SYNCHRONIZATION AND DECOMPOSITION OF FINITE AUTOMATA

Thesis for the Degree of Ph. D.

MICHIGAN STATE UNIVERSITY
WILLIAM FREDERICK CUTLIP

1968

This is to certify that the

thesis entitled

Synchronization and Decomposition
of Finite Automata

presented by

William Frederick Cutlip

**has been accepted towards fulfillment
of the requirements for**

Ph.D. **degree in** Mathematics

C. H. Langley, William Kilmer
Major professor

Date September 3, 1968

ABSTRACT

SYNCHRONIZATION AND DECOMPOSITION OF FINITE AUTOMATA

by William Frederick Cutlip

An input sequence x synchronizes a finite automaton A if the state of A after receiving x is independent of the state of A before receiving x. The automaton A is synchronizable if such a sequence x exists. The questions of synchronizability and properties of the set of synchronizing sequences, both for arbitrary and particular classes of automata, motivate much of the present work.

In Chapter 3 (the first two chapters being background material) the homing and distinguishing problems are briefly discussed, with references to some of the related published research. The synchronization problem is then defined and related to the preceding problems, and algebraic properties of the set of synchronizing sequences of any automaton are investigated. It is shown that the sequence x synchronizes the automaton A if and only if the Myhill class of sequences containing x is a right zero of the monoid of A. Several bounds in terms of the number of states of A are established on the length of a shortest synchronizing sequence for A. Sharper bounds are established for particular classes of automata.

Chapter 4 contains most of the major results of the thesis. The set of tapes which synchronize a purely k-definite automaton is characterized. This characterization is shown to carry over, but with a quite different proof, to ultimate-definite automata; and it is shown that every ultimate-definite automaton is synchronizable. Synchronization of reverse ultimate-definite automata is investigated, and a characterization is obtained for the synchronizing sequences of a subclass of such machines.

Also in Chapter 4 we observe that the set of synchronizing sequences of an automaton is an event which is both ultimate-definite and reverse ultimate-definite. We define such an event to be central-definite, show that such an event has a unique canonical form, and

characterize the set of synchronizing sequences of a central-definite automaton in terms of this new canonical form.

In Chapter 5, Zeiger's procedure for decomposing a finite automaton into a cascade of permutation-reset machines is reviewed. Several flaws in the procedure are illustrated by examples and then remedied. It is shown that an automaton A is definite if and only if any Zeiger decomposition of A is a cascade of reset machines. The last part of the chapter is devoted to an examination of ways in which permutation inputs to A are manifested in the covers and machines associated with any Zeiger decomposition of A.

Chapter 7 suggests several lines for further inquiry.

SYNCHRONIZATION AND DECOMPOSITION OF FINITE AUTOMATA

By

William Frederick Cutlip

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirement
for the degree of

DOCTOR OF PHILOSOPHY

Department of Mathematics

1968

ACKNOWLEDGMENTS

Convention does not permit me to list all of the persons to whom I have become indebted during the past four years of study. The mention of only a few must convey my feelings of appreciation for the many.

Professor Charles P. Wells, Chairman of the Department of Mathematics, at several critical junctures made the decisions which permitted me to initiate, modify and complete what must be described as a somewhat non-standard program and thesis. I take great pleasure in acknowledging the contribution which only he could have made.

I wish to express my profound thanks to Professors William L. Kilmer and Patrick H. Doyle, co-chairmen of my guidance committee. The variety and depth of their interests and their outstanding personal qualities are at once an inspiration and a challenge. I am especially indebted to Professor Kilmer, to whom I owe my present acquaintance with and interest in automata theory.

Thanks are also due Professors Richard J. Dubes, Edgar M. Palmer and James Stapleton, who served on my committee and who made helpful comments during the writing of this thesis.

Finally I wish to express my appreciation to Mr. Bob G. Reynolds, who stood ever ready to read my latest proof with a careful eye or puncture a fond but foolish conjecture with a counterexample. Discussions with him helped greatly in shaping some of the more intricate proofs.

To my wife and best friend,

Jean.

TABLE OF CONTENTS

TABLES

FIGURES

1. INTRODUCTION

Little more than a decade has elapsed since the appearance of most of the seminal papers in what is now called automata theory. Beginning in an attempt to formalize the study of the structure and function of neural systems [6,11], automata theory has evolved as a discipline in its own right, replete with the structure and intrinsic fascination so characteristic of other areas of mathematics.

Links between automata theory and the theory of semigroups have been discovered and exploited, with this exploitation taking place in both directions. The elegant decomposition theory for finite automata set forth by Krohn and Rhodes [7] and further developed by Zeiger [20] has its counterpart in semigroup-theoretic results due to Frobenius; and a continuing stream of papers [8,9,17,18] indicates that the study of automata has suggested new lines of inquiry with respect to semigroups. In the American Mathematical Monthly, Hartmanis and Stearns [5] investigated some topological properties of sets of real numbers defined by finite automata; and Rabin [15] constructed one of his more elegant proofs by recasting and then solving the problem in terms of simplices.

Particular subclasses of automata have been examined, a situation not without its parallel in, for example, classical group theory. Among these classes are definite automata, introduced by Kleene [6] and studied intensively by Perles, Rabin and Shamir [14]; and ultimate-definite, reverse ultimate-definite and symmetric definite automata, introduced as a logical extension of the work of the latter authors by Paz and Peleg [13].

Much of the present work was motivated by the problem of driving a finite automaton from an arbitrary unknown state to a known state by means of a predetermined input sequence, without regard to the corresponding output sequence. This notion is made precise in Chapter 3 with the definition of synchronization of a finite automaton, Chapter 2 being devoted to laying the necessary conceptual and notational foundations. Also in Chapter 3 the algebraic properties of such synchronizing sequences are explored, and bounds are obtained for the minimal length of synchronizing sequences for both arbitrary and particular classes of automata.

We examine the definite automata of Perles, Rabin and Shamir [14] in Chapter 4, and obtain a characterization for the set of sequences which synchronize a reduced, purely k-definite automaton. Looking at the more general ultimate-definite automata of Paz and Peleg [13] we discover that the characterization of synchronizing sequences for a purely k-definite automaton carries over, but with a quite different proof, to the ultimate-definite case. We also treat synchronization of reverse ultimate-definite automata, and close the chapter with the definition of and some results concerning central-definite events and automata.

The central concern of Chapter 5 is the decomposition theory of finite automata as treated by Zeiger [20,21]. Some general results are obtained, in addition to a characterization of cascade decompositions of k-definite prefix automata. The section also illustrates and corrects errors which appeared in Zeiger's most recent treatment of this subject [21].

The original results of the present work are confined to Chapters 3, 4 and 5. Results due to other authors are clearly indicated wherever cited.

2. FUNDAMENTAL CONCEPTS

2.1. Events and Regular Expressions

Most of the material of this and the following section was stated or is implicit in Kleene [6] and Rabin and Scott [16], and is included so that the present work may be essentially self-contained. The development follows Rabin and Scott for the most part.

Definition 2.1.1: Given a finite nonempty alphabet or set of symbols $\Sigma = \{\sigma_1, \ldots \sigma_r\}$ let Σ^* denote the set of all finite, but not necessarily nonempty, sequences of symbols from Σ. Let Λ denote the empty sequence.

Each finite sequence in Σ^* is also called a tape, with Λ denoting the empty tape. The lower case letters u, v, w, x, y, z, with or without subscripts, will be used to denote tapes; for example, $x = \sigma_1\sigma_2\sigma_7\sigma_1, \Lambda$, and $y_1 = \sigma_2\sigma_5$ are elements of Σ^*.

Definition 2.1.2: An event over the alphabet Σ is any subset of Σ^*.

The upper case letters R, U, V, W, X, with or without subscripts, will be used to denote events; for example, if $\Sigma = \{0, 1\}$, then $U = \{010, 111\}$ and $V_2 = \{1, 11, 111, 1111, \ldots\}$ are respectively a finite and an infinite event over Σ.

Definition 2.1.3: The operations $\cdot$, $\cup$, and $*$ are defined as follows: If x and y are tapes, then $x \cdot y$ is the tape obtained by concatenating x and y. xy denotes $x \cdot y$. If U and V are events, then $U \cdot V = UV = \{uv : u \in U \text{ and } v \in V\}$.

If U and V are events, $U \cup V$ is the set-theoretic union of U and V.

For any event U, U^* is the set containing Λ, the tapes in U, and all tapes which may be obtained by concatenation of finitely many tapes in U.

To avoid tedious nesting of grouping symbols, various notational liberties are commonly taken with the above definition. For example, $101 \cup 110$ may be written instead of $\{101\} \cup \{110\} = \{101, 110\}$; 10^*1 denotes $\{1\} \cdot \{0\}^* \cdot 1$, and both mean "the set of all finite sequences of 0's and 1's which begin and end with 1, all intervening symbols (if any) being 0's."

Definition 2.1.4: If U is an event, then U^n, where n is a non-negative integer, is defined recursively by:

$$U^0 = \{\Lambda\} = \Lambda \text{ (the latter by notational convention);}$$

$$U^n = U \cdot U^{n-1} \text{ for each positive integer } n.$$

In the class of all events in Σ^*, a certain subcollection has been the object of considerable interest. The events of this class are called regular events, and were first characterized by Kleene [6]. Of the several characterizations which have since appeared, the following is most useful for our purposes.

Definition 2.1.5: A regular expression over the alphabet Σ is any expression which can be written using only Λ, the symbols in Σ, and finitely many applications of $\cup$, $\cdot$ and $*$ in accordance with Definition 2.1.3 and the notational conventions described thereafter. A regular event is any event (set of tapes) which can be denoted by a regular expression. Two expressions are equivalent if they denote the same event.

Example 2.1.1:

(a) The event U described by $1(00)^*11$ is regular according to Definition 2.1.5. $U = 1(00)^*11 = \{111, 10011, 1000011, \ldots\}$.

(b) Although the expression $\{0, 11, 010\} \cup \{111\}$ is not regular, an equivalent expression is $0 \cup 11 \cup 010 \cup 111$, which is regular. Thus both expressions describe a regular event.

(c) The event $V = \{1^p : p \text{ is a positive prime integer}\}$ is not regular. An attempt to write a regular expression for V results in something of the form $1 \cup 1^2 \cup 1^3 \cup 1^5 \cup \ldots$, and it is the "..." which prevents this expression from being regular according to Definition 2.1.5.

2.2. Finite Automata

Speaking intuitively, one might view an automaton as a device A having finitely many internal configurations or states (positions of internal switches, for example), operating in discrete time steps, and capable of receiving any one of a finite collection Σ of stimuli at a given time, such that

(1) A is in only one state at a given time t,

(2) the state of A at time t+1 is completely determined by the state of A at time t and the stimulus received at time t, and

(3) an indicator tells at each time step whether the state of A at that time is final or non-final.

These concepts are made precise in the following definition.

Definition 2.2.1: A finite automaton over the alphabet Σ is a system $A = (S, M, s_0, F)$ where S is a finite nonempty set (the set of internal states of the automaton), $s_0 \in S$ is the fixed starting state, $F \subseteq S$ is the set of final ("yes") states, and $M : S \times \Sigma \to S$ is the transition function of A. For each $s \in S$, let $M(s, \Lambda) = s$. Automaton, machine, and device will be used interchangeably with finite automaton.

There are several schemes for representing a finite automaton A, each having its particular advantages. These will be introduced by example rather than by formal definition; and the fact that they all represent the same automaton will be asserted without proof.

Example 2.2.1: Let $S = \{s_0, s_1, s_2\}$. Let $\Sigma = \{0, 1\}$. Let $F = \{s_2\}$.

(a) M may be specified in a transition table.

Table 2.2.1. Transition table of $A = (S, M, s_0, F)$

M	0	1
s_0	s_1	s_2
s_1	s_1	s_1
s_2	s_2	s_1

(b) M may be specified by transition matrices.

For each symbol σ in alphabet Σ, construct a corresponding matrix $M(\sigma)$ of 0's and 1's, with a 1 in location (i, j) if and only if $M(s_{i-1}, \sigma) = s_{j-1}$. The transition matrix specification of the same function M specified in tabular form in part (a) is as follows:

$$M(0) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad M(1) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

(c) M may be specified by state diagram.

Construct a directed graph with one vertex corresponding to each state, and a directed line or arrow emanating from each vertex for each alphabet symbol. Let the arrow from vertex s_i corresponding to symbol σ terminate at vertex s_j if and only if $M(s_i, \sigma) = s_j$. One says in such a case that σ takes automaton A from state s_i to state s_j, or that σ takes s_i to s_j. The state diagram specifying the same function M as in parts (a) and (b) is the following:

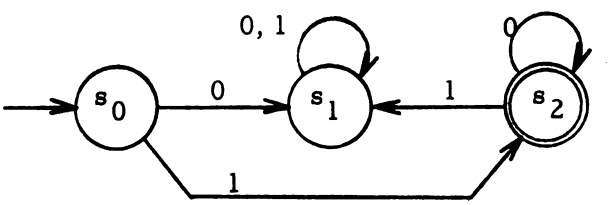

Figure 2.2.1. State diagram of $A = (S, M, s_0, F)$

Note that the start state is distinguished with a single arrow entering it from no preceding state, and each state in F is distinguished with two concentric circles.

The function M may be extended in obvious fashion to map $S \times \Sigma^*$ into S as follows.

Definition 2.2.2: For any tape $x \in \Sigma^*$, any symbol σ in Σ, and any state $s \in S$, $M(s, x\sigma) = M(M(s, x), \sigma)$.

Example 2.2.2: We compute $M(s_0, 101)$ using each of the specification schemes of Example 2.2.1.

(a) Using Definition 2.2.2 and the transition table for M, we obtain

$$M(s_0, 101) = M(M(s_0, 10)\ 1) = M(M(M(s_0, 1), 0), 1)$$
$$= M(M(s_2, 0), 1) = M(s_2, 1) = s_1.$$

(b) In terms of symbol matrices, if we let $M(101) = M(1)\ M(0)\ M(1)$, we obtain

$$M(101) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

It is easily shown that the matrix M(x) obtained in this fashion for any x in Σ^* has a 1 in location (i, j) if and only if $M(s_{i-1}, x) = s_{j-1}$. Thus, the 1 in location (1, 2) of M(101) reveals that $M(s_0, 101) = s_1$.

(c) Finally, the path obtained by beginning at vertex s_0 in the state diagram of A and following in turn arrows labelled 1, 0, and 1 terminates in vertex s_1.

Thus, all three representations of A show that $M(s_0, 101) = s_1$. The state behavior of A (i.e., the sequence of states through which A passes starting from any particular state and in response to any input sequence x from Σ^*) is completely specified by each of the three representations.

So far, no particular significance has been attached to the subset F of the state set S.

Definition 2.2.3: For each tape x in Σ^*, for each finite automaton $A = (S, M, s_0, F)$, A accepts x if $M(s_0, x)$ is in F. T(A) denotes the set of all tapes accepted by automaton A. If U is the set of tapes accepted by automaton A, we write A = A(U) to denote that A is an automaton whose set of accepted tapes is U.

It is apparent from the state diagram that a tape x is accepted by the automaton of Example 2.2.1 if and only if x consists of a single one or a single one followed by finitely many 0's. Symbolically, $x \in T(A)$ if and only if $x \in 10^*$. Thus, T(A) is described by the regular expression 10^*. This is no mere happenstance, for Kleene [6] established that an event U is regular if and only if it is precisely T(A) for some finite automaton A.

Thus, there is a one-to-one correspondence between the set of distinct events accepted by finite automata and the set of non-equivalent regular expressions. McNaughton and Yamada [12] contributed a procedure for obtaining a state diagram for an automaton which accepts the event defined by a given regular expression, and for writing down a regular expression corresponding to a given state diagram. In neither direction is this correspondence unique. Once an automaton corresponding to a given regular expression has been obtained, a unique reduced automaton, described in Definitions 2.2.4

through 2. 2. 6, can be obtained. However, there is at present no technique for transforming a regular expression corresponding to a given automaton into some unique canonical form. The regular expression notation adapted in McNaughton and Yamada [12] is used frequently in the present work.

One additional notion regarding arbitrary finite automata will be useful. Unless we specify otherwise, we consider only automata with no superfluous states, i. e., with no states which are never entered under any input sequence, or which duplicate exactly the behavior of other states. More precisely:

Definition 2. 2. 4: Given the finite automaton $A = (S, M, s_0, F)$, the state $s \in S$ is said to be accessible if there is a tape x such that $M(s_0, x) = s$.

Definition 2. 2. 5: States s_i and s_j of finite automaton $A = (S, M, s_0, F)$ are equivalent if, for any tape x, $M(s_i, x) \in F$ if and only if $M(s_j, x) \in F$.

The motivation for this last definition becomes clearer if we think of A as being able to tell an observer only whether or not it is in a final state, and not what particular state is occupied. For example, the automaton may emit a "1" if in a final state, a "0" if in a non-final state. If s_i and s_j are to be distinguishable to an observer under this restriction, then there must be some tape x which takes s_i to a state in F and s_j to a state not in F, or vice-versa. Such a tape x, if it exists, is said to distinguish between s_i and s_j; and if no such tape exists, then s_i and s_j are indistinguishable or equivalent.

There is a straightforward method for merging a set of equivalent states of a finite automaton into one state; and one may simply delete all inaccesssible states from the automaton. The net result is an automaton which accepts the same set of tapes as the original machine and uses a minimal number of states in order to perform this tape-sorting function.

Definition 2. 2. 6: The finite automaton $A = (S, M, s_0, F)$ is said to be reduced or minimum-state if all states of A are accessible and no two states of A are equivalent.

2.3. Some Elements of the Algebraic Theory of Automata

With each tape $x \in \Sigma^*$ we may associate a mapping x_A of the state set S of automaton $A = (S, M, s_0, F)$ back into S, according to the rule

$$x_A(s) = M(s, x) \qquad \text{for each } s \in S.$$

Since S is finite, there are finitely many such mappings for a given automaton; and we view as equivalent those tapes which induce the same mapping of S into S.

Definition 2.3.1: If x and y are tapes, then we say that x and y are equivalent and we write $x \equiv y$ if for each $s \in S$, $M(s, x) = M(s, y)$. It is clear that $\equiv$ is an equivalence relation. Let $[x]_{\equiv}$ denote the equivalence class containing $x \in \Sigma^*$.

N. B. The relation $\equiv$ is due to J. R. Myhill. (See [16] Theorem 1 and related discussion.) We shall occasionally refer to it as "Myhill equivalence."

The relation $\equiv$ on Σ^* is always defined with reference to a particular automaton, and varies from one automaton to another. For a given automaton, the Myhill classes have the following well-known property.

Lemma 2.3.1: Given $A = (S, M, s_0, F)$, the equivalence classes $[x]_{\equiv}$ form a monoid under the operation $[x]_{\equiv}\ [y]_{\equiv} = [xy]_{\equiv}$.

Proof: If $x_1 \equiv x$ and $y_1 \equiv y$, then for each $s \in S$, $M(s, xy)$ $= M(M(s, x), y) = M(M(s, x_1), y_1) = M(s, x_1 y_1)$. Hence $xy \equiv x_1 y_1$; so the operation is well-defined. Closure and associativity are obvious, and $[\Lambda]_{\equiv}$ is the identity element.

Definition 2.3.2: Given $A = (S, M\ s_0, F)$, let $\mathcal{M}(A)$ denote the monoid of A, generated in accordance with the preceding definition and lemma, i.e., $\mathcal{M}(A) = (\{ [x]_{\equiv} \}, \cdot)$.

We shall occasionally write "[x]" instead of "$[x]_{\equiv}$" where no misunderstanding is likely.

2.4. Definite Events and Definite Automata

The definitions and notation of this section follow Perles, Rabin and Shamir [14] with a few modifications.

Definition 2.4.1: For any tape x, the length of x, denoted by $\ell(x)$, is the number of symbols from Σ which were concatenated in forming x. Put another way, $\ell(x) = n$ if and only if $x \in \Sigma^n$ (n a non-negative integer). A tape of length n is an n-tape.

Definition 2.4.2: If x and y are tapes, x is a subtape of y if there exist tapes u and v such that y = uxv. Tapes u and v are not necessarily nonempty. The tape x is a suffix of the tape y if there is a tape u such that y = ux, and is a proper suffix of y if $u \neq \Lambda$. If k is a non-negative integer, x is a k-suffix of y if x is a suffix of y and $\ell(x) = k$. The conditions under which x is a prefix, proper prefix, and k-prefix of y are analogous to the corresponding suffix concepts.

Definition 2.4.3: For any event U and for each non-negative integer k, U is weakly k-definite if for each tape x such that $\ell(x) \geq k$, x is in U if and only if the k-suffix of x is in U.

That is, if $\ell(x) \geq k$, it can be decided whether or not x is in U by examining the last k symbols of x. Note that U being weakly k-definite does not preclude the possibility of U containing tapes of length less than k.

Definition 2.4.4: For each non-negative integer k, for each event U, U is k-definite if U is weakly k-definite but not weakly (k-1)-definite.

Example 2.4.1: If $\Sigma = \{0, 1\}$ and if U is the set of all tapes which end in 01, 101, or 001 (i.e., $U = \Sigma^*\{101, 01, 001\}$), then U is weakly 3-definite, since examination of the last 3 symbols of any tape x whose length exceeds 3 will reveal whether or not x belongs to U. Indeed, x of length exceeding 2 is in U if x ends in 01, whatever the symbol preceding the final 2-suffix might be. Therefore, U is weakly 2-definite.

However, U is not weakly 1-definite, for the tape 0101 is in U but its 1-suffix is not in U. Since U is weakly 2-definite but not weakly 1-definite, U is a 2-definite event.

Definition 2.4.5: For each event U, U is definite if U is k-definite for some non-negative integer k.

We observe that if U is weakly k-definite, then U is weakly m-definite for any integer $m > k$, and U is j-definite for some non-negative integer $j \leq k$.

A k-definite event U may be described by listing both the "short" tapes of length less than k which are in U and the tapes of length k which are in U; for, any tape x whose length exceeds k is in U if and only if its k-suffix is in U. Thus

$$\text{if } V = \bigcup_{i=0}^{k-1} (U \cap \Sigma^i) \text{ and } W = U \cap \Sigma^k, \text{ then } U = V \cup \Sigma^* W.$$

U is evidently regular since V and W are finite. Indeed, if V and W are finite, U is definite, independent of whether $W \subseteq \Sigma^k$ for some positive integer k.

One well-known procedure for constructing an automaton A such that $T(A) = U = V \cup \Sigma^* W$ involves constructing two automata, one accepting V and the other accepting $\Sigma^* W$. The former construction is straightforward but varied, dependent for its particular form on V. The latter construction may be described in quite general terms, and leads to several results of the next chapter. We therefore center our attention for the time being on events of the form $\Sigma^* W$, where $W \subseteq \Sigma^k$.

Definition 2.4.6: The event U is purely k-definite if $U = \Sigma^* W$ for some $W \subseteq \Sigma^k$.

Note that if U is purely k-definite, then U is k-definite.

Definition 2.4.7: A finite automaton A is weakly k-definite (k-definite) [purely k-definite] if T(A) is weakly k-definite (k-definite) [purely k-definite].

Perles, Rabin and Shamir [14] showed that the state of a reduced k-definite automaton is uniquely determined by the k most recent input symbols received, and that an automaton whose state is uniquely specified by the last k inputs is m-definite for some positive integer $m \leq k$.

2.5. Prefix Automata

Perles, Rabin and Shamir [14] detailed the construction, used in obtaining one of our results, of an automaton which accepts an event of the form $\Sigma^* W$ ($W \subseteq \Sigma^k$). The resulting prefix automaton (so called because of the method of construction, not because it accepts tapes on the basis of their prefixes, which is not the case) will be denoted by $P(\Sigma^* W)$. The construction is as follows.

Given $W \subseteq \Sigma^k$, let $S = \{[\Lambda]\} \cup \{[s] : s \text{ is a prefix of some } y \in W\}$ be the set of states of P, where Λ is the null tape and $[\Lambda]$ denotes s_0. Let $M : S \times \Sigma^* \to S$ be defined by $M([s], x) = [u]$ if and only if u is the longest suffix of sx such that $[u] \in S$. Let $F = \{[w] \in S : w \in W\}$ be the set of designated final states of P, i.e., given $x \in \Sigma^*$, $x \in T(A)$ if and only if $M([\Lambda], x) \in F$.

