SARANAKOMANA, SUCHADA. Experiments with Finite State Machines. (1975)
Directed by: Dr. Richard Michael Willett. Pp. 42

The purpose of this study is to introduce and illustrate the various types of experiments with finite state machines. A finite state machine is an abstract object composed of a finite number of input, output and state symbols. The behavior of the machine is described by a functional relationship between input, output and state. In designing a finite state machine it often happens that two states represent the same internal condition. Therefore it is desirable to develop a technique for transforming one machine into another which has no redundant states, so that both have the same behavior. The definition of k-equivalent and k-distinguishable are useful in an algorithm which is developed to determine which states of the machine are equivalent. The machine with no two equivalent states is a reduced machine. An experiment on a reduced state machine consists of applying an input sequence and observing the output. The classification of experiments is (1) simple and multiple experiments; (2) preset and adaptive experiments; (3) distinghishing and homing experiments. The successor tree is a useful device in designing the experiment. The successor tree is terminated by specific rules in each experiment. Homing and distinguishing experiments can be either preset and adaptive. All reduced machines have a homing sequence. A distinguishing sequence is sometimes possible in both preset and adaptive form. However, some reduced machines have only an adaptive experiment and some do not have any simple distinguishing sequence at all.

EXPERIMENTS WITH FINITE STATE MACHINES

by

Suchada Saranakomana

A Thesis Submitted to
the Faculty of the Graduate School at
The University of North Carolina at Greensboro
in Partial Fulfillment
of the Requirements for the Degree
Master of Arts

Greensboro
1975

Approved by

Michael Willett

Thesis Adviser

APPROVAL SHEET

This thesis has been approved by the following committee of the Faculty of the Graduate School at the University of North Carolina at Greensboro.

Thesis Adviser	Michael Willett
Committee Members	David S. Herr
	William A. Powers

August 12, 1975
Date of Acceptance by Committee

ACKNOWLEDGMENTS

The writer wishes to express her deep appreciation to Dr. Michael Willett for his constant patience, assistance, and understanding throughout this study.

Appreciation is extended to Dr. David Herr and Dr. William Powers for their cooperation as members of the examining committee.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In a short time man has found himself quite dependent on machines. Machines perform many of our routine labors for us under automatic control. When the word "machine" is used, most people seem to think that machines can do only what they are told, and that they will do it relentlessly. This is why it seems to most people that it is not possible to build machines capable of imaginative or creative activity. To make a theoretical study of machines it is necessary to ignore many physical details of mechanical systems. To arrive at a precise definition of a mathematical object which approximates our concept of a machine, we will concentrate on the function of a machine and not its physical structure.

The object of this study is the kind of machine that can be built from a finite number of simple parts. These machines are called finite state machines. The behavior of a finite state machine can be represented in terms of mathematical relations between three variables: input, output and state.

Before we give a precise mathematical definition of a machine, we will attempt to motivate the various ingredients. A machine may be represented as a black-box with input and output terminals which has the schematic representation shown below:

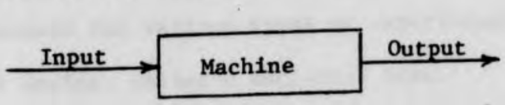

Figure 1.1: Machine representation.

Any one of a finite number of symbols can be presented at the input terminal. The machine acts on this symbol and produces an output symbol at the output terminal. If the machine represents a functional relationship between input and output symbols, that is, a given input symbol always produces the same output symbol, then the realization of such devices constitutes switching circuit theory. But we can imagine examples of such devices pictured above in which there is no functional relationship between input and output. Each time we pull on a light string (input) we alternately produce light then dark (output). In such devices the output is additionally a function of some changing internal condition or state of the device.

The object in this study is to devise ways to determine the unknown initial state of any machine presented to us. The internal state can be determined in certain cases by external experiments on the machine; that is, selected input is fed to the machine and the output is observed. This thesis is based on the work of Moore [8], Gill [3], Ginsburg [4], and Hibbard [6] on finite state machines.

Before we discuss experiments we will precisely define our class of machines in Chapter II.

In Chapter III, a kind of experiment is discussed. There we will observe that certain formal state symbols may actually represent the same internal state or configuration of the machine. The definition of state equivalence is given and will be used to determine which states of the machine are indistinguishable.

Chapter IV presents the various types of experiments and also introduces a useful device, called a successor tree.

In Chapter V the special case of a homing experiment is discussed. Additionally we will discuss a restricted version of this experiment known as the synchronizing experiment.

Chapter VI is a discussion of the distinguishing experiment. A summary of the major ideas of this study is presented in Chapter VII.

CHAPTER II

FINITE STATE MACHINES

As we indicated in Chapter I, the behavior of a finite state machine can be represented in terms of mathematical relations between input, output, and state. In the definition below we have presented each notion discussed in Chapter I symbolically, but the reader can easily imagine physical realizations of the various components.

Definition 2.1: The 4-tuple $M = (I,A,S,f)$ is a finite state machine (FSM) if:

(a) I,A,S are each finite sets of symbols with the elements of I called the input symbols, the elements of A called output symbols, and the elements of S called the internal states of the machine.

(b) $f: I \times S \to A \times S$ is a function called the computing function.

Some state $s \in S$ is selected and designated as the initial state. This designation is denoted symbolically by $M|s$, which is read "M started in state s". The state of the machine after an input has been applied will be called the final state.

Without any loss in generality we will usually denote the symbols in S by $1,2,\cdots,N$.

Our version of the definition of a finite state machine is called a Mealy machine. If the output depends only on the present state of the machine then the model is called a Moore machine.

If M is presented with the symbol $i \in I$ at the input terminal then the machine evaluates $f(i,s) = (a,s')$. The symbol $a \in A$ is produced at the output terminal and the internal state is changed to s'. If the machine is presented with another input symbol i', then $f(i',s')$ is evaluated and output-state transition proceeds as above.

A finite state machine is completely described by its computing function. Since each such function has only a finite number of arguments, we can present the function in table form. Consider machine M_1 in Table 2.1 below. ($I = \{1,2,3\}$, $S = \{1,\cdots,9\}$, $A = \{0,1\}$)

Table 2.1: Machine M_1

State	Next state			Output		
	1	2	3	1	2	3
1	2	2	5	1	0	0
2	1	4	4	0	1	1
3	2	2	5	1	0	0
4	3	2	2	0	1	1
5	6	4	3	1	0	0
6	8	9	6	0	1	1
7	6	2	8	1	0	0
8	4	4	7	1	0	0
9	7	9	7	0	1	1

The output and state transition is shown for each input symbol and present state. The transition table of this machine shows that with input 1 to present state 1 the output is 1 and the state 1 is changed to state 2. Thus, if we know the initial state of the machine, then we can determine the output sequence response to a given input sequence. Consider $M_1|1$ presented with input 121211. The output is 110001 and the state transitions are 243212.

