{p\}}$ consists of $\sum_{j=1}^{p} C_j^n$ states. If there exists an input sequence t that is such that $M^{\{p\}}(\{s_i, s_j, s_k \ldots\}, t) = \{s_u\}$, then there exists an input sequence $L(t') \leq \sum_{j=1}^{p} C_j^n - 1$ that is such that $M^{\{p\}}(\{s_i, s_j, s_k \ldots\}, t') = \{s_u\}$. Q.E.D.

COROLLARY 4: If a set of p states $s_i, s_j, s_k \cdots$ in an automaton A is synchronizable, the upper bound on the length of the shortest synchronizing sequence is $\sum_{j=2}^{p} C_j^n$.

PROOF OF COROLLARY 4: There are n simple states in $A^{\{p\}}$. If we consider these n states to be combined as a single state, the resulting automaton will have $\sum_{j=1}^{p} C_j^n - n + 1 = \sum_{j=2}^{p} C_j^n + 1$ states. Any sequence that leads $\{s_i, s_j, s_k \ldots\}$ to this combined state is a synchronizing sequence. Q.E.D.

Better bounds on the length of the synchronizing sequence can be obtained for some special automata.

THEOREM 17: In a synchronizable automaton, an upper bound on the length of the shortest synchronizing sequence that synchronizes any set of p states is $(p-1)\, C_2^n$.

PROOF 17: Let $s_1, s_2 \ldots s_p$ be the set of states to be synchronized. By the argument of necessity in Proof 15, every pair of states in this set is synchronizable. We then have

$$\exists t_1: \; L(t_1) \leq C_2^n \qquad M(s_1, t_1) = M(s_2, t_1)$$

$$\exists t_2: \; L(t_2) \leq C_2^n \qquad M(s_1, t_1 t_2) = M(s_2, t_1 t_2) = M(s_3, t_1 t_2)$$

$$\vdots$$

$$\exists t_{p-1}: \; L(t_{p-1}) \leq C_2^n \qquad M(s_1, t_1 t_2 \ldots t_{p-1}) = M(s_2, t_1 t_2 \ldots t_{p-1}) = \ldots = M(s_p, t_1 t_2 \ldots t_{p-1}).$$

Here, $L(t_1t_2\ldots t_{p-1}) \leq (p-1)\,C_2^n$, and $t_1t_2\ldots t_{p-1}$ is certainly a synchronizing sequence. $L(t_1)$, $L(t_2)$, $\ldots$ $L(t_{p-1})$ are all less than or equal to C_2^n, since there are $n(n-1)/2 = C_2^n$ compound states in the automaton $A^{\{2\}}$ and the length of the shortest input sequence that will synchronize a compound state should be less than or equal to C_2^n. Q.E.D.

We want to prove the following lemmas.

LEMMA 6: If s_x is a persistent state and there exists an input sequence t that is such that $M(s_x, t) = s_y$, then there exists an input sequence t' that is such that $M(s_y, t') = s_x$.

PROOF OF LEMMA 6: Write $t = \sigma_1\sigma_2\sigma_3\ldots\sigma_p\sigma_q\sigma_r$. By the definition of a persistent state (Definition 12), $M(s_x, \sigma_1)$, $M(s_x, \sigma_1\sigma_2)$, $\ldots$ $M(s_x, \sigma_1\sigma_2\sigma_3\ldots\sigma_p\sigma_q)$, $M(s_x, \sigma_1\sigma_2\sigma_3\ldots\sigma_p\sigma_q\sigma_r)$ are all persistent states. Moreover, there exists an input sequence t_1 that is such that $M(M(M(s_x, \sigma_1\sigma_2\sigma_3\ldots\sigma_p\sigma_q), \sigma_r), t_1) = M(s_x, \sigma_1\sigma_2\sigma_3\ldots\sigma_p\sigma_q)$. That is,

$$\exists t_1\colon\ M(s_y, t_1) = M(s_x, \sigma_1\sigma_2\sigma_3\ldots\sigma_p\sigma_q).$$

Similarly,

$$\exists t_2\colon\ M(M(M(s_x, \sigma_1\sigma_2\sigma_3\ldots\sigma_p), \sigma_q), t_2) = M(s_x, \sigma_1\sigma_2\sigma_3\ldots\sigma_p)$$

$$\vdots$$

$$\exists t_{r-1}\colon\ M(M(M(s_x, \sigma_1), \sigma_2), t_{r-1}) = M(s_x, \sigma_1)$$

$$\exists t_r\colon\ M(M(s_x, \sigma_1), t_r) = s_x.$$

Therefore, $M(s_y, t_1t_2\ldots t_{r-1}) = s_x$. Q.E.D.

DEFINITION 23: Two states s_x, s_y are connected if there exist input sequences t_1, t_2 that are such that $M(s_x, t_1) = s_y$ and $M(s_y, t_2) = s_x$.

LEMMA 7: If two persistent states s_x and s_y are synchronizable, they are connected.

PROOF OF LEMMA 7:

CASE I: There exists an input sequence t that is such that $M(s_x, t) = M(s_y, t) = s_x$ (or s_y). By Lemma 6, there exists an input sequence t' that is such that $M(s_x, t') = s_y$ (or $M(s_y, t') = s_x$).

CASE II: There exists an input sequence t that is such that $M(s_x, t) = M(s_y, t) = s_z$. By Lemma 6 there exist t' that is such that $M(s_z, t') = s_x$, and t" that is such that $M(s_z, t") = s_y$. Therefore $M(s_x, tt") = s_y$ and $M(s_y, tt') = s_x$. Q.E.D.

We now have the following theorem:

THEOREM 18: In a synchronizable automaton, if a set of p states s_i, s_j, s_k $\ldots$ is synchronizable with a persistent state s_u, an upper bound on the length of the shortest synchronizing sequence that synchronizes this set of states with s_u is $(p-1)\,C_2^n + (n-1)$.

PROOF 18: By Theorem 16, there exists an input sequence t with $L(t) \leq (p-1)\,C_2^n$ that will synchronize the set of states s_i, s_j, s_k $\ldots$ with an arbitrary state s_v. s_u and s_v are synchronizable because the automaton A is synchronizable. If s_v is a persistent

state, by Lemma 7, s_u and s_v are connected. There exists an input sequence t' that is such that $M(s_v, t') = s_u$. tt' will then be the synchronizing sequence that synchronizes s_i, s_j, s_k ... with s_u. If s_v is a transient state, there exists an input sequence t" that is such that $M(s_v, t") = s_y$, where s_y is a persistent state. However, s_y and s_u are connected; thus there exists an input sequence t"' that is such that $M(s_y, t"') = s_u$. That is, $M(s_v, t"t"') = s_u$, and tt"t"' is then the synchronizing sequence. Since both L(t') and L(t"t"') are less than or equal to n-1, we have proved the theorem. Q. E. D.

We finally can establish the bounds for absolutely synchronizable automata.

THEOREM 19: In an absolutely synchronizable automaton, an upper bound on the length of the shortest synchronizing sequence is n-1.

PROOF 19: This bound is obvious because an absolutely synchronizable automaton has a finite memory span. Q. E. D.

THEOREM 20: In an absolutely synchronizable automaton, an upper bound on the length of the shortest synchronizing sequence with a specific state s_u is n-1.

PROOF 20: By Theorem 14, we know that s_u must be a persistent state. By the definition of a persistent state (Definition 12), we know that there exists an input sequence t' with $L(t') \leq n - 1$ that is such that $M(M(s_u, \sigma), t') = s_u$. Because the automaton has a finite memory span, t' is a synchronizing sequence that synchronizes the automaton with s_u. Q. E. D.

4.6 COMBINED AUTOMATA

We shall introduce here some ideas that should help to reduce the labor in checking the synchronizability of sets of states in an automaton and may give a better bound on the length of the shortest synchronizing sequence. In Section III we defined a series of combined automata A_{c1}, A_{c2}, A_{c3} ... for an automaton A, where A_{c1} is derived from A by combining all structurally indistinguishable states in A, and A_{c2} is derived from A_{c1} by combining all structurally indistinguishable states in A_{c1}, and so forth. We called A_{cx} the exhaustively combined automaton of A, if A_{cx} is the last member of the series A_{c1}, A_{c2}, A_{c3} ... so that there will be no more combinable sets of indistinguishable states. In other words, for any pair of states s_i and s_j in A_{cx}, there exists an input symbol σ that is such that $M_{cx}(s_i, \sigma) \neq M_{cx}(s_j, \sigma)$. Notice that in the series of automata A, A_{c1}, A_{c2}, A_{c3} ... A_{cx}, every automaton is a homomorphic image of each of the previous automata. In particular, A_{cx} is an homomorphic image of A. Let c_x denote the mapping that maps the states in S of A into the states in S_{cx} of A_{cx}. From the previously adopted notations, for a state s_j in S_{cx} of A_{cx}, $c_x^{-1}(s_j)$ denotes the set of states in A which are mapped into s_j by c_x. We then have the following theorem:

THEOREM 21: For any state s_j in A_{cx}, the set of states $c_x^{-1}(s_j)$ in A is absolutely synchronizable by any input sequence of a length equal to or larger than a constant x. Moreover, $x \leq n - 1$, where n is the number of states in A.