Example 2.5.1: Let $U = \Sigma^* W$, where $W = \{100, 010\}$. Then $S = \{[\Lambda], [0], [1], [01], [10], [010], [100]\}$, $F = \{[010], [100]\}$, and the mapping M is described in tabular form by

Table 2.5.1. Next-state function of $A(\Sigma^* W)$

M		0	1
s_0 =	[Λ]	[0]	[1]
s_1 =	[0]	[0]	[01]
s_2 =	[1]	[10]	[1]
s_3 =	[01]	[010]	[1]
s_4 =	[10]	[100]	[01]
s_5 =	[010]	[100]	[01]
s_6 =	[100]	[0]	[01]

The state diagram of P is shown in Figure 2.5.1.

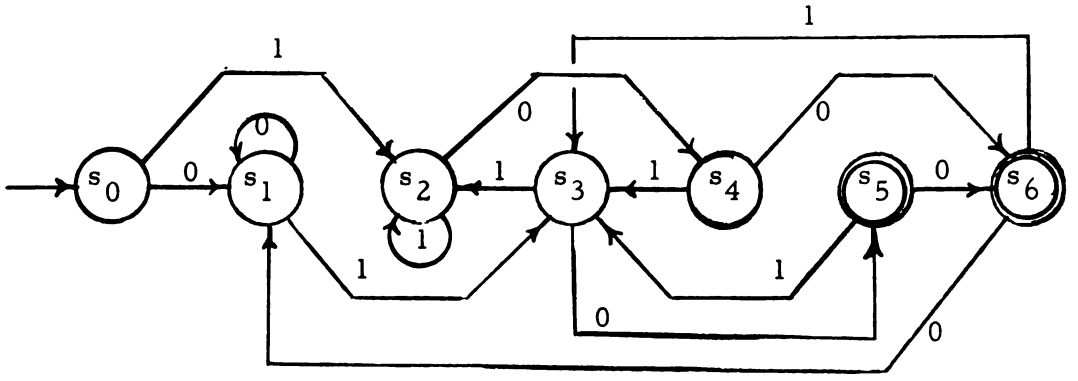

Figure 2.5.1. State diagram of $P(\Sigma^* W)$

2.6. Output Behavior and State Behavior

There are two common formulations which associate an output symbol with each input symbol received by an automaton A. The Mealy model of a finite automaton associates, with each state and input symbol, one of a finite collection of output symbols. One may think of the output symbol as being emitted by A during the transition from a given state to the next state. Such a device is depicted in Figure 2.6.1. (a). Each transition arrow is labelled with an ordered pair (i, j), where i denotes the input causing the transition and j is the emitted output symbol.

The Moore model of a finite automaton A associates an output symbol with each state of A, the output symbol corresponding to state s being emitted whenever A goes into state s. Such an automaton is depicted in Figure 2.6.1. (b). Each state is labelled with an ordered pair (s, j), where s names the state and j the output symbol. It is readily seen that, by associating an output symbol 1 with each state in F and a 0 with each state in S-F, the machine $A = (S, M, s_0, F)$ is a Moore machine and tape x is accepted by A if and only if, when A is started in s_0 and receives x, the final output symbol is a 1.

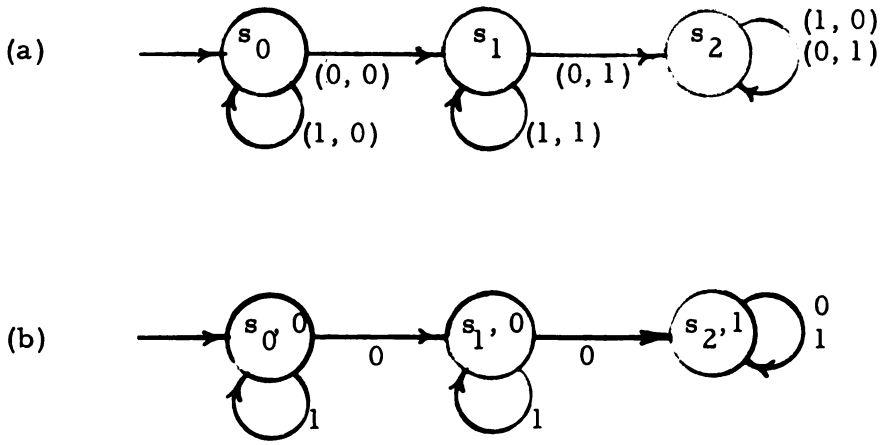

Figure 2.6.1. (a) A Mealy machine, and
(b) a Moore machine

In either case, the output behavior of A may be described by listing, for each state and input symbol, the corresponding output symbol. We may, however, be interested only in the way S is mapped into S under various inputs, without regard to tape-acceptance or output behavior of A. In particular the next chapter concentrates on state transitions (state behavior), ignoring outputs.

3. THE SYNCHRONIZATION PROBLEM: GENERAL RESULTS

3.1. Background and Related Problems

Our use of the terms finite automaton, automaton and machine as synonyms bespeaks the physical world origins which automata theory shares with much of classical mathematics. The tone of the preceding chapter was mathematical, notwithstanding the occasional use of "machine," for we spoke there in terms of tables, graphs, functions, sequences and the like.

If, for a moment we sharpen our focus on the machine-like aspects of automata, we must admit the possibility of error or malfunction of the devices with which we are dealing. If our principle concern is with state transition behavior, we see that errors may occur either due to a faulty state transition following correct reception of an input symbol, or to incorrect reception of the input symbol. Continuing the machine analogy, the latter may occur when a burst of noise on the input line causes signal σ_i to be interpreted as signal σ_j $(i \neq j)$.

Whatever the cause of our ignorance of the state of the automaton, an obvious task confronts us: to identify the present state or some future state of the automaton. The former is called the distinguishing problem, the latter the homing problem.

The solutions of the distinguishing and homing problems might be described as follows. Given a finite automaton of known structure but in an unknown state, the observer supplies an input sequence. This procedure is called an experiment. At the conclusion of the experiment, the observer may be able to announce either the state of the automaton at the beginning of the experiment, solving thereby the distinguishing problem, or the state of the automaton at the conclusion of the experiment, solving the homing problem.

If the input sequence supplied by the observer is completely determined before the experiment begins, the experiment is said to be preset; if at some time during the experiment the inputs and corresponding outputs which occurred earlier in the experiment are examined in order to determine the next input symbol, the experiment is said to be adaptive. The experiment is further described as being

simple or multiple according to whether it is carried out using one or more than one copy of the automaton in question.

Gill [2] discusses the solution of the distinguishing and homing problems by means of preset or adaptive, simple or multiple experiments. Ginsburg [3] and Even [1] also attack the homing problem. All three authors deal with experiments in which, following each input symbol, the corresponding output symbol is observed, with all or part of the input sequence and the corresponding output sequence being used to identify the state of the automaton at the start or conclusion of the experiment.

3.2. The Synchronization Problem; Algebraic Properties of Synchronizing Sequences

Our attention shall be confined to automata which undergo correct state transitions. We shall be dealing with a restricted version of the homing problem, in which it is our task to identify the state of the automaton at the end of a preset, simple experiment without either prior information or consideration of the corresponding output sequence. In effect, we suppress outputs and deal only with state machines.

Throughout the following, let $A = (S, M, s_0, F)$ be a finite automaton over the input alphabet Σ.

Definition 3.2.1: The sequence (tape) x is a synchronizing sequence (tape) for A if $M(S, x)$ is a singleton subset of S. Such a tape x is said to synchronize A. A is synchronizable if there is a tape x which synchronizes A.

Put another way, a synchronizing tape x will drive A from an arbitrary unknown state to a unique known state. Several equivalent formulations are immediate and are stated without proof in the following result.

Lemma 3.2.1: For any finite automaton $A = (S, M, s_0, F)$ and any tape $x \in \Sigma^*$, the following statements are equivalent:

(1) x synchronizes A.

(2) $M(s, x) = M(s_0, x)$ for all $s \in S$.

(3) $M(s, x) = M(t, x)$ for all s and t in S.

(4) M(T. x) is a singleton for any $T \subseteq S$.

The automata represented in Figure 3. 2. 1 show that some, but not all, finite automata are synchronizable. A synchronizable automaton has infinitely many synchronizing sequences; for, if x synchronizes A, so does wxy for any tapes w and y in Σ^*. Several questions are thus suggested: What are the properties of the set of all synchronizing tapes for a given automaton? How can one tell whether or not a given automaton is synchronizable? We first deal with the former question.

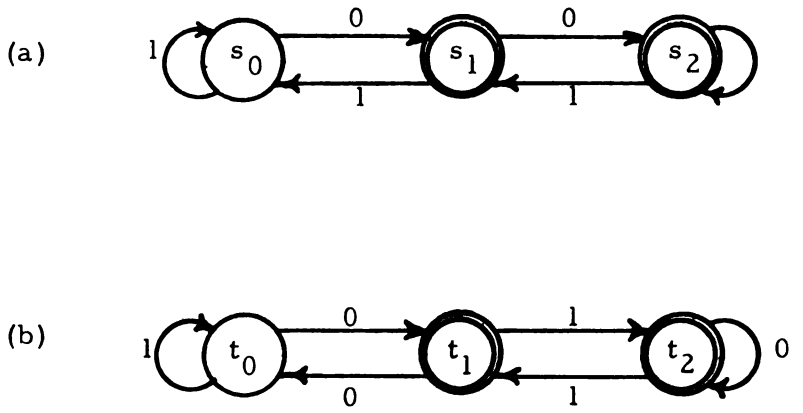

Figure 3. 2. 1. Automaton (a) is synchronizable, e. g. by 00. Automaton (b) is not synchronizable, since both 0 and 1 permute the state set.

Two proofs are supplied for the following result because each introduces notions which will be useful in subsequent work. The first is analytic, the second is constructive.

Theorem 3. 2. 2: The set of all synchronizing sequences for $A = (S, M, s_0, F)$ is a regular set.

Proof 1: Let R denote the set of all tapes which synchronize A. For each s_i and s_j in $S = \{s_0, \ldots, s_{n-1}\}$ we define a new automaton $A_{ij} = (S, M, s_i, \{s_j\})$ having the same transitions but perhaps not the same start and final state as A. Then $T(A_{ij}) = \{x \in \Sigma^* : M(s_i, x) = s_j\}$ is the event accepted by A_{ij}.

For fixed j, $R_j = \bigcap_{i=0}^{n-1} T(A_{ij})$ is the set of all tapes which take every $s \in S$ to s_j, i.e., is the set of all tapes which synchronize A to s_j. Since R_j is a finite intersection of regular sets, R_j is regular.

Thus $R = \bigcup_{j=0}^{n-1} R_j$, showing that the set of all tapes which synchronize A is the union of finitely many regular sets. This completes the proof that R is regular.

We could also show R to be regular by constructing a finite automaton whose set of accepted tapes is R. The construction of one such automaton is straightforward, emulating techniques used by Hartmanis and Stearns [19] and may be considered an alternate proof of the preceding theorem.

Proof 2 (of Theorem 3.2.2): Given $A = (S, M, s_0, F)$, let $S' = \{S_i : S_i \subseteq S\}$, $s_0' = S$, $M'(S_i, x) = M(S_i, x)$ for each tape x and each $S_i \in S'$, and $F = \{\{s\} : s \in S\}$. Let $A' = (S', M', s_0', F')$. The following statements are then equivalent.

(1) The tape x synchronizes A.

(2) For some $t \in S$, $M(S, x) = \{t\}$.

(3) The tape x is accepted by A'.

Hence the tapes accepted by A' are precisely the synchronizing tapes of A. By a previously cited result of Kleene [6], T(A') is regular and therefore R is regular.

For future use, we formalize the notation introduced in Proof 1.

Definition 3.2.2: If $A = (S, M, s_0, F)$ and $S = \{s_0, \ldots, s_{n-1}\}$, let $\underline{R} = \{x \in \Sigma^* : x \text{ synchronizes } A\} = \{x \in \Sigma^* : M(S, x) \text{ is a singleton}\}$. For each i $(0 \le i \le n-1)$ let $R_i = \{x \in \Sigma^* : M(S, x) = s_i\}$. That is, $\underline{R}_i$ is the set of all tapes which synchronize A to state s_i.

The synchronizing sequences for a particular automaton A are reflected in the monoid $\mathfrak{M}(A)$, as is brought out in the next two theorems.

Theorem 3.2.3: $R_i = \{x \in \Sigma^* : M(S, x) = s_i\}$ is a Myhill equivalence class (see Definition 2.3.1) of tapes.

Proof: For each $s \in S$ and any tapes x and y in R_i, $M(s, x) = s_i = M(s, y)$. By Definition 2.3.1, $x \equiv y$. Furthermore, if $z \in \Sigma^*$ and $z \equiv x$, then for every $s \in S$, $M(s, z) = M(s, x) = s_i$; hence $z \in R_i$.

Corollary to Theorem 3.2.3: $R_i \in \mathcal{M}(A)$.

Definition 3.2.3: The element b of monoid $\mathcal{M}$ is a right zero of if, for each $m \in \mathcal{M}$, $mb = b$.

Theorem 3.2.4: The tape x synchronizes automaton A if and only if $[x]_{\equiv} = [x]$ is a right zero of $\mathcal{M}(A)$.

Proof: If x synchronizes A then there is a state $s_i \in S$ such that $M(s, x) = s_i$ for any $s \in S$. Hence for any $y \in \Sigma^*$ and each $s \in S$, $M(s, yx) = M(M(s, y), x) = s_i$; so $yx \in [x] = R_i$. Thus for any $[y] \in \mathcal{M}(A)$, $[y][x] = [yx] = [x]$, and $[x]$ is a right zero of $\mathcal{M}(A)$.

Conversely, let $[x]$ be a right zero of $\mathcal{M}(A)$. Let $s_i \in S$ be an arbitrary state of A. Since A is reduced, there is a tape x_i such that $M(s_0, x_i) = s_i$. Because $[x]$ is a right zero of $\mathcal{M}(A)$, $[x_i][x] = [x]$. Hence $M(s_i, x) = M(M(s_0, x_i), x) = M(s_0, x_i x) = M(s_0, x)$. Since s_i was arbitrary, x synchronizes A.

3.3. A Test for Synchronizability; Bounds on the Length of a Shortest Synchronizing Tape

Liu [10] also considered the synchronization problem and presented the next three results with proofs resembling those shown here.

Definition 3.3.1: For any finite set B, let $|B|$ denote the cardinal number of, or number of elements in, B.

Definition 3.3.2: If $T \subseteq S$, the tape x synchronizes T if M(T, x) is a singleton. If s and t are states, x merges s and t if x synchronizes $\{s, t\}$.

Lemma 3.3.1: The finite automaton A is synchronizable if and only if for each pair of distinct states s and t in S there is a tape x such that $M(s, x) = M(t, x)$.

Proof: If A is synchronizable, there is a tape x such that $M(S, x) = M(s_0, x)$. For any pair of states s and t, then, $M(s, x) = M(s_0, x) = M(t, x)$.

Conversely, if for each pair of distinct states s and t there is a tape x such that $M(s, x) = M(t, x)$, one may proceed as follows. (Recall $S = \{s_0, \ldots, s_{n-1}\}$.)

Select x_1 so that $M(s_1, x_1) = M(t_1, x_1)$ for distinct states s_1 and t_1 in S. Let $M(S, x_1) = S_2$. $|S_2| \leq n-1$, since $|S| = n$ and x_1 merges s_1 and t_1.

If $|S_2| > 1$, select x_2 to merge distinct states s_2 and t_2 of S_2. Let $M(S_2, x_2) = S_3$. $|S_3| \leq n-2$, since x_2 merged s_2 and t_2.

Proceeding in this fashion, since $|S| = n$ there is an integer $k \leq n-1$ such that $M(S_k, x_k) = S_{k+1}$ and $|S_{k+1}| = 1$. Thus $M(S, x_1 x_2 \ldots x_k)$ is a singleton, and $x_1 \ldots x_k$ synchronizes A.

Lemma 3.3.2: If $A = (S, M, s_0, F)$ and $|S| = n$, and if the distinct states s_1 and t_1 can be merged, they can be merged by some tape x such that $\ell(x) \leq \binom{n}{2}$. ("$\binom{n}{2}$" denotes "the number of combinations of n things taken two at a time.")

Proof: Let $y = \sigma_1 \sigma_2 \ldots \sigma_r$ merge s_1 and t_1. Since $s_1 \neq t_1$ and $M(s_1, y) = M(t_1, y)$, there is a least integer j, $0 < j \leq r$, such that $M(s_1, \sigma_1 \ldots \sigma_j) = M(t_1, \sigma_1 \ldots \sigma_j)$. Let $y' = \sigma_1 \ldots \sigma_j$.

The symbol-by-symbol effect of y' on states s_1 and t_1 may be depicted as follows:

$$\{s_1, t_1\} \xrightarrow{\sigma_1} \{s_2, t_2\} \xrightarrow{\sigma_2} \cdots \rightarrow \{s_j, t_j\} \xrightarrow{\sigma_j} \{s_{j+1}\},$$

where $s_i \neq t_i$ for $1 \leq i \leq j$.

If any pair $\{s_p, t_p\}$ is identical to another (unordered) pair $\{s_q, t_q\}$ for $1 \leq p < q \leq j$, then the segment of tape $\sigma_p \ldots \sigma_{q-1}$ may be removed from x, since $M(\{s_p, t_p\}, \sigma_p \ldots \sigma_{q-1}) = \{s_q, t_q\} = \{s_p, t_p\}$.

One may obtain, by performing all such deletions of segments of y', a tape x which sends $\{s_1, t_1\}$ through a sequence of state pairs to $\{s_{j+1}\}$ without repetition of a pair. Since there are $\binom{n}{2} - 1$ pairs of distinct states different from the pair $\{s_1, t_1\}$, $\ell(x) \leq \binom{n}{2}$. The desired bound is thus established.

A more elegant, though less instructive, proof could be constructed using the automaton A' of Proof 2, Theorem 3.2.2. Note

that A' has $\binom{n}{2}$ states corresponding to two-element subsets of S, and that, for each input symbol σ and state S_i of A', $|M(S_i, \sigma)| \leq |S_i|$. Thus state $\{s_1, t_1\}$ of A' can pass through at most $\binom{n}{2}$ -1 other states without repetition before reaching a state of A' corresponding to a singleton subset of S.

The next theorem, also due to Liu, is the first of several results which bound the length of a shortest synchronizing sequence for a given synchronizable automaton.

Theorem 3.3.3: If A (S, M, s_0, F) is synchronizable and $|S| = n$, then A has a synchronizing tape whose length does not exceed $(n-1)\binom{n}{2}$.

Proof: By Theorem 3.3.2, the tapes $x_1, x_2, \ldots, x_k$ of the proof of Theorem 3.3.1 can be chosen so that $\ell(x_i) \leq \binom{n}{2}$, $1 \leq i \leq k$. Since $k \leq n-1$, x can be chosen so that $\ell(x) = \ell(x_1 x_2 \ldots x_k) = \ell(x_1) + \ldots + \ell(x_k) \leq (n-1)\binom{n}{2}$.

Definition 3.3.3: Let Liu's bound = $L(n) = (n-1)\binom{n}{2}$.

In the establishment of this and other bounds on the length of the shortest synchronizing sequence of A, several notions recur. We might view the synchronization of an automaton A as starting with total ignorance of the present state of A, presenting to A a preset sequence, and then being able to announce the exact state of A. Thus we pass from a situation where the state of A may be any element of S to one in which the state of A is isolated in a singleton subset of S, by successively reducing the size of the subsets of S which might contain the present state of A.

Since A proceeds from state to state in discrete time steps, we can at any time t during the presentation of such an input sequence decide for each state $s \in S$, on the basis of input symbols received up to time t, whether or not s may be the state of A at time t. This partitions S into two subsets; the set of all states which A may be in at time t, and the set of states A cannot be in at time t. We refer to the former subset of S as the occupied set at time t; the elements thereof are occupied states. Note that at any time the occupied set is the smallest subset of S known to contain the state of A at that time.

For the sake of brevity, let "O" be an abbreviation for the phrase "the occupied set." Note that O does not denote a fixed subset of S, but rather a set which may change as A receives inputs. The following example illustrates the concepts just described.

Example 3.3.1: If an observer is presented with the automaton A of Figure 3.3.1 and has no information concerning its present state, then $O = S = \{s_0, s_1, s_2, s_3\}$.

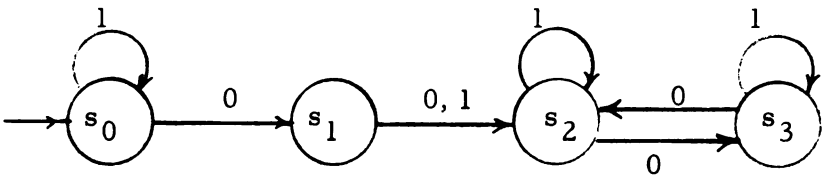

Figure 3.3.1. State diagram of A

If an input 0 is supplied, $M(S, 0) = \{s_1, s_2, s_3\}$. That is, O becomes $\{s_1, s_2, s_3\}$. A 1 input then changes O to $\{s_2, s_3\}$. Since 0 and 1 permute $\{s_2, s_3\}$, the occupied set remains $\{s_2, s_3\}$ under all subsequent inputs.

Definition 3.3.4: If $P \subseteq S$ and the tape x has the property that $|M(P, x)| < |P|$, x is said to produce a contraction in P. (Note that P does not necessarily contain M(P, x).)

Thus Lemma 3.3.2 established that, if $|S| = n$ and $|O| > 1$, $|O|$ can be reduced by some tape of length not exceeding $\binom{n}{2}$. The application of this result at most (n-1) times will reduce $|O|$ from n to 1, as in Theorem 3.3.3.

We now establish a second bound on the length of a shortest synchronizing sequence of A.

Theorem 3.3.4: If A is synchronizable and $|S| = n$, then A has a synchronizing sequence whose length does not exceed $2^n - n - 1$.

Proof: Recall the automaton A' of Proof 2, Theorem 3.2.2. Synchronizing A is equivalent to driving A' from s'_0 to one of the states $\{s_j\}$. corresponding to a singleton subset of S. A' has 2^n states, n of which are of the form $\{s_j\}$. Without entering the same state twice, A' can undergo at most 2^n - n-1 state transitions before entering one of the states $\{s_j\}$, since A' initially occupies the state $s'_0 = S$.

Definition 3.3.5: Let $B(n) = 2^n - n - 1$ for integers $n \geq 1$.

Theorem 3.3.4 could also have been proved noting the following. No portion of a shortest synchronizing sequence $x = \sigma_1 \ldots \sigma_r$ maps an occupied set onto itself. Therefore σ_1 takes S properly into S, so that following the application of σ_1, $|O| \leq n-1$. If $|O| = n-1$, there are $\binom{n}{n-1}$ subsets of S which may be occupied before an input symbol again reduces the size of the occupied set; hence at most $\binom{n}{n-1}$ input symbols are required to make $|O| \leq n-2$. In general, if $|O| = k$, $n \geq k > 1$, then no more than $\binom{n}{k}$ symbols are required to make $|O| \leq k-1$. A shortest synchronizing sequence, then, has a length not exceeding $\sum_{j=n}^{2} \binom{n}{j} = 2^n - n - 1$.

The point of this discussion is not to verify the bound established in Theorem 3.3.4, but rather to point out the bound on the number of symbols required to reduce $|O|$ from k ($n \geq k > 1$) to something not exceeding k-1. These step-by-step reductions of $|O|$ are different from those of Liu's bound (Results 3.3.1 through 3.3.3), which guarantee that $|O|$ can be reduced by merging two occupied states, using at most $\binom{n}{2}$ symbols.

The bound B(n) can be realized for n = 1, 2, and 3 as is shown in the following example. Thus B(n) is the best possible bound for n = 1, 2, 3.

Example 3.3.2:

(a) n = 1. $B(1) = 2^1 - 1 - 1 = 0$. Trivially a one-state automaton is synchronizable by any sequence x such that $\ell(x) \geq 0$.