Another way of representing a machine is by using a state transition diagram. We will use circles to represent states. We represent the function value $f(i,s) = (a,s')$ by the following schematic:

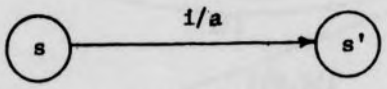

Figure 2.1: Mealy machine representation

If the machine is a Moore machine then the state output symbol is usually included in the circle with the state.

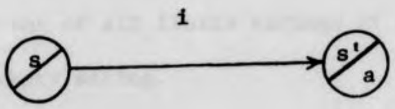

Figure 2.2: Moore machine representation

The machine M_1 can be converted to a diagram as in Figure 2.3.

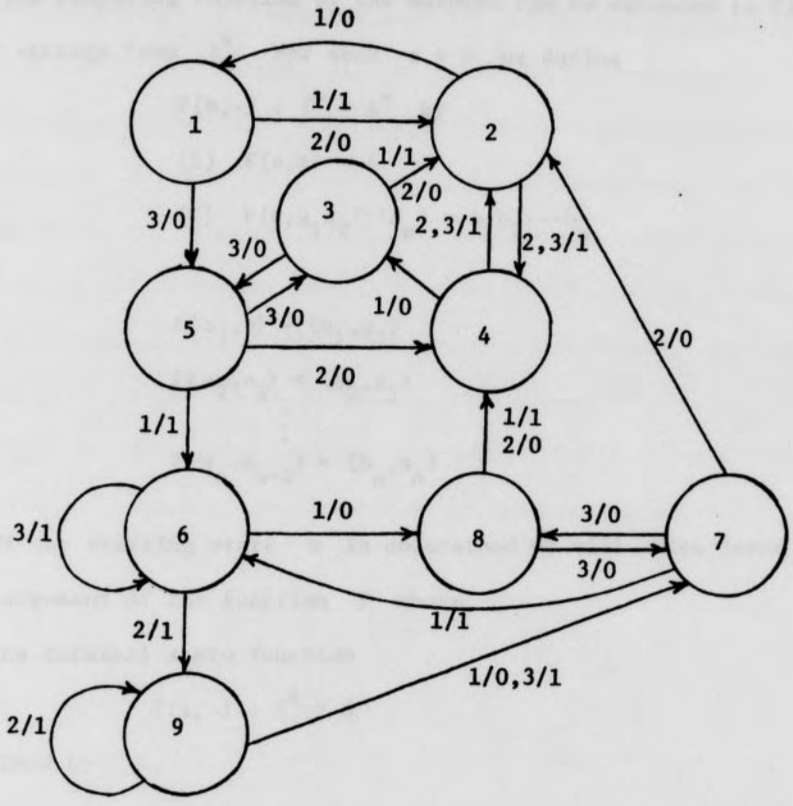

Figure 2.3: Machine M_1

Define A^* as the set of all finite strings of symbols from A. Let ϕ denote the empty string.

The computing function of the machine can be extended to finite input strings from I^*. For each $s \in S$ we define

$$F(s,\cdot) : I^* \to A^* \text{ by}$$

$$(1) \quad F(s,\phi) = \phi$$

$$(2) \quad F(s,a_1a_2\cdots a_n) = b_1b_2\cdots b_n$$

where

$$f(a_1,s) = (b_1,s_1)$$
$$f(a_2,s_1) = (b_2,s_2)$$
$$\vdots$$
$$f(a_n,s_{n-1}) = (b_n,s_n)$$

If the starting state s is understood we will often leave off the first argument of the function F above.

The terminal state function

$$T(s,\cdot) : I^* \to S$$

is defined by

$$T(s,\phi) = i$$

$$T(s,a_1a_2\cdots a_n) = s_n$$

where s_n is the last state transitioned into by the machine when processing the input string.

<u>Definition 2.2</u>: For each $x \in I^*$ and $s \in S$ let $T(s,x) = s'$. Then s' is called the x-successor of s.

That is, if an input sequence $x \in I^*$ causes a finite state machine to make a transition from s to s' then s' is said to be the x-successor of s. If for every pair of states s and s' of M, there exists an input sequence $x \in I^*$ which takes M from s to s' then M is said to be strongly connected.

In the machine M_1 the sequence 121 takes the machine from state 1 into state 3. The machine M_1 is also strongly connected.

We wish to illustrate how a machine U may be designed to perform a given task. Suppose input strings will consist of zeros and/or ones. We wish to design a machine which will examine the string sequentially and print 0 if there has been an even number of ones in the string up to and including the present position and 1 otherwise. We will need two states: one to "remember" that there has been an even number of ones prior to the present position and another for the possibility of an odd number of ones. Such a machine is shown in Figure 2.4.

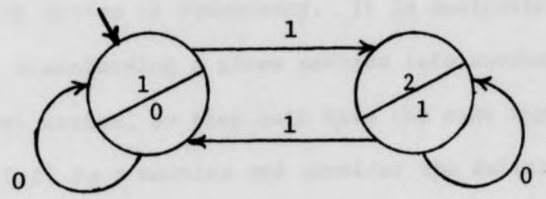

Figure 2.4: Representation of machine U.

CHAPTER III

STATE EQUIVALENCE

In constructing the state diagram (or table) for a finite state machine it often happens that some states in the diagram are redundant, in the sense that they are not essential in defining the input/output relationships of the machine. Intuitively, the role of such a redundant state could be absorbed by another state. The minimization of the number of states reduces the complexity and cost of realizing the machine. We will introduce a concept called state equivalence which will be a formal statement of the notion of redundancy. It is desirable to develop techniques for transforming a given machine into another machine which has no redundant states, so that both have the same input/output behavior. Let $M = (I,A,S,f)$ be a machine and consider the definition.

Definition 3.1: Two states i and j of M are equivalent, denoted $i = j$ if $F(i,x) = F(j,x)$ for all $x \in I^*$; that is, state i and j are equivalent if and only if $M|i$ and $M|j$ produce the same output regardless of input.

If i and j are not equivalent these states are called distinguishable. This means that there exists at least one finite input sequence which when applied as an input sequence to M yields different output sequences, depending on whether i or j is the initial state. If the input sequence contains k symbols, the states are said to be distinguishable by an experiment of length k. We are thus led to the following definition.

Definition 3.2: Two states i and j are called k-equivalent, denoted $i \underset{k}{=} j$, when $F(i,x) = F(j,x)$ for all $x \in I^*$ with $L(x) \le k$ where $L(x)$ denotes the length of the string x.

In words, state i and j are k-equivalent when any input sequence of length $\le k$ applied to $M|i$ and $M|j$ produces the same output sequence.

The definition of k-equivalent and k-distinguishable are useful in an algorithm which we shall develop to determine which states of the machine are equivalent. It is easy to see that each $\underset{k}{=}$ and $=$ are equivalence relations on the set of states. Thus, for each equivalence relation the set of states of the machine is partitioned into disjoint subsets, known as the equivalence classes, where each such class contains all the states (k-) equivalent to any state in the class. Let P be the set of equivalence classes for $=$ and let P_k be the set of equivalence classes for $\underset{k}{=}$.

