F A U L T D E T E C T I O N E X P E R I M E N T S

I N S E Q U E N T I A L M A C H I N E S

by

Jacques Andre Rivierre

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June, 1968

Signature of Author _____
Department of Electrical Engineering, May 17, 1968

Certified by _____
Thesis Supervisor

Accepted by _____
Chairman, Departmental Committee on Graduate Students

F A U L T - D E T E C T I O N E X P E R I M E N T S

I N S E Q U E N T I A L M A C H I N E S

by

JACQUES ANDRE RIVIERRE

Submitted to the Department of Electrical Engineering on May 17, 1968 in partial fulfillment of the requirements for the Degree of Master of Science

A B S T R A C T

The design of fault-detection experiments for sequential machines is based upon the use of distinguishing or characterizing sequences. Systematic procedures to build fault-detection experiments from these sequences have been devised by Hennie.

Our objective in this thesis is to show that fault-detection experiments can be greatly improved by an appropriate choice of the above sequences.

First, we will discover that a sensible improvement can be achieved by the use of adaptive sequences rather than preset ones. Second, for machines which do not have ditinguishing sequences, we will investigate the relationship between the structure of the partition associated with a set of characterizing sequences, and the complexity of the generated locating sequences. A procedure to select the best partition will be given.

THESIS SUPERVISOR: Professor Zvi Kohavi

TITLE: Assistant Professor of Electrical Engineering

A C K N O W L E D G M E N T S

The author wishes to express his sincere appreciation to Dr. Zvi Kohavi. His excellent guidance and suggestions were extremely valuable and most welcome.

C O N T E N T S

I. INTRODUCTION

The ability to determine whether or not a given machine operates correctly, from terminal measurements alone, becomes increasingly important as more and more machines are being manufactured in integrated form. Moreover, it is useful to try to shorten the length of such terminal measurements, especially when many units of the same machine have to be tested.

The first method to determine whether or not a finite-state machine operates in accordance with its state-table, has been devised by Moore [1]. However, Moore's method, which is based on the construction of the direct sum table, leads to extremely long experiments and is therefore impractical.

A more efficient approach has been developed by Hennie [2]. Hennie used two slightly different procedures according to whether or not the machine possesses distinguishing sequences. A summary of these procedures is given herein.

Kohavi [3] proposes still another procedure for the design of checking experiments. His method consists of modifying the original design of the machine as well as introducing additional output logic so as to provide the machine with special distinguishing sequences which lead to short fault-detection experiments.

The overall design of fault-detection experiments in this thesis will follow Hennie's procedures. The objective here is to show how the use of more appropriate sequences than those used up to now can lead to much shorter fault-detection experiments and a more flexible design.

II. B A C K G R O U N D

II. 1. Basic Definitions:

An experiment on a machine is the application of input sequences to the input terminals and the recording of the corresponding responses from its output terminals. If the experiment is designed to take the machine through all possible transitions in such a way that a definite conclusion can be reached as to whether or not the machine operates correctly, it is said to be a fault-detection experiment.

An experiment is said to be preset when the entire input sequence is specified in advance. An experiment is said to be adaptive when the choice of the input symbols to apply next is influenced by the way in which the machine has responded to the previous input symbols.

An experiment that can be used to determine the machine's initial state is called a distinguishing experiment. A distinguishing experiment can be preset or adaptive. The input sequence used in a preset distinguishing experiment is called preset distinguishing sequence (PDS). When we use an adaptive distinguishing experiment, the total input sequence used to pin down the initial state depends upon the initial state itself. We then have a set of sequences that we will call a set of adaptive distinguishing sequences (ADS). Not all machines have distinguishing experiments. It is sometimes possible to find both preset distinguishing experiments (each of which is associated with one PDS) and adaptive distinguishing experiments (each of which is associated with a set of ADS) for a given machine. Some machines have only adaptive distinguishing experiments, and some machines have no distinguishing experiment at all.

A sequence whose application forces the machine into a specific final state is called a synchronizing sequence. Some machines have synchronizing sequences, others do not.

A sequence whose application makes it possible to determine the final state of the machine by observing the output sequence is called a homing sequence. Every reduced machine has a homing sequence.

An extensive discussion of various sequences and experiments can be found in Gill [4] or Hennie [5].

II. 2. Summary of Hennie's Paper:

Since we will follow the general design of fault-detection experiments developed by Hennie, it is appropriate to summarize the general ideas and the different steps of the procedure.