(b) n = 2. $B(2) = 2^2 - 2 - 1 = 1$. If a two-state automaton is synchronizable, the minimum length of a synchronizing sequence is one. The state diagram of Figure 3.3.2 depicts such an automaton.

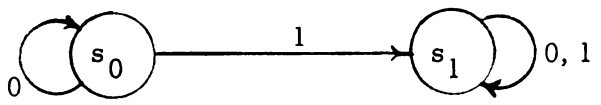

Figure 3.3.2. A two-state, synchronizable automaton

(c) n = 3. $B(n) = 2^3 - 3 - 1 = 4$. 0110 is a shortest synchronizing sequence for the automaton of Figure 3.3.3.

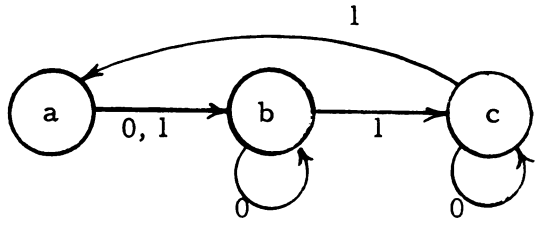

Figure 3.3.3. A three-state automaton requiring four symbols for synchronization

The action of 0110 on the state set $\{a, b, c\}$ is shown in Figure 3.3.4.

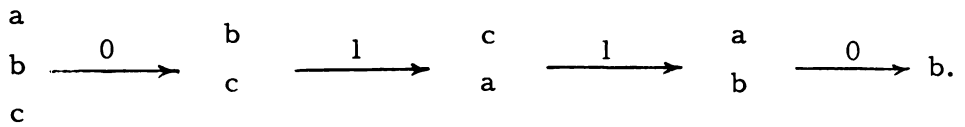

Figure 3.3.4. Action of the sequence 0110 on the state set of the automaton of Figure 3.3.3.

The bound $B(n) = 2^n - n - 1$ is better than Liu's bound $L(n) = (n-1)\binom{n}{2}$ for $n \leq 7$, while Liu's bound is lower for $n > 7$, with the difference between bounds increasing rapidly as n increases. Table 3.3.1 on page 27 lists values of B(n) and L(n) for $1 \leq n \leq 10$.

We now make one more approach to bounding the length of a shortest tape which synchronizes an n-state automaton. Following two preliminary results (Lemmas 3.3.5 and 3.3.6), we establish a new bound on the length of tape necessary to reduce $|O|$ (Lemma 3.3.7). The results of these three theorems are incorporated into the bound of Theorem 3.3.8.

Lemma 3.3.5: If A is synchronizable with n states and $|O| = n$, there is an input symbol which reduces $|O|$.

Proof: If A is synchronizable, every pair of states $s_i \neq t_1$ in S can be merged by some sequence $x_1 = \sigma_1 \ldots \sigma_r$, with no proper prefix of x_1 merging s_1 and t_1. Thus if $M(s_1, \sigma_1 \ldots \sigma_{r-1}) = s_2$ and $M(t_1, \sigma_1 \ldots \sigma_{r-1}) = t_2$, then $s_2 \neq t_2$ and $M(s_1, \sigma_r) = M(s_2, \sigma_r)$. If $|O| = n$, then $|M(S, \sigma_r)| < n$; i.e., σ_r reduces $|O|$.

Theorem 3.3.6: In a synchronizable n-state automaton A, if $|O| = n-1$, three symbols are sufficient to produce a contraction in O.

Proof: There is a pair $\{s_1, s_2\}$ of states which will merge under one (properly chosen) symbol. If O contains such a pair, then one symbol will yield a contraction. If O contains no such pair, then either $s_1 \notin O$ or $s_2 \notin O$.

Since A is synchronizable, there is an input sequence which will bring $\{s_1, s_2\}$ into O. There are only two choices of O for which $\{s_1, s_2\} \notin O$; and since one of them is currently O, at most two symbols are required to (perhaps) first change O to the other configuration and then to a configuration which contains $\{s_1, s_2\}$, upon which a third symbol merges s_1 and s_2, producing a contraction in O.

Pursuing the ideas of the preceding proof we obtain the following.

Lemma 3.3.7: In a synchronizable n-state automaton A, if $2 \leq |O| = p \leq n-2$, then $\binom{n}{p} - \binom{n-2}{p-2} + 1$ symbols (properly chosen) are sufficient to produce a contraction.

Proof: There is a pair $\{s_1, s_2\}$ of states which merge under one properly chosen input symbol. If this is the only such pair in the state set of A, there are $\binom{n-2}{p-2}$ ways for this pair to lie in O, with the remaining (p-2) states in O being chosen from among the n-2 states different from s_1 and s_2.

There are $\binom{n}{p}$ choices of O for which $|O| = p$. Thus there are $\binom{n}{p} - \binom{n-2}{p-2}$ choices of O for which $|O| = p$ and $\{s_1, s_2\} \not\subseteq O$; hence at most $\binom{n}{p} - \binom{n-2}{p-2} + 1$ symbols are required to bring O to a configuration containing $\{s_1, s_2\}$ and then merge s_1 and s_2. The proof is now complete.

In the preceding proof, if $\{s_1, s_2\}$ is not the only pair of states which merge under one symbol, a contraction may be produced using fewer than $\binom{n}{p} - \binom{n-2}{p-2} + 1$ symbols.

Using the preceding results, we construct a new bound on the length of the shortest synchronizing sequence for a synchronizable n-state automaton.

Theorem 3.3.8: If A is an n-state synchronizable automaton ($n \geq 4$), a shortest synchronizing sequence for A has length not exceeding

$$C(n) = 1 + 3 + \sum_{j=2}^{n-2} \left[\binom{n}{n-j} - \binom{n-2}{n-j-2} + 1 \right] .$$

Proof: The first two terms are consequences of Lemmas 3.3.5 and 3.3.6, respectively. The summation over $2 \leq j \leq n-2$ yields the terms corresponding to $|O| = n-2, n-3, n-4, \ldots, 2$, in that order, as derived in Lemma 3.3.7. These observations comprise the proof.

A form of C(n) more suited to computation is

$$C(n) = 3n-2 + 3 \sum_{j=1}^{n-4} \binom{n-2}{j} , \quad n \geq 4.$$

Values of C(n) for $1 \leq n \leq 10$ are listed in Table 3.3.1 on page 27.

Examination of Table 3.3.1 reveals that no one of the three bounds L(n), B(n), C(n) is least for all values of n, $1 \leq n \leq 10$. Each bound was arrived at by examining in a particular way the reduction of $|O|$ from p to some value not exceeding p-1 ($n \geq p > 1$). These examinations yielded $\binom{n}{p}$, $\binom{n}{2}$ and $\binom{n}{p} - \binom{n-2}{p-2} + 1$, respectively.

In Table 3.3.2 on page 28, we make a term-by-term comparison of the bounds L(n), B(n) and C(n), each term being the number of symbols sufficient to reduce $|O|$. Since a best bound has been achieved for $n < 4$, by L(n) and C(n), we concern ourselves in Table 3.3.2 only with $n \geq 4$.

Table 3.3.1. A comparison of bounds on the length of a shortest synchronizing sequence for an n-state (synchronizable) automaton

n	B(n)	L(n)	C(n)	Minsum(n)	D(n)	$D_1(n)$
1	0	0	0	-	-	-
2	1	1	1	1	1	1
3	4	6	4	4	4	4
4	11	18	10	10	11	10
5	26	40	22	22	24	22
6	57	75	46	44	45	42
7	120	126	94	79	76	72
8	247	196	184	130	119	114
9	502	288	382	200	176	170
10	1013	405	748	292	249	242

Table 3.3.2. A term-by-term comparison of L(n), B(n), C(n), and D(n). Each column heading L, B, C, and D identifies source of all terms in that column.

n	|O|	No. symbols to reduce |O| B	L	C	D
4	4	1	6	1*	1
	3	4	6	3*	4
	2	6	6	6*	6
5	5	1	10	1*	1
	4	5	10	3*	5
	3	10	10	8*	8
	2	10	10*	10	10
6	6	1	15	1*	1
	5	6	15	3*	6
	4	15	15	10*	10
	3	20	15*	17	13**
	2	15	15*	15	15
7	7	1	21	1*	1
	6	7	21	3*	7
	5	21	21	12*	12
	4	35	21*	26	16**
	3	35	21*	31	19**
	2	21	21*	21	21
8	8	1	28	1*	1
	7	8	28	1*	1
	6	28	28	14*	14
	5	56	28*	37	19**
	4	70	28*	56	23**
	3	56	28*	51	26**
	2	28	28*	28	28
9	9	1	36	1*	1
	8	9	36	3*	9
	7	36	36	16*	16
	6	84	36*	50	22**
	5	126	36*	92	27**
	4	126	36*	106	31**
	3	84	36*	78	34**
	2	36	36*	36	36
10	10	1	45	1*	1
	9	10	45	31*	10
	8	45	45	18*	18
	7	120	45*	65	25**
	6	210	45*	141	31**
	5	252	45*	197	36**
	4	210	45*	183	40**
	3	120	45*	113	43**
	2	45	45*	45	45

A single asterisk is used in Table 3.3.2 to indicate, in each row, a minimal entry among columns B, L, and C. Each entry in these columns is valid for an arbitrary synchronizable automaton. We may therefore select the least entry in each row for a given value of n and sum these least entries to obtain a new bound, Minsum(n), on the length of a shortest synchronizing sequence. Minsum(n), $1 \leq n \leq 10$, is entered in Table 3.3.1 for ease of comparison with other bounds.

We note that, at least within the scope of Table 3.3.2, Minsum(n) may be obtained by using the last (n-4) terms of L(n) and the first three terms of C(n). This observation holds for all $n \geq 4$, as we shall now verify.

Theorem 3.3.9: For all $n \geq 4$, each of the first three terms of C(n) is less than or equal to the corresponding term of B(n) or L(n).

Proof: The first terms of B(n), L(n), and C(n) respectively are 1, $\binom{n}{2}$ and 1. The theorem thus holds for the first terms of the three series.

The second terms of B(n), L(n) and C(n) are $\binom{n}{n-1}$, $\binom{n}{2}$ and 3, respectively. For $n \geq 4$, $\binom{n}{n-1} = n > 3$ and $\binom{n}{2} = \frac{n(n-1)}{2} \geq 6 > 3$.

The third terms of B(n), L(n) and C(n) respectively are $\binom{n}{n-2}$, $\binom{n}{2}$ and $\binom{n}{n-2} - \binom{n-2}{n-4} + 1$. For all integers $n > 1$, $\binom{n}{n-2} = \binom{n}{2}$. For all $n \geq 4$, $\binom{n}{2} - \left[\binom{n}{n-2} - \binom{n-2}{n-4} + 1\right] = \binom{n-2}{n-4} - 1 = \binom{n-2}{2} - 1 = \frac{(n-2)(n-3) - 2}{2} \geq \frac{2 \cdot 1 - 2}{2} = 0$. Hence $\binom{n}{2} \geq \binom{n}{n-2} - \binom{n-2}{n-4} + 1$ for $n \geq 4$. This concludes the proof.

Theorem 3.3.10: For $n \geq 5$ and for all integers j such that $4 \leq j \leq n-1$, the j^{th} term of L(n) does not exceed the corresponding term of B(n) or C(n).

Proof: For $4 \leq j \leq n-1$, the j^{th} terms of B(n), L(n) and C(n) are respectively $\binom{n}{n-j+1}$, $\binom{n}{2}$ and $\binom{n}{j-1} - \binom{n-2}{j-1} + 1$. For the values of j being considered, $3 \leq j-1 \leq n-2$; and so $\binom{n}{n-2} = \binom{n}{2} \leq \binom{n}{j-1} = \binom{n}{n-j+1}$ for all such j. That is, the j^{th} term of L(n) does not exceed the j^{th} term of B(n).

Still bearing in mind $3 \leq j-1 \leq n-2$, we have the following:

$$j^{th} \text{ term of } C(n) = \binom{n}{j-1} - \binom{n-2}{j-1} + 1$$

$$= \frac{n(n-1)\ldots(n-j+2)}{(j-1)!} - \frac{(n-2)(n-3)\ldots(n-j)}{(j-1)!}$$

($j-1$) factors; largest is $(n-2)$.

($j-2$) factors; largest $\leq n-2$.

$$\geq \frac{n(n-1)\ldots(n-j-2)}{(j-1)!} - 1 + 1$$

$j-1$ factors; smallest ≥ 3.

($j-2$) factors; smallest $= 2$.

$$= \frac{n(n-1)}{2} \quad \frac{(n-2)(n-3)\ldots(n-j+2)}{(j-1)\ldots(3)} \geq \frac{n(n-1)}{2} = \binom{n}{2} = j^{th} \text{ term of } L(n).$$

The proof is now concluded.

One might wonder at this point what the lower limit of such bounds for arbitrary synchronizable automata might be. The question is partially resolved by the existence of a class of automata in which an n-state machine requires $(n-1)^2$ steps for synchronization. The construction is as follows.

Let $\Sigma = \{0, 1\}$ and $S = \{s_1, \ldots, s_n\}$. Let the input 0 take s_1 and s_2 to s_2, fixing $s_3, \ldots, s_n$; let 1 permute the state set cyclically, sending s_i to s_{i+1} $(1 \leq i \leq n-1)$ and s_n to s_1. The shortest synchronizing sequence for such an automaton is $(01^{n-1})^{n-2}0$; and $\ell[(01^{n-1})^{n-2}0]$ $= (n-2)(n) + 1 = n^2 - 2n + 1 = (n-1)^2$.

The assertions of the preceding paragraph are made without proof. The two automata of the following example are offered as supportive evidence. A useful notion, that of the synchronization tree of a synchronizable automaton, is also introduced in Example 3.3.2. The tree is read from top to bottom; the path labelled with σ and emanating from vertex $S_i \subseteq S$ leads to $M(S_i, \sigma)$. Other features of the tree are pointed out within the example.

Example 3.3.2: Let A_n be the n-state automaton described above, with next state function M_n.

(a) n = 3

Table 3.3.3. Transition table for A_3

M_3	0	1
s_1	s_2	s_2
s_2	s_2	s_3
s_3	s_3	s_1

To avoid crowding, the subscripts of the s_i, with the letter s omitted, are shown at each vertex.

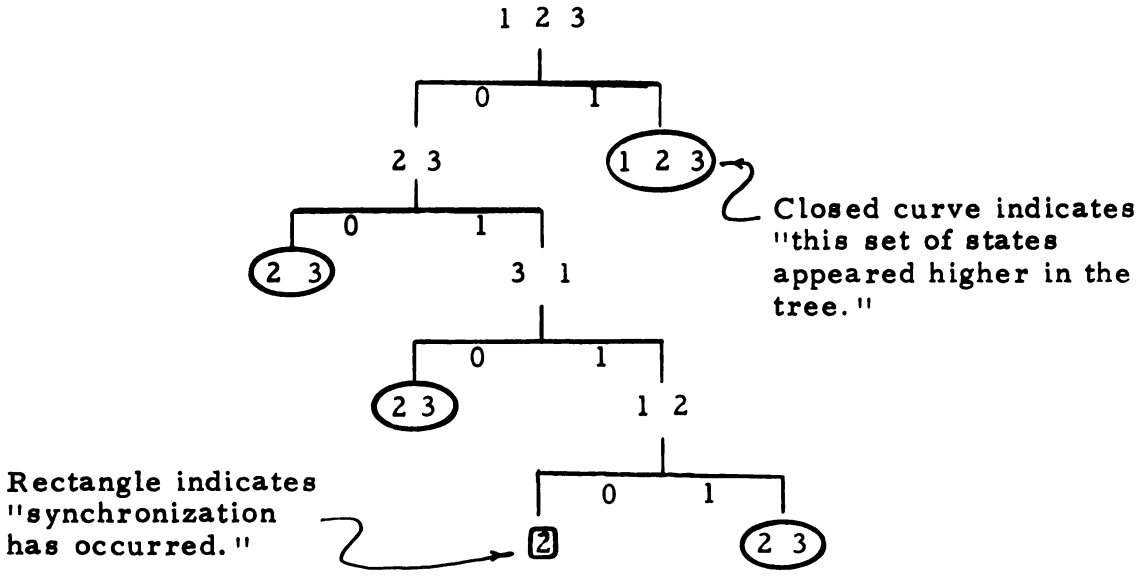

Figure 3.3.5. Synchronization tree for A_3

Inspection of the tree shows that, for n = 3, earliest synchronization occurs under $0110 = (01^2)^1 0 = (01^{3-1})^{3-2} 0$.

(b) n = 4

Table 3.3.4. Transition table for A_4

M_4	0	1
s_1	s_2	s_2
s_2	s_2	s_3
s_3	s_3	s_4
s_4	s_4	s_1

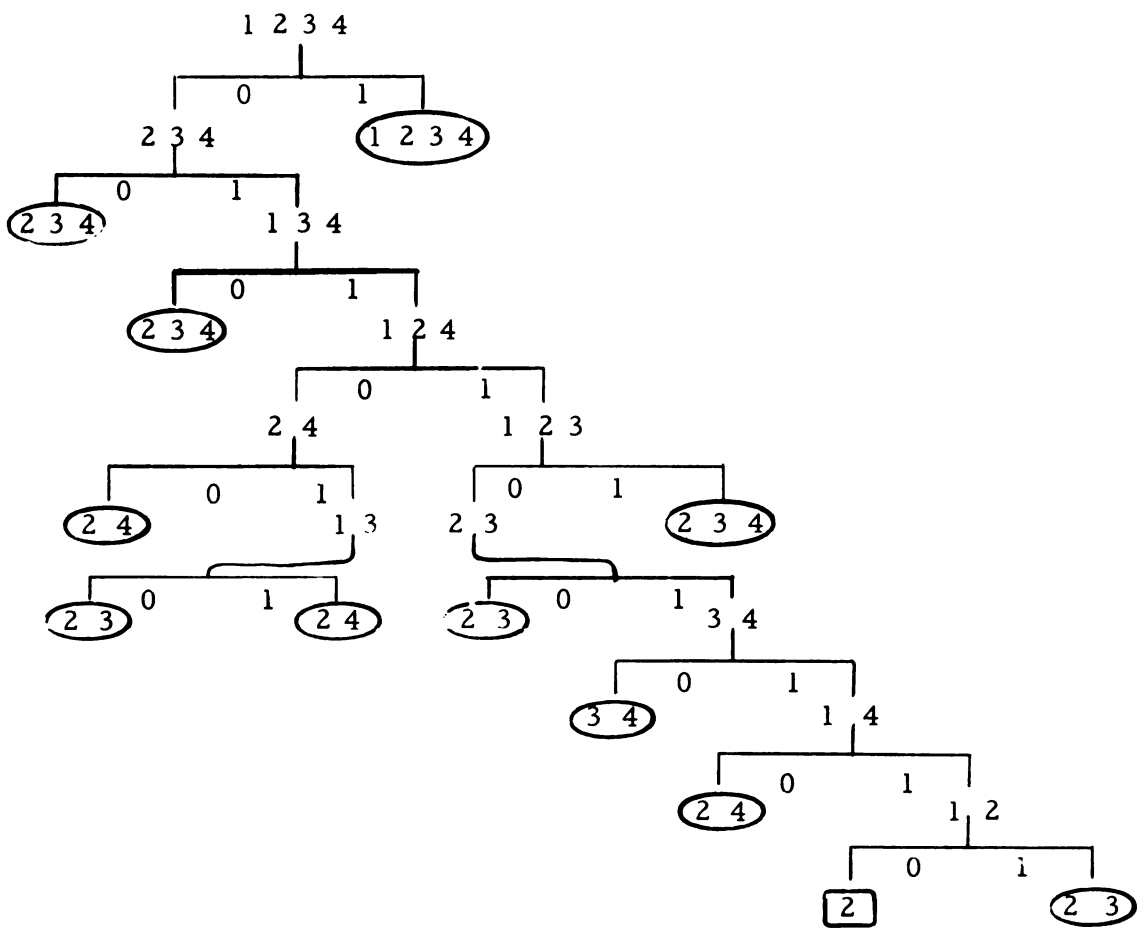

Figure 3.3.6. Synchronization tree for A_4

Thus for n = 4, synchronization occurs first under 011101110 $= ((01)^{4-1})^{4-2}0$.

The preceding example shows that, for a given $n \geq 1$, the best bound on the length of shortest synchronizing sequences for n-state synchronizable automata cannot be less than $(n-1)^2$. It should be noted in connection with Example 3.3.2 that construction of a synchronization tree for an arbitrary automaton, with each path leading to a vertex which appears earlier in the tree being terminated, constitutes a bounded experiment to determine whether that automaton is synchronizable. The functions L(n), B(n), C(n) and Minsum (n), for example, bound the number of steps from the top of the tree within which synchronization will occur if at all possible.

3.4. Synchronization Bounds for Several Classes of Automata

The bounds which were established in the preceding section for the length of a shortest synchronizing sequence are valid if no information is available concerning the automaton in question, save that it is synchronizable. Placing conditions on the automaton yields tighter bounds, as one would expect. We examine two restricted classes of automata in the present section.

Liu's bound L(n) was predicated on the fact that any two states which can be synchronized can be merged by some tape of length not exceeding $\binom{n}{2}$ (see Lemma 3.3.2). If an automaton has a pair of states requiring exactly $\binom{n}{2}$ symbols for synchronization, additional information emerges concerning synchronization of the automaton and is set forth in the following results.

Theorem 3.4.1: If the n-state automaton A has a pair of states which require $\binom{n}{2}$ symbols for synchronization, then

(1) A is synchronizable, and

(2) for each integer j such that $1 \leq j \leq \binom{n}{2}$ there is a unique pair $P_j = \{ s_{i_j}, s_{k_j} \}$ of states of A which requires j symbols for synchronization.

Proof: (1) Let $p_{\binom{n}{2}}$ denote a pair of states requiring $\binom{n}{2}$ symbols for synchronization. Then $p_{\binom{n}{2}}$ is carried through all remaining state pairs $\{s, t\}$, $s \neq t$, before a singleton set of states is reached. If $x = \sigma_1 \sigma_2 \ldots \sigma_{\binom{n}{2}}$ is a sequence of length $\binom{n}{2}$ which synchronizes $p_{\binom{n}{2}}$ to s, and if we let $M(p_{\binom{n}{2}-1}, \sigma_1) = p_{\binom{n}{2}-1}$, $M(p_{\binom{n}{2}-1}, \sigma_2) = p_{\binom{n}{2}-2}, \ldots,$

then the set of all $\binom{n}{2}$ pairs of distinct states is enumerated in the list $p_{\binom{n}{2}}, p_{\binom{n}{2}-1}, \ldots p_2, p_1$. Every pair is synchronizable, since $M(p_{\binom{n}{2}-j}, \sigma_{j+1}\sigma_{j+2}\ldots\sigma_{\binom{n}{2}}) = s_j$; therefore A is synchronizable.

(2) In the listing $p_{\binom{n}{2}}, \ldots, p_2, p_1$, the pair p_j requires j or fewer symbols for synchronization. That j symbols (properly chosen) are sufficient was shown in Part (1) of this proof. If h symbols, $h \leq j$, will synchronize p_j, then $\binom{n}{2} - j + h$ symbols will synchronize $p_{\binom{n}{2}}$, contrary to hypothesis; therefore j symbols are required to synchronize p_j. The proof is now complete.

In the event that some pair of states of A requires $\binom{n}{2}$ symbols for synchronization, the preceding result yields a bound on the minimum number of symbols which will synchronize A.

Theorem 3.4.2: If the n-state automaton A has two synchronizable states which require $\binom{n}{2}$ symbols for synchronization, then the shortest sequence which will synchronize A has a length not exceeding $\sum_{j=2}^{n} \left[\binom{n}{2} - \binom{j}{2} + 1 \right] = D(n)$.