If $F(i,x) = F(j,x)$ for all $x \in I^*$ then it is true for all input sequences of length $\le k$. Thus, from the definition of $i \underset{k}{=} j$ and $i = j$ we see that $i = j$ if and only if $i \underset{k}{=} j$ for all k.

Let $P = \{A_1, A_2, \cdots, A_m\}$ be a collection of non-empty subsets of a set S. P is a partition of S if $A_i \subset S$, $\bigcup_{i=1}^{m} A_i = S$ and $A_i \cap A_j$ is empty for $i \neq j$.

Definition 3.3: Let P, Q be partitions of a set S. Then P is a refinement of Q when for each $C \in P$ there exists $B \in Q$ so that $C \subseteq B$.

Lemma 3.1: If $i \underset{k+1}{=} j$ then $i \underset{k}{=} j$.

Proof: Choose $i, j \in S$ so that $i \underset{k+1}{=} j$.

This means that $F(i,x) = F(j,x)$ for all $x \in I^*$ with $L(x) \leq k + 1$. But this implies the output is identical whenever $L(x) \leq k$.

Therefore, $i \underset{k}{=} j$. Q.E.D.

By Lemma 3.1 we can say that P_{k+1} is a refinement of P_k.

Since we are dealing with machines with a finite state set, there can only be a finite number of P_i for which P_{i+1} is a proper refinement of P_i; that is, for which $P_{i+1} \neq P_i$. Therefore there must be an integer K such that $P_K = P$. We now proceed to find a bound on K.

Lemma 3.2: If $P_k = P_{k+1}$, then $P_{k+1} = P_{k+2}$.

Proof: Assume $P_k = P_{k+1}$.

Consider any two states i and j such that $i \underset{k+1}{=} j$. Then the successor states $i' = T(i,a)$ and $j' = T(j,a)$ where $a \in I$ satisfy $i' \underset{k}{=} j'$. But $P_k = P_{k+1}$ implies that $i' \underset{k+1}{=} j'$. Since i, j are 1-equivalent and the successor states are $(k+1)$-equivalent regardless of input a, this implies that i, j are $(k+1) + 1 = (k+2)$-equivalent; that is, $i \underset{k+2}{=} j$. Since we already know that $(k+2)$-equivalent implies $(k+1)$ - equivalent, we have that $P_{k+1} = P_{k+2}$. Q.E.D.

Lemma 3.3: If $P_k = P_{k+1}$ then $P_k = P$.

Proof: If $P_k = P_{k+1}$ then by Lemma 3.2 $P_k = P_{k+m}$ for all $m > 0$. Because P is a refinement of P_k for all k, then $P_k = P_{k+1}$ implies $P_k = P$. Q.E.D.

In other words, if two consecutive P_k are equal then there will be no further refinement of the partition P_k. The next lemma establishes a bound on when this must occur.

<u>Lemma 3.4</u>: $P_{N-1} = P_N$ where N is the number of states.

<u>Proof</u>: Assume $P_{N-1} \neq P_N$.

By Lemma 3.2, this implies that $P_k \neq P_{k+1}$ for $1 \le k \le N - 1$. Let $| P_k |$ denote the number of sets in the partition P_k. First, we claim that $| P_1 | \ge 2$. If $| P_1 | = 1$ then all transitions produce the same output which implies that all states are equivalent. Then $P_1 = P_2 = \cdots = P_{N-1} = P_N$. But we are assuming that $P_{N-1} \neq P_N$, so $| P_1 | \ge 2$. Since P_{k+1} is a proper refinement of P_k for $1 \le k \le N -1$ this implies that

$$\begin{aligned} | P_2 | &\ge 3 \\ | P_3 | &\ge 4 \\ &\vdots \\ | P_N | &\ge N + 1 \end{aligned}$$

But this would indicate that we have partitioned N objects into at least $N + 1$ disjoint sets. This is obviously impossible. So we must have $P_{N-1} = P_N$. Q.E.D.

This discussion has provided an algorithm for locating equivalent states in a given machine. We proceed as follows:

(a) Determine $P_1 (i \underset{1}{=} j$ when $f(a,i) = f(a,j)$ for all $a \in I)$

(b) Determine P_{k+1} from P_k as follows: Choose $i \underset{k}{=} j$. If $T(i,a) \underset{k}{=} T(j,a)$ for all $a \in I$, then $i \underset{k+1}{=} j$.

(c) If $P_k = P_{k+1}$, then $P_k = P$.

This must happen for some $k \le N - 1$.

The following example will illustrate these procedures. Consider the machine M_1 with the transition table in Table 2.1.

The first step is to partition the states of M_1 into subsets such that all states in the same subset are 1-equivalent. This is accomplished by placing those initial states together which produce the same output for each input. We then proceed to calculate successive P_k. For the machine M_1 we have

$$P_1 = \{(1,3,5,7,8)(2,4,6,9)\}$$
$$P_2 = \{(1,3,5,7,8)(2,4,6)(9)\}$$
$$P_3 = \{(1,3,5,7,8)(2,4)(6)(9)\}$$
$$P_4 = \{(1,3,8)(5,7)(2,4)(6)(9)\} = P$$
$$P_5 = \{(1,3,8)(2,4)(5,7)(6)(9)\}$$

For this case the equivalence classes are (1,3,8)(2,4)(5,7)(6)(9). By arbitrarily selecting one state from each equivalence class, the machine M_1 can be transformed into another machine M_2 with five states such that M_1 and M_2 have the same input/output behavior. The transition table of machine M_2 is shown in Table 3.1. We have merged states 3, 8 into 1; 4 into 2; and we have relabelled 6 as 4, 9 as 5, and 7,5 as 3.

Table 3.1: Machine M_2

State	Next State			Output		
	1	2	3	1	2	3
1	2	2	3	1	0	0
2	1	2	2	0	1	1
3	4	2	1	1	0	0
4	1	5	4	0	1	1
5	3	5	3	0	1	1

The machine M_2 is called a reduced state machine, since no two states of the machine are equivalent.

For any two states i, j in a reduced machine with N states, there is an input sequence x, $L(x) \leq N - 1$ so that $M|i$, $M|j$ yield different output. This is an example of an experiment which can be performed on the machine. In Chapter IV we shall introduce various types of experiments with finite state machines. For the remainder of this study, we shall assume that all machines are reduced.

CHAPTER IV

MACHINE EXPERIMENTS

In this chapter we will present various classifications of machine experiments. An experiment consists of applying an input sequence and observing the output. In each of the experiments the machine M is assumed to be reduced and completely specified, but we do not know the initial state. The experimenter has access to the input and output terminals and the state transition table of the machine, but may not inspect the internal structure of the machine during the course of the experiment.