PROOF 21: Recalling the series of partitions P_o, P_1 ... defined in Section III, we

see that $c_x^{-1}(s_j)$ is a set of states in the same block in the partition P_x. After x steps of transition, all states of this set will enter the same state, that is, they are absolutely synchronizable by input sequences of length x. It was shown in Section III that $x \leq n - 1$; thus the proof of the theorem is completed. Q. E. D.

THEOREM 22: If a set of states $\{s_i, s_j, s_k \ldots\}$ in A_{cx} is synchronizable, then the set of states $\{c_x^{-1}(s_i), c_x^{-1}(s_j), c_x^{-1}(s_k) \ldots\}$ in A is synchronizable.

PROOF 22: By assumption, there exists an input sequence t that is such that $M_{cx}(s_i, t) = M_{cx}(s_j, t) = M_{cx}(s_k, t) = \ldots = s_u$. Let $c_x^{-1}(s_i) = s_{i1}, s_{i2} \ldots$, $c_x^{-1}(s_j) = s_{j1}, s_{j2} \ldots$, $c_x^{-1}(s_k) = s_{k1}, s_{k2} \ldots$. Because c_x is an automaton homomorphism, $c_x(M(s_{i1}, t)) = M_{cx}(c_x(s_{i1}), t) = M_{cx}(s_i, t) = s_u$, or $M(s_{i1}, t) \in c_x^{-1}(s_u)$. Similarly,

$$M(s_{i2}, t) \in c_x^{-1}(s_u) \quad \ldots$$

$$M(s_{j1}, t) \in c_x^{-1}(s_u) \qquad M(s_{j2}, t) \in c_x^{-1}(s_u) \quad \ldots$$

$$M(s_{k1}, t) \in c_x^{-1}(s_u) \qquad M(s_{k2}, t) \in c_x^{-1}(s_u) \quad \ldots .$$

By Theorem 21, there exists an input sequence t' that is such that $M(s_{i1}, tt') = M(s_{i2}, tt') = \ldots M(s_{j1}, tt') = M(s_{j2}, tt') = \ldots M(s_{k1}, tt') = M(s_{k2}, tt') = \ldots = s_{u1}$. Q. E. D.

COROLLARY 5: If $s_{i1}, s_{j1}, s_{k1} \ldots$ in A are synchronizable, then $s_i, s_j, s_k \ldots$ in A_{cx} are synchronizable.

COROLLARY 6: If s_i and s_j in A_{cx} are not synchronizable, then no pair of states from $c_x^{-1}(s_i)$ and $c_x^{-1}(s_j)$ is synchronizable, and vice versa.

PROOF OF COROLLARY 6: Assume that there is a pair of synchronizable states from $c_x^{-1}(s_i)$ and $c_x^{-1}(s_j)$. By Corollary 5, s_i and s_j are synchronizable, a contradiction of the conditions of Corollary 6.

Conversely, assume that s_i and s_j are synchronizable. By Theorem 22 any pair of states from $c_x^{-1}(s_i)$ and $c_x^{-1}(s_j)$ is synchronizable, also an obvious contradiction.

Q. E. D.

From Corollaries 5 and 6, we see that the synchronizability of sets of states in A can be tested in A_{cx}, which consists of fewer states. Moreover, the bounds proposed in Theorems 16-18 can be improved too, as Figs. 45-48 illustrate. (Note that in Fig. 45 the states in A_{cx} are primed for clarity.) To check the synchronizability of A, we only have to check the synchronizability of A_{cx}. Since A_{cx} is synchronizable, A is synchronizable. Applying Theorem 17 directly to the automaton A, we have $(9-1)\, C_2^9 = 288$ as an upper bound on the length of the shortest synchronizing sequence. Applying Theorem 17 to A_{cx}, however, we have, for an upper bound, $(n'-1)\, C_2^{n'} + (n-1) = (4-1)\, C_2^4 + (9-1) = 26$, where n' is the number of states in A_{cx}. Our reasoning is as follows. In A_{cx} there is a synchronizing sequence of length $(n'-1)\, C_2^{n'}$ or less. From Proof 22, we see that this sequence will also lead the set of all states in A to a set of states that is

S \ Σ	0	1
s_1	s_8	s_3
s_2	s_5	s_7
s_3	s_9	s_2
s_4	s_1	s_4
s_5	s_5	s_1
s_6	s_4	s_3
s_7	s_8	s_3
s_8	s_7	s_9
s_9	s_1	s_4

S_{cx} \ Σ	0	1
s'_1	s'_4	s'_3
s'_2	s'_2	s'_1
s'_3	s'_4	s'_2
s'_4	s'_1	s'_4

A_{cx}

$c_x^{-1}(s'_1) = s_1, s_6, s_7$

$c_x^{-1}(s'_2) = s_2, s_5$

$c_x^{-1}(s'_3) = s_3$

$c_x^{-1}(s'_4) = s_4, s_8, s_9$

Figure 45

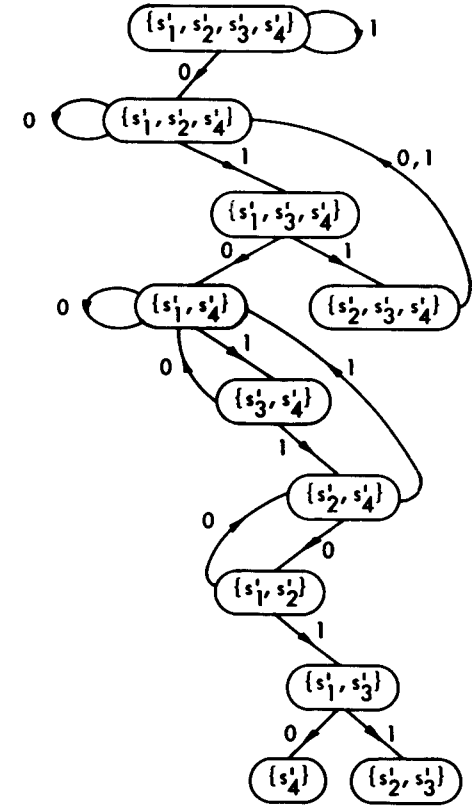


Figure 46

absolutely synchronizable. Thus by Theorem 21 any sequence of length n-1 will synchronize this set of states. Moreover, we can actually find a synchronizing sequence for A by finding one for A_{cx} first. Construct the essential part of $A_{cx}^{\{4\}}$ as shown in Fig. 46. We see that the shortest synchronizing sequence for A_{cx} is 01011010, which will synchronize all states to s_4'. Since $c_x^{-1}(s_4') = \{s_4, s_8, s_9\}$, we want to find a synchronizing sequence to synchronize s_4, s_8, s_9 in A. A portion of the essential part of $A^{\{9\}}$ is shown in Fig. 47. From Fig. 47, 0101101000, 0101101001, 0101101010, and 0101101011 are synchronizing sequences for A. Notice that, in some cases, the synchronizing sequence obtained in this manner is not necessarily the shortest possible synchronizing sequence. For example, in Fig. 46, the sequence 01011011 leads all states in A_{cx} to $\{s_2', s_3'\}$. It might be possible (but not in this example) that the set of states $\{c_x^{-1}(s_2'), c_x^{-1}(s_3')\} = \{s_2, s_5, s_3\}$ has a synchronizing sequence shorter than that of the set $\{s_4, s_8, s_9\}$. Then there would be synchronizing sequences for A which are shorter than those found above. In this example, however, the sequences that we found are really the shortest synchronizing sequences.

Furthermore, we can tell immediately that s_1, s_6, s_7 in A are synchronizable by a synchronizing sequence of a length equal to or less than 9 - 1 = 8 (compared with the bound (3-1) $C_2^9 = 72$ given by Theorem 17). The set of states s_1, s_3, s_6, s_7 is synchronizable, and any input sequence of length 3 is one of the shortest synchronizing sequences (s_1' and s_3' in A_{cx} are synchronized with s_4' by the input sequence 0, and the states of the set $c_x^{-1}(s_4')$ are, in turn, synchronized by 00, 01, 10, 11).