Throughout this thesis we will make the following assumptions: the correctly operating machine is reduced, strongly-connection and does not suffer any malfunction that increases the number of its states.

The design of fault-detection experiments is divided into two parts. The first part brings the machine into some desired starting state. This part can be preset if the state table has a synchronizing sequence. Otherwise, we apply a homing sequence, deduce the present state and then apply an additional input sequence which will bring the machine into the desired starting state.

The choice of method for the second and main part of the experiment depends on whether or not the machine has a PDS.

In both methods the overall design is the same: first we identify all the states, then check all transitions. Although these two steps can be intermingled, we will keep them in that order for the sake of

simplicity.

Before explaining the methods used, let us introduce some notation. $S_1, S_2, \ldots, S_n$ denote the states of a n-state machine--the application of the input sequence X_k to this machine in state S_i leaves the machine in state Q_{ik}. $T(S_i, S_j)$ denotes an input sequence that takes the machine from state S_i to state S_j.

II. 2a. First Method (Machine with preset distinguishing sequence)

Step 1: Choose a particular PDS X_0, and begin the experiment by the following sequence, the machine being assumed to be in state S_1:

$$X_0\ T(Q_{10}, S_2)\ X_0\ T(Q_{20}, S_3) \ldots T(Q_{n-1,0}, S_n)\ X_0\ T(Q_{n0}, S_1)\ X_0$$

If the machine operates correctly, the corresponding output sequence Z will contain the output responses X_0 for all the n states of the machine, which we can then identify. If the output is different from Z, the machine is faulty.

Step 2: We now have to check, for each state, the transitions under all the input symbols. To check the x-transition from state S_j, when the machine is in state Q_{i0}, we use the following sequence:

$$T(Q_{i0}, S_{j-1})\ X_0\ T(Q_{j-1,0}, S_j) \times X_0$$

The sequence $T(Q_{i0}, S_{j-1})$ brings the machine in state S_{j-1}, then by the first step, the machine will certainly be in state S_j if it has just received $X_0\ T(Q_{j-1,0}, S_j)$ and responded to the X_0 part of this sequence by producing Z_{j-1}.

II. 2b. Second Method (Machines not having preset distinguishing sequences)

Even if the machine does not have PDS's, it is always possible to identify its states by their responses to an appropriately chosen set of input sequences, called characterizing sequences. The number of sequences in this set does not have to exceed n-1.

i) Machines having two characterizing sequences (X_1 and X_2)

Step 1: We build from X_1 and X_2 a locating sequence (LS) for each state - the LS for state S_i is the following:

$$\left[X_1 \; T(Q_{i1}, S_i) \right]^{n+1} \; X_2$$

From the corresponding output sequence, it can be shown that the machine, just before the application of X_2, was in the same state as before at least one of the n+1 applications of X_1.

A locating sequence built up from two characterizing sequences will be called a locating sequence of second order (A PDS is a locating sequence of first order).

Step 2: We then begin the fault-detection experiment by a sequence which is built up of alternating LS's and transfer sequences

$$L_1 \; T(Q_{12}, S_2) \; L_2 \; T(Q_{22}, S_3) \; L_3 \ldots T(Q_{n-1,2}, S_n) \; L_n$$

The transfer sequences bring the machine to the proper state for the next LS. At the end of this sequence, the machine is in Q_n

Step 3: Select a convenient LS, say L_k (associated with S_k) to be used in the remainder of the experiment.

The machine being in state Q_n, we determine the state it will be in at the end of L_k by the sequence

$$T(Q_n, S_k)\ L_k\ X_1\ T(Q_{i1}, S_k)\ L_k\ X_2$$

This sequence enables us to identify the state S_i in which the machine is at the end of L_k by the responses of S_i to X_1 and X_2.

Step 4: Check the transitions, beginning by those from S_i - the following sequence performs this function:

$$L_k \times X_1\ T(Q_j, S_k)\ L_k \times X_2$$

According to step 3, the machine is in S_i just before each application of x. S_j, the x-successor of S_i, is then determined by its responses to X_1 and X_2.

We can now check the transitions from S_j in the same manner, the same procedure applies then for the remaining transitions.

ii) Machines having many characterizing sequences

The design of fault-detection experiments, in this case, differs from the process described in the preceding section in two ways. First, the design of LS's becomes more complicated. Second, each transition has to be examined k times, k being the number of characterizing sequences.