The identification experiments can be classified as follows:

(1) <u>The distinguishing experiment</u>: a given machine M is in one of the states $s_1, s_2, \cdots, s_n$. This experiment is designed to unambiguously identify the initial state at the start of the experiment. Note that the machine state changes during the course of the experiment.

(2) <u>The homing experiment</u>: a given machine M is in one of the states $s_1, s_2, \cdots, s_n$. This experiment is designed to identify the final state at the end of the experiment.

According to how the experiment is performed, the identification experiments are further divided into

(1) <u>Preset experiment</u>: the applied input sequence is completely determined in advance.

(2) <u>Adaptive experiment</u>: the applied input is composed of two or more sequences, each input sequence depending on the previous output.

A preset experiment is easier to implement than an adaptive one. It requires no decision making before the final decision is made, while the adaptive experiment requires a number of decisions before termination. However, for some machines an adaptive experiment is easier to design than a preset experiment.

Identification experiments can also be classified according to the number of copies of the machine available.

(1) Simple experiment is performed on a single copy of the machine.

(2) Multiple experiment is performed on two or more copies of the machine all in the same initial state.

In practice, the simple experiment is preferable since most machines are available in only one copy.

All experiments can be schematically presented as in Figure 4.1.

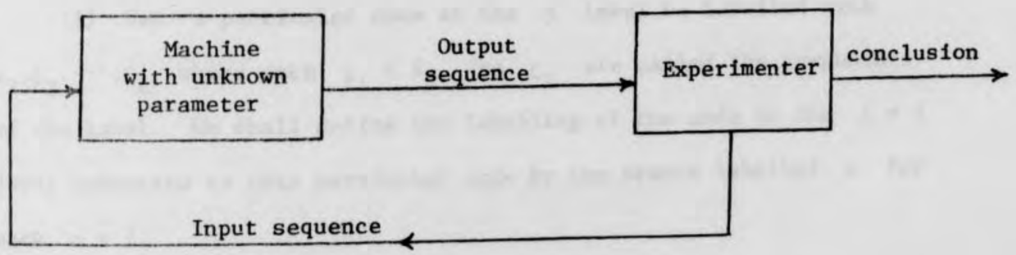

Figure 4.1: Illustration of general experimental process

We will now develop a general procedure which will be useful in designing the various experiments. Assume that we are presented with a machine M in some initial state. The initial uncertainty is defined to be the minimal subset of S known to contain the initial state of M.

A good experiment is one which reduces the initial uncertainty. A useful device in designing the experiment is called the successor tree.

Definition 4.1: The successor tree is a structure which consists of nodes and branches connecting the nodes. The nodes of the tree correspond to the possible states that the machine can be in after the application of an input. The nodes are arranged in levels labelled $j = 0,1,\cdots$. There is one node at the $j = 0$ level. From each node at the j^{th} level there is exactly one branch for each possible input symbol connected to a node at the $j + 1$ level. Each node at the $j + 1$ level has only this one branch connecting it to the j level.

We now proceed to inductively define the labelling of each node by a collection of subsets of the state set:

(1) label the $j = 0$ node with the initial uncertainty.

(2) Let a particular node at the j level be labelled with $L_1,L_2,\cdots,L_k$ where each $L_i \subset S$. The L_i are called the components of the label. We shall define the labelling of the node in the $j + 1$ level connected to this particular node by the branch labelled a for each $a \in I$.

For each L_i define: (recall that $A = \{b_1,b_2,\cdots,b_m\}$)

$$B_{iL} = \{s \mid T(t,a) = s \text{ where } F(t,a) = b_L \text{ for some } t \in L_i\}.$$

That is, each B_{iL} groups together transition states from a particular previous component for the same output during transition.

We want to emphasize that the B_{iL} (and the L_i) are not sets in the strict sense because we require that s be included in B_{iL} as many times as there is a $t \in L_i$ satisfying the definition.

Consider the reduced machine M_2 in Table 3.1. The successor tree can be constructed as follows.

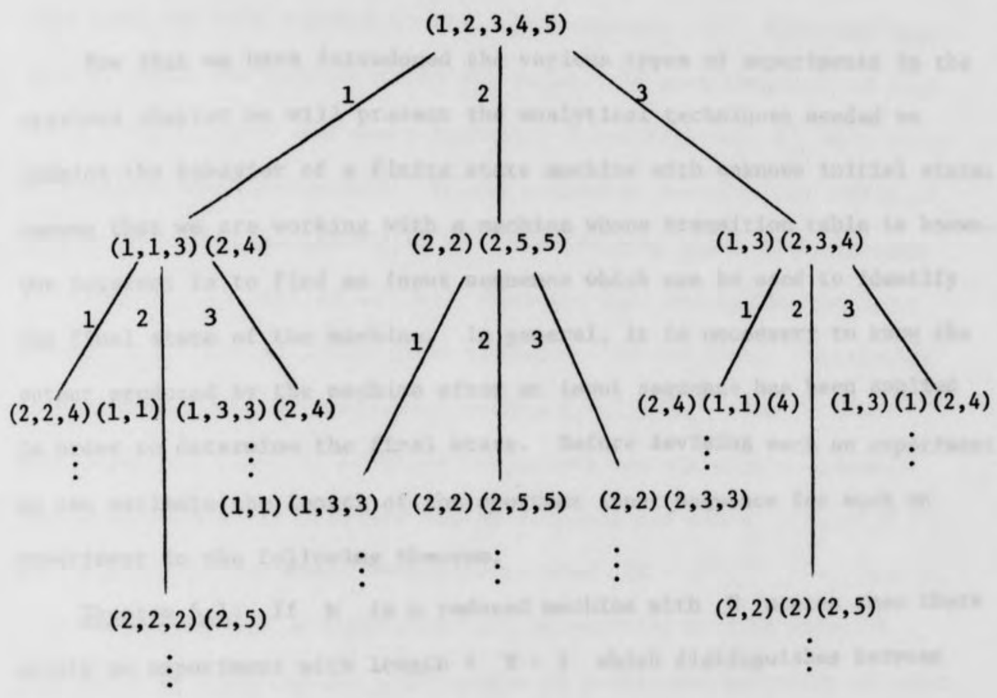

Figure 4.2: The successor tree of machine M_2

The successor tree may be continued as far as is necessary, but for each experiment the successor tree will be terminated by various rules for that experiment. Notice that each node contains the same number of state symbols but they are divided into (possibly) more components as we proceed to the lower levels.

CHAPTER V

HOMING EXPERIMENTS

Now that we have introduced the various types of experiments in the previous chapter we will present the analytical techniques needed to examine the behavior of a finite state machine with unknown initial state. Assume that we are working with a machine whose transition table is known. Our interest is to find an input sequence which can be used to identify the final state of the machine. In general, it is necessary to know the output produced by the machine after an input sequence has been applied in order to determine the final state. Before devising such an experiment we can estimate the length of the shortest input sequence for such an experiment in the following theorem.