Proof: If $|O| = k$, where O is the occupied set of states and $2 \leq k \leq n$, then there are represented in O exactly $\binom{k}{2}$ pairs of states from the exhaustive list $p_{\binom{n}{2}}, \ldots, p_1$ of the proof of Theorem 3.4.1. Among the pairs of states in O, there is a pair requiring no more than $\binom{n}{2} - \binom{k}{2} + 1$ symbols for synchronization, since in the most extreme instance the pairs of states in O are exactly the first $\binom{k}{2}$ pairs in the list $p_{\binom{n}{2}} \ldots p_1$ of all pairs of states. Thus to change $|O|$ successively from n to 1, assuming the worst case where $|O|$ decreases by one at each contraction, requires at most $\sum_{j=2}^{n} \left[\binom{n}{2} - \binom{j}{2} + 1 \right]$ symbols. This is the desired conclusion.

For the sake of comparison, terms of D(n) ($n \leq 4 \leq 10$) are entered in Table 3.3.2 on page 28. A double asterisk ("**") indicates those terms of D(n) which are less than the corresponding terms (marked with "*" in Table 3.3.2) of Minsum(n). The terms of D(n)

are valid only under the hypothesis "some pair of states of A requires $\binom{n}{2}$ symbols to merge," while the terms of Minsum(n) are valid for any synchronizable automaton. For each n, we may thus obtain a new bound $D_1(n)$ for this restricted class of automata by selecting, for each value of $|O|$, either the corresponding Minsum(n) or D(n) term, whichever is less. Values of D(n) and the improved bound $D_1(n)$, $1 \leq n \leq 10$, are entered in Table 3.3.1.

Lest it be thought we operate in a vacuum when we speak of n-state automata having a pair of states requiring $\binom{n}{2}$ steps for their synchronization, we observe that the machines of Example 3.3.2 are such devices. More formally, keeping in mind that each integer $n > 1$ has a representation of the form $2k$ or $2k + 1$, where k is a positive integer, we have the following result.

Theorem 3.4.3: Let A be an automaton with state set $S = \{s_1, s_2, \ldots, s_n\}$, $|S| = n > 1$, over the alphabet $\Sigma = \{0, 1\}$ such that $M(s_1, 0) = s_2$, $M(s_j, 0) = s_j$ for $2 \leq j \leq n$, and $M(s_j, 1) = s_{j+1}$ for $1 \leq j \leq n-1$ with $M(s_n, j) = s_1$. If $n = 2k$ or $n = 2k+1$, then $\{s_2, s_{k+2}\}$ is a pair of states requiring $\binom{n}{2}$ steps for synchronization.

Proof: The state diagram of such an n-state automaton is supplied (Figure 3.4.1) in order that the proof may be carried on in visual, rather than symbol-manipulative, terms. In the diagram, (j) corresponds to s_j.

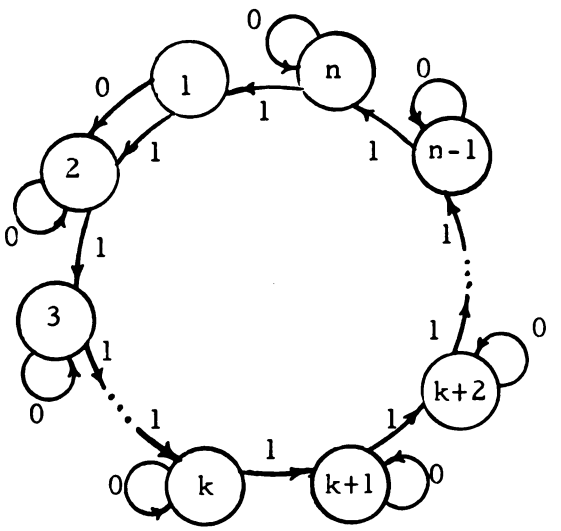

Figure 3.4.1. State diagram of n-state automaton requiring $(n-1)^2$ symbols for synchronization

Examining the state diagram, we see that the only instance in which an input will alter the relative positions of two states on the circle occurs when one of those states is $s_1 = 1$ and a 0 is received. In such an instance, the counterclockwise "distance" between the two states, starting with $s_2 = 2 = M(s_1, 0)$, is one less than the corresponding distance before the input 0. For $n = 2k+1$, Figure 3.4.2 depicts the synchronization of the pair $\{2, 2k+2\} = \{s_2, s_{k+2}\}$.

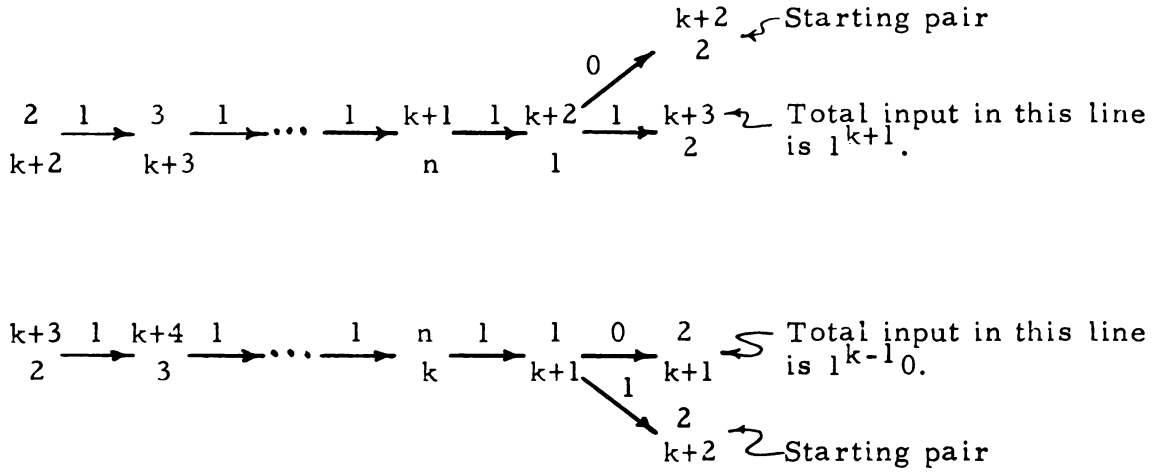

Figure 3.4.2. Shortest transition sequence merging $\{s_2, s_{k+2}\}$

In Figure 3.4.2, a transition under 0 is shown only when it is not the identity mapping on the pair of states in question. At the end of the first line, a 0 input would return O to the starting pair of states; therefore a transition under 1 is selected and the second line continues from that point. For like reasons, a transition under 0 is selected at the end of the second line.

Thus the sequence 1^k0 takes $\{2, k+2\}$ to $\{2, k+1\}$, decreasing the distance between the occupied states by 1, from k to k-1. The next decrease occurs when $\{2, k+1\}$ has been taken to $\{1, k\}$ by input sequence 1^{2k}, at which point an input 0 yields $O = \{2, k\}$. Since the initial gap was k units on the state circle, and since each application of $1^{2k}0$ decreases the gap by one, the tape $(1^{2k}0)^k$ synchronizes $\{2, k+2\}$, is the shortest tape to do so, and $\ell[(1^{2k}0)^k] = k(2k+1)$ $= \frac{(2k+1)(2k)}{2} = \binom{n}{2}$.

This completes the proof for the case n = 2k+1. A similar proof holds for the case n = 2k, and is left to the reader.

3.5. Synchronization of Automata Having Dead States

In this section we examine another restricted class of automata, those having a state which, once entered, is never escaped. Following preliminary definitions and results, we establish a bound on the length of shortest synchronizing sequences for this type of automaton and show by example that the bound is minimal.

Definition 3.5.1: If $A = (S, M, s_0, F)$ and $S_1 \subseteq S$, S_1 is a dead subset of S if $S_1 \neq S$ and for each $\sigma \in \Sigma$, $M(S_1, \sigma) \subseteq S_1$. The state $s \in S$ is a dead state if $\{s\}$ is a dead subset of S.

Some observations about an automaton with a dead state are given without proof in the next result.

Lemma 3.5.1: If A has a dead state s_D, then the tape x synchronizes A if and only if $M(S, x) = s_D$. A synchronizable automaton has no more than one dead state. If in such a case $s_D \in F$, then $\Sigma^* x \Sigma^* \subseteq T(A)$. If $s_D \notin F$, $\Sigma^* x \Sigma^* \cap T(A) = \phi$.

Definition 3.5.2: A is strongly connected if, for any states s_i and s_j of A, there is a tape x such that $M(s_i, x) = s_j$.

Theorem 3.5.2: A has no dead subset of states if and only if A is strongly connected.

Proof: Sufficiency of the latter condition is apparent. We establish necessity by contraposition.

Let $s_i \neq s_j$ be two states of A. Let $r(s_i)$ denote the set of states which can be reached from s_i. For any state $t \in r(s_i)$ and any input symbol σ, there is a tape x_t such that $M(s_i, x_t) = t$; hence $M(t, \sigma) = M(s_i, x_t\sigma) \in r(s_i)$. That is to say, $M(r(s_i), \sigma) \subseteq r(s_i)$.

If $s \notin r(s_i)$, then $r(s_i) \neq S$, and is hence a dead subset of S. That is, A not strongly connected implies A has a dead subset of states. We have thus proved the contrapositive of the statement to be proved.

The next result is not restricted to automata having dead states, but is included in this section because it greatly simplifies the proof of Theorem 3.5.4.

Theorem 3.4.3: If $A = (S, M, s_0, F)$, $|S| = n$, $T \subsetneq S$ and $s \in r(t)$ for some $t \in T$, then there exist a state $t_1 \in T$ and a tape x_1 such that $\ell(x_1) \leq |S| - |T|$ and $M(t_1, x_1) = s$.

Proof: If $s \in T$, the theorem is trivially fulfilled by taking $t_1 = s$ and $x_1 = \Lambda$.

If $s \notin T$, a sequence $x = \sigma_1 \sigma_2 \ldots \sigma_k$ may be chosen so that $M(t, x) = s$ and no state appears twice in the sequence of states t, $M(t, \sigma_1)$, $M(t, \sigma_1 \sigma_2), \ldots, M(t, \sigma_1 \ldots \sigma_k) = s$. Since $t \in T$ and $s \in S$, there is a greatest integer $j (1 \leq j \leq k)$ such that $M(t, \sigma_1 \ldots \sigma_j) \in T$.

Then each state $M(s, \sigma_1 \ldots \sigma_{j+1}), \ldots, M(t, \sigma_1 \ldots \sigma_k) = s$ is in $S - T$. Since no duplications appear in this latter sequence, $|\{M(t, \sigma_1 \ldots \sigma_{j+1}), \ldots, M(, \sigma_1 \ldots \sigma_k)\}| \leq |S| - |T|$. Choosing $t_1 = M(t, \sigma_1 \ldots \sigma_j)$ and $x_1 = \sigma_{j+1} \ldots \sigma_k$ fulfills the conclusion of the present theorem and completes the proof.

We now establish the bound described in the opening paragraph of this section.

Theorem 3.5.4: If $A = (S, M, s_0, F)$ is synchronizable and has a dead state s_D, then A is synchronizable by some tape of length not exceeding $\binom{n}{2}$.

Proof: Since A is synchronizable (and only to s_D), for each nonempty subset T_j of $T = S - \{s_D\}$ and for any state $t \in T_j$, $s_D \in r(t)$. Thus any nonempty set $T_j \subseteq T$ and the state $s_D \in S - T_j$ satisfy the hypothesis of Theorem 3.5.3.

Let $T_1 = T \neq \phi$. Then $|T_1| = n-1$. By Theorem 3.5.3 there is a state $t_1 \in T_1$ and a tape w_1 such that $\ell(w_1) \leq |S| - |T_1|$ and $M(t_1, w_1) = s_D$. Let $T_2 = M(T_1, w_1) \cap T$. If $T_2 = \phi$, then w_1 synchronized A, since $M(S, w_1) = s_D$ in this case. If $T_2 \neq \phi$, proceed.

Now $|T_2| < |T_1|$, since $M(t_1, w_1) = s_D \notin T$. There is a state $t_2 \in T_2$ and a tape w_2 such that $\ell(w_2) \leq |S| - |T_2|$ and $M(t_2, w_2) = s_D$. Let $T_3 = M(T_2, w_2) \cap T$. If $T_3 = \phi$, then $w_1 w_2$ synchronized A. If $T_3 \neq \phi$, proceed.

Continuing in this fashion, one obtains a sequence of nonempty sets $T_1, T_2, \ldots$ of T and a set of corresponding tapes $w_1, w_2, \ldots$ such that $n-1 = |T_1| > |T_2| > \ldots$ and $\ell(w_j) \leq |S| - |T_j|$, $j = 1, 2, \ldots$. For some integer $k \leq n$, it must be the case that $|T_{k-1}| > 0$ and $|T_k| = 0$; hence $w_1 w_2 \ldots w_k = w$ synchronizes A; and $\ell(w) = \sum_{j=1}^{k-1} \ell(w_j) \leq \sum_{j=1}^{k-1} (|S| - |T_j|) = \sum_{j=1}^{k-1} (n - |T_j|) \leq \sum_{\ell=1}^{n-1} \ell = \binom{n}{2}$. The desired bound is thus established.

The bound established in Theorem 3.5.4 can be realized for arbitrary n by allowing $|\Sigma| = n-1$. This will not be proved here, but the following instances for n = 5 and n = 6 suggest a scheme for generating an example for arbitrary n.

Example 3.5.1:

(a) $n = 5$. $\binom{n}{2} = 10$. $\Sigma = \{0, 1, 2, 3\}$.

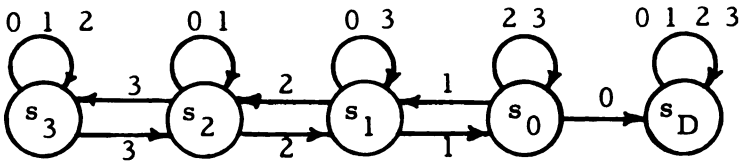

Figure 3.5.1. A 5-state automaton requiring $\binom{5}{2} = 10$ symbols for synchronization

A shortest synchronizing tape for the automaton of Figure 3.5.1 is 0102103210, with length 10.

(b) $n = 6$. $\binom{n}{2} = 15$. $\Sigma = \{0, 1, 2, 3, 4\}$.

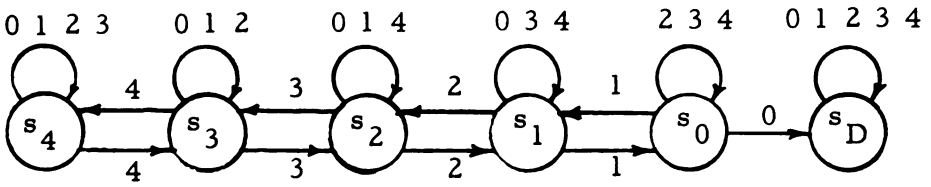

Figure 3.5.2. A 6-state automaton requiring $\binom{6}{2} = 15$ symbols for synchronization.

A shortest synchronizing tape for the automaton of Figure 3.5.2 is 010210321043210, with length 15.

4. SYNCHRONIZABILITY OF PARTICULAR CLASSES OF AUTOMATA, WITH CHARACTERIZATIONS OF THEIR SYNCHRONIZING TAPES

In the present chapter we examine several classes of automata, establishing conditions under which an automaton in a given class is synchronizable and characterizing, for such a machine, the set of all sequences which synchronize it. We begin with definite automata as treated by Perles, Rabin and Shamir [14] and then take up in turn the ultimate-definite and reverse ultimate-definite automata of Paz and Peleg [13]. Lastly, a new form of regular event emerges quite naturally. It turns out to be a specialization of events of Paz and Peleg; we close the chapter with a treatment of some of its properties.

Although several early results in the chapter are later lumped into one, and a theorem occuring within the chapter subsumes some earlier results, the presentation here is meant to preserve a spirit of discovery which is among the earliest casualties of a more terse style.

4.1. Synchronization of Definite Automata

Perles, Rabin and Shamir ([14], Theorem 2) showed that the state of a reduced k-definite automaton depends only on the past k input symbols. Thus such a machine $A = (S, M, s_0, F)$ is trivially synchronizable; for, any sequence x such that $\ell(x) \geq k$ has the property that, for any $s \in S$, $M(s_0, x) = M(s, x)$. We have shown the following.

Lemma 4.1.1: If A is any reduced k-definite automaton and x is a tape such that $\ell(x) \geq k$, x synchronizes A.

The preceding result of course applies to the smaller class of purely k-definite automata $A(\Sigma^* W)$, where $W \subseteq \Sigma^k$. We narrow our attention to the latter class for the time being. The next example reveals that such automata may have synchronizing sequences of length less than k.

Example 4.1.1: Let $W_1 = \{100, 101, 001\} \subseteq \Sigma^3$, where $\Sigma = \{0, 1\}$. The prefix automaton $P(\Sigma^* W_1)$ (see Chapter 2, section 2.5) is depicted in Figure 4.1.1.

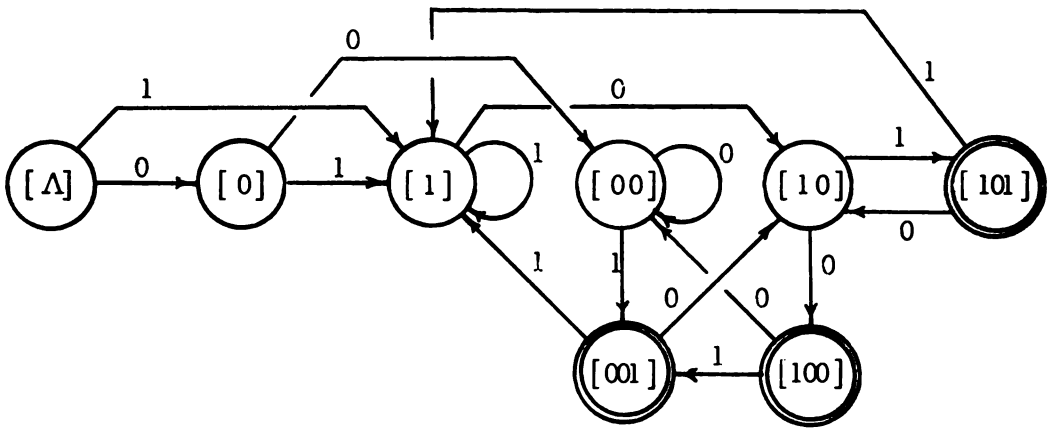

Figure 4.1.1. State diagram of $A(\Sigma^* W_1)$

The synchronizing tapes for P are 10, 11 and any tape whose length exceeds 2.

Note that, by its very construction, the prefix automaton of a purely k-definite event is synchronized by any tape of length not less than k. In addition to such tapes, the automaton of Example 4.1.1 has the shorter synchronizing tapes 10 and 11, and neither of these is a 2-suffix of a 3-tape in W_1.

Theorem 4.1.2: If $\ell(x) = k-1$, x is a synchronizing sequence for $P(\Sigma^* W)$ $(W \subseteq \Sigma^k)$ if and only if x is not a (k-1)-suffix of any k-tape in W.

Proof: (Necessity, by contraposition) If $\ell(x) = k-1$ and x is a suffix of some $y = \sigma_1 \sigma_2 \ldots \sigma_k \in W$, then $[\sigma_1]$ is a state of the corresponding prefix automaton P (see Chapter 2, section 2.5), as is $[\Lambda]$; hence $M([\sigma_1], x) = [y]$ with $\ell(y) = k$, and $M([\Lambda], x) = [t]$ with $\ell(t) \leq k-1$. Therefore x is not a synchronizing sequence for the prefix automaton $P(\Sigma^* W)$. Since $[\Lambda]$ and $[\sigma_1]$ are distinguishable states (by the tape $\sigma_2 \ldots \sigma_k$), both are also states of the reduced version of $P(\Sigma^* W)$. The same may be said for states $M([\Lambda], x)$ and $M([\sigma_1], x)$. Therefore x fails to synchronize the reduced automaton $A(\Sigma^* W)$.

(Sufficiency) If $\ell(x) = k-1$ and x is not a (k-1)-suffix of any $y \in W$, consider any $[w] \in S$ such that $w \neq \Lambda$. Now $M([w], x) = [u]$ if and only if u is the longest suffix of wx such that $[u] \in S$. The length of u does not exceed k-1; for, if $\ell(u) = k$ then x is a (k-1)-suffix of the tape $u \in W$, contrary to hypothesis.

Since $\ell(u) \leq k-1$, u is a suffix of x and is the longest suffix of x such that $[u] \in S$. Since [w] was arbitrary except for the restriction $w \neq \Lambda$, $M([w], x) = [u]$ for any $[w] \in S$ such that $w \neq \Lambda$. Finally, $M([\Lambda], x) = [v]$ if and only if v is the longest suffix of $\Lambda x = x$ such that $[v] \in S$; hence v = u and for any $[w] \in S$, $M([w], x) = [u]$. That is, x synchronizes $P(\Sigma^* W)$.

Corollary to Theorem 4.1.2: If $\ell(y) \leq k-1$ and y is a synchronizing sequence for $P(\Sigma^* W)$, then y is not a subtape of a (k-1)-suffix of any tape in W.

Proof: (By contraposition) If y is a subtape of a (k-1)-suffix of some tape in W, then there exist tapes t and u such that tyu is a (k-1)-suffix of a tape in W. By Theorem 4.1.2, tyu does not synchronize $P(\Sigma^* W)$; hence y does not synchronize P. The proof is complete.

One might conjecture that, if $\ell(y) \leq k-1$ and y is not a suffix of any tape in W, then y synchronizes $P(\Sigma^* W)$. This is not the case; for, if $W = \{000, 100, 010\}$, investigation of $P(\Sigma^* W)$ reveals the "short" synchronizing sequences to be 01, 11 and no others. Although 1 is not a suffix of any tape in W, 1 fails to synchronize P. The precise conditions under which a tape of length $\leq k-1$ is a synchronizing tape for $P(\Sigma^* W)$ are suggested by the above corollary and verified in the next result.

Theorem 4.1.3: If $\ell(y) \leq k-1$ and y is not a subtape of any (k-1)-suffix of a tape in W, then y is a synchronizing sequence for $P(\Sigma^* W) = (S, M, s_0, F)$.

Proof: (Notation: If α is a description of a prefix of a tape in W, let $[\alpha]$ denote the corresponding state in S.)

Recall that, for any tape y, $M([\Lambda], y) = [$ longest suffix of y which is also a prefix of some $w \in W]$, and, if $[x] \neq [\Lambda]$, $M([x], y) = [$ longest suffix of xy which is also a prefix of some $w \in W]$. Now suppose y

meets the hypothesis of the present theorem. If $M([x], y) = [vy]$ for some $v \neq \Lambda$, then $\ell(vy) \leq k$ and y is a subtape of the (k-1)-suffix of some tape $ryz \in W$, since vy must be a prefix of a tape in W. This is contrary to hypothesis.

Therefore $v = \Lambda$, and $M([x], y) = [$longest suffix of y which is a prefix of some $w \in W] = M([\Lambda], y)$, for any $[x] \neq [\Lambda]$. Hence y synchronizes $P(\Sigma^* W)$.

The preceding theorems and corollary are summarized in the following statement.

Theorem 4.1.4: If $W \subseteq \Sigma^k$, then the tape y synchronizes $P(\Sigma^* W)$ if and only if y is not a subtape of any (k-1)-suffix of a tape in W.

An alternate form, based on previously defined terms, is the following.

Theorem 4.1.4-a: If $W \subseteq \Sigma^k$, then y is a synchronizing sequence for $A(\Sigma^* W)$ if and only if y is not a prefix of any proper suffix of a tape $w \in W$.

This formulation, which matches nicely the statement of a later result, was not used in the preceding development due to a feeling that "subtape of a (k-1)-suffix" conveys a more vivid mental picture than "prefix of a proper suffix." The two notions are equivalent.

Save for Lemma 4.1.1, the results of this chapter have been stated and proved in terms of prefix automata. Not every prefix automaton is reduced; for instance, states [001] amd [101] of Example 4.11 are equivalent. Certainly each tape which synchronizes a given prefix automaton P also sychronizes the reduced version, P', thereof. The question is: Are there sequences which synchronize the reduced machine P' but which fail to synchronize the k-definite prefix automaton P because of its extra, redundant states?

Theorem 4.1.5: Let $A(\Sigma^* W)$, $W \subseteq \Sigma^k$, be the reduced automaton whose set of accepted tapes is $\Sigma^* W$. The tape x synchronizes A if and only if x is not a prefix of a proper suffix of a tape in W.

Proof: We establish the contrapositive of the theorem.