For a machine having 3 characterizing sequences X_1, X_2, X_3, the LS for state S_i is

$$(Y_1^{n+1}\ Y_2)^{n+1}\ Y_1^{n+1}\ Y_3$$

where Y_j (j = 1, 2, 3) is the sequence that has X_j as its initial portion and takes the correctly operating machine from S_i back to S_i.

The above sequence is obtained by repeating $Y_1^{n+2}\ Y_2$ n+2 times, Y_2 being replaced by Y_3 in the last repetition.

The process repeats itself to obtain LS's of higher order.

III MACHINES HAVING A SET OF ADAPTIVE DISTINGUISHING SEQUENCES

The main tool used in the design of preset checking experiments is the P.D.S., for each state produces, in response to this sequence, a different output sequence. We can then discover the state the machine was in just before the application of the PDS by looking at the response. We also know this result can be achieved with adaptive distinguishing sequences (ADS), and that these sequences are more powerful than preset ones, requiring in general less input symbols.

It is thus appropriate to try to design fault detection experiments with ADS rather than with PDS. At first sight, the use of adaptive DS may seem incompatible with preset fault-detection experiments. We will see in the discussion to follow that this is not so, and that we actually obtain better results.

III. 1. An Analysis Example

Suppose we are given a machine and that the only information available about this machine is that it has at most four internal states.

We now apply the input sequence X to this machine which responds by producing the output sequence Z (fig. 1).

Time:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
X:	1	1	1	0	1	0	1	1	1	1	0	0	1	0	1	1	0	1	0	1	0	1	1
Z:	0	0	1	1	1	0	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0	0	1

- fig. 1 -

The notation S(t) will denote the state of the machine at time t.

From this experiment, what can we say about the state - table that represents the machine?

First, we observe that at times t=1 and t=2 the machine is supplied with the input sequence 11 and produces two different output sequences: 00 and 01. Therefore S(1) and S(2) must be distinct states.

At times t=3 and t=5, the machine is supplied with the input sequence 10, but again produces different output sequences: 11 and 10. Therefore S(3) and S(5) must be distinct states.

Moreover S(3) and S(5) respond to a 1 by producing a 1, while S(1) and S(2) respond to a 1 by producing a 0. Therefore S(1), S(2), S(3), and S(5) must be distinct from each other.

At this point we have identified four distinct states, and because the machine is assumed to have at most four states, we have identified all its states. Let us name the states A, B, C and D.

Further let S(2) be state A, S(3) state B, S(1) state C and S(5) state D.

Suppose then that at some time t, later in the experiment, the machine is presented with the input sequence 11 and produces the output sequence 01. Then S(t) can not be B or D, these two states responding to a 1 by a 1. Neither can S(t) be C, for C responds to 11 by 00. Hence, S(t) must be A. So S(8) $\equiv$ S(22) $\equiv$ A. By similar reasoning S(7) $\equiv$ C,

$S(13) \equiv S(16) \equiv B$, $S(10) \equiv S(18) \equiv D$.

From this it is now a simple matter to recover the states at each step of the experiment and deduce the complete state-table (fig. 2) of the machine (see details in appendix I).

	0	1
A	A, 0	B, 0
B	C, 0	D, 1
C	B, 0	A, 0
D	D, 1	B, 1

- fig. 2 -

The above reasoning is very close to the reasoning used to recover the state-table of a machine from a fault-detection experiment built up from a PDS.

The main difference stems from the fact that we have been able to identify the four states of the machine with the use of two different sequences rather than simply one (namely a PDS): 11 to identify A and C, 10 to identify B and D. Our experiment requires 23 input symbols.