Theorem 5.1: If M is a reduced machine with N states then there exists an experiment with length $\leq N - 1$ which distinguishes between any two initial states. (Moore [8])

This theorem is implied by the algorithm of Chapter III which reduces a machine.

Theorem 5.2: For every reduced N-state machine M there exists a preset homing sequence whose length is at most $(N - 1)^2$.

Proof: Let the initial uncertainty be $(1,2,3,\cdots,N)$.

Since M is reduced, then for every pair of states i and j there exists an experiment of length $\leq N - 1$ which distinguishes i from j as initial states.

Choose any two states i, j in the initial uncertainty and let $E_1 \in I^*$ distinguish i from j as initial states. In the successor tree find the node reached by the input sequence E_1. This node must have at least two components in its label. If each component of this label contains only one state (possibly repeated) then E_1 will serve as a preset homing sequence. If one component contains two different states i', j' then let $E_2 \in I^*$ be a sequence which distinguishes i' from j' as initial states. Then the node reached by the input string E_1E_2 has at least three components in its label. Continuing in this manner we create an input string $E_1E_2E_3\cdots E_k$, some $k \leq N - 1$, that reaches a node all of whose components contain single states. Then this sequence is a preset homing sequence and since $L(E_i) \leq N - 1$ and $k \leq N - 1$ the total length is at most $(N - 1)^2$. Q.E.D.

The preset homing experiment can be modified into an adaptive experiment by applying one subsequence at a time and selecting the next subsequence by observing the previous output. We thus have:

<u>Theorem 5.3</u>: Let M be a reduced N-state machine. Then there exists an experiment of length $N(N - 1)/2$ which can determine the final state of M at the end of the experiment.

<u>Proof</u>: We claim that there is an adaptive experiment such that for each k, when the set of possible final states of M contains at most $N - k$ members, then at most $k(k + 1)/2$ input symbols have been applied.

By mathematical induction: if $k = 1$, then the result is obvious because not all state/input pairs produce the same output in a reduced machine. Assume that the statement is true for $k - 1 < N - 1$.

Let G_{k-1} be the set of states with at most $N-(k-1)$ members. The set P_k partitions states of S into at least $k+1$ classes, because M is reduced.

So, G_{k-1} has members from at least two different classes of P_k. To show that this is true, assume not. Then one class in P_k has at least $N-k+1$ members and the other k have at least k total. Together there would be $N-k+1+k = N+1$ states, which is a contradiction to the total number of states.

Assume states $s_i, s_j \in G_{k-1}$ and $s_i \neq s_j$. Then there exists an experiment of length k so that s_i, s_j are distinguished. Perform this experiment to build G_k. At most k more steps reduce G_{k-1} to $N-k$ states. Then the total sequence consists of $\leq k(k-1)/2 + k = k(k+1)/2$ elements.

Now let $k = N-1$. Then the length of the sequence is $\leq N(N-1)/2$ which would be required to reduce the uncertainty to a single state.

Q.E.D.

Now we can streamline this experiment by observing the homing tree which is defined below.

Definition 5.1: A homing tree is a successor tree in which a j^{th} level node is terminated if any one of the two following conditions occur.

(1) A node of the j^{th} level is composed of components containing only one element each (possibly multiple).

(2) The node is labelled identically to some node in the preceding levels.

The presence of single state components indicates that from the output sequence the final state can be uniquely determined. We observe that the necessary information for determining a homing sequence is not lost if we streamline the homing tree as follows. For each B_{iL}, discard duplicate states and label the node in question with the subsets of S which remain. This will define our construction of the homing tree.

The procedure for solving the homing problem by preset experiment can be outlined as follows.

(1) Construct a homing tree.

(2) Choose any of the paths in the tree which is terminated with all single element components, called a homing path, and apply the input corresponding to that path to the machine M.

(3) Determine the final state of M by examining the output.

The input sequence applied to M is called a homing sequence if the final state can be determined from the output sequence.

Segment a given homing sequence E into some number n of subsequences so that $E = E_1E_2\cdots E_n$. Then a simple adaptive experiment can be performed as follows.

(1) Let $k = 1$. Apply E_1 to the machine M and observe the output.

(2) If the output from input sequence $E_1E_2\cdots E_k$ allows one to determine the final state then halt the input process.

(3) If not, then increase k by 1 and repeat (2).

Consider the machine M_2 in Table 3.1. The homing tree can be constructed as in Figure 5.1.

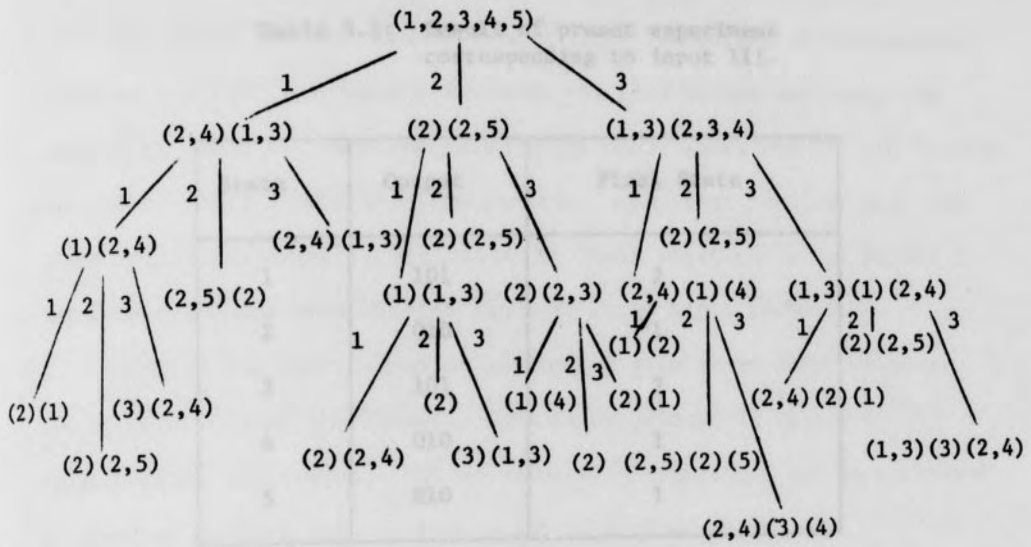

Figure 5.1: Homing tree for machine M_2

By examining the homing tree we see that one of the homing sequences is 111. Since the termination of this path is all single element components then we can determine the final state of machine M_2 as in Table 5.1.

Table 5.1: Result of preset experiment corresponding to input 111.

State	Output	Final State
1	101	2
2	010	1
3	101	2
4	010	1
5	010	1

As in Table 5.1, we see that the output 101 and 010 tell exactly what will be the final state regardless of the initial state.

The machine in Table 3.1 will also be used for an example of an adaptive experiment. Suppose that after applying E_1 we know the machine is in state 1 or 3. Then the partial homing tree is as in Figure 5.2.