In Fig. 48, A is not synchronizable because A_{cx} is not synchronizable. Moreover, none of the sets of states $\{s_1, s_2\}$, $\{s_1, s_4, s_5\}$, $\{s_1, s_5\}$... are synchronizable because

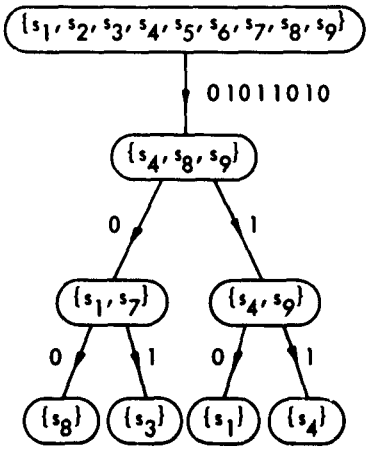


Figure 47

S \ Σ	0	1
s_1	s_5	s_4
s_2	s_1	s_3
s_3	s_3	s_2
s_4	s_5	s_4
s_5	s_4	s_3

A

S_{cx} \ Σ	0	1
s'_1	s'_2	s'_1
s'_2	s'_1	s'_3
s'_3	s'_3	s'_2

A_{cx}

$c_x^{-1}(s'_1) = s_1, s_4$

$c_x^{-1}(s'_2) = s_2, s_5$

$c_x^{-1}(s'_3) = s_3$

Figure 48

$\{s'_1, s'_2\}$ is not synchronizable.

4.7 SOME APPLICATIONS

Suppose that there is an automaton that is in an unknown initial state and we want to reset the automaton to a preselected state s_i, but there is no resetting switch available. If the automaton is synchronizable with s_i, we need only apply an appropriate synchronizing sequence. As another example, suppose that several copies of identical automata in different unknown initial states are connected in parallel (they are accepting identical input sequences) and we want to reset them to the same initial state or to a certain specific initial state. The principles of synchronization can be applied immediately.

Synchronization also can be applied to an automaton used as a decoder. The automaton starts from a certain initial state s_1, receives an input code word, decodes it, gives the corresponding outputs, returns to s_1, and waits for the next input code word. If $t_1, t_2, t_3 \ldots$ are code words, it is certainly true that $M(s_1, t_1) = s_1$, $M(s_1, t_2) = s_1$, $M(s_1, t_3) = s_1 \ldots$. However, if some errors occur during the transmission of a string of code words, for example, some input symbols are received incorrectly or some input symbols are erased, then the normal decoding procedure would be disturbed. If the automaton does not return to s_1 after decoding a certain code word because of some error occurring in that code word, the subsequent code word will not be decoded correctly. This effect might propagate, and the automaton thus would give many successive meaningless output symbols. We naturally ask the question: How can we prevent the normal decoding procedure from being disturbed, although errors may occur during the transmission? The answer is that the code words $t_1, t_2, t_3 \ldots$ must be synchronizing sequences that will synchronize the automaton with s_1. Suppose that a string of

code words $t_i t_j t_k \ldots$ is transmitted and some error occurs during the transmission of t_i. $t_i' t_j t_k \ldots$ is then received. t_i' will not be decoded correctly. Since $M(s_1, t_i')$ is not necessarily equal to s_1, the code word t_j may be decoded incorrectly too. However, $M(s_1, t_i' t_j) = M(M(s_1, t_i'), t_j) = s_1$ because t_j is a synchronizing sequence with s_1. Therefore, t_k and the subsequent code words will be decoded correctly. We see that in this example the effect of errors will never propagate beyond a correctly received code word.

4.8 SYNCHRONIZATION OF DIFFERENT AUTOMATA

In two automata $A = \{\Sigma, S, M\}$ and $B = \{\Sigma, S', M'\}$, a pair of states s_i and s_i' is said to be synchronizable simultaneously with another pair of states s_u and s_u' if there exists an input sequence t of finite length which is such that $M(s_i, t) = s_u$ and $M'(s_i', t) = s_u'$, where $s_i, s_u \in S$ and $s_i', s_u' \in S'$. By extending this definition, a set of states $s_i, s_j, \ldots \in S$ and a set of states $s_i', s_j' \ldots \in S'$ are said to be synchronizable simultaneously (with s_u and s_u') if there exists an input sequence t that is such that $M(s_i, t) = M(s_j, t) = \ldots (= s_u)$ and $M'(s_i', t) = M'(s_j', t) = \ldots (= s_u')$. Furthermore, A and B are said to be synchronizable simultaneously (with s_u and s_u') if the set of all states in S and the set of all states in S' are synchronizable simultaneously (with s_u and s_u'). To investigate the synchronizability of sets of states in different automata, we, again, can employ the idea of product automata which we developed before. In general, for the synchronization of a set of p states in A and a set of q states in B, we can study the automaton $A^{\{p\}} \times B^{\{q\}}$. With respect to the length of the synchronizing sequence, we have the following bounds.

THEOREM 23: If a set of p states in A and a set of q states in B are simultaneously synchronizable with s_u and s_u', the upper bound on the length of the shortest synchronizing sequence is

$$\sum_{i=1}^{p} C_i^n \times \sum_{j=1}^{q} C_j^{n'} - 1,$$

where n and n' are the number of internal states in A and B, respectively.

PROOF 23: $A^{\{p\}}$ has $\sum_{i=1}^{p} C_i^n$ states and $B^{\{q\}}$ has $\sum_{j=1}^{q} C_j^{n'}$ states; therefore, $A^{\{p\}} \times B^{\{q\}}$ has $\sum_{i=1}^{p} C_i^n \times \sum_{j=1}^{q} C_j^{n'}$ states. Q.E.D.

COROLLARY 7: If a set of p states in A and a set of q states in B are synchronizable simultaneously, the upper bound on the length of the shortest synchronizing sequence is

$$\sum_{i=1}^{p} C_i^n \times \sum_{j=1}^{q} C_j^{n'} - n \times n' + 1.$$

PROOF OF COROLLARY 7: In $A^{\{p\}} \times B^{\{q\}}$, there are $n \times n'$ states that are ordered pairs like $(\{s_u\},\{s'_u\})$. Q.E.D.

THEOREM 24: If A is synchronizable and B is synchronizable, then A and B are synchronizable simultaneously.

PROOF 24:

$$\exists t: \; M(s_1, t) = M(s_2, t) = \ldots M(s_n, t) = s_v$$

$$\exists t': \; M'(s'_1, t') = M'(s'_2, t') = \ldots \; M'(s'_n, t') = s'_v.$$

Then,

$$M(s_1, tt') = M(s_2, tt') = \ldots \; M(s_n, tt') = M(s_v, t')$$

$$M'(s'_1, tt') = M'(s'_2, tt') = \ldots \; M'(s'_n, tt') = s'_v. \qquad \text{Q.E.D.}$$

COROLLARY 8: If A and B are synchronizable simultaneously, the upper bound on the length of the shortest synchronizing sequence is $(n-1)\,C_2^n + (n'-1)\,C_2^{n'}$.

PROOF OF COROLLARY 8: By Theorem 17, the bounds on the lengths of the shortest possible t and t' are $(n-1)\,C_2^n$ and $(n'-1)\,C_2^{n'}$, respectively. Q.E.D.

THEOREM 25: If A is absolutely synchronizable by any sequence of length equal to or larger than m+1 and B is absolutely synchronizable by any sequence of length equal to or larger than m'+1, then any set of states in A and any set of states in B can be synchronized simultaneously by any sequence of length equal to or larger than m+1 or m'+1, whichever is the larger.

PROOF 25: Suppose that $m > m'$ (the proof for $m' > m$ is identical). For any sets of states $s_i, s_j \ldots \in S$ and $s'_i, s'_j \ldots \in S'$, and for any t with $L(t) \geq m + 1$, by the definition of absolute synchronizability

$$M(s_i, t) = M(s_j, t) = \ldots$$

$$M'(s'_i, t) = M'(s'_j, t) = \ldots . \qquad \text{Q.E.D.}$$

In order to reduce the labor of checking the synchronizability and to obtain a better estimation of the bounds on the length of the shortest synchronous sequences, we can combine indistinguishable states in each individual automaton as we did in section 4.6. The ideas introduced previously can be applied directly to the synchronization of two different automata.

The synchronization of more than two different automata is a direct extension of the synchronization of two different automata.

V. MEMORY ORDER OF STATES

5.1 MEMORY ORDER OF STATES WITH RESPECT TO THE INPUT

In Sections III and IV we discussed the cases in which the input sequence to an automaton is known but the initial state of the automaton is unknown. In this section we shall investigate the case in which a portion of the input sequence to the automaton is unknown although the initial state is known. We start with the following definitions.