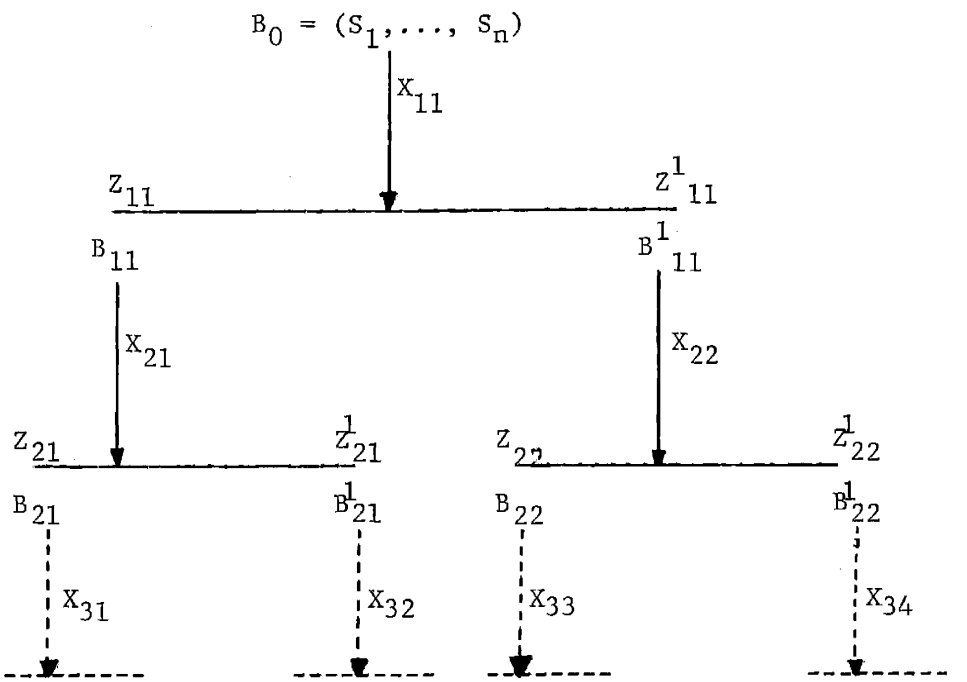
The shortest experiment I have been able to construct for this machine with a PDS(011) requires 44 symbols. The question then arises as to what property of the sequences 10 and 11 makes them particularly useful. The answer to this is that 11 and 10 form a set of ADS, as the reader may verify from the state table in fig. 2.

III. 2. A General Property

We will now turn to prove that this feature is a general property: for a machine having a set of ADS, a fault detection experiment

can be designed with Method 1, in which X_0 (which stood for a PDS) will now stand for ADS.

Proof: Let us consider a strongly-connected, reduced, N-state machine, and assume there exists an adaptive distinguishing experiment for this machine. This experiment can be represented by the tree in fig. 3.



- fig. 3 -

The letters B_{ij} or B^1_{ij} represent blocks of states. Each block represents our knowledge of the initial state of the machine at each step of the experiment. For example if $B^1_{21} = (S_1, S_4, S_9)$, this means that after applying the input sequence $X_{11}\ X_{21}$ (see fig. 3), if the machine responds by producing the output sequence $Z_{11}\ Z^1_{21}$, it must have been in S_1 or S_4 or S_9 at the beginning of the experiment.

Recall that X_{ij} is an input sequence which may have more than one input symbol. Recall also that each branch in the tree is terminated

when the block associated with it is a single state block.

According to this tree, the adaptive distinguishing experiment is to be executed in the following order:

Apply X_{11}

1. If the output sequence is Z_{11}:

   Apply X_{21}

   1.1 If the output sequence is Z_{21}:

   Apply X_{31}

   1.1.1 If .....

   1.1.2 If .....

   1.2 If the output sequence is Z^1_{21}:

   Apply X_{32}

   1.2.1 .....

   1.2.2 .....

2. If the output sequence is Z^1_{11}:

   Apply X_{22}

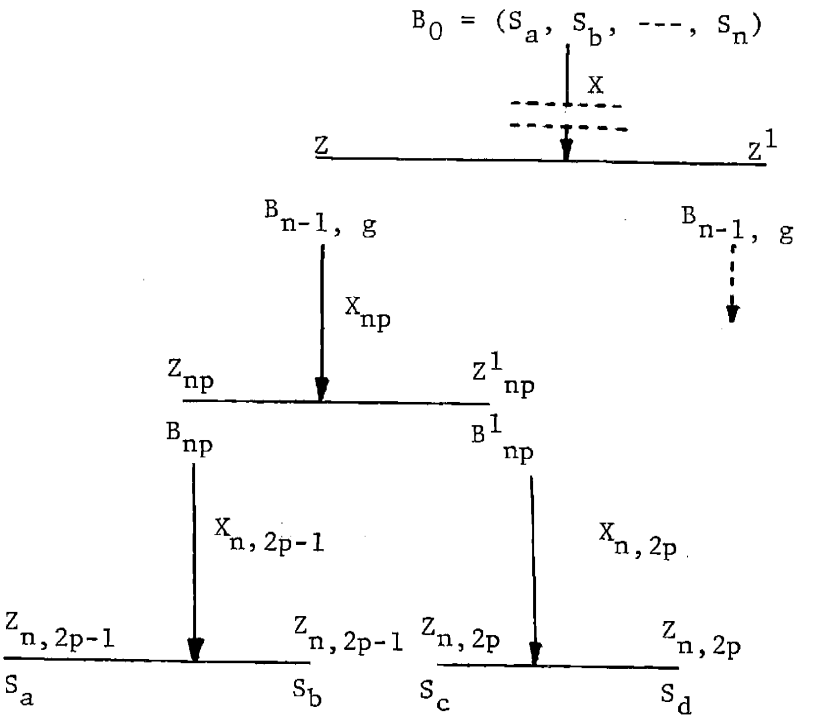
   2.1 If the output sequence is Z_{22}:
   .....
   2.1 If the output sequence is Z^1_{22}:
   .....