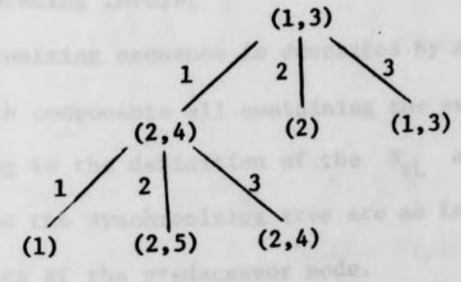

Figure 5.2: Homing tree with initial uncertainty (1,3)

Now select another input sequence E_2. Regardless of the present state we see that the input 2 produces the same output and leads the machine to state 2. Thus the final state is 2. Note that if the machine was led to state 1 or 3 then the adaptive experiment requires only two input symbols in order to determine the final state while the preset experiment for the same machine requires three input symbols.

Consider the restriction of the homing problem in which we are not allowed to view the output. Such an experiment is called a synchronizing experiment. The synchronizing experiment can be performed by means of a synchronizing tree which is defined below.

Definition 5.2: A synchronizing tree is a successor tree which is constructed by ignoring the output. A node in the j^{th} level will be terminated whenever one of the following occurs.

(1) Some node of the j^{th} level is labelled with components all containing the same state.

(2) The node is labelled identically to some node in the preceding levels.

A synchronizing sequence is described by a path which terminates at a node with components all containing the same state.

Referring to the definition of the B_{iL} above, we see that the node components for the synchronizing tree are as follows. The L_1 are the node components of the predecessor node.

For each L_i define:

$$C_i = \{s \mid T(t,a) = s \text{ and } t \in L_i\}.$$

A synchronizing tree is formed from the successor tree as follows.

(1) A node is terminated when its components C_i all contain the same state.

(2) If condition (1) does not occur, generate the next level branches and go back to (1).

The procedure for solving the synchronizing problem is as follows.

(1) Construct a synchronizing tree.

(2) Choose a path which terminates at a node as in (1) above.

(3) Observe the final state after applying the associated input sequence called a synchronizing sequence.

The machine M_2 will be used for an example of the synchronizing problem. The synchronizing tree is shown in Figure 5.2 (partially).

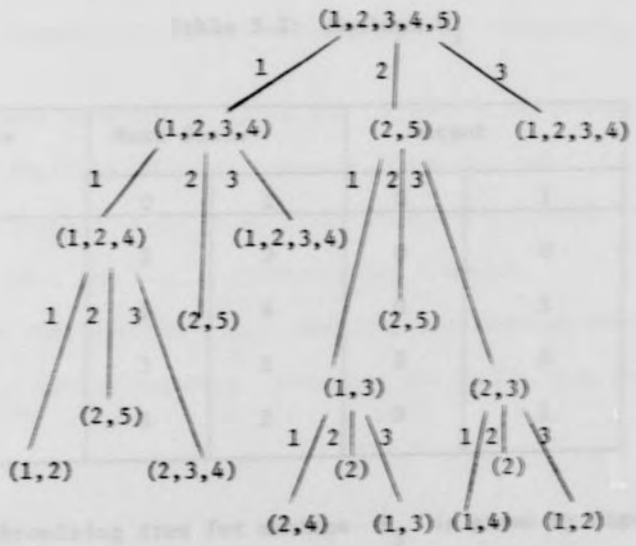

Figure 5.3: Synchronizing tree for machine M_2

The synchronizing sequence is described by a path in the tree leading from the initial uncertainty to a one state node. For machine M_2 synchronizing sequences are 212 or 232; both lead the machine to state 2 regardless of the output or the initial state.

The length of the synchronizing experiment can also be estimated by the following theorem.

<u>Theorem 5.4</u>: If a synchronizing sequence for an N-state machine M exists, then its length is at most $N(N-1)^2/2$. (Kohavi [7]).

Even though all reduced machines have a homing sequence, not all machines have a synchronizing sequence.

Consider machine (M_3) as shown in Table 5.2.

Table 5.2: Machine M_3

State	Next State		Output	
	0	1	0	1
1	2	3	0	0
2	1	4	0	1
3	3	1	0	0
4	3	2	0	1

The synchronizing tree for machine M_3 is given by Figure 5.4.

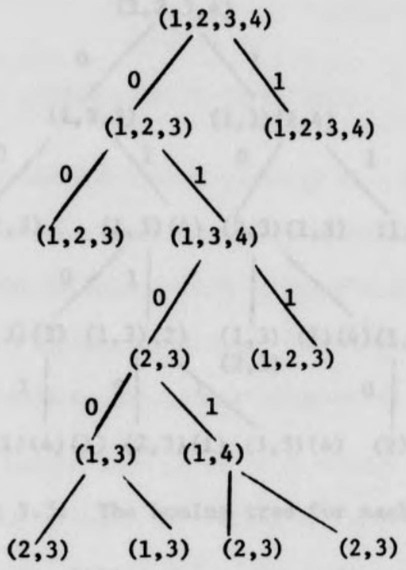

Figure 5.4: Synchronizing tree for machine M_3

We see that none of the paths are terminated with same state components. There is no input sequence which will take the machine from any possible initial state to an identifiable final state. Thus the machine M_3 does not have a synchronizing sequence.

Although this machine (M_3) does not have a synchronizing sequence, it does have a homing sequence. Consider the homing tree in Figure 5.5.

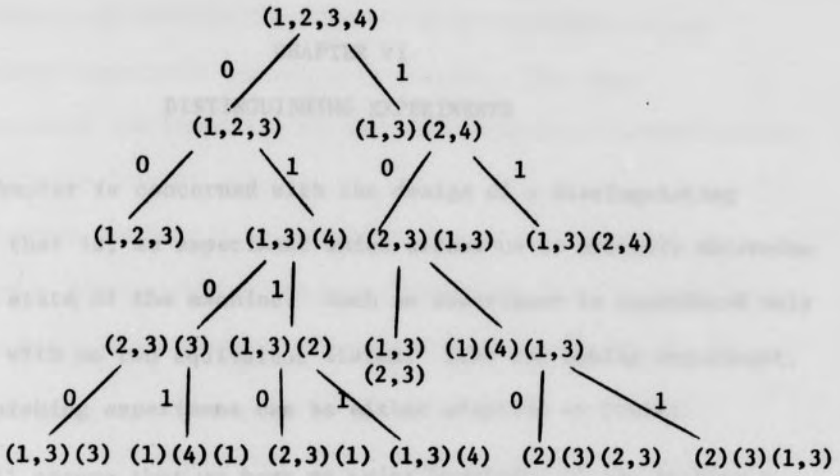

Figure 5.5: The homing tree for machine M_3

A homing sequence is 0101. The corresponding output and final state are shown in Table 5.3.

Table 5.3: The output and final state response of M_3

State	Output	Final State
1	0100	1
2	0000	1
3	0001	4
4	0001	4

The results in Table 5.3 uniquely determine the final state of machine M_3.