If x fails to synchronize A, there are states $s_1, s_2 \in S$ such that $M(s_1, x) \neq M(s_2, x)$. Since A is reduced, there are tapes v_1 and v_2

such that $M(s_0, v_1) = s_1$ and $M(s_0, v_2) = s_2$; hence $M(s_0, v_1x) \neq M(s_0, v_2x)$. Again because A is reduced, there is a tape z which distinguishes between $M(s_0, v_1x)$ and $M(s_0, v_2x)$.

Without loss of generality, suppose $M(s_0, v_1xz) \in F$ and $M(s_0, v_2xz) \notin F$. Since A is k-definite, the preceeding statement implies $\ell(xz) \leq k-1$; so the k-suffix of v_1xz is a tape in W in which x is a prefix of a proper suffix. The contrapositive of the reverse implication in the theorem is now established.

Conversely, if x is a prefix of a proper suffix of a tape $w \in W$, suppose $w = zxy$ $(z \neq \Lambda)$. Then $M(s_0, xy) \notin F$, since every tape accepted by A must be at least k symbols in length; and $M(s_0, zxy) \in F$. Thus xy fails to merge s_0 and $M(s_0, z)$, as does x; therefore x fails to synchronize A. The contrapositive of the forward implication in the theorem is now established, and the proof is complete.

We have shown that the set of sequences which synchronize the reduced automaton $A(\Sigma^* W)$ is precisely the set of synchronizing sequences of the prefix automaton $P(\Sigma^* W)$, where $W \subseteq \Sigma^k$.

4.2. Ultimate Definite, Reverse Ultimate-Definite and Symmetric-Definite Automata

Paz and Peleg [13] generalized the notion of a purely k-definite event and considered several closely related forms of events as follows.

Definition 4.2.1: The event $P \subseteq \Sigma^*$ is ultimate-definite if and only if there is an event $Q \subseteq \Sigma^*$ such that $P = \Sigma^* Q$.

The event $P \subseteq \Sigma^*$ is reverse ultimate-definite if and only if P^r is ultimate-definite. (The set of tapes obtained by reversing all tapes in P is denoted by P^r.)

The event $P \subseteq \Sigma^*$ is symmetric-definite if and only if there are events $Q, R \subseteq \Sigma^*$ such that $P = Q\Sigma^* R$. [Note that P, Q and R need not be finite or even regular.]

The authors define a representation $\Sigma^* Q$ of the ultimate-definite event U to be canonical if and only if no tape in Q is a proper suffix of another tape in Q. A constructive procedure for obtaining such a representation is used to prove the following theorem.

Theorem 4.2.1: (Paz and Peleg) If P is ultimate-definite, then P has a unique canonical representation.

We omit the proof of the theorem, but show the construction of the canonical form. Let $P_0 = P \cap \Sigma^0$, and for each non-negative integer j, let $P_{j+1} = P \cap \Sigma^{j+1} - \bigcup_{k=0}^{j} \Sigma^{j+1-k} P_k$. That is, P_0 is the set of all tapes in P of length zero, P_1 is the set of all tapes in P of length 1 which do not have as suffixes any tape in P_0,

. . .

P_{j+1} is the set of all tapes in P of length j+1 which do not have suffixes in P_j, P_{j-1}, ..., P_0. Finally let $P^t = \bigcup_{0}^{\infty} P_j$. Thus P^t is a listing of all tapes in P which do not have as proper suffixes other tapes in P.

In the case of a purely k-definite event $P = \Sigma^* Q (Q \subseteq \Sigma^k)$, it is evident that $P_k = Q$ and, for $j < k$, $P_j = \phi$. Thus $\Sigma^* Q$ is the canonical form of the purely k-definite event P.

Definition 4.2.2: Let $\Sigma^* P^t$ denote the canonical representation of the ultimate-definite event P.

It was noted earlier that $P = \Sigma^* Q$ being ultimate-definite does not require Q to be finite or even regular. For example, if $Q_1 = \{10^p : p \text{ is a positive prime integer}\}$, Q_1 is a well-known non-regular set, and $P_1 = \Sigma^* Q_1$ is not recognizable by any finite automaton, hence is also non-regular. (The truth of the latter assertion is evident if one observes that such an automaton would be able to recognize the set of positive prime integers, a task beyond the capabilities of a finite-state device.) In contrast, if $P_2 = \Sigma^* Q_2$ where $Q_2 = \{0^p : p \text{ is a positive prime integer}\}$, Q_2 is not regular but P_2 contains $\Sigma^* 00$ and any tape in $\Sigma^* Q_2$ is in $\Sigma^* 00$. Therefore $P_2 = \Sigma^* 00$ and is regular. Thus Q non-regular and $P = \Sigma^* Q$ may yield either a regular or non-regular P.

However, Paz and Peleg [13] showed that the regularity or non-regularity of an ultimate-definite event P is reflected in the canonical representation $\Sigma^* P^t$.

Theorem 4.2.2: (Paz and Peleg) If P is a regular ultimate-definite event with canonical representation $\Sigma^* P^t$, then P^t is regular.

This result, coupled with the observation that P^t being regular implies $\Sigma^* P^t$ is regular, establishes regularity of P^t as necessary and sufficient for regularity of $P = \Sigma^* P^t$.

A canonical form for reverse ultimate-definite events is now stated in terms of that of ultimate-definite events.

Definition 4.2.3: The canonical representation of a reverse ultimate-definite event $P = Q\Sigma^*$ is denoted $Q^t\Sigma^*$ and is defined to be the reverse of the canonical representation of P^r.

Thus if $P = Q\Sigma^*$, the canonical form of P is obtained as follows: $P^r = (Q\Sigma^*)^r = \Sigma^*\Sigma^r = \Sigma^*(Q^r)^t$, the last expression being the unique canonical representation for the ultimate-definite event P^r. The canonical representation of P, then, is $(\Sigma^*(Q^r)^t)^r = ((Q^r)^t)^r\Sigma^*$.

It is readily seen that uniqueness of the canonical representation $P^r = \Sigma^*(Q^r)^t$ of the ultimate-definite event P^r renders $((Q^r)^t)^r\Sigma^*$ the unique canonical representation of P, and also that $Q\Sigma^*$ is the canonical representation of P if and only if no tape in Q is a proper prefix of another tape in Q.

We now consider finite automata corresponding to regular events of the forms just discussed.

Definition 4.2.5: The finite automaton $A = (S, M, s_0, F)$ is ultimate-definite (reverse ultimate-definite) if and only if T(A) is ultimate-definite (reverse ultimate-definite).

4.3. Synchronization of Ultimate-Definite Automata

We have shown (Lemma 4.1.1 -- Theorem 4.1.5) the existence and characterization of synchronizing sequences for purely k-definite automata $A(\Sigma^* W)$, $W \subseteq \Sigma^k$. Recognizing that such automata are a special class of ultimate-definite automata, we turn now to an investigation of the latter class.

The proof of the theorem characterizing synchronizing sequences for purely k-definite automata made extensive use of the k-definite properties, i.e., of the facts that tapes were accepted or rejected on the basis of their last k symbols, and that the state of the corresponding automaton was determined by the last k inputs. The general proof, below, makes use of the canonical form for ultimate-definite events

P, and is valid whether the event P^t such that $P = \Sigma^* P^t$ is finite or infinite.

Theorem 4.3.1: If $\Sigma^* P^t$ is regular, then the tape x synchronizes $A(\Sigma^* P^t)$ if and only if x is not a prefix of a proper suffix of any tape y in P^t.

Proof: We first prove the forward implication, by contraposition. Suppose x is a prefix of a proper suffix of tape in P^t. Then there are tapes u and w ($u \neq \Lambda$) such that $uxw \in P^t$, whence $M(s_0, uxw) \in F$.

If $M(s_0, x) = M(s_0, ux)$, then $M(s_0, xw) = M(s_0, uxw) \in F$. This implies that some suffix v of xw is in P^t and, since $u \neq \Lambda$, $v \in P^t$ is a proper suffix of $uxw \in P^t$, contrary to the definition of P^t. This contradiction stems from taking $M(s_0, x) = M(s_0, ux)$; hence $M(s_0, x) \neq M(s_0, ux) = M(M(s_0, u), x)$, and x fails to synchronize A.

The reverse implication is again proved by contraposition. If x fails to synchronize A, then for some $s_i \in S$, $M(s_0, x) \neq M(s_i, x)$. Since A is reduced, there is a tape w_i such that $M(s_0, w_i) = s_i$, and there is a tape u (possibly empty) such that u distinguishes between $M(s_0, x)$ and $M(s_i, x)$. Thus one and only one of $M(s_0, xu)$ and $M(s_i, xu)$ is in F.

If $M(s_0, xu) \in F$, then some suffix of xu is in P^t. Therefore the same suffix of $w_i xu$ is in P^t, and $M(s_i, xu) = M(s_0, w_i xu) \in F$. Hence u fails to distinguish between $M(s_0, x)$ and $M(s_i, x)$, contrary to the selection of u. This rules out the case $M(s_0, xu) \in F$.

We are forced to conclude $M(s_0, xu) \notin F$ and $M(s_0, w_i xu) = M(s_i, xu) \in F$. Hence some suffix v of $w_i xu$ is in P^t, but no suffix of xu is in P^t. Therefore xu is a proper suffix of $v \in P^t$, and x is a prefix of a proper suffix of a tape in P^t. The proof is now complete.

Theorem 4.3.1 characterizes those sequences which will synchronize an ultimate-definite automaton. Is it possible for the set of synchronizing sequences so neatly described to be empty? That is, need $A(\Sigma^* P^t)$ be synchronizable?

If P^t is finite, with a longest tape in P^t having length k, then any tape whose length exceeds k-1 will do and there may be other, shorter synchronizing tapes. If P^t is infinite, $A(\Sigma^* P^t)$ exists and is not synchronizable if and only if $\Sigma^* P^t$ is regular and every tape in Σ^*

is contained in a proper suffix of a tape in P^t. This latter condition can indeed occur as is shown in the next example.

Example 4.3.1: Let t_1 = 1 1 0,

t_2 = 1 11 10 01 00,

t_3 = 1 111 110 101 100 011 010 001 000,

⋮

$t_j = 1 \cdot s_j$, where s_j is a tape of length $2^j \cdot j$ obtained by concatenating all tapes in Σ^j, the only stipulation being that the concatenation end in 10^j.

Let $Q = \{t_i : i = 1, 2, 3\ldots\}$, and let $U = \Sigma^* Q$. The representation $\Sigma^* Q$ is seen to be canonical and every tape in Σ^* is contained in many proper suffixes of tapes in Q.

It remains for us to show whether or not $\Sigma^* Q$ is regular or if a regular set exists having the property we built into Q. These question are resolved by the following theorem and corollary.

Theorem 4.3.2: All ultimate-definite automata are synchronizable.

Proof: We show that any two states of an ultimate-definite automaton can be merged and apply Lemma 3.3.1.

Consider any two states s_1 and s_2 of $A(\Sigma^* P^t) = (S, M, s_0, F)$. Since A is reduced, there are tapes x_1 and x_2 such that $M(s_0, x_1) = s_1$ and $M(s_0, x_2) = s_2$. Furthermore, there is a tape z_1 of length $\leq n-1$ such that $M(s_1, z_1) = M(s_0, x_1 z_1) \in F$ or $M(s_2, z_1) = M(s_0, x_2 z_1) \in F$ but not both. That is $M(s_{i_1}, z_1) \in F$ for $i_1 = 1$ or 2 but not both.

Then for some non-null suffix $x^1_{i_1}$ of x_{i_1}, $x^1_{i_1} z_1 \in P^t$. ($x^1_{i_1} = \Lambda$ implies some suffix of z_1 is in P^t, whence $M(s_1, z_1) \in F$ and $M(s_2, z_1) \in F$, contrary to selection of z_1.)

If $M(s_1, x^1_{i_1}) \neq M(s_2, x^1_{i_1})$ then, proceeding as above, there is a tape $z_2 (\ell(z_2) \leq n-1)$ such that $M(s_1, x^1_{i_1} z_2) \in F$ or $M(s_2, x^1_{i_1} z_2) \in F$ but not both. Denote the one in F by $M(s_{i_2}, x^1_{i_1} z_2)$. As before, for some non-null suffix $x^2_{i_2}$ of x_{i_2}, $x^2_{i_2} x^1_{i_1} z_2 \in P^t$.

Proceeding in this fashion, for each positive integer j such that $M(s_1, x^j_{i_j} \ldots x^2_{i_2} x^1_{i_1}) \neq M(s_2, x^j_{i_j} \ldots x^2_{i_2} x^1_{i_1})$ we obtain a corresponding

tape $x_{i_{j+1}}^{j+1} x_{i_j}^{j} \ldots x_{i_2}^{2} x_{i_1}^{1} z_{j+1} \in P^t$, with $\ell(z_{j+1}) \leq n-1$. Consider the list of such tapes.

$$
\begin{array}{llll}
 & & x_{i_1}^{1} & z_1 \\
 & x_{i_2}^{2} & x_{i_1}^{1} & z_2 \\
 & & \vdots & \\
 & x_{i_j}^{j} \cdots x_{i_2}^{2} & x_{i_1}^{1} & z_j \\
x_{i_{j+1}}^{j+1} & x_{i_j}^{j} \cdots x_{i_2}^{2} & x_{i_1}^{1} & z_{j+1} \\
 & & \vdots &
\end{array}
$$

This process cannot continue indefinitely for there are finitely many tapes z_j since $\ell(z_j) \leq n-1$; and no tape z_j can be used twice to generate tapes in the above list. (Otherwise for some $j < k$, $z_j = z_k$, and $x_{i_j}^{j} \ldots x_{i_1}^{1} z_j$ is a proper suffix of $x_{i_k}^{k} \ldots x_{i_j}^{j} \ldots x_{i_1}^{1} z_k$, which cannot occur in P^t.)

The above process terminates only when, for some integer r, $M(s_1, x_{i_r}^{r} \ldots x_{i_1}^{1}) = M(s_2, x_{i_r}^{r} \ldots x_{i_1}^{1})$. Such termination is assured.

We have shown that for any states s_1 and $s_2 \in S$, s_1 and s_2 can be merged. Thus by Lemma 3.3.1, $A(\Sigma^* P^t)$ is synchronizable.

Corollary to Theorem 4.3.2: There is no regular set Q such that

(1) no tape in Q is a suffix of another tape in Q, and

(2) every tape $x \in \Sigma^*$ is a prefix of a proper suffix of a tape in Q.

Proof: We show that a regular set Q satisfying 1 violates 2. If Q is regular and satisfies condition 1, then $P = \Sigma^* Q$ is regular, ultimate-definite and in canonical form. The reduced automaton $M(\Sigma^* Q)$ is synchronizable by Theorem 4.3.2. There is therefore a tape x which synchronizes A, and by Theorem 4.3.1 x is not a prefix of a proper suffix of a tape in Q. Hence Q fails to satisfy condition 2.

N. B. The set Q of Example 4. 3. 1 is thus revealed to be not regular.

4. 4. Synchronization of Reverse Ultimate-Definite Automata

Given a reverse ultimate-definite event $P = Q^t\Sigma^*$ in canonical form, the set Q^t may or may not be finite. In the former case it is clear that P is regular, while in the latter instance there is no such assurance. However, regularity of P is equivalent to regularity of Q^t. That is,

Lemma 4. 4. 1: The event $P = Q^t\Sigma^*$ in canonical form is regular if and only if Q^t is regular.

Proof: The reverse implication is immediate. The forward implication follows easily from Definition 4. 2. 1 and Theorem 4. 2. 1; verification is left to the reader.

Theorem 4. 4. 2: If $P = Q^t\Sigma^*$ is regular, then $A(P) = (S, M, s_0, F)$ has exactly one final state s_F. Furthermore s_F is a dead state.

Proof: Let s_F be a final state of A. There is a tape x such that $M(s_0, x) = s_F$; hence there are tapes w and y such that $x = wy$ and $w \in P^t$.

For any tape z, $xz = wyz \in P^t\Sigma^*$; hence $M(s_F, z) = M(s_0, xz) \in F$. That is, any state reachable from s_F is also a final state.

Since s_F was an arbitrary final state of A, no two final states of A are distinguishable by any tape z. Since A is reduced, there is but one final state s_F.

Furthermore, since $M(s_F, z) \in F$ for all tapes z, $M(s_F, z) = s_F$ for all $z \in \Sigma^*$. This shows s_F to be a dead state.

The next result characterizes synchronizable reverse ultimate-definite automata $A(Q^t\Sigma^*)$ in terms of their state sets.

Theorem 4. 4. 3: If $Q^t\Sigma^*$ is regular, $A(Q^t\Sigma^*)$ is synchronizable if and only if A has no non-final dead state.

Proof: The forward implication is immediate, since no two dead states can be merged and A has a final dead state. The reverse implication is established by showing that the (dead) final state s_F is reachable from every state in $S - \{s_F\}$.

For any state $t \in S - \{s_F\}$, if $s_F \notin r(t)$ then r(t) consists entirely of states from which s_F is inaccessible. Therefore no two states in r(t) are distinguishable, since no final state is in r(t) or accessible from a state in r(t). Thus r(t) is a singleton, i.e., r(t) = t. However this implies t is a non-final dead state, contrary to hypothesis; so $s_F \in r(t)$ for any $t \in S - \{s_F\}$.

The proof is completed with the observation that A can be synchronized by mapping occupied states in $S - \{s_F\}$ to s_F, until the occupied set if reduced to $\{s_F\}$.

We note in passing that Theorem 3.5.4 established the bound $\binom{n}{2}$ on the length of a shortest synchronizing sequence for $A(Q^t\Sigma^*)$, if A is synchronizable and has n states. Furthermore, Example 3.5.1 shows that this bound is minimal for reverse ultimate-definite automata; one need only construct an n-state machine in the manner suggested by the example, taking s_0 as the start state and s_D as the final state.

The next theorem establishes a criterion for synchronizability of $A(Q^t\Sigma^*)$, where Q^t is finite, in terms of tapes in Q^t. If A is synchronizable in such a case, the event $P = Q^t\Sigma^*$ is shown to be m-definite for some positive integer m (see Definition 2.4.4), and the synchronizing tapes of length less than k are characterized in a subsequent result.

Theorem 4.4.4: If $P = Q^t\Sigma^*$ and a longest tape in Q^t has length k, then A(P) is synchronizable if and only if every tape in Σ^k has a prefix in P^t.

Proof: The reverse implication is proved directly. If every tape in Σ^k has a prefix in Q^t, then every tape in Σ^k (and hence each tape of length exceeding k) is accepted by A. For any state s of A there is a tape x such that $M(s_0, x) = s$. For any tape z of length k, $\ell(xz) \geq k$; hence $xz \in P$. Therefore $M(s, z) = M(s_0, xz) = s_F$, the lone final state of A. Since s was arbitrarily chosen, z synchronizes A to s_F.

The forward implication is established by contraposition. If not every tape in Σ^k has a prefix in Q^t, there is a tape y of length k which has no prefix in Q^t. There is a tape x of length k in Q^t by hypothesis. Thus for any tape z, $M(M(s_0, x), z) = M(s_0, xz) \in F$, since

$x \in Q^t$; and $M(M(s_0, y)z) = M(s_0, yz) \notin F$, since no prefix of yz is in Q^t. (This is assured, since $\ell(y) = k$, no prefix of y is in Q^t and no tape longer than y is in k.) That is, for arbitrary $z \in \Sigma^*$, $M(M(s_0, x)z) \neq M(M(s_0, y), z)$, and z thus fails to synchronize A.

Corollary to Theorem 4.4.4: If Q^t is finite and a longest tape in Q^t has length k, and if $A(Q^t\Sigma^*)$ is synchronizable, then A is m-definite for some positive integer m.

Proof: Since any tape of length k synchronizes A to s_F, the state of A is uniquely determined by any tape x such that $\ell(x) \geq k$. By the remark following Definition 2.4.7, A is m-definite for some positive integer m, and the desired conclusion is obtained.

Thus if Q^t is finite and $A(Q^t\Sigma^*)$ is synchronizable, $Q^t\Sigma^*$ is in reality a definite event. The following result is of course an immediate consequence of the preceding corollary. The proof supplied, however, illuminates the structure of $P = Q^t\Sigma^*$ as a definite event.

Theorem 4.4.5: If Q^t is finite, then $A(Q^t\Sigma^*)$ is synchronizable if and only if there exist finite events U and V such that $Q^t\Sigma^* = U \bigcup \Sigma^* V$.

Proof: The reverse implication is an immediate consequence of the discussion following Definition 2.4.5.

The forward implication will be proved constructively. It was observed in the proofs of Theorem 4.4.4 and its corollary that $A(Q^t\Sigma^*)$ is synchronizable if and only if A accepts all tapes of length exceeding k-1, where k is the length of a longest tape in Q^t. Thus, $\Sigma^*\Sigma^k \subseteq Q^t\Sigma^*$. The tapes of length less than k accepted by A are of the form $Q_0(\Sigma^0 \bigcup \Sigma^1 \bigcup \ldots \bigcup \Sigma^{k-1}) \bigcup Q_1(\Sigma^0 \bigcup \ldots \bigcup \Sigma^{k-2}) \bigcup \ldots \bigcup Q_{k-1}(\Sigma^0) = \bigcup_{j=0}^{k-1} [Q_j(\bigcup_{i=0}^{m-j-1} \Sigma^i)]$, where $Q_j = Q^t \bigcap \Sigma^j$. Taking $U = \bigcup_{j=0}^{k-1} Q_j (\bigcup_{i=0}^{m-j-1} \Sigma^i)]$ and $V = \Sigma^k$ yields the desired result.

We have seen that, for finite $Q^t \subseteq \Sigma^*$ such that k is the length of a longest tape in Q^t, if $A(Q^t\Sigma^*)$ is synchronizable then any tape of length exceeding k-1 synchronizes A. The following example reveals

that shorter synchronizing tapes may exist, and the next result characterizes such short synchronizing tapes.

Example 4.4.1: Consider $A_1(Q^t\Sigma^*)$, where $Q^t = \{00,01,11,100,101\}$. A longest tape in Q^t has length 3. Since every tape in Σ^3 has a prefix in Q^t, A_1 is synchronizable.

Examination of the state diagram of A_1 (Figure 4.4.1) reveals the "short" synchronizing sequences 00, 01, and 11.

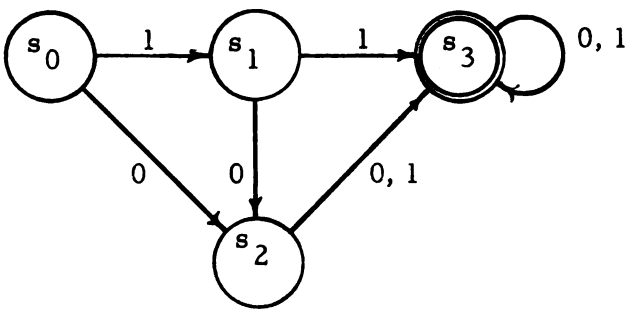

Figure 4.4.1. State diagram of A_1

It might seem on examining Example 4.4.1 that short synchronizing sequences must also be elements of Q^t. This is not the case, as is shown by the following example.

Example 4.4.2: Examination of $A_2(P^t\Sigma^*)$ where $P^t = \{0,100,101,11\}$ discloses the short synchronizing tapes 00, 01 and 11.

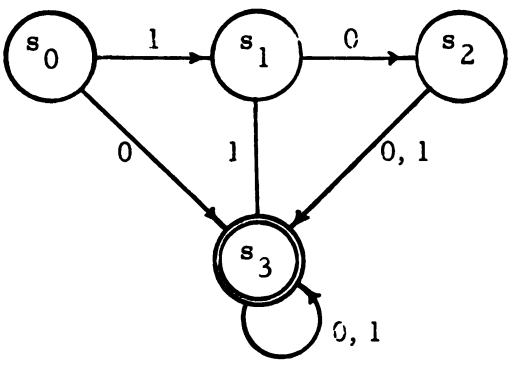

Figure 4.4.2. State diagram of A_2

We now describe the short synchronizing tapes of $A(Q^t\Sigma^*)$ for finite Q^t.