DEFINITION 24: In an automaton $A = \{\Sigma, S, M\}$, a state s_i has a memory order equal to p, if $M(s_i, t_1 t) = M(s_i, t_2 t)$ for any input sequences t_1, t_2 with $L(t_1) = L(t_2) = p$ and t with $L(t) \geq K$, where K is a constant for this specific automaton, and if there exist input sequences t_3, t_4 with $L(t_3) = L(t_4) = p + 1$, and t' of any arbitrary length which is such that $M(s_i, t_3 t') \neq M(s_i, t_4 t')$.

Definition 24 means that if an automaton is in a state having a memory order equal to p, then p successive unknown input symbols can be applied without jeopardizing our ability to discover eventually the state of the automaton by observing the next K input symbols. From this definition, we see that if a state s_i does not have a memory order p, it will not have memory order larger than p.

DEFINITION 25: In an automaton $A = \{\Sigma, S, M\}$, a state s_i has an infinite memory order, if $M(s_i, t_1 t) = M(s_i, t_2 t)$ for any input sequences t_1, t_2 with $L(t_1) = L(t_2)$ and t with $L(t) \geq K$, where K is a constant for this specific automaton.

a. Memory Order of States in Exhaustively Combined Automata

From Definitions 24 and 25, the memory order of a state in a given automaton may be found by enumerating all possible input sequences and examining every possible value of p. This method is obviously very tedious. Furthermore, we shall not know whether or not we should stop testing if this state has an infinite memory order. An effective method to determine the memory order of a state will be introduced, but the simpler case of an exhaustively combined automaton is studied first. As defined previously, an exhaustively combined automaton $A_{cx} = \{\Sigma, S_{cx}, M_{cx}\}$ is one in which for all s_i, $s_j \in S_{cx}$ there exists an input symbol σ that is such that $M_{cx}(s_i, \sigma) \neq M_{cx}(s_j, \sigma)$.

THEOREM 26: In an exhaustively combined automaton $A_{cx} = \{\Sigma, S_{cx}, M_{cx}\}$, a state s_i has a memory order equal to p if and only if (a) $M_{cx}(s_i, t_1) = M_{cx}(s_i, t_2)$ for all t_1 and t_2 whose lengths are equal to p and (b) there exist t_3 and t_4 whose lengths are equal to p + 1 which are such that $M_{cx}(s_i, t_3) \neq M_{cx}(s_i, t_4)$.

PROOF 26: First, we prove the necessity. Suppose that there exist t_1' and t_2' whose lengths are equal to p so that $M_{cx}(s_i, t_1') \neq M_{cx}(s_i, t_2')$. Because A_{cx} is exhaustively combined, there exists σ_a that is such that $M_{cx}(M_{cx}(s_i, t_1'), \sigma_a) \neq M_{cx}(M_{cx}(s_i, t_2'), \sigma_a)$. Similarly, there exists σ_b that is such that $M_{cx}(M_{cx}(s_i, t_1' \sigma_a), \sigma_b) \neq M_{cx}(M_{cx}(s_i, t_2' \sigma_a), \sigma_b)$. Repeating this argument, we can have an input sequence $t = \sigma_a \sigma_b \ldots$ of any arbitrary length so that $M_{cx}(s_i, t_1' t) \neq M_{cx}(s_i, t_2' t)$. Therefore s_i has a memory order smaller than

p. Suppose that for all t_3' and t_4' whose lengths are equal to p + 1, $M_{cx}(s_i, t_3') = M_{cx}(s_i, t_4')$. For any t, $M_{cx}(s_i, t_3't) = M_{cx}(s_i, t_4't)$. Therefore, s_i has a memory order larger than p.

Second, we prove the sufficiency. Since $M_{cx}(s_i, t_1) = M_{cx}(s_i, t_2)$ for all t_1 and t_2 whose lengths are equal to p, $M_{cx}(s_i, t_1t) = M_{cx}(s_i, t_2t)$ for any t. Since there exist t_3 and t_4 whose lengths are equal to p + 1 so that $M_{cx}(s_i, t_3) \neq M(s_i, t_4)$, by the same argument used above there exists an input sequence t of any arbitrary length which is such that $M_{cx}(s_i, t_3t) \neq M(s_i, t_4t)$. Q. E. D.

COROLLARY 9: In an exhaustively combined automaton $A_{cx} = \{\Sigma, S_{cx}, M_{cx}\}$, a state has an infinite memory order if and only if for all t_1 and t_2 whose lengths are equal, $M_{cx}(s_i, t_1) = M_{cx}(s_i, t_2)$.

By Theorem 26, we can easily recognize a state having a zero memory order in an exhaustively combined automaton. If a state s_i has a memory order equal to zero, there exist input symbols σ_1 and σ_2 that are such that $M_{cx}(s_i, \sigma_1) \neq M_{cx}(s_i, \sigma_2)$. On the other hand, if a state s_i has a nonzero memory order, we shall have $M_{cx}(s_i, \sigma_1) = M_{cx}(s_i, \sigma_2)$ for any input symbols σ_1 and σ_2. The following theorem will be helpful in determining the memory orders of states.

THEOREM 27: In an exhaustively combined automaton $A_{cx} = \{\Sigma, S_{cx}, M_{cx}\}$, if a state s_i has a nonzero finite memory order equal to p, that is, $\infty > p > 0$, then for any input symbol σ_1, $M(s_i, \sigma_1)$ has a memory order equal to (p-1).

PROOF 27: (a) For any input sequences t_1 and t_2 of length p - 1, $M_{cx}(M_{cx}(s_i, \sigma_1), t_1) = M_{cx}(M_{cx}(s_i, \sigma_1), t_2)$, since σ_1t_1 and σ_1t_2 are input sequences of length p.

(b) There exist t_3 and t_4 of length p + 1 so that $M_{cx}(s_i, t_3) \neq M_{cx}(s_i, t_4)$. Writing $t_3 = \sigma_a t_3'$ and $t_4 = \sigma_b t_4'$, we have $M_{cx}(M_{cx}(s_i, \sigma_a), t_3') \neq M(M_{cx}(s_i, \sigma_b), t_4')$. Since $M_{cx}(s_i, \sigma_1) = M_{cx}(s_i, \sigma_a) = M_{cx}(s_i, \sigma_b)$, then $M_{cx}(M_{cx}(s_i, \sigma_1), t_3') \neq M_{cx}(M_{cx}(s_i, \sigma_1), t_4')$. Q. E. D.

COROLLARY 10: For any state s_i having a finite memory order equal to p in an exhaustively combined automaton $A_{cx} = \{\Sigma, S_{cx}, M_{cx}\}$ consisting of n states, $p \leq n - 1$.

COROLLARY 11: In an exhaustively combined automaton $A_{cx} = \{\Sigma, S_{cx}, M_{cx}\}$, if a state s_i has an infinite memory order, then $M_{cx}(s_i, \sigma_1)$ has an infinite memory order for any input symbol σ_1.

To determine the memory order of a certain state s_i, we shall trace the step-by-step transitions from s_i. If a state having a memory order equal to zero is entered, we know that s_i has a finite memory order. If a state having a memory order equal to zero has not been entered after n-1 steps of transition, s_i has an infinite memory order. Figure 49 illustrates the manner in which we find the memory orders of the states of an exhaustively combined automaton. In A_{cx}, s_4 has a memory order equal to zero. Since $M_{cx}(s_2, 0) = M_{cx}(s_2, 1) = s_4$, s_2 has a memory order equal to 1; then s_3 has a memory order equal to 2; s_1 and s_5 have infinite memory orders. If we draw the transition diagram as that shown in Fig. 49b, the memory order of each state is evident.