CHAPTER VI
DISTINGUISHING EXPERIMENTS

This chapter is concerned with the design of a distinguishing experiment; that is, an experiment which allows us to uniquely determine the initial state of the machine. Such an experiment is considered only on machines with no two equivalent states. Like the homing experiment, the distinguishing experiment can be either adaptive or preset.

We shall assume that we have no prior knowledge of the machine's initial state. Then our problem is to find an input sequence that produces different output sequences for each choice of initial state. Such an input sequence is called a distinguishing sequence.

The experiment can be designed by using the distinguishing tree which is defined below.

<u>Definition 6.1</u>: A <u>distinguishing</u> <u>tree</u> is a successor tree in which a node in the j^{th} level becomes terminal when any of the following occur.

(1) The node contains single element components (no multiple states).

(2) The node is labeled the same as a node of some branch in a preceding level.

(3) The node contains a component with repeated states.

The following procedure is for solving the distinguishing problem by preset experiment.

(1) Construct the successor tree; the tree will be terminated by the rules described above.

(2) Choose any path in the tree which is terminated by one state components (no multiple states). The input sequence corresponding to that path produces different output for each initial state.

(3) Observe the output to determine the initial state of M.

Consider a distinguishing sequence, broken up into k segments. Denote the input subsequence described by the kth segment by E_k. An adaptive experiment can be conducted as follows.

(1) Let k = 1, apply E_1 to M and observe the output.

(2) If the output is attributable to a single initial state in M, this state is the initial state sought.

(3) If not, increase k by 1 and repeat (2).

The following theorem indicates that not all machines have a distinguishing sequence.

<u>Theorem 6.1</u>: There is a machine M for which no simple experiment can distinguish the initial state.

<u>Proof</u>: Consider a machine M as shown in Figure 6.1.

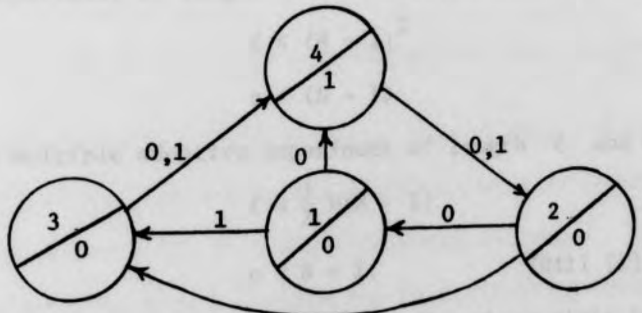

Figure 6.1: State diagram of machine M

(1) If the experiment starts with the input 1 we can not distinguish between states 1 and 2.

(2) If the input 0 is applied to state 1 or 3 we can not distinguish state 1 from 3. (Moore [8]) Q.E.D.

Theorem 6.2: Given any machine M and any multiple experiment on M then there exists another machine different from M for which the original experiment would have had the same outcome. (Moore [8])

That is, it will never be possible to perform a machine identification experiment on a completely unknown machine.

A length estimate for distinguishing experiments, when they exist, can be found by the following.

Theorem 6.3: Let M be a N-state machine with a known transition table. The initial state of M, if at all identifiable by simple experimentation, is identifiable by a simple preset or a simple adaptive experiment of length ℓ where

$$\ell \leq (N - 1)N^N.$$

The initial state of M is always identifiable by a multiple (c copies) preset experiment of length ℓ and multiplicity c, where

$$\ell \leq (N - 1)^2$$

$$c \leq (N - 1)$$

and by a multiple adaptive experiment of length ℓ and multiplicity c, where

$$\ell \leq \frac{1}{2} N(N - 1)$$

$$c \leq N - 1. \qquad \text{(Gill [2])}$$

Consider a reduced state machine M_4 whose state transition table is shown in Table 6.1.

Table 6.1: Machine M_4

State	Next state		Output	
	0	1	0	1
1	1	3	0	0
2	1	2	1	0
3	2	2	1	1
4	3	2	1	1

The distinguishing tree is as follows.

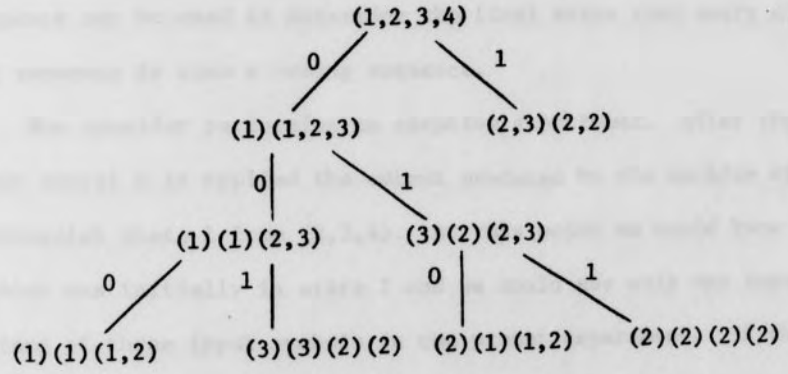

Figure 6.2: Distinguishing tree for machine M_4

By examining the distinguishing tree of Figure 6.2, we see that the node label (2)(2)(2)(2) consists of single element components. Therefore the input sequence 011 is a distinguishing sequence. There is no input sequence starting with a 1 which can be a distinguishing sequence since the branch produced by the first input 1 is terminated.

Table 6.2: Output response of M_4 to distinguishing sequence 011

State	Output	Final state
1	001	2
2	101	2
3	100	2
4	110	2

As shown in Table 6.2, the output sequence that the machine produces uniquely determines the initial state. Since the initial state and input sequence can be used to determine the final state then every distinguishing sequence is also a homing sequence.

Now consider performing an adaptive experiment. After the first input symbol 0 is applied the output produced by the machine will distinguish state 1 from (2,3,4). At this point we would know if the machine was initially in state 1 and we would use only one input symbol instead of three input symbols in the preset experiment. If the first output does not determine the initial state then we have to consider another input sequence. As in Figure 6.2 the adaptive experiment consists of applying a 0 then applying 1 or 11 depending on the previous output. The results of this experiment are shown in Table 6.3.

Table 6.3: Result of the adaptive experiment on machine M_4

State	Input sequence	Output sequence
1	0	0
2	011	101
3	011	100
4	01	11

Thus for the machine M_4 both preset and adaptive experiments can be found. However, some reduced machines have only an adaptive distinguishing experiment and some do not have any distinguishing experiments at all.

Consider the reduced machine M_2 shown in Table 3.1 which has the distinguishing tree as shown in Figure 6.3. The terminal nodes of the distinguishing tree do not contain single element components, which means there is no way to identify the initial state by observing the output of some preset experiment.