Theorem 4.4.6: If Q^t is finite, $A(Q^t\Sigma^*)$ is synchronizable, and a longest tape in Q^t has length k, then the tape z of length $m < k$ is a synchronizing tape for A if and only if $z \in Q^t\Sigma^*$ and for any tape y such that $\ell(y) < k-m$, $yz \in Q^t\Sigma^*$.

Proof: The forward implication is established by the following observations. If z synchronizes A, then for each state s of A, $M(s,z) = s_F$. For any tape y (hence in particular for y such that $\ell(y) < k-m$), $M(s_0, yz) = M(M(s_0, y)) = s_F$. Thus $z \in Q^t\Sigma^*$ and $yz \in Q^t\Sigma^*$.

To establish the converse, it is useful to recall that A accepts all tapes of length exceeding k-1. Thus for each $s_i \in S - \{s_F\}$ there is a tape y_i of length less than k such that $M(s_0, y_i) = s_i$. By hypothesis, if $\ell(y_i) < k-m$, then $M(s_i, z) = M(s_0, y_i z) = s_F$, since $y_i z \in Q^t\Sigma^*$. If $\ell(y_i) \geq k-m$, then $M(s_i, z) = M(s_0, y_i z) = s_F$ since $\ell(y_i z) \geq k$. Thus in any case, $M(s_i, z) = s_F$, and the proof is complete.

Acknowledgedly, since all synchronizing sequences for $A(Q^t\Sigma^*)$ send A into its final state s_F, from which it cannot escape, such tapes are of little worth in the practical sense. For finite Q^t we have, however, achieved a clear mathematical solution to questions of their existence and characterization.

4.5. Central-Definite Events and Automata

In this section we define a form of event which springs naturally from our consideration of synchronizing sequences, relate the new form to the ultimate-definite and reverse ultimate-definite events of Paz and Peleg [13], and obtain a canonical form for the event. Finally we give a procedure for constructing automata for a particular class of central-definite events.

Recall that, for any automaton A and any tape x, x synchronizes A if and only if the Myhill equivalence class $[x]_{\equiv}$ is a right zero of $\mathcal{M}(A)$, the monoid of A. (See section 3.2.) A consideration of the algebraic properties of right zeros leads to a new form of event.

Lemma 4.5.1: The set of right zeros of a semigroup S is a two-sided ideal in S.

Proof: Let $R \subseteq S$ be the set of right zeros of S. Then for each $r \in R$ and s, $t \in S$, $s(rt) = (sr)t = rt$; hence $rt \in R$. Trivially, $tr \in R$. Thus for any $t \in S$, $tR \subseteq R$ and $Rt \subseteq R$.

The preceding result prompts a further observation.

Lemma 4.5.2: If $A = (S, M, s_0, F)$ and r is a synchronizing sequence for A, then for any tapes x and y, xry is a synchronizing sequence for A.

Proof: For any state $s \in S$, $M(s, xry) = M(M(M(s, x), r), y)$ $= M(M(s_0, r), y) = M(s_0, ry)$. The first and third equalities follow from definitions; the second from the fact that $M(t, r) = M(s_0, r)$ for all $t \in S$, hence in particular for $t = M(s, x)$. Thus xry takes all states of A to $M(s_0, ry)$ which, by Definition 3.2.1, completes the proof.

Corollary to Lemma 4.5.2: If R is the set of all synchronizing sequences for A, then $R = \Sigma^* R \Sigma^*$.

Proof: The inclusion $\Sigma^* R \Sigma^* \subseteq R$ is immediate from Theorem 4.5.2. The observation that $\Lambda \in \Sigma^*$, hence $R = \Lambda R \Lambda \subseteq \Sigma^* R \Sigma^*$, completes the proof.

Definition 4.5.1: For any event $P \subseteq \Sigma^*$, the event $U = \Sigma^* P \Sigma^*$ is a central-definite event. If U is regular, A(U) is a central-definite automaton.

Thus by the preceding corollary, the set of all synchronizing sequences of a finite automaton is a central-definite event. The next result relates central-definite events to events previously considered.

Theorem 4.5.3: The event U is central-definite if and only if U is both ultimate-definite and reverse ultimate-definite.

Proof: If U is central-definite, there is an event $P \subseteq \Sigma^*$ such that $U = \Sigma^* P \Sigma^*$; hence $U = \Sigma^* (P\Sigma^*) = (\Sigma^* P)\Sigma^*$, the first equality showing U to be ultimate-definite, the second showing U to be reverse ultimate-definite.

Conversely, if U is ultimate-definite and reverse ultimate-definite, there are events P and Q such that $U = \Sigma^* P = Q\Sigma^*$. Then $\Sigma^* U = \Sigma^* (Q\Sigma^*) = \Sigma^* (\Sigma^* P) = \Sigma^* P = U$. The proof is completed by noting that $\Sigma^* Q \Sigma^* = U$.

We now show that $U = \Sigma^* P \Sigma^*$ has a canonical form analogous to those constructed for ultimate-definite and reverse ultimate-definite automata.

Definition 4.5.2: If U is central-definite, a representation of the form $U = \Sigma^* Q \Sigma^*$ is canonical if no tape in Q is a proper subtape of another tape in Q. A canonical representation of U will be denoted by $\Sigma^* Q^t \Sigma^*$.

Theorem 4.5.4: If U is central-definite, then U has a unique canonical representation.

Proof: If $U = \Sigma^* Q \Sigma^*$, we first show the existence of an event Q^t such that $U = \Sigma^* Q^t \Sigma^*$ is a canonical representation of U. We then show that the set Q^t is unique.

Let $Q_0 = Q \bigcap \Sigma^0$. For any integer $j \geq 1$,

let $Q_j = Q \bigcap \Sigma^j - \bigcup_{i=0}^{j-1} \left(\bigcup_{\ell=0}^{j-1} \Sigma P_i \Sigma^{j-\ell-i} \right)$.

That is, Q_j is the set of all tapes in Q of length j which do not have proper subtapes in Q.

Finally, let $Q^t = \bigcup_{j=0}^{\infty} Q_j$.

The construction of Q^t assures that no tape in Q^t is a proper subtape of another tape in Q^t. It must still be shown that $U = \Sigma^* Q^t \Sigma^*$.

If $x \in U$, then there are tapes u and w in Σ^* and v in Q such that x = uvw. Since $v \in Q$, either $v \in Q^t$ or some proper subtape of v is in Q^t. In either case, $v \in \Sigma^* Q^t \Sigma^*$, hence $uvw \in \Sigma^* Q^t \Sigma^*$. Thus $\Sigma^* Q \Sigma^* \subseteq \Sigma^* Q^t \Sigma^*$. The reverse inclusion is immediate, since $Q^t \subseteq Q$.

To see that Q^t is a unique set, suppose $\Sigma^* P^t \Sigma^*$ is another canonical representation of U. Without loss of generality, suppose the tape x is in P^t and not in Q^t. Then $x \in \Sigma^* P^t \Sigma^* = \Sigma^* Q^t \Sigma^*$, and there are tapes u and w in Σ^*, not both empty, and a tape $v \in Q^t$ such that x = uvw. Since no tape in P^t is a subtape of another tape in P^t, $v \notin P^t$.

Since $v \in Q^t$, $v \in \Sigma^* Q^t \Sigma^* = \Sigma^* P^t \Sigma^*$; so there are tapes t and z in Σ^* and $y \in P^t$ such that v = tyz.

Thus $x \in P^t$, $y \in P^t$, and $x = uvw = utyzw$. Since not both u and w are empty, y is a proper subtape of x, contrary to the supposition that $\Sigma^* P^t \Sigma^*$ is canonical. The supposition is thus false and the uniqueness of the set Q^t is established.

The unique canonical representations of $U = \Sigma^* Q \Sigma^*$, viewed as an ultimate-definite, reverse ultimate-definite and central-definite event, may be quite different from one another. For example, $U = \Sigma^* (10 \bigcup 011) \Sigma^*$ is a central-definite event in canonical form. Its canonical representations as an ultimate-definite and a reverse ultimate-definite event are $\Sigma^* [100^* (0^* \bigcup 1) \bigcup 0111^*]$ and $[(0^* \bigcup 1^*) 10 \bigcup 0^* 011] \Sigma^*$, respectively.

(Lest this example be misleading, we observe that in the canonical form $U = \Sigma^* P^t \Sigma^*$, P^t need not be finite. Consider $P^t = 1\ 0^* 11$.)

The three possible canonical representations of $U = \Sigma^* Q \Sigma^*$ are not totally unrelated, however. Their relationship is brought out in the next two results.

Theorem 4.5.5: Given the event $U = \Sigma^* Q \Sigma^*$, the tape x is in Q and has no proper subtape in Q if and only if $x \in U$ and has no proper subtape in U.

Proof: If $x \in Q$ and has no proper subtape in Q, it is clear that $x \in U$. If x has a proper subtape $u \in U$, then there are tapes w and v, not both empty, such that $x = wuv$. Since $u \in U$, there is a tape $q \in Q$ such that q is a subtape of u. Since w and v are not both empty, q is a proper subtape of x, contrary to hypothesis. Thus x has no proper subtape in U, and the desired forward implication is obtained.

Conversely, if $x \in U$ and has no proper subtape in U, then in particular x has no proper subtape in Q. Furthermore, $x \in Q$; for if not, there are tapes w and y, not both empty, and a tape $q \in Q$ such that $x = wqy$. The tape $q \in U$ is then a proper subtape of x, contrary to hypothesis.

Theorem 4.5.6: If the canonical representations of $U = \Sigma^* Q \Sigma^*$ as a reverse ultimate-definite, central-definite, and ultimate-definite event, respectively, are $P^t \Sigma^*$, $\Sigma^* Q^t \Sigma^*$, and $\Sigma^* R^t$, then $Q^t = P^t \bigcap R^t$.

Proof: As a consequence of Theorem 4.5.5, x is an element of Q^t if and only if $x \in U$ and x has no proper subtape in U. Results analogous to Theorem 4.5.5 for ultimate-definite events yield the following.

The tape x is in P^t if and only if $x \in U$ and x has no proper prefix in U, and $x \in R^t$ if and only if $x \in U$ and x has no proper suffix in U.

It is thus apparent that $Q^t \subseteq P^t \cap R^t$. The reverse inclusion is established by contraposition. Consider any tape $y \in U$. If $y \notin Q^t$, then y has a proper subtape in U, and there exist tapes u, v and w such that $v \in U$, not both u and w are empty, and $y = uvw$. Also, both $uv \in U$ and $vw \in U$.

If $w \neq \Lambda$, then y has a proper prefix in U; if $v \neq \Lambda$, then y has a proper suffix in U. Thus either $y \notin P^t$ or $y \notin R^t$; hence $y \notin P^t \cap R^t$. That is, $y \notin Q^t$ implies $y \notin P^t \cap R^t$. The proof is now complete.

Theorem 4.5.6 shows that, of the possible canonical representations of $U = \Sigma^* Q \Sigma^*$, the central-definite one is most compact in terms of tapes which must be specified.

Since $U = \Sigma^* Q \Sigma^*$ is ultimate-definite, the synchronizing tapes for A(U) have been given one characterization in Theorem 4.3.1. There is a simpler description if U is viewed as central-definite.

Theorem 4.5.7: If $U = \Sigma^* Q^t \Sigma^*$ is regular, then the tape x synchronizes $A(U) = (S, M, s_0, F)$ if and only if x has a subtape in Q^t.

Proof: Since U is ultimate-definite, A(U) is synchronizable. Since U is reverse ultimate-definite, A(U) has a single, dead final state s_F; and for any tape x which synchronizes A(U), $M(S, x) = s_F$.

Thus if x synchronizes A(U), then $M(s_0, x) = s_F$ and $x \in U$. Therefore x has a subtape in Q^t. Conversely, if x has a subtape in Q^t, and $s \in S$, then there is a tape y such that $M(s_0, y) = s$; hence yx has a subtape in Q^t, $yx \in U$, and $M(s, x) = M(s_0, yx) = s_F$. Since s is arbitrary, $M(S, x) = s_F$. The proof is now complete.

Of special interest are events of the form $U = \Sigma^* Q^t \Sigma^*$ where Q^t is finite. Such events are regular; and if a longest tape in Q^t has length k, then the membership or non-membership in $\Sigma^* Q^t \Sigma^*$ of a tape can be ascertained by examining all subtapes having length k.

For example, if $Q^t = \{1111\}$, if is only necessary to scan y taking in four (or more) symbols at a time to see if y belongs to U.

Definition 4.5.3: If $U = \Sigma^* Q^t \Sigma^*$ and a longest tape in Q^t has length k, U is a central k-definite event. If $Q^t \subseteq \Sigma^k$, U is a central purely k-definite event. A(U) is correspondingly said to be a central k-definite or central purely k-definite automaton.

We close the present section with a procedure for constructing a central k-definite automaton $A(\Sigma^* Q^t \Sigma^*) = (S, M, s_0, F)$. The scheme is an adaptation of the prefix automaton construction of Perles, Rabin and Shamir [14].

Given $U = \Sigma^* P^t \Sigma^*$, let $S = \{[\Lambda]\} \bigcup \{[x] : x \text{ is a proper prefix of a tape in } P^t\} \bigcup \{s_F\}$. Take $s_0 = [\Lambda]$ and $F = \{s_F\}$. Let the next-state function M be defined as follows: for each state $[x] \neq s_F$ and each symbol $\sigma \in \Sigma$,

$$M([x], \sigma) = \begin{cases} [y] \neq s_F & \text{if and only if y is the longest prefix of some } z \in P^t \text{ which is a suffix of } x\sigma, \text{ and } y \notin P^t. \\ s_F & \text{if and only if } x\sigma \in P^t. \end{cases}$$

and let $M(s_F, \sigma) = s_F$.

Example 4.5.1: If $U = \Sigma^*(10 \bigcup 011)\Sigma^*$, then $S = \{[\Lambda], [0], [1], [01], s_F\}$, and M is specified in Table 4.5.1. The state diagram of A(U) is given in Figure 4.5.1.

Table 4.5.1. Transition function of $A(\Sigma^*(10 \bigcup 011)\Sigma^*)$

M	0	1
$[\Lambda]$	[0]	[1]
[0]	[0]	[01]
[1]	s_F	[1]
[01]	s_F	s_F
s_F	s_F	s_F

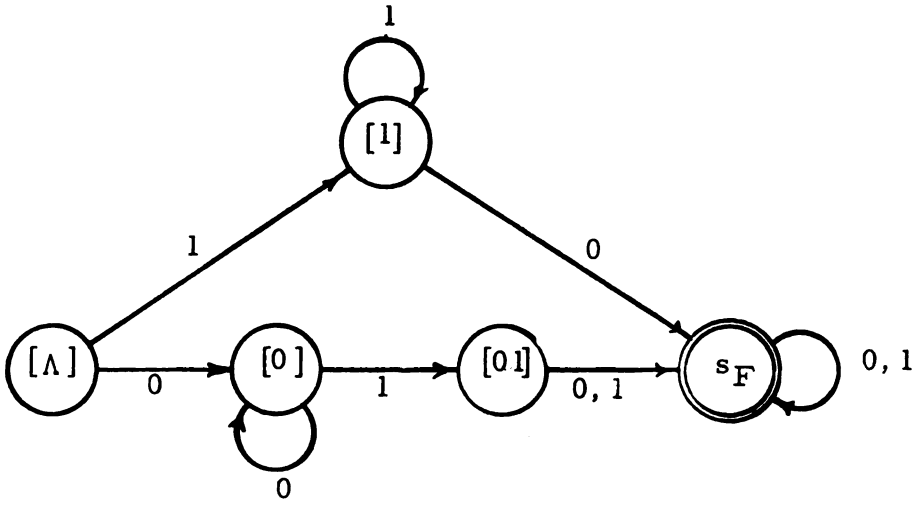

Figure 4.5.1. State diagram of $A(\Sigma^*(10 \cup 011)\Sigma^*)$

5. TOPICS IN AUTOMATON DECOMPOSITION

5.1. Introduction

Several authors have attacked the problems of expressing the structure and function of a finite automaton in terms of component devices which are in some way less complicated than the original machine. One such measure of a machine's complexity is its number of states.

Instead of concentrating wholly on fragmenting or decomposing automata into devices each of which has fewer states than the original, some writers have placed constraints on the kinds of component machines to be allowed, or on the ways in which they may be interconnected. For example, Krohn and Rhodes [7] and Zeiger [20] described techniques for decomposing an automaton A into a set of automata $K_1, K_2, \ldots, K_r$ wherein each machine K_i is one of several canonical types (to be further discussed in Section 5.3) and such that the output of K_i is allowed to act as part of the input to K_j if and only if $i < j$. There is thus a one-way flow of information from machines with lower to machines with higher indices. Such an arrangement is called a cascade of automata.

Hartmanis and Stearns [5], on the other hand, described a decomposition procedure in which any component machine may send its output to any other component machine, resulting in a network of automata. (A cascade is thus a special network.) The procedure yields component machines each having strictly fewer states than the original machine. Figure 5.1.1 shows a cascade and a network which is not a cascade. As suggested in the drawings, the input to the net may be supplied to any or all of the component machines.

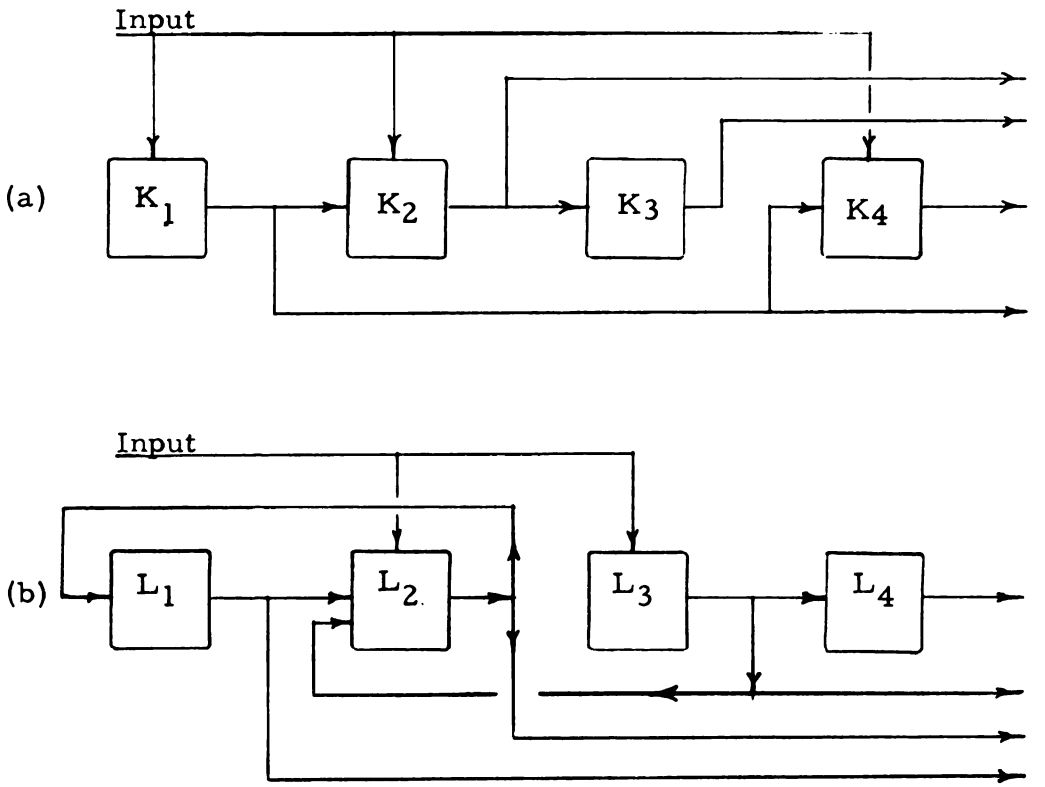

Figure 5.1.1. A cascade of automata (a) and a network which is not a cascade (b)

The results of the present chapter were obtained while investigating cascade decompositions of the classes of automata treated in Chapter 4. Some pertain directly to definite automata, while others are of a more general nature.

Zeiger's procedure, correct in concept, has undergone several modifications to remedy errors in the rather intricate details. Some faults persist in that author's most recent exposition [21]. Before exhibiting our original results we review that exposition, point out the errors, and suggest corrections. The balance of the chapter is devoted to general results concerning the ways in which permutations of states in a given automaton are reflected in its Zeiger decomposition and to results on the decomposition of definite automata.

5.2. State Behavior Simulation and Decomposition

Before discussing the decomposition procedure we formalize several ideas introduced in Section 5.1 and establish the notation to be used in the remainder of the present chapter. Conforming to Zeiger [21], the notation is somewhat different from that of preceding chapters, and is used because it was designed to efficiently display the necessary details. The material of the present section is for the most part due to Zeiger [20], [21]. We review the essentials and elaborate where needed in order to provide the background for subsequent results and to correct several errors.

Definition 5.2.1: Given a finite automaton M, let Q_M denote its set of states, I_M denote the input alphabet of M, and S_M denote the semigroup of all transformations of Q_M induced by nonempty sequences of symbols from I_M. Let $\tilde{S}_M = T_M \cup S_M$, where T_M consists of the identity transformation on Q_M and all transformations which map Q_M to a single state. If x is an input sequence, let x_M be the transformation induced by x on Q_M. Note that, if x and y are tapes, then $(xy)_M = y_M x_M$. That is, for each $s \in Q_M$, $(xy)_M(s) = y_M(x_M(s))$.

Since the concern of the present chapter is with state behavior, no reference need be made to a starting state or a set of final states of M.

Example 5.2.1: If M is the automaton whose state diagram appears in Figure 5.2.1, then $Q_M = \{a, b, c\}$ and $I_M = \{1, 2\}$. S_M contains, among others, the function $f = (12)_M$, with $(12)_M(a) = c$, $(12)_M(c) = a$, and $(12)_M(b) = b$.

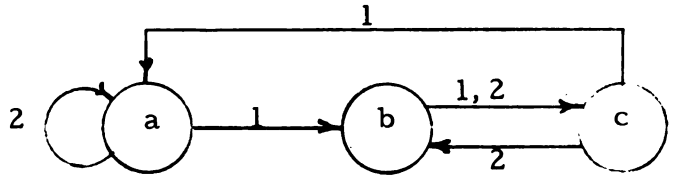

Figure 5.2.1. State diagram of M

The concept of an automaton N being able to simulate an automaton M is essential to any decomposition theory. In non-technical terms, we say that N simulates the state behavior of M if there is a way to view the states of N as if they were states of M and if, on so doing, we see that the transitions of N under all inputs are the same as those of M.

Definition 5.2.2: The finite automaton N simulates the state behavior of the finite automaton M under the correspondence Z_N mapping Q_N onto Q_M if $I_N = I_M$ and, for each $s \in Q_N$ and $x \in I_N = I_M$, $Z_N(x_N(s)) = x_M(Z_N(s))$.

Example 5.2.2: The automaton N of Figure 5.2.2 simulates the state behavior of automaton M, Figure 5.2.2 under the correspondence Z_N shown in Table 5.2.1.

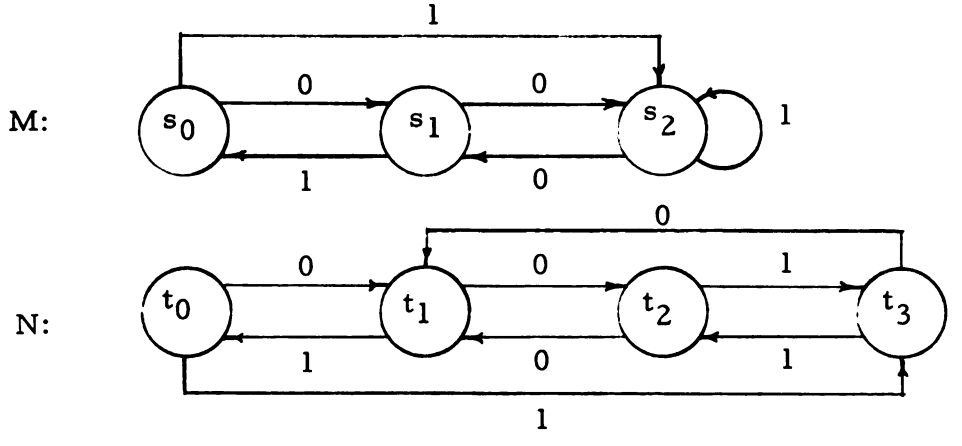

Figure 5.2.2. Automata M and N such that N simulates the state behavior of M

Table 5.2.1 The function Z_N mapping Q_N onto Q_M so that N simulates the state behavior of M

$t_i \in Q_N$	$Z_N(t_i)$
t_0	s_0
t_1	s_1
t_2	s_2
t_3	s_2

The essential features of the decomposition procedure will now be discussed. A fundamental notion is that of a cover of the state set of an automaton.