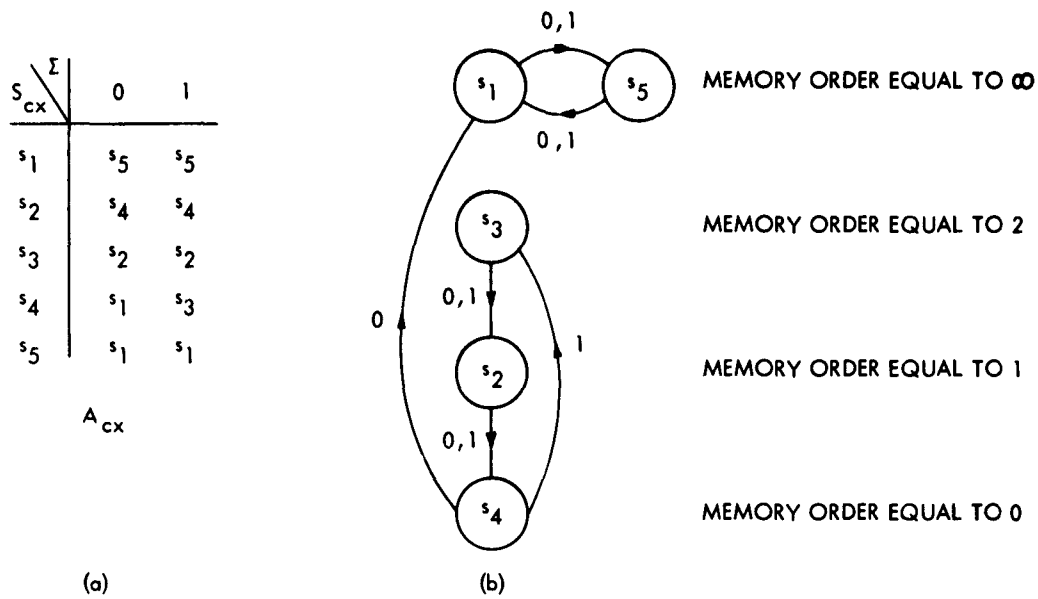


Figure 49

b. The General Case of All Automata

The result obtained above can be extended immediately to all automata. We have the following theorems:

THEOREM 28: Let $A_{cx} = \{\Sigma, S_{cx}, M_{cx}\}$ be the exhaustively combined automaton of $A = \{\Sigma, S, M\}$. Any state s_i in S has the same memory order as $c_x(s_i)$ in S_{cx}.

PROOF 28: (a) Suppose that $c_x(s_i)$ has a memory order p. That is, for any input sequences t_1 and t_2 of length p, $M_{cx}(c_x(s_i), t_1) = M_{cx}(c_x(s_i), t_2) = s'_u$. Because A_{cx} is a homomorphic image of A, $M(s_i, t_1) \in c_x^{-1}(s'_u)$ and $M(s_i, t_2) \in c_x^{-1}(s'_u)$. But by Theorem 21 $M(s_i, t_1)$ and $M(s_i, t_2)$ are absolutely synchronizable by any input sequence t of length equal to or larger than n-1, where n is the number of states in A. Therefore, $M(s_i, t_1 t) = M(s_i, t_2 t)$.

(b) Since the state $c_x(s_i)$ has a memory order p, there exist input sequences t_3 and t_4 of length p + 1, so that $M_{cx}(c_x(s_i), t_3) \neq M_{cx}(c_x(s_i), t_4)$. Also, as shown in Proof 26, there exists an input sequence t of arbitrary length so that $M_{cx}(c_x(s_i), t_3 t) \neq M_{cx}(c_x(s_i), t_4 t)$. Therefore, $M(s_i, t_3 t) \neq M(s_i, t_4 t)$. Q.E.D.

COROLLARY 12: In Definitions 24 and 25 $K \leq n - 1$, where n is the number of states of the automaton A.

Figure 50 shows the manner in which we find the memory order of each state of a given automaton. From the state diagram of A_{cx} in Fig. 50, we see that s'_1 has a memory order equal to 3, s'_2 has a memory order equal to 2, s'_4 has a memory order equal to 1, and s'_3 has a memory order equal to zero. In A, s_1 has a memory order equal

S \ Σ	0	1
s_1	s_2	s_5
s_2	s_4	s_8
s_3	s_1	s_4
s_4	s_3	s_7
s_5	s_9	s_4
s_6	s_1	s_8
s_7	s_1	s_4
s_8	s_7	s_7
s_9	s_6	s_3

S_{cx} \ Σ	0	1
s'_1	s'_2	s'_2
s'_2	s'_4	s'_4
s'_3	s'_1	s'_4
s'_4	s'_3	s'_3

A_{cx}

$c_x^{-1}(s'_1) = s_1$

$c_x^{-1}(s'_2) = s_2, s_5$

$c_x^{-1}(s'_3) = s_3, s_6, s_7$

$c_x^{-1}(s'_4) = s_4, s_8, s_9$

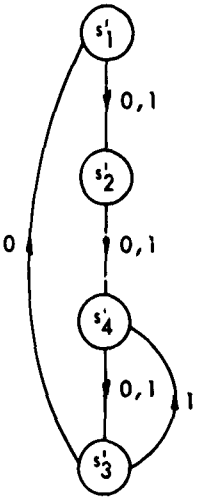


Figure 50

to 3; s_2 and s_5 have memory orders equal to 2; s_4, s_8, and s_9 have memory orders equal to 1, and s_3, s_6, and s_7 have memory orders equal to zero.

Theorem 27 and Corollaries 10 and 11 can be extended as follows:

THEOREM 29: In an automaton $A = \{\Sigma, S, M\}$, if a state s_i has a nonzero finite memory order equal to p, then $M(s_i, \sigma)$ has a memory order equal to (p-1) for any input symbol σ.

PROOF 29: By Theorems 27 and 28. Q. E. D.

COROLLARY 13: For any state s_i having a finite memory order equal to p in an automaton A consisting of n states, $p \leq n - 1$.

PROOF OF COROLLARY 13: The number of states of the exhaustively combined automaton A_{cx} is less than or equal to the number of states of A. Q. E. D.

COROLLARY 14: In an automaton $A = \{\Sigma, S, M\}$, if a state s_i has an infinite memory order, then for any input symbol σ, $M(s_i, \sigma)$ has an infinite memory order.

Notice that in an automaton having a finite memory span with respect to the input, every state has an infinite memory order. This fact follows from the discussion in Section III that the exhaustively combined automaton of an automaton having a finite memory span with respect to the input consists of only one state. However, the converse of this statement is not necessarily true. That is, an automaton does not necessarily have a finite memory span if each of its states has an infinite memory order.

From Theorem 29, we can develop a very illustrative pictorial representation of the memory order of a state in an automaton, as illustrated in Fig. 51. Consider a

staircase of p steps. An initial state having a memory order equal to p will correspond to a ball sitting on the p^{th} step of the staircase (see Fig. 51a). For any input symbol, known or unknown, the transition will lead to a state having a memory order equal to p - 1. This corresponds to moving the ball one step down to the $(p-1)^{th}$ step of the staircase. For an automaton in a state having an infinite memory order, the correspondence is to a ball sitting on a platform of infinite length on which the ball rolls with no change of its height (Fig. 51b). For an automaton in a state having a zero memory order, the correspondence is to a ball sitting at the end of a staircase. For a known input symbol, the ball will be moved to some state having a memory order equal to ∞, p or 0. For an unknown input symbol, the ball will drop to minus infinity (Fig. 51c). This indicates that if the automaton is in a state having a memory order equal to zero and the next input is unknown, we shall lose track of the final state of the automaton.

If the initial state of the automaton in Fig. 49 is s_2, then Fig. 52a illustrates the effect of an input sequence – 1 – – 0 – –, and Fig. 52b illustrates the effect of an input sequence – 1 – – – (the – are unknown input symbols). We can see that, for an initial state s_i, if we want to find $M(s_i, t)$, all that we have to know about the input sequence t are the last n-1 input symbols and those "critical input symbols" that specify the transitions from states having a memory order equal to zero. For the example in Fig. 52a, among the symbols in the sequence – 1 – – 0 – –, only 1 and 0 are critical input symbols. In other words, if an automaton starts from a certain initial state and if the critical input symbols and the last n-1 input symbols of two input sequences to the automaton are identical, the final state of the automaton must be the same for both sequences.

In a practical situation, when the input symbols are sent through a binary erasure channel to an automaton, all of the symbols other than the critical input symbols and the last n-1 symbols of the sequence can be erased as long as we are concerned with only the final state of the automaton (Fig. 53). Furthermore, if the transmission channel is a binary symmetric channel, then all input symbols other than the critical ones and the last n-1 can tolerate an error.

5.2 MEMORY ORDER OF STATES WITH RESPECT TO THE INPUT-OUTPUT SEQUENCE

We can now extend our discussion to the case in which we have access to both the input and the output symbols. Similar to the definitions for the case of the memory orders of states with respect to the input symbols are Definitions 26 and 27.

DEFINITION 26: In an automaton $A = \{\Sigma, Z, S, M, N\}$, a state s_i has a memory order equal to p with respect to the input-output sequence, if (a) $R(s_i, w_1 w) \neq R(s_j, w_2 w)$ for any input-output sequences w_1 and w_2 with $L(w_1) = L(w_2) = p$ and w with $L(w) \geq K$, where K is a constant for this specific automaton, and (b) there exist input-output sequences w_3 and w_4 with $L(w_3) = L(w_4) = p + 1$, and exists input-output sequence w of any arbitrary length, so that $R(s_i, w_3 w)$ and $R(s_j, w_4 w)$ are incompatible.