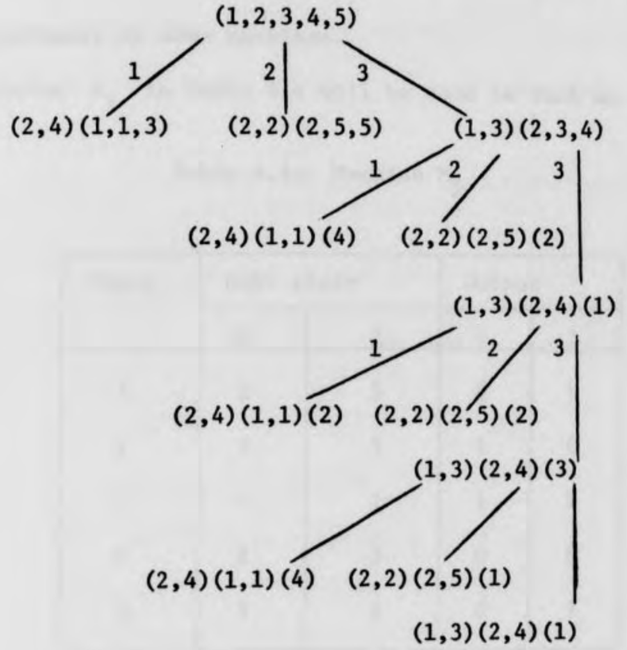

Figure 6.3: Distinguishing tree for machine M_2

Let us try to find an adaptive distinguishing experiment for machine M_2. If we apply the first input 3 we then obtain the output of 0 or 1. The output 0 can distinguish states (1,3) from (2,4,5). Assume that the machine is initially in states 1 or 3. Select another input 3 and observe the output. We see that we would have no way to decide whether the initial state is 1 or 3. The experiment on the machine M_2 shows that there is no distinguishing experiment whether preset and adaptive.

Many machines have all paths through the distinguishing tree which terminate in not necessarily all single element components. Thus we know that we can not find a preset distinguishing experiment for these machines. We now show that we can find the initial state by using an adaptive experiment on some machines.

The machine M_5 in Table 6.4 will be used as such an example.

Table 6.4: Machine M_5

State	Next state		Output	
	0	1	0	1
1	2	5	0	0
2	5	2	1	0
3	4	1	1	1
4	2	3	0	0
5	5	4	0	1

The distinguishing tree is shown in Figure 6.4.

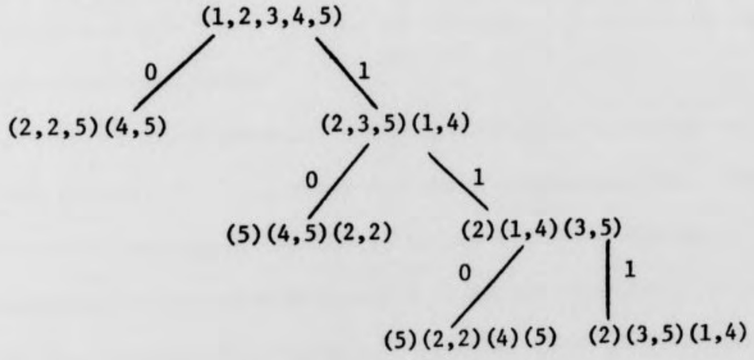

Figure 6.4: The distinguishing tree for M_5

Since there is no path terminating with single element components, then there is no preset distinguishing experiment. But we can find an adaptive experiment on this machine. The results are shown in Table 6.5.

Table 6.5: Result of adaptive distinguishing experiment on machine M_5

State	Input	Output
1	1010	0010
2	101	011
3	110	100
4	1010	0101
5	110	101

From Table 6.5, we see that by observing the output corresponding to each input symbol the initial state of the machine is uniquely determined.

CHAPTER VII

SUMMARY

A finite state machine is an abstract object composed of a finite number of input, output and state symbols. Any functional relationship between input, output and state describes the behavior of a machine. A finite state machine can be presented by either state table or state diagram which indicates the output and state transition for each input symbol and present state.

In designing a finite state machine it often happens that two states represent the same internal condition. These two states are said to be equivalent. If two states are not equivalent they are distinguishable. For each input of length $\le k$ we say that two states are k-equivalent if and only if $M|i$ and $M|j$ produce the same output sequence. Both equivalence and k-equivalence are equivalence relations. These equivalence relations can be used to partition all states of the machine into equivalence classes.

The length of an input sequence which can distinguish between two states need not exceed $N - 1$ symbols for each N-state machine. The machine which has no equivalent states is called a reduced machine.

The application of an input sequence to a machine and the observation of the corresponding output is referred to as an experiment. Machine experiments are used to solve the state-identification problem. In performing an experiment, if the input sequence is determined in advance the experiment is called a preset experiment. It is an adaptive experiment if each input sequence selected is based on the previous output. According to the number of copies of the machine, the experiment is called a simple experiment if the experiment is performed on one copy of the machine and is called a multiple experiment if the experiment is performed on more than one copy.

Two major classes of experiments are the homing experiment and the distinguishing experiment. The homing experiment is designed to identify the final state of the machine; and the distinguishing experiment is designed for identifying the initial state. The special case of a homing experiment in which the final state can be identified without knowing the output sequence is called a synchronizing emperiment. A useful device for designing an experiment is called a successor tree which can be constructed in a rather routine, step by step manner.

The successor tree is terminated by specific rules in each experiment. A preset experiment is described by a path of the tree which leads to certain terminal nodes; an adaptive experiment is described by a set of paths which are selected after observing the previous output. Both homing and distinguishing experiments can be either preset or adaptive. Since all states of a reduced machine are distinguishable then there is always a homing sequence. In the case of a distinguishing experiment, it is sometimes possible that both preset and adaptive experiments can be found. However, some reduced machines have only an adaptive experiment and some do not have any simple distinguishing experiment at all.

Length estimates for the various experiments were presented.

BIBLIOGRAPHY

1. Booth, Taylor. Sequential Machines and Automata Theory. New York: John Wiley and Sons, Inc., 1968.

2. Gill, Arthur. Introduction to the Theory of Finite State Machines. New York: McGraw Hill Book Company, 1962. pp. 49-135.

3. Gill, Arthur. "State-Identification Experiments in Finite Automata." Information and Control, vol. 4, 1961. pp. 132-154.

4. Ginsburg, Seymour. "On the Length of the Smallest Uniform Experiment Which Distinguishes the Terminal States of a Machine." J. Assoc. Comput. Mach., vol. 5, 1958. pp. 266-280.

5. Hennie, Frederick. Finite State Models for Logical Machines. New York: John Wiley and Sons, Inc., 1968. pp. 1-53, 96-139.

6. Hibbard, Thomas. "Least Upper Bounds on Minimal Terminal State Experiments for Two Classes of Sequential Machines." J. Assoc. Computing Machinery, vol. 8, 1961. pp. 601-612.

7. Kohavi, Zvi. Switching and Finite Automata Theory. New York: McGraw Hill Book Company, 1970. pp. 261-313, 407-443.

8. Moore, E. F. "Gedanken-experiments on Sequential Machines." Automata Studies, Annuals of Mathematics Studies, no. 34. N.J.: Princeton University Press, 1956. pp. 129-153.