Definition 5.2.3: Given a finite automaton M, a collection $C = \{b_1, b_2, \ldots, b_r\}$ of subsets of Q_M is a cover of Q_M if $\bigcup_{i=1}^{r} b_i = Q_M$ and for each $x_m \in S_M$ and $b_i \in C$ there is a $b_j \in C$ such that $x_M(b_i) \subseteq b_j$. A cover C of $Q_M = \{s_1, \ldots, s_n\}$ will be denoted by listing the states in the cover elements in groups separated by commas; e.g., $C = \{s_1 s_3, s_2 s_3 s_4\}$.

That is, C is a collection of "blocks" of states of M which exhaust Q_M, such that each input sequence takes elements of C into elements of C. Thus C mirrors, in a coarse manner, the actual state behavior of M. If an element of C is known to contain the present state of M, then it is possible to specify an element of C containing the next state of M following any input x. Indeed, it is possible to construct an automaton N which, for each cover element in C and input symbol x, specifies a next cover element in C.

Definition 5.2.4: Given a finite automaton M and a cover C of Q_M, the automaton N tells where M is in C if $I_N = I_M$ and there is a mapping Z_N of Q_N onto C such that, for every $x \in I_M$ and $s \in Q_N$, $x_M(Z_N(s)) \subseteq Z_N(x_N(s))$.

Example 5.2.3: Let M be the automaton whose state diagram is given in Figure 5.2.3. $C_1 = \{ac, bd\}$ is a cover of Q_M, as is $C_2 = \{abc, bd\}$. Trivial covers are $I = Q_M$ and the discrete cover $O = \{a, b, c, d\}$, which are respectively the coarsest and finest covers of Q_M. A collection of subsets which exhausts Q_M but fails to be a cover is $\{ab, cd\}$, for $1_M(ab) = bd$ is not contained in either $\{ab\}$ or $\{cd\}$. It is not necessary for all elements of a cover to be of the same size. A cover element may map onto or properly into another cover element.

Table 5.2.2 specifies a two-state machine N which tells where M is in C_1. The correspondence Z_N mapping Q_N onto C_1 is also shown.

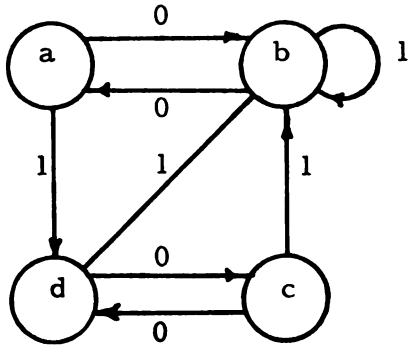

Figure 5.2.3. State diagram of M

Table 5.2.2. Transition table for N, which tells where M is in C_1

$s \in Q_N$	$Z_N(s)$	$0_N(s)$	$1_N(s)$
s_1	ac	s_2	s_2
s_2	bd	s_1	s_2

If the automaton M has n states, there are n^n ways in which a given input symbol x can map Q_M into Q_M. The mappings allowed in the machines of a cascade decomposition are sharply restricted in the following sections, to types defined below.

Definition 5.2.5: The input symbol $x \in I_M$ is a permutation input to machine M if x_M permutes Q_M, and a resetting input or a reset if

$x_M(Q_M)$ is a singleton. M is a permutation-reset machine if every input to M is either a permutation input or a reset, and is a permutation machine if every input is a permutation input.

5.3. Zeiger's Decomposition Procedure

Examination of Definitions 5.2.2 and 5.2.4 reveals that, if C is the discrete cover of Q_M and N tells where M is in C, then N simulates the state behavior of M. Briefly stated, given machine M which is not a permutation-reset machine, Zeiger's procedure [21] generates a sequence of successively finer covers $C_1, C_2, \ldots, C_r$ and a corresponding sequence of permutation-reset machines $L_1, L_2, \ldots, L_r$ such that

(a) Each machine L_i has strictly fewer states than M has.

(b) A cascade N_j (specified by the decomposition procedure) of $L_1, \ldots, L_j$ tells where M is in C_j, by means of the mapping $Z_{N_j}: Q_{N_j} \longrightarrow C_j$.

(c) The group of permutations in S_{L_i} is a homomorphic image of the group of permutations induced on some subset of Q_M by elements of S_M.

(d) C_r is the discrete cover of Q_M. Thus N_r simulates the state behavior of M.

The decomposition procedure owes much of its intricacy to the constraints on the L_i and the requirements (a) -- (d) above.

The proof of Proposition 1 in Zeiger [21] describes the first step in the decomposition process, whereby one generates a cover C of the state set Q_M and obtains a permutation-reset machine N that tells where M is in C. The construction is as follows:

(1) Let $C = \max [\{w(Q_M): w \in \tilde{S}_M\} - Q_M]$. (If B is a collection of sets, max (B) is the set of elements of B which are maximal with respect to set inclusion.)

(2) Let $Q_N = C$.

(3) Let Z_N, the correspondence between states of N and elements of C, be the identity mapping.

(4) For any input $x \in I_N = I_M$, let x_N be defined by:

a. If $x_M(Q_M) \subsetneq Q_M$, then select any $R' \in C = Q_N$ such that $x_M(Q_M) \subseteq R'$, and let $x_N(b) = R'$ for each $b \in Q_N$.

b. If $x_M(Q_M) = Q_M$, then x_M permutes the blocks of C; let $x_N(b) = x_M(b)$ for each $b \in Q_N$.

It is thus seen that (4)-a produces reset mappings on Q_N, while (4)-b produces permutations on Q_N. Furthermore, the group of permutations on Q_N is a homomorphic image of a subgroup of S_M; and N does tell where M is in C.

The proof of Proposition 2 tells how this process may be continued. Step 1 of the proof tells how, given a cover C of Q_M and a machine K which tells where M is in C, a finer cover C' of Q_M is obtained. Step 2 details the construction of a corresponding permutation-reset machine L and a cascade N of K and L such that N tells where M is in C', via mapping Z_N: Q_N onto C'. (See figure 5.3.1.) Steps 1 and 2 are shown following the figure.

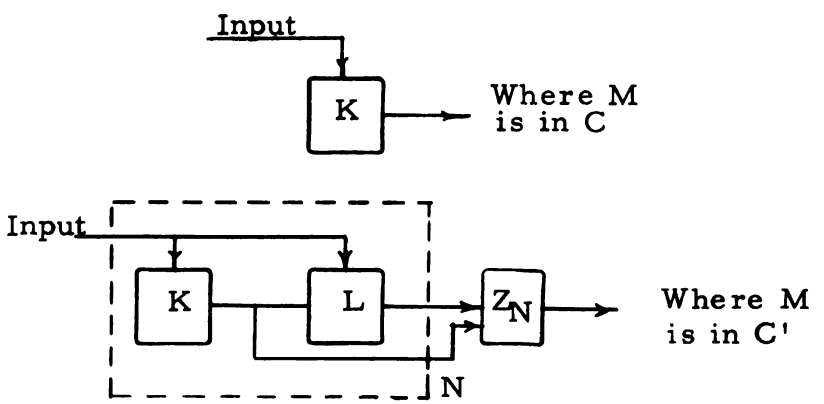

Figure 5.3.1. Diagram of the inductive step in Zeiger's decomposition process

Step 1: Given a cover C of Q_M, two elements of C are similar if each is the image of the other under appropriate transformations from S_M. (Similarity is an equivalence relation on the elements of C, and partitions C into similarity classes, each of which contains

cover elements of a single cardinality. There may be several similarity classes whose cover elements have the same cardinality.) An element $b \in C$ is initial in C if b is similar to every element of C which maps onto b under some transformation from $\tilde{S}_M$. A refined cover C' is obtained by replacing each b in one initial similarity class $D \subseteq C$ with $R_b = \max[\{w(R) : w(R)$ is a proper subset of b, $w \in \tilde{S}_M$, and $R \in C\}]$.

Step 2: Pick one $q \in D$. For any $p \in D$, there are transformations v_p^q and v_q^p in S_M such that v_p^q maps p onto q, v_q^p maps q onto p, and the composition of v_p^q and v_q^p is the identity transformation on q. We indicate this by

$$q \xrightarrow{v_q^p} p \xrightarrow{v_p^q} q \quad (\mathrm{id}).$$

Let the state set of L be R_q, the set of elements which replaced q in passing from C to C'. Let $I_L = Q_K \times I_K$. The state set of N, which is a cascade of K and L, is $Q_N = Q_K \times Q_L$; and $I_N = I_K = I_M$. The function Z_N, mapping Q_N onto C', is defined by:

For each $(p, r) \in Q_N$,

(1) If $p \notin D$, then $Z_N(p, r) = p$.

(2) If $p \in D$, then $Z_N(p, r) = v_q^p(r)$.

The state transitions of N, and hence of L, are defined as follows:

For each $x \in I_N$ and $(p, r) \in Q_N$, define $x_N(p, r) = (s, t)$ by:

(1) $s = x_K(p)$.

(2) If $s \in C \cap C'$, let $t = r$.

(3) If $p \in C \cap C'$, and $s \in D$, let $t = v_s^q x_M(p)$.

(4) If $p \in D$ and $s \in D$, let $t = v_s^q x_M v_q^p(r)$.

Iteration of Steps 1 and 2 yields the aforementioned sequence of successively finer covers $C_1, C_2, \ldots$ and corresponding permutation-reset machines $L_1, L_2, \ldots$ such that the cascade N_j of $L_1, \ldots, L_j$ tells where M is in C_j. Finally the discrete cover C_r is obtained, with a cascade N_r of N_{r-1} and L_r telling where M is in C_r. Machine N_r is a decomposition of M into a cascade of permutation-

reset machines. These machines may be further decomposed into the canonical machines of Krohn and Rhodes [7], each of which is either a two-state permutation-reset machine with no non-identity permutations or a permutation machine whose group is simple and is a composition factor of a subgroup of S_M.

5.4. Flaws in the Decomposition Procedure

To show that the specification of state transitions of L (items (1)-(4), Step 2, Section 5.3) makes L a permutation-reset machine, Zeiger [21] notes that (2) produces the identity permutation on the states of L. He claims that (3) produces resets. It should be observed, however, that $v_s^q x_M(p)$ may be a proper subset of one of the blocks of states with which q was replaced in obtaining C'. This is illustrated in Example 5.5.1.

Also, it is claimed that (4) always produces permutations on the state set of L "since $v_s^q x_M v_q^p$ permutes q." The mappings v_s^q and v_q^p are indeed one-to-one transformations from s onto q and q onto p, respectively, but x_M may or may not be a one-to-one mapping of $p = v_q^p(q)$ onto s. This is also shown in example 5.5.1. Consequently, in a situation where (4) applies, a permutation of states of L is not forced. As in (3), such exceptions occur only when $x_M(p)$ is properly contained in s. Both of the situations just described may be resolved by applying the following lemma.

Lemma 5.4.1: If $p \in C$ and $x_M(p)$ is properly contained in s, then there is a state $u \in Q_L$ (not necessarily unique) such that $v_s^q x_M(p)$ is contained in u.

Proof: If $x_M(p)$ is properly contained in s, then $v_s^q x_M(p)$ is properly contained in q, since v_s^q maps s one-to-one onto q. Therefore $v_s^q x_M(p) \in \{w(R) : R \in C,\ w \in \tilde{S}_M,\ \text{and } w(R) \subsetneq q\}$. The proof is completed by recalling that the maximal elements of this last set are taken to be the states of L.

Thus if $x_M(p)$ is properly contained in s, in (3) we may always select $t = u \in Q_L$, since $v_s^q x_M(r) \subseteq u$. In (4), if $x_M(p)$ is properly contained in s, then for any state $r \in Q_L$ we may let t = u, since $v_s^q x_M v_q^p(r) \subseteq v_s^q x_M(p) \subseteq u$. Note that in the latter case, (4) produces a reset of L to state t rather that a permutation on Q_L.

Finally, we note that in (1), (2) and (1)--(4) of Step 2, the states of K are also treated as elements of C, being interpreted in the latter sense in the construction of N and L. Steps (1)--(4), as given above, imply that the next-state assignment of L may be made using only the present input to K and the information "where M is in C" supplied by K. In applying (2)--(4), this state assignment is dependent upon knowing the next cover element, s, to be specified by K, and no problem arises so long as s is uniquely specified by the present cover element p and transformation x_K.

However, in following the above procedure for the automaton M' of Example 5.5.2 we encounter a situation where K specifies at time t that M' is in element B_i of cover C, receives an input x, and may then specify at time t + 1 that M' is in element B_k or B_l of C ($B_k \neq B_l$), depending on the exact state of K at time t. This is not totally unexpected if we note that in general many states of K may correspond to a single element B_i of C. Among these, some may map under input x to states corresponding to B_k, others to states corresponding to B_l; and input x to the original machine M' may indeed map B_i into $B_k \cap B_l$. Thus, either B_k or B_l is an accurate description of "where M' is in C". In such a case, the next-state assignment in L requires state of K information rather than element of C information from K.

5.5. Examples Illustrating Flaws in the Decomposition Procedure

Our general scheme in the examples is as follows:

1. Obtain a cover C_1 of Q_M and a permutation-reset machine K which tells where M is in C_1. Let $K = N_1$, and define $Z_K = Z_{N_1}$ to be the identity map from Q_K to C_1.

2. Having cover C_i not consisting entirely of singletons and a machine N_i telling where M is in C_i, obtain a refined cover C_{i+1}, a permutation-reset machine L_{i+1}, and a machine N_{i+1} as a cascade of N_i and L_{i+1}, so that N_{i+1} tells where M is in C_{i+1}. Let D_i denote the initial similarity class of elements of C_i replaced in obtaining C_{i+1}.

We leave it to the reader to verify that the constructions in the examples are carried out according to the procedures of Section 5.3. In both examples, the decomposition is incomplete, being carried only far enough to illustrate the observations made in Section 5.4.

Example 5.5.1: Let M be the automaton whose transition table is shown in Table 5.5.1. Then $C_1 = \{abc, bcde\}$. Table 5.5.2 shows the transitions of $K = N_1$ which tells where M is in C_1.

Table 5.5.1. State transitions of M

M	0	1	2
a	a	b	c
b	b	c	d
c	c	d	d
d	c	e	d
e	c	e	d

Table 5.5.2. Transitions of $K = N_1$

K	0	1	2
abc	abc	bcde	bcde
bcde	abc	bcde	bcde

To obtain C_2, take $D_1 = \{bcde\}$. Replace D_1 with $\{bcd, cde\}$. Then $Q_{L_2} = \{bcd, cde\}$, and $Q_{N_2} = Q_K \times Q_{L_2}$.

To obtain the transition table of N_2 we show the states of N_2 as ordered pairs (p, r) where $p \in Q_K$ and $r \in Q_{L_2}$. If N_2 is in state (p, r) and input x is received, the first component of the next state of N_2 is $x_K(p)$. This accounts for all first components in the body of Table 5.5.3. To obtain second components, we follow parts (2), (3) and (4) of Step 2 of Zeiger's construction. Noting that $bcde \xrightarrow{\Lambda_M} bcde \xrightarrow{\Lambda_M} bcde$ (id) where Λ_M is the identity mapping

produced by the null input sequence, we obtain the second components in the table body in the manner described following Table 5.5.3.

Table 5.5.3. State transition table for N_2. Starred entries required modification of Zeiger's procedure.

N_2	0	1	2
(abc, bcd)	(abc, bcd)	(bcde, bcd)	(bcde, bcd*)
(abc, cde)	(abc, cde)	(bcde, bcd)	(bcde, bcd*)
(bcde, bcd)	(abc, bcd)	(bcde, cde)	(bcde, bcd*)
(bcde, cde)	(abc, cde)	(bcde, cde*)	(bcde, bcd*)

By (2), the second components in the 0 column are unchanged from the corresponding entries in the present state column.

By (3), 1_{N_2}(abc, bcd) = (bcde, t_1) where $t_1 = \Lambda_M 1_M$(abc) = bcd. Similarly, 1_{N_2}(abc, cde) = (bcde, bcd). In the same manner, 2_{N_2}(abc, bcd) = (bcde, t_2) where $t_2 = \Lambda_M 2_M$(abc) = cd. But cd $\notin Q_{L_2}$. This illustrates the first difficulty discussed in Section 5.4. Observing that $\{cd\} \subseteq \{bcd\}$ ($\{cde\}$ would do as well), we let t_2 = bcd. In like fashion, we arrive at 2_{N_2}(abc, cde) = (bcde, bcd).

By (4), 1_{N_2}(bcde, bcd) = bcde, t_3) where $t_3 = \Lambda_M 1_M \Lambda_M$(bcd) = cde. Also, 1_{N_2}(bcde, cde) = (bcde, t_4) where $t_4 = \Lambda_M 1_M \Lambda_M$(cde) = de. However, de $\notin Q_{L_2}$, illustrating the second difficulty discussed in Section 5.4. Noting that 1_M(bcde) = cde, we select t_4 = cde. The entries 2_{N_2}(bcde, bcd) = (bcde, bcd) and 2_{N_2}(bcde, cde) = (bcde, bcd) are similarly arrived at. The table for L_2 can be obtained from the table for N_2 and is shown in Table 5.5.4.

Table 5.5.4. State transition table of L_2

L_2	(abc, 0)	(abc, 1)	(abc, 2)	(bcde, 0)	(bcde, 1)	(bcde, 2)
bcd	bcd	bcd	bcd	bcd	cde	bcd
cde	cde	bcd	bcd	cde	cde	bcd

Example 5.5.2: Let M' be the automaton depicted in Table 5.5.5. Cover $C_1 = \{abc, abd, acd, bcd\}$. Table 5.5.6 shows the state transitions of $K'_1 = N'_1$ which tells where M' is in C_1.

Table 5.5.5. State transition table of M'

M'	0	1
a	b	b
b	c	c
c	c	d
d	d	a

Table 5.5.6. State transition table of $K' = N'_1$

K'	0	1
abc	bcd	bcd
abd	bcd	abc
acd	bcd	abd
bcd	bcd	acd

Since all elements of C_1 are similar, $D_1 = \{abc, abd, acd, bcd\}$. Note that:

$$abc \xrightarrow{\Lambda_{M'}} abc \xrightarrow{\Lambda_{M'}} abc \text{ (id)}, \quad abc \xrightarrow{111_{M'}} abd \xrightarrow{1_{M'}} abc \text{ (id)},$$

$$abc \xrightarrow{11_{M'}} acd \xrightarrow{11_{M'}} abc \text{ (id)}, \text{ and } abc \xrightarrow{1_{M'}} bcd \xrightarrow{111_{M'}} abc \text{ (id)}.$$

Replace {abc} with {ab, ac, bc}, {abd} with {ab, ad, bd}, {acd} with {ac, ad, cd}, and {bcd} with {bc, bd, cd}, obtaining $C_2 = \{ab, ac, ad, bc, bd, cd\}$. Take $Q_{L'_2} = \{ab, ac, bc\}$; and then $Q_{N'_2} = Q_{N'_1} \times Q_{L'_2}$. Part of the correspondence $Z_{N'_2}$ mapping $Q_{N'_2}$ onto C_2 is shown in Table 5.5.7. Sample computations in obtaining $Z_{N'_2}$ are the following: Since $abc \xrightarrow{\Lambda_{M'}} abc \xrightarrow{\Lambda_{M'}} abc$ (id), $Z_{N'_2}((abc, ad)) = \Lambda_{M'}(ab) = ab$; and since $abc \xrightarrow{111_{M'}} abd \xrightarrow{1_{M'}} abc$ (id), $Z_{N'_2}((abd, ab)) = 111_{M'}(ab) = ad$.

Table 5.5.8 depicts some of the state transitions of N'_2. A sample computation is: $0_{N'_2}((abd, ab)) = (bcd, ac)$. The first component results from $0_{K'}(abd) = bcd$. The second follows from the fact that $abc \xrightarrow{111_{M'}} abd \xrightarrow{1_{M'}} abc$ (id), $abc \xrightarrow{1_{M'}} bcd \xrightarrow{111_{M'}} abc$ (id), and $111_{M'} 0_{M'} 111_{M'}(ab) = ac$.

Table 5.5.7. Partial list of values of function $Z_{N'_2}$

State of N'_2: (K' state, L'_2 state)	Corresponding element of C_2
(abc, ab)	ab
(abd, ab)	ad
(bcd, ab)	bc
(bcd, ac)	bd
(abc, bc)	bc
(abd, bc)	ab
.	.
.	.
.	.

Table 5.5.8. Partial table of state transitions of N'_2

N'_2		0		1	
[ab]	(abc, ab)	[bc]	(bcd, ab)	[bc]	(bcd, ab)
[ad]	(abd, ab)	[bd]	(bcd, ac)	[ab]	(abc, ab)
[ab]	(abd, bc)	[bd]	(bcd, ac)	[bc]	(abc, bc)
	.		.		.
	.		.		.
	.		.		.

Elements of C_2 corresponding to states of N'_2 are shown in brackets in Table 5.5.8. Note that N'_2 may say "M' is in [ab]", receive an input 0, and then say "M' is in [bc]" or "M' is in [bd]", depending on the exact state of N'_2 before receiving the 0. This illustrates the third difficulty described in Section 5.4.

5.6. Suggested Remedies

In the light of the observations of Section 5.4, we suggest that Step 2 of Section 5.3 be modified to read as follows:

> . . . Let Z_N map Q_N onto C' so that, for each $(p_1, r) \in Q_N$:
>
> (1'') If $Z_K(p_1) = p \notin D$, then $Z_N(p_1, r) = Z_K(p_1)$.
>
> (2'') If $Z_K(p_1) = p \in D$, then $Z_N(p_1, r) = v_q^r(r)$, where. . .

...For each $x \in I_N$ and $(p_1, r) \in Q_N$, define $x_N(p_1, r) = (s_1, t)$ by:

(1') $s_1 = x_K(p_1)$.

(2') If $Z_K(s_1) = s \in C \cap C'$, let $t = r$.

(3') If $Z_K(p_1) = p \in C \cap C'$ and $Z_K(s_1) = s \in D$, choose any t such that $v_s^q x_M(p) \subseteq t$.

(4') If $Z_K(p_1) = p \in D$ and $Z_K(s_1) = s \in D$, choose t so that $v_s^q x_M v_q^p(r) \subseteq t$.

These suggestions subsume some of those contained in an erratum note [22] published after the completion of the present work. In that same note, and to correct still another subtle failing of the Zeiger procedure, it was suggested that the initial similarity class D consist of elements of C which are of maximal size among the elements of C. It is easily shown that such a choice of D is always possible.

Lemma 5.6.1: If C is a cover of Q_M and a largest element of C contains k states, then there is an initial similarity class D of cover elements each of which contains k states.

Proof: Let $D_1, D_2, \ldots, D_n$ be the distinct similarity classes of elements of C such that each element of D_i $(1 \leq i \leq n)$ contains k states. Since similarity is an equivalence relation, each element of D_i $(1 \leq i \leq n)$ is mapped by elements of $\tilde{S}_M$ onto every element of D_i. Thus, for $i \neq j$, if any element of D_i maps onto an element of D_j, then

a. every element of D_i maps onto every element of D_j under appropriate transformations from $\tilde{S}_M$, and

b. no element of D_j maps onto any element of D_i. (Otherwise, an element of D_i and an element of D_j would be similar, making $D_i = D_j$.)

The relation # defined by $D_i \# D_j$ if and only if some element of D_i maps onto an element of D_j is a partial ordering on the set $\{D_1, D_2, \ldots, D_n\}$. Since the latter collection is finite, it contains a

maximal element D with respect to #. The similarity class D is an initial similarity class, since no element not in D maps onto any element in D under any transformation in $\tilde{S}_M$.