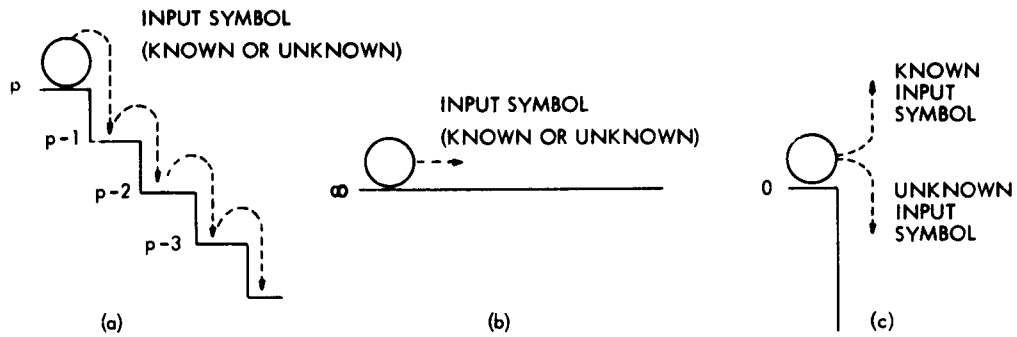


Figure 51

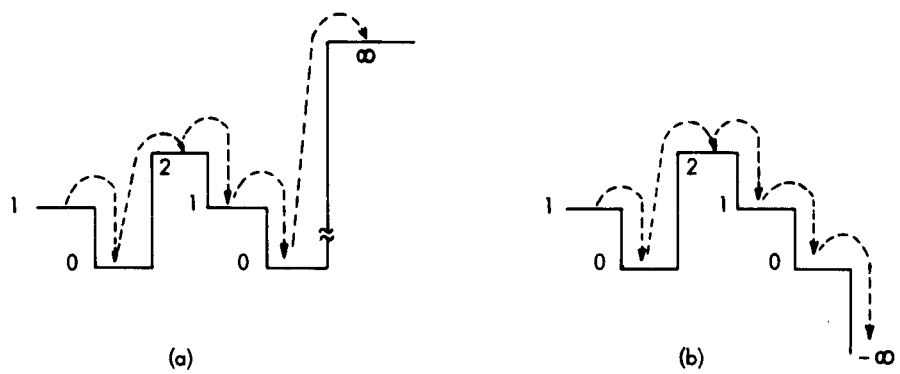


Figure 52

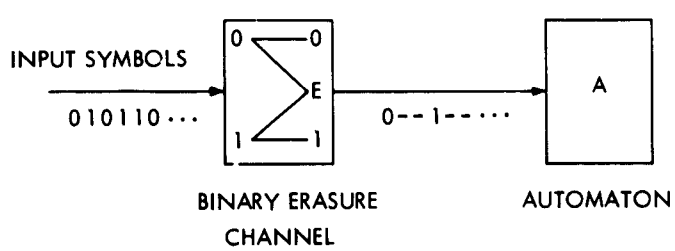


Figure 53

DEFINITION 27: In an automaton $A = \{\Sigma, Z, S, M, N\}$, a state s_i has an infinite memory order with respect to the input-output sequence, if $R(s_i, w_1 w) * R(s_j, w_2 w)$ for any input-output sequences w_1 and w_2 with $L(w_1) = L(w_2)$ and w with $L(w) \geq K$, where K is a constant for this specific automaton.

As we can see, the method developed above to find the memory order with respect to the input of the states of an automaton does not apply in the present case, since the compatibility relation of the function R is not transitive. (See discussion following Corollary 1.) We have another method to check the memory order of a state with respect to the input-output sequence. Following the notations and definitions in Section III, we have

DEFINITION 28: A set of states $s_i, s_j, s_k \ldots$ in an automaton $A = \{\Sigma, Z, S, M, N\}$ has a unique final state with respect to the input-output sequence if, for all s_u and s_v in the set and for all input-output sequences w of length equal to or larger than a constant K, $R(s_u, w) * R(s_v, w)$.

THEOREM 30: A set of states $s_i, s_j, s_k \ldots$ in an automaton $A = \{\Sigma, Z, S, M, N\}$ will have a unique final state with respect to the input-output sequence if and only if every pair of states in this set has a unique final state with respect to the input-output sequence. (Theorem 30 is a slightly more general form of Theorem 10.)

PROOF 30: First, we prove the necessity. Suppose that there is a pair of states s_i and s_j which does not have a unique final state with respect to the input-output sequence, that is, there exists an input-output sequence w of length equal to or larger than any constant K, so that $R(s_i, w)$ and $R(s_j, w)$ are incompatible. It is obvious that this set of states does not satisfy Definition 28 as a set having a unique final state with respect to the input-output sequence.

Second, we prove the sufficiency. For any pair of states s_i and s_j, and for any (σ, z) in $\Sigma \times Z$, if $R(s_i, (\sigma, z)) * R(s_j, (\sigma, z))$, then s_i and s_j obviously have a unique final state with respect to the input-output sequence. If the values of $R(s_i, (\sigma, z))$ and $R(s_j, (\sigma, z))$ are not equal, they must, in turn, have a unique final state with respect to the input-output sequence. We can repeat this argument. The repetition, however, will not go beyond $n(n-1)/2$ steps, since there are a total of $n(n-1)/2$ distinct pairs of states in A. Otherwise, s_i and s_j will not have a unique final state with respect to the input-output sequence. Such a result would be a contradiction to the conditions of the theorem. Therefore, for any input-output sequence w with $L(w) \geq n(n-1)/2$, $R(s_u, w) * R(s_v, w)$ for all s_u and s_v in the set.

COROLLARY 15: In Definitions 26-28 $K \leq n(n-1)/2$.

THEOREM 31: Let U_k denote a set of states $\{s_i, s_j, s_k \ldots\}$ and U_{k+1} denote the set of states $\{R(s_i, (\sigma_1, z_1)), R(s_i, (\sigma_1, z_2)) \ldots R(s_j, (\sigma_1, z_1)) \ldots\}$ for all s in U_k and all (σ, z) in $\Sigma \times Z$ which are such that $R(s, (\sigma, z))$ is defined. (a) If U_k does not have a unique final state with respect to the input-output sequence, then U_{k+1} will not have a unique final state with respect to the input-output sequence. (b) If U_{k+1} has a unique final state with respect to the input-output sequence, then U_k has a unique final state with respect

to the input-output sequence.

PROOF 31: (a) Suppose that there is a pair of states s_i and s_j in the set U_k and an input-output sequence $(\sigma_1, z_1)w$ of any arbitrary length which are such that $R(s_i, (\sigma_1, z_1)w)$ and $R(s_j, (\sigma_1, z_1)w)$ are incompatible. That is, $R(R(s_i, (\sigma_1, z_1)), w)$ and $R(R(s_j, (\sigma_1, z_1)), w)$ are incompatible. $R(s_i, (\sigma_1, z_1))$ and $R(s_k, (\sigma_1, z_1))$ must be defined; otherwise they will be compatible. $R(s_i, (\sigma_1, z_1))$ and $R(s_j, (\sigma_1, z_1))$ are in U_{k+1}. Therefore, U_{k+1} does not have a unique final state with respect to the input-output sequence.

(b) Suppose that U_{k+1} has a unique final state with respect to the input-output sequence. If U_k does not have a unique final state with respect to the input-output sequence, by the argument in the preceding paragraph, U_{k+1} will not have a unique final state with respect to the input-output sequence. This result would be a contradiction to our assumption. Q. E. D.

THEOREM 32: For any state s_i having a finite memory order equal to p with respect to the input-output sequence in an automaton A consisting of n states, p is less than or equal to $n + [n(n-1)/2] - 2$.

PROOF 32: Let U_o denote the set $\{s_i\}$. Let U_1 denote the set of states $\{R(s_i, (\sigma_1, z_1)), R(s_i, (\sigma_1, z_2)) \ldots\}$ for all (σ, z) in $\Sigma \times Z$ which are such that $R(s_i, (\sigma, z))$ is defined. Recursively, let U_{k+1} denote the set of states $\{R(s_x, (\sigma_1, z_1)), R(s_x, (\sigma_1, z_2)), \ldots R(s_y, (\sigma_1, z_1)), R(s_y, (\sigma_1, z_2)) \ldots\}$ for all s_x and s_y in U_k and for all (σ, z) in $\Sigma \times Z$ which are such that the function R is defined.

First, we claim that, at most, n sets of states, from U_o up to U_n, would consist of only one state in each set, for if there are n+1 or more sets consisting of only one state in each set, two or more such sets would be identical. That is, there are loops around these sets, and s_i will have an infinite memory order with respect to the input-output sequence.

Second, we claim that if U_{k+1} has a unique final state with respect to the input-output sequence (that is, if each of $U_o, U_1 \ldots U_k, U_{k+1}$ has a unique final state with respect to the input-output sequence by Theorem 31), then there must be a pair of states in U_{k+1} which have not occurred in the same set of the collection $U_o, U_1 \ldots U_k$. Suppose that the converse is true. For a pair of states s_x and s_y in U_{k+1}, any pair of states from the set $\{R(s_x, (\sigma_1, z_1)), R(s_x, (\sigma_1, z_2)) \ldots R(s_y, (\sigma_1, z_1)), R(s_y, (\sigma_1, z_2)) \ldots\}$ must have a unique final state with respect to the input-output sequence because s_x and s_y have appeared in previous sets. Moreover, this relationship is true for all pairs of states in U_{k+1}. Thus U_{k+2} will have a unique final state with respect to the input-output sequence. However, every pair of states in U_{k+2} must have appeared, in pair, in the previous sets $U_o, U_1 \ldots U_{k+1}$. This argument can be repeated, and s_i has an infinite memory order with respect to the input-output sequence. Since the sufficient condition for every state in an automaton to have an infinite memory order with respect to the input-output sequence is that every pair of states has a unique final state (see Theorem 10), if s_i has a finite memory order, then there must be at least one pair of states that does not have a unique final state. That is, there are at most $[n(n-1)/2] - 1$ sets among $U_o, U_1, U_2 \ldots U_k$

that will consist of two or more states and will have a unique final state with respect to the input-output sequence because there are n(n-1)/2 different pairs of states altogether.

Therefore, we can conclude that if s_i has a finite memory order p with respect to the input-output sequence, then $p \leq (n-1) + [n(n-1)/2] - 1$; otherwise, s_i would have an infinite memory order. Q. E. D.

Now we are able to determine the memory order of a state s_i with respect to the

S \ Σ	0	1
s_1	$s_2, 0$	$s_2, 0$
s_2	$s_3, 1$	$s_3, 1$
s_3	$s_4, 0$	$s_4, 1$
s_4	$s_2, 1$	$s_1, 1$

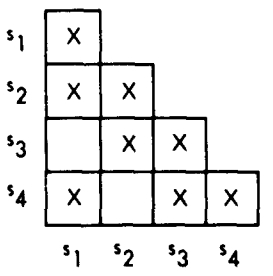
A

(a)

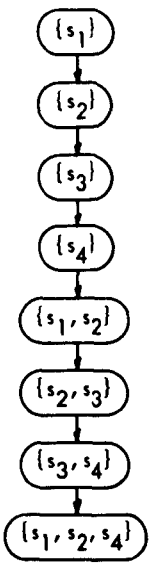
S \ Σ × Z	0,0	0,1	1,0	1,1
s_1	s_2	-	s_2	-
s_2	-	s_3	-	s_3
s_3	s_4	-	-	s_4
s_4	-	s_2	-	s_1

A_r

(b)



(c)



(d)

Figure 54

input-output sequence. If the automaton starts from $U_0 = \{s_i\}$ and if the present input and output symbols are unknown, U_1 will be the set of all possible next states. If U_1 does not have a unique final state with respect to the input-output sequence, then s_i has a zero memory order with respect to the input-output sequence. If U_1 does have a unique final state with respect to the input-output sequence, we can determine whether or not U_2 has a unique final state with respect to the input-output sequence. If not, s_i has a memory order equal to 1 with respect to the input-output sequence; if so, we shall continue to repeat this procedure. From Theorem 32 we know that this procedure will terminate after, at most, $n + [n(n-1)/2] - 2$ repetitions. Figure 54 illustrates this procedure. To find the memory order of s_1 with respect to the input-output sequence, we start from s_1 and step by step list the possible next states for unknown input and output symbols (Fig. 54d). This procedure terminates when we come to the set $\{s_1, s_2, s_4\}$, which does not have a unique final state with respect to the input-output sequence (Fig. 54c). Therefore, s_1 has a memory order equal to 6. In general, we have to follow the same procedure to find the memory orders of s_2, s_3, and s_4. However, it happens that the memory orders of s_2, s_3, and s_4 can be observed immediately to be 5, 4, and 3, respectively, from Fig. 54d. Therefore, no further repetition of this procedure for each state is needed.

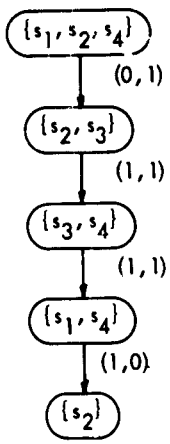


Figure 55

Suppose that we start from s_1 and have an input-output sequence: (–,–) (–,–) (–,–) (–,–) (–,–) (–,–) (0, 1) (1, 1) (1, 1) (1, 0), where the first six input-output symbol pairs are unknown. We can determine the final state by constructing a portion of $A_r^{\{3\}}$ as shown in Fig. 55.

5.3 MEMORY ORDER WITH RESPECT TO THE OUTPUT

As pointed out in Section III, when we study the memory aspects of an automaton with respect to the output symbols, under certain special conditions we can apply the results from the study of the memory aspects of an automaton with respect to the input symbols or with respect to the input and output symbols. Here, we shall study the memory order of states with respect to the output for all automata.

DEFINITION 29: In an automaton $A = \{\Sigma, Z, S, M, N\}$, a state s_i has a memory order equal to p with respect to the output if, for any output sequence y with L(y) greater than or equal to a specific constant K for the automaton A, the set of states $\{Q(s_i, y_1), Q(s_i, y_2) \ldots\}$ (where y_1, y_2 ... are all output sequences of length p so that the function Q is defined), has a unique intermediate state with respect to the output. Moreover, for some output sequence y' of any arbitrary length the set of states $\{Q(s_i, y'_1), Q(s_i, y'_2) \ldots\}$ (where y'_1 and y'_2 ... are output sequences of length p + 1 so that

the function Q is defined), does not have a unique intermediate state with respect to the output.

DEFINITION 30: In an automaton $A = \{\Sigma, Z, S, M, N\}$, a state s_i has an infinite memory order with respect to the output, if for any output sequence y, with L(y) greater than or equal to a specific constant K for the automaton A, the set of states $\{Q(s_i, y_1), Q(s_i, y_2) \ldots\}$ (where $y_1, y_2 \ldots$ are all output sequences of the same length so that the function Q is defined), has a unique intermediate state with respect to the output.

DEFINITION 31: In an automaton $A = \{\Sigma, Z, S, M, N\}$, a state s_i has a minus infinity memory order with respect to the output if there exists a nonempty output sequence y that is such that for some output sequence y' of any arbitrary length $Q(s_i, y)$ does

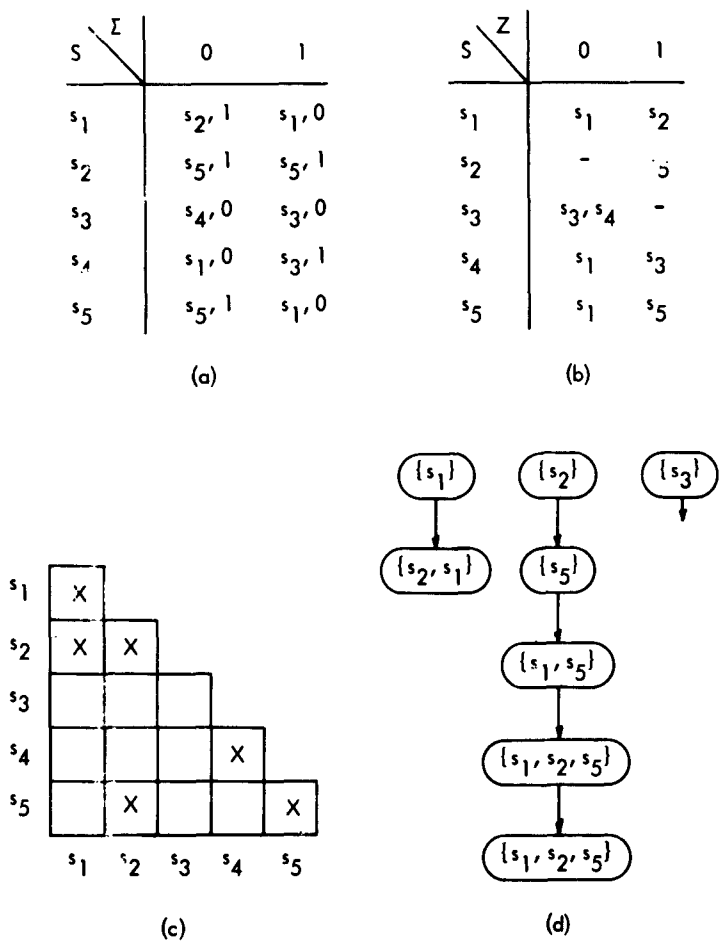


Figure 56

not have a unique intermediate state.