5.7. Decomposition of Definite Automata

In the present section we characterize definite automata as those in which no non-singleton subset of the state set is permuted by any nonempty input sequence. Using this result and examining the decomposition procedure of preceding sections, we show that a Zeiger decomposition of a definite automaton is essentially a cascade of reset machines--"essentially" because some of the component machines may receive permutation inputs, but only of a trivial sort. A result concerning the way in which permutations of any non-singleton subset of a machine's state set are manifested in decompositions of that machine yields the converse result: an automaton whose decomposition consists essentially of resets is a definite automaton.

Lemma 5.7.1: The finite automaton M is definite if and only if no non-singleton subset of Q_M is permuted by any nonempty input sequence.

Proof: Suppose M is k-definite for some positive integer k. Then M is synchronized by any sequence whose length exceeds k. If $P \subseteq Q_M$, $x \neq \Lambda$, and $x_M(P) = P$, then $(x^k)_M(P) = P$. Since $\ell(x^k) \geq k$, P is a singleton. Thus no non-singleton subset of Q_M is permuted by any input sequence $x \neq \Lambda$.

Conversely, if $|Q_M| = n$, $P \subseteq Q_M$ and $|P| = j > 1$ implies P is not permuted by any input sequence $x \neq \Lambda$, then for any sequence x such that $\ell(x) \geq \binom{n}{j}$, it follows that $|x_M(P)| < |P|$. (Otherwise, some subset R of Q_M with cardinality j appears twice in the sequence of subsets through which P is mapped as the input sequence x is received by M. Such a set R is therefore permuted by a nonempty subtape of x, contrary to hypothesis.) Therefore if x is any tape of length not

less than $c = \sum_{j=2}^{n} \binom{n}{j}$, then $x_M(Q_M)$ is a singleton. The automaton M is hence k-definite for some integer $k \leq c$, and the proof is complete.

It would seem to follow immediately from Lemma 5.7.1. that no machine L_i in a cascade decomposition of M receives permutation inputs, since the group of permutations on Q_{L_i} is a homomorphic image of the group of permutations on some subset of Q_M. However, singleton subsets of Q_M may be permuted (that is, fixed) by nonempty input sequences x; so machine L_i may receive an input which induces the identity mapping on Q_{L_i}.

Consider now the induction step in the decomposition of an automaton M. Given a cover C of Q_M and a machine K which tells where M is in C, one obtains a finer cover C' of Q_M and a machine L such that a cascade, N, of K and L tells where M is in C'.

A close examination of statements (1')--(4') of section 5.6 reveals that, for any $p \in Q_K$ and $x \in I_M$, (p, x) is a permutation input to L if and only if (2') holds, or (4') holds and $x_M(p) = s$. (In this case, p and s correspond to elements of the initial similarity class D of elements of C replaced in forming C'.) The next results and the related discussion rule out the latter possibility for definite automata. In case (2') applies, the resulting identity mapping on Q_L is trivial in the sense that it represents the arbitrary filling of a "don't care" entry in the transition table for L. We shall refer to this as the trivial identity and to any other permutations induced on machines in the cascade (when (4') applies) as nontrivial permutations. The trivial identity permutations assigned under (2') could as easily have been resets. We therefore shall refer to a permutation-reset machine whose only permutations are trivial identities as a reset machine. The next three results show that a decomposition of a definite automaton is a cascade of such reset machines.

Lemma 5.7.2: The first permutation-reset machine K_1 in a decomposition of the k-definite automaton M has no permutation inputs.

Proof: Since M is k-definite, no non-singleton subset B of Q_M is permuted by any function $x_M \in S_M$. Therefore no input symbol permutes Q_M. According to the procedure for constructing K_1 (see Zeiger's Proposition 1 construction as described in Section 5.3), every input to K_1 is a resetting input.

Lemma 5.7.3: If M is a k-definite automaton, C is a cover of Q_M, $B_i \in C$ and $|B_i| > 1$, then $\{B_i\}$ is a similarity class of elements of C.

Proof: Since M is k-definite, any tape x whose length exceeds k-1 synchronizes M. If B_i and B_j are distinct elements of the same similarity class of elements of C, then there are nonempty tapes u and v such that $B_i \xrightarrow{u} B_j \xrightarrow{v} B_i$ (id), and hence $(uv)_M$ permutes B_i, contrary to Lemma 5.7.1. This contradiction completes the proof.

In connection with the preceding lemma, note in particular that if C is a cover of Q_M not consisting entirely of singletons, then an initial similarity class D of elements of C contains but one cover element q. This observation simplifies the proof of the following result.

Lemma 5.7.4: If M is a definite automaton, $C_1, C_2, \ldots, C_r = 0$ (the discrete cover) is the sequence of covers associated with a Zeiger decomposition of M, and $L_1, L_2, \ldots, L_r$ is the corresponding sequence of permutation-reset machines, then no machine L_i $(2 \leq i \leq r)$ receives a nontrivial permutation input.

Proof: For $2 \leq i \leq r$ a cascade K of $L_1, \ldots, L_{i-1}$ tells where M is in C_{i-1}. Let $D_{i-1} = \{q\}$ be the initial similarity class of elements of C_{i-1} replaced in obtaining C_i. In the discussion preceding Lemma 5.7.2 it was observed that a nontrivial permutation occurs for the machine L_i under the cascade input symbol x if and only if x_M maps an element p of D_i onto another element s of D_i. In the present instance this means $x_M(q) = q$, which would be contrary to Lemma 5.7.1.

The next result is of sufficient intrinsic interest to be called a

theorem, and is not restricted to definite automata. It is also used in establishing the main result of the present section, Theorem 5.7.6.

Theorem 5.7.5: If $x \in I_M$ and x_M permutes a non-singleton subset S_x of Q_M, then in any cascade decomposition of M there is a machine L_i which receives a nontrivial permutation input.

Proof: Since S_x is split into singletons by $C_r = 0$, and S_x is contained in some element of C_1, there is a greatest integer $j < r$ such that S_x is contained in an element of C_j. Then for some element p of D_j, $S_x \subseteq p$ and S_x is not contained in any element of R_p. Since x_M permutes S_x, $x_M(S_x) = S_x \subseteq p$; thus either $S_x = p$ or S_x is not maximal in $\{w(R) : w \text{ is in } \widetilde{S}_M, R \text{ is in } C_j, \text{ and } w(R) \subsetneq p\}$. The latter possibility is ruled out by the fact that maximality of S_x would cause S_x to be an element of C_{j+1}, contrary to the selection of j; therefore $S_x = p$. Thus x_M fixes $p \in D_j$, hence permutes the elements of R_p. Moreover, since p is maximal in C_j, if machine N_j is started in a state s corresponding to p and receives input symbol x, the next state of N_j also corresponds to p. By step (4') of Section 5.6., (s, x) is a permutation input to L_{j+1}.

We are now in a position to state and prove the main result of this section.

Theorem 5.7.6: The machine M is a definite automaton if and only if any Zeiger decomposition of M is a cascade of reset machines.

Proof: The forward implication follows immediately from Lemmas 5.7.2. and 5.7.4. To establish the reverse implication, we apply the contrapositive of the preceding result and the characterization of definite automata of Lemma 5.7.1. The proof is as follows.

If any Zeiger decomposition of M is a cascade of reset machines, then there is a cascade decomposition of M in which no component machine L_i receives a nontrivial permutation input. By the contrapositive of Theorem 5.7.5., no non-singleton subset of Q_M is permuted

by any input symbol $x \in I_M$. By Lemma 5.7.1 , M is a definite automaton.

5.8. Decomposition Effects of Q_M

In this section we consider automata having permutation inputs, and examine some of the ways in which such inputs are manifested in cascade decompositions. We continue to let $C_1, C_2, \ldots, C_r = 0$ denote the sequence of covers associated with a particular but arbitrary decomposition. The following lemma is due to Zeiger [21].

Lemma 5.8.1: If $x \in I_M$ and x_M permutes Q_M then x_M permutes C_1.

That is to say, x_M maps each element of C_1 onto (not into) an element of C_1, the result being a one-to-one mapping of C_1 onto C_1. That this result may be extended to any of the covers C_i $(1 \leq i \leq r)$ may be intuitively appealing but is not immediate. The extended result depends on the particular way in which cover C_{i+1} is generated from cover C_i $(1 \leq i < r)$ and not just on the fact that C_{i+1} is a cover of Q_M. This is shown in Example 5.8.1.

Example 5.8.1: Let M be the machine represented in Figure 5.8.1.

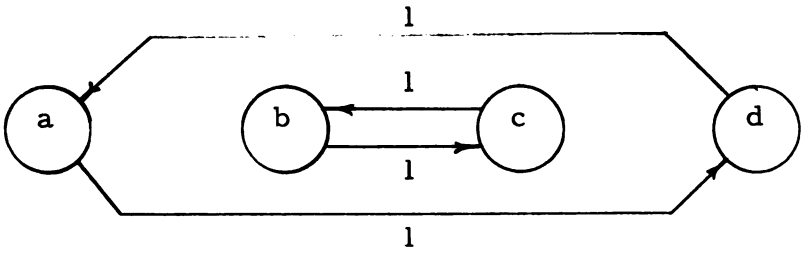

Figure 5.8.1. State Diagram of M

Let C = {ab, c, cd}. Then C is a cover of Q_M, and I_M permutes Q_M, but I_M does not permute C.

It may appear that we have somehow cheated by allowing one element of C to be properly contained in another element of C. However, the procedure by which the sequence $C_1, \ldots, C_r$ is obtained does not preclude such inclusions, for in obtaining C_{i+1} each element q in

D_i is replaced with $R_q = \max(\{ w(R) : R \in C_i, w \in \tilde{S}_M$, and $w(R) \supsetneq q \})$, letting $C_{i+1} = C_i - D_i \bigcup \{R_q : q \in D_i\}$, not $C_{i+1} = \max(C_i - D_i \bigcup \{R_q : q \in D_i\})$.

Lemma 5.8.2: If x_M permutes Q_M and x_M permutes C, then for each $b \in C$, $x_M(b) \in C$ (and not just contained in an element of C).

Proof: Place the elements of C into sets according to their cardinalities. Let A_1 be the set of elements of C of maximal cardinality, A_2 the set of elements of next smaller cardinality, and so on.

Since x_M permutes Q_M, $|x_M(b)| = |b|$ for any $b \in C$. Accordingly, no element of A_i is mapped by x_M into an element of A_j for $j > i$. Hence x_M maps each element of A_1 onto an element of A_1 and, since x_M permutes C, this mapping of A_1 into A_1 is one-to-one and onto.

The mapping x_M takes each element of A_2 either onto another element of A_2 or into an element of A_1. Since each element of A_1 has a pre-image in A_1 under x_M, the latter possibility is ruled out, and x_M maps A_2 one-to-one onto A_2. Proceeding in this fashion, x_M maps A_i on-to-one onto A_i for each class A_i.

Corollary 5.8.3: If x_M permutes Q_M and permutes C, then the orbit of each element b of C under repeated applications of x_M consists of cover elements having the same cardinality as b.

Corollary 5.8.4: If x_M permutes Q_M and cover C, if D is any similarity class of elements of C, and if $b \in D$, then $x_M(b) \in D$. Thus, x_M permutes D and permutes C - D.

Proof: For some positive integer j, $(x_M)^j$ is the identity mapping on Q_M. Then $(x_M)^{j-1}$ maps $x_M(b)$ onto b. Therefore $x_M(b)$ is similar to b.

Lemma 5.8.5: For any integer i ($1 \leq i \leq r$), if y_M permutes Q_M and y_M permutes C_i then y_M permutes C_{i+1}.

Proof: If y_M permutes Q_M then for some positive integer j, $(y_M)^j$ is the identity mapping on Q_M. By the preceding corollary, y_M permutes D_i.

Let p be any element of D_i, and let $y_M(p) = s \in D_i$. For any cover element $b \in R_p$, $y_M(b)$ is contained in some element $c \in R_s$. If $y_M(b) \subsetneq c$, then $b = (y_M)^{j-1}(y_M(b)) \subsetneq (y_M)^{j-1}(c) \subsetneq (y_M)^{j-1}(s) = p$. Since $c \in R_s$, $(y_M)^{j-1}(c) \in \{x_M(R) : x \in S_M, R \in C \text{ and } x_M(R) \subsetneq p\}$; and since $b \subsetneq (y_M)^{j-1}(c)$, $b \notin R_p$; but this is contrary to the selection of b. Therefore $y_M(b) = c \in R_s$.

That is, if $y_M(p) = s$ then y_M maps each element of R_p onto an element of R_s; hence y_M maps $\bigcup_{p \in D_i} R_p$ into $\bigcup_{p \in D_i} R_p$. Furthermore, this mapping is one-to-one, since y_M is one-to-one on Q_M. Therefore y_M permutes $C_i - D_i$, permutes $\bigcup_{q \in D_i} R_q$, and hence permutes $C_{i+1} = C_i - D_i \bigcup_{q \in D_i} R_q$.

Lemmas 5.8.1 and 5.8.5 are sufficient to establish the following major result.

Theorem 5.8.6: If x permutes Q_M and if $C_1, \ldots, C_r = 0$ is the sequence of covers associated with a particular decomposition of M, then x_M permutes C_i $(1 \leq i \leq r)$.

The remaining results of this section deal with ways in which permutations of Q_M show up in the machines of cascade decompositions of M. Recall that to each C_i $(1 \leq i \leq r)$ there corresponds a permutation-reset machine L_i and a cascade N_i of $L_1, \ldots, L_i$ such that N_i tells where M is in C_i.

Lemma 5.8.7: If x_M permutes Q_M and N_i tells where M is in C_i, and if t is a state of N_i corresponding to $p \in D_i$, then $x_{N_i}(t)$ corresponds to $x_M(p) \in D_i$.

Proof: It has been shown (Corollary 5.8.4) that x_M permutes D_i; hence $x_M(p) \in D_i$. Since N_i tells where M is in C_i, $Z_{N_i}(x_{N_i}(t))$

$\supseteq x_M(Z_{N_i}(t)) = x_M(p)$, where Z_{N_i} is the function mapping Q_{N_i} onto C_i. The containment shown is actually equality, due to the maximal size of $x_M(p)$ in C_i.

Lemma 5.8.7 facilitates the proof of the following theorem, which characterizes decompositions of automata having permutation inputs.

Theorem 5.8.8: The automaton M has a permutation input if and only if every permutation-reset machine in any decomposition of M receives a permutation input.

Proof: The reverse implication is immediate, for if every machine L_i ($1 \leq i \leq r$) receives a permutation input, then in particular L_1 does. By the construction of L_1, Q_{L_1} is permuted by x_{L_1} only if Q_M is permuted by x_M.

The forward implication follows from Lemma 5.8.7. If x_M permutes Q_M, then x_{L_1} permutes $Q_{L_1} = Q_{N_1}$. For any integer i ($1 \leq i \leq r$) there is a state $t_i \in Q_{N_i}$ corresponding to cover element $p_i \in D_i$; hence $x_{N_i}(t_i)$ corresponds to another element $v_i \in D_i$ by Lemma 5.8.7. Therefore (p_i, x) is a permutation input to L_{i+1}. Thus for all i ($1 \leq i \leq r$), L_i receives a permutation input, and the proof is complete.

In Theorem 5.8.8 it is possible for any or all of the permutations in question to be identity mappings. The next result establishes that a non-identity permutation on the state set of an automaton M is reflected in a non-identity permutation on the state set of some machine in any decomposition of M.

Theorem 5.8.9: If x_M is a non-identity permutation on Q_M, then at least one machine L_i in any cascade decomposition of M receives a non-identity permutation input.

Proof: Since C_r consists of singletons, x_M induces a non-identity

permutation on C_r. There is a least index j ($1 \leq j \leq r$) such that x_M induces a non-identity permutation on C_j.

If j = 1, then x_{L_1} is a non-identity permutation on the state set of L_1, the first machine in the cascade.

If j > 1, then N_{j-1}, a cascade of $L_1, \ldots, L_{j-1}$, tells where M is in C_{j-1}. Let $q \in D_{j-1}$ be the initial element such that $R_q = Q_{L_j}$. Since x_M fixes C_{j-1}, the elements of C_j which are moved by x_M are not in C_{j-1}. Let p_1 be a moved element of C_j, with $p_2 = x_M(p_1) \neq p_1$. Since x_M fixes p, $x_M(p_1) = p_2 \subseteq p$; hence $p_2 \in R_p$. The elements of R_p correspond one-to-one to those of R_q under the mapping $p \xrightarrow{v_p^q} q$; and, since $p_1 \neq p_2$, $q_1 = v_p^q(p_1) \neq v_p^q(p_2) = q_2$.

The machine N_{j-1} tells where M is in C_{j-1}. There is a state s of N_{j-1} corresponding to cover element $p \in D_{j-1}$; and, since x_M fixes C_{j-1}, $x_{N_{j-1}}(s) = t$ also corresponds to cover element p. Starting machine N_{j-1} in state s and machine L_j in state q_1 and applying input x to the cascade sends N_{j-1} to state t. Since both s and t correspond to $p \in D_{j-1}$, the input (s, x) to L_j is a permutation input. If L_j is in state q_1 and receives input (s, x) the next state of L_j is, by step (4') of Section 5.6, $v_p^q x_M v_q^p(q_1) = v_p^q x_M(p_1) = v_p^q(p_2) = q_2 \neq q_1$. That is, (s, x) is a non-identity permutation input to L_j.

Corollary 5.8.10: If x_M permutes Q_M and j is the least integer such that x_M induces a non-identity permutation on C_j, then L_j is the first machine in the cascade such that the input x is part of a non-identity permutation input to L_j.

Proof: If i < j then x_M fixes C_i and C_{i-1}. For any $p \in D_{i-1}$, $x_M(p) = p$. If s is a state of N_{i-1} corresponding to p, then for any present state t of L_i the next state of L_i is $v_p^q x_M v_q^p(t) = v_p^q v_q^p(t) = t$, where the first equality holds because x_M fixes C_i and the second holds because $v_p^q v_q^p$ is the identity mapping on q. Therefore x can produce only the identity permutation on Q_L.

It has been shown (Theorem 5.8.6) that if x_M permutes Q_M then x_M permutes every cover C_i $(1 \leq i \leq r)$ associated with any particular Zeiger decomposition of M. If x_M is not the identity permutation on Q_M, then x_M produces a non-identity permutation on one or more covers C_i $(1 \leq i \leq r)$. The next results show that the covers permuted in non-identical fashion by x_M occur consecutively and at the terminal end of the sequence $C_1, \ldots, C_r$.

Theorem 5.8.11: If for some integer j $(1 \leq j \leq r)$ x_M permutes Q_M and fixes the elements of C_j, then x_M fixes the elements of C_i for $1 \leq i \leq j$.

Proof: We show that if x_M fixes C_j then x_M fixes C_{j-1}. The result follows by iteration.

Since $C_j = C_{j-1} \bigcup_{q \in D_{j-1}} R_q$, if x_M fixes C_j then x_M fixes $C_{j-1} - D_{j-1}$ and also fixes the elements of R_q for each $q \in D_{j-1}$. Since $q = \bigcup_{a \in R_q} a$, x_M fixes q for each $q \in D_{j-1}$. Thus x_M fixes $(C_{j-1} - D_{j-1}) \bigcup D_{j-1} = C_{j-1}$.

Corollary 5.8.12: If for some integer j $(1 \leq j \leq r)$ x_M permutes Q_M and moves an element of C_j, then x_M moves an element of C_i for all integers i such that $j < i \leq r$.

Proof: If, for some $i > j$, x_M fixes the elements of C_i, then by Theorem 5.8.11, x_M fixes the elements of C_j, contrary to hypothesis.

It should be pointed out that, even though x_M moves an element of C_i, x is not necessarily part of a non-identity permutation input to L_i. For example, it may be the case that, for each $p \in D_i$, x_M fixes the elements of R_p.

6. SUGGESTIONS FOR FURTHER INVESTIGATION

There are several unresolved or untouched issues which might be considered in connection with the present work. Among the former are questions concerning the machine-independent bounds of Chapter 3. Can a sharp bound be obtained for the length of a shortest synchronizing sequence of an arbitrary n-state automaton? What sort of bounds can be established in terms of both the number of states and the number of symbols in the input alphabet?

It would also be logical to investigate cascade decompositions of arbitrary ultimate-definite automata in an effort to characterize such decompositions.

It is evident that, if a cascade decomposition of M is synchronizable, then M is synchronizable. In the examples constructed by the author, synchronizable automata yielded synchronizable decompositions. Is this always the case?

Much work is now appearing concerning cellular arrays of automata. What can be said concerning arrays of synchronizable automata? What sorts of arrays are also synchronizable?

Little mention has been made of the kinds of events accepted by synchronizable machines. The author has examined some aspects of such events, and there seems to be enough structure to warrant further investigation.

The decomposition procedure described and modified herein has an ad hoc flavor, its features arising from particular constraints which might also be satisfied by other constructions. A modification which shortened the sequence $C_1, C_2, \ldots, C_r$, perhaps by discarding more elements of C_i in obtaining C_{i+1}, would be of interest.

REFERENCES

1. EVEN, S. Test for synchronizability of finite automata and variable length codes. IEEE Trans. Inf. Theory IT-10 (1964), 185-189.

2. GILL, A. State-identification experiments in finite automata. Inf. Contr. 4 (1961), 132-154.

3. GINSBURG, S. An Introduction to Mathematical Machine Theory, Addison Wesley, Inc., Reading, Mass., 1962.

4. HARTMANIS, J. and Stearns, R. E. Algebraic Structure Theory of Sequential Machines, Prentice-Hall, Englewood Cliffs, New Jersey, 1966.

5. _____, Sets of numbers defined by finite automata. Am. Math. Monthly 74 (1967), 539-542.

6. KLEENE, S. C. Representation of events in nerve-nets and finite automata. Automata Studies, Ann. Math. Studies No. 34. C. E. Shannon and J. McCarthy (Eds.), Princeton University Press, Princeton, New Jersey (1956), 3-41.

7. KROHN, K. and Rhodes, J. Algebraic theory of machines: I. Prime decomposition theorem for finite semigroups and machines. Trans. Am. Math. Soc. 116 (1965), 450-464.

8. _____, Results on finite semigroups derived from the algebraic theory of machines. Proc. Nat. Acad. Sci. U. S. A. 53 (1965), 499-501.

9. KROHN, K., Mateosian, R. and Rhodes, J. Complexity of ideals in finite semigroups and finite state machines. Math. Systems Theory 1 (1967), 59-66.

10. LIU, C. L. Some memory aspects of finite automata. Tech. Rept. No. 411, M. I. T. Research Lab. of Electronics, Cambridge, Mass., May 31, 1963.

11. MCCULLOCH, W. S. and Pitts, W. A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophysics 5 (1943), 115-133.

12. MCNAUGHTON, R. E. and Yamada, H. Regular expressions and state graphs for automata. Trans. IRE EC EC-9 (1960), 39-47.

13. PAZ, A. and Peleg, B. Ultimate-definite and symmetric-definite events and automata. J. ACM 12 (1965), 399-410.

14. PERLES, M., Rabin. M. O. and Shamir, E. The theory of definite automata. Trans. IRE E C-12 (1963), 233-243.

15. RABIN, M. O. Probabilistic automata. Inf. Contr. 6 (1963), 230-245.

16. RABIN, M. O. and Scott, D. Finite automata and their decision problems. IBM J. Res. Develop. 3 (1959), 114-125.

17. RHODES, J. Complexity and characters of finite semigroups. Submitted to J. Algebra.

18. _____. A homomorphism theorem for finite semigroups. Math. Systems Theory 1 (1967), 289-304.

19. STEARNS, R. E. and Hartmanis, J. Regularity preserving modifications of regular expressions. Inf. Contr. 6 (1963), 55-69.

20. ZEIGER H. P. Cascade synthesis of finite-state machines. In Proc. of the Sixth Annual Conf. on Switching Theory and Automata, Institute of Electrical and Electronics Engineers, New York, 1965.

21. _____. Cascade synthesis of finite-state machines Inf. Contr. 10 (1967), 419-433.

22. ERRATUM, Inf. Contr. 11 (1967), 471.