THEOREM 33: For any output sequence y with $L(y) \geq n(n-1)/2$, a set of states $s_i, s_j, s_k \ldots$ in an automaton $A = \{\Sigma, Z, S, M, N\}$ will have a unique intermediate state with respect to the output if every pair of states in this set has a unique next state with respect to the output.

COROLLARY 16: In Definition 29 $K \leq n(n-1)/2$.

The proof of Theorem 33 follows immediately from Lemmas 4 and 5.

THEOREM 34: Let V_k denote the set of states $\{s_i, s_j, s_k \ldots\}$, and V_{k+1} denote the set of states $\{Q(s_i, z_1), Q(s_i, z_2), \ldots Q(s_j, z_1), Q(s_j, z_2), \ldots\}$ for all s in V_k and all z in Z so that the function Q is defined. If V_k does not have a unique intermediate state for some output sequence y' of any arbitrary length, then neither does V_{k+1}. If V_{k+1} has a unique intermediate state for any output sequence y with L(y) larger than or equal to a specific constant K for the automaton A, then so does V_k.

THEOREM 35: For any state s_i having a finite memory order equal to p with respect to the output in an automaton A consisting of n states, $p \leq [n(n-1)/2] - 2$.

The proofs of Theorems 34 and 35 are similar to those of Theorems 31 and 32 and will not be repeated here.

Figure 56 illustrates the procedure for finding the memory order of a state with respect to the output. From Fig. 56, we know that s_1 has a memory order equal to zero with respect to the output, s_2 has a memory order equal to infinity, and s_3 has a memory order equal to minus infinity. A state having a memory order equal to minus infinity is detected easily in the checking table by the unchecked diagonal box.

VI. CONCLUSION

We have discussed several classes of finite automata whose final behaviors can be determined uniquely, even if their initial states or input sequences are not completely known. In Section III we studied the dependence of the final states of finite automata upon their initial states. For an automaton having a finite memory span m with respect to the input, its final state can be determined uniquely whenever the last m+1 input symbols are known. That is, after more than m+1 input symbols are received, the behavior of an automaton having a finite memory span m is affected neither by the initial state of the automaton nor by any input symbol preceding the most recent m+1 symbols.

Our investigation of an automaton having a finite memory span with respect to the input was extended to include an automaton having a finite memory span with respect to the input-output sequence, where the final state of an automaton can be determined when the last (m+1) input symbols to the automaton and their corresponding (m+1) output symbols are known. However, although the information about the initial state of the automaton is not explicitly needed in the determination of the final state, the output sequence corresponding to the input sequence does provide some information about the initial state of the automaton. The output sequence provides this information because for a given input sequence, only the states in a certain subset are the possible initial states that will generate that given output sequence.

A further extension is to automata having a finite memory span with respect to the output. Since knowing the present state of an automaton and the output symbol from the automaton is not always sufficient to determine the next state of the automaton if the input symbol is unknown, the memory span of an automaton with respect to the output is defined somewhat differently.

In Section IV, the dependence of the final states of finite automata upon their initial states is studied from another point of view. In this section, we are interested in the existence of one or more input sequences, called synchronizing sequences, that will lead an automaton to the same final state for a set of different initial states. An automaton having a finite memory span with respect to the input which is discussed in Section III is obviously a special case in which all input sequences of length m+1 or more are synchronizing sequences for the set of all states of the automaton (m is the value of the memory span with respect to the input).

In Section V, we studied the dependence of the final state of an automaton upon the input sequence. We found that for a special class of automata, their final states can be determined uniquely when their initial states are known but some of the input symbols are unknown. The possibility of determining the final state of an automaton when some input symbols are unknown depends upon the initial state of the automaton, as well as the number of unknown input symbols. For a given initial state, the unique determination of the final state is possible only when the number of consecutive unknown input symbols does not exceed a certain constant p. The constant p is defined as the memory order

of the initial state with respect to the input and is different, in general, for different states. Also, an automaton having a finite memory span with respect to the input is a special case in which every state has an infinite memory order. The notion of the memory order of a state with respect to the input has also been extended to the notion of the memory order of a state with respect to the input-output sequence and with respect to the output. We investigated the possibility of determining the final state of an automaton when some input symbols are unknown but the subsequent input symbols and/or the corresponding subsequent output symbols are known.

The study of the possibility of determining the final behavior of an automaton when some of its past history is unknown is by no means completed with this report. There are interesting cases about which we still have not been able to reach a general conclusion, for example, the determination of the final state of an automaton when only a portion of the input sequence and some other portion of the output sequence (not the portion corresponding to the known portion of the input sequence) are known. Another investigation may enable us to understand more about these, and it is believed that the techniques developed here will be useful in the study of related topics.

APPENDIX

DEFINITIONS OF MATHEMATICAL TERMS

Term	Definition
Set	A well-defined collection of objects.
Cartesian product of sets	The Cartesian product $A \times B$ of two sets A and B are all ordered pairs in the form of (a,b) where a is in the set A and b is in the set B. If A has m elements and B has n elements, $A \times B$ will have mn ordered pairs.
Relation between two sets A and B	A subset S of the ordered pairs in $A \times B$. For $a \in A$ and $b \in B$, a R b (a is related to b) if and only if $(a,b) \in S$.
Function	A relation between two sets A and B. A is called the domain of the function; B, the range of the function. We also say that a function defines a mapping that maps the elements in its domain into the elements in its range.
Simple Algebra	A system consists of (i) a set A, and (ii) a mapping of $(A \times A)$ into A. (Such a mapping is also called a binary composition in the set A.)
Semigroup	A simple algebra with an associative binary composition. A binary composition, $\circ$, is associative if $(a \circ b) \circ c = a \circ (b \circ c)$.
Monoid	A semigroup with a neutral element. An element e in the set A is called a neutral element for a binary composition $\circ$ in the set A, if $e \circ a = a \circ e = a$ for all $a \in A$.
Monoid Homomorphism	A mapping f from a monoid A into another monoid B so that (i) f maps the neutral element e_A of A into the neutral elements e_B of B. (ii) If a_1, a_2 are elements in A, then $f(a_1 \circ a_2) = f(a_1) \bullet f(a_2)$, where $\circ$ is the binary composition in A and $\bullet$ is the binary composition in B.
Free Monoid	Let Σ be any set. Assume that we are given a monoid F and a mapping ψ of Σ into F. We shall say that F (together with ψ) is a free monoid on the set Σ, if for any mapping ϕ of Σ into any monoid A, there exists a unique monoid homomorphism f of F into A which is such that $f(\psi(s)) = \phi(s)$. (See Fig. 57.)

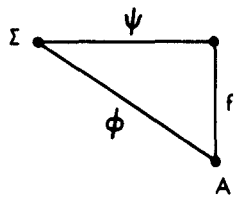


Figure 57

Term	Definition
Equivalence Relation	A relation R in A × A which is such that for all a_1, a_2, a_3 in A (i) a_1 R a_1 (reflexive property) (ii) a_1 R a_2 implies that a_2 R a_1 (symmetric property) (iii) a_1 R a_2 and a_2 R a_3 implies that a_1 R a_3 (transitive property).
Equivalence Class	Let R be an equivalence relation in the set A and a_1 be any element in A. The equivalence class of a_1 under R is the largest subset of elements a_2, a_3 ... in A which are such that a_1 R a_2, a_1 R a_3 ....
Congruence Relation	Let A be a simple algebra. Let • denote its binary composition. An equivalence relation R in the set of elements of A is called a congruence relation if, for any a, x, y in the set of elements of A, (i) x R y implies that a • x R a • y (left invariance) (ii) x R y implies that x • a R y • a (right invariance).
Congruence Class	An equivalence class of elements under a congruence relation over a simple algebra.
Partition of a set	Subdivision of a set into subsets that are disjoint and exhaustive.

Acknowledgment

The author wishes to thank his supervisor, Professor Dean N. Arden, for many helpful discussions and constant encouragement. Thanks are also due to Professor David A. Huffman and Professor Frederick C. Hennie III for their suggestions and criticisms.

References

1. D. A. Huffman, The Synthesis of Sequential Switching Circuits, Technical Report 274, Research Laboratory of Electronics, M.I. T., January 10, 1954.

2. S. C. Kleene, Representation of events in nerve nets and finite automata, Automata Studies, edited by C. E. Shannon and J. McCarthy (Princeton University Press, 1956), pp. 3-41.

3. E. F. Moore, Gedanken-experiments on sequential machines, Automata Studies, edited by C. E. Shannon and J. McCarthy (Princeton University Press, 1956). pp. 129-153.

4. S. Seshu, R. E. Miller, and G. Metze, Transition matrices of sequential machines, IRE Transactions on Circuit Theory, Vol. CT-6, No. 1, pp. 5-12, March 1959.