Every state of the machine, then is associated with a sequence of input sequences X_{ij}, and a corresponding sequence of output sequences Z_{ij} or Z^1_{ij}.

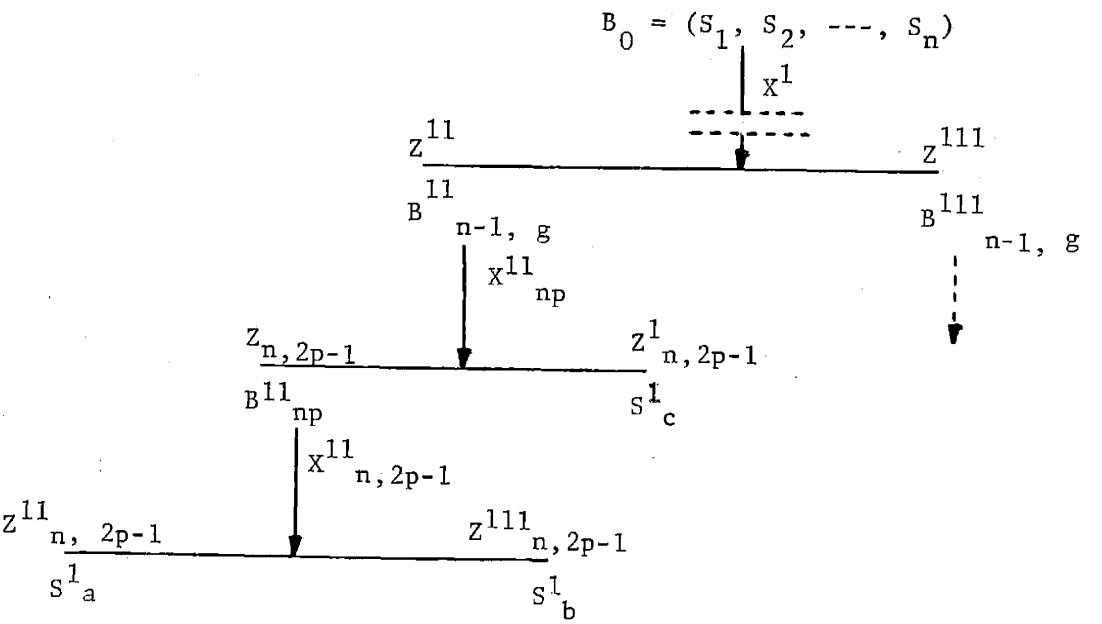
For our proof, we have now to consider the extremities of the tree of fig. 3. They can be of the form represented either in fig. 4a or in fig. 4b, in which X (resp. X^1) is the input sequence which leads to the block $B_{n-1,g}$ (resp. $B^{11}_{n-1,\,g}$) from the block B_0, the machine producing

the output sequence Z (resp. Z^{11}).

Let us consider the case of fig. 4a. The ADS's associated with S_a, S_b, S_c, and S_d and their corresponding output sequences are:



- fig. 4a -



- fig. 4b -

$$\text{ADS}(S_a): \quad \begin{matrix} X & X_{np} & X_{n,2p-1} \\ Z & Z_{np} & Z_{n,2p-1} \end{matrix} \qquad \text{ADS}(S_b): \quad \begin{matrix} X & X_{np} & X_{n,2p-1} \\ Z & Z_{np} & Z^1_{n,2p-1} \end{matrix}$$

$$\text{ADS}(S_c): \quad \begin{matrix} X & X_{np} & X_{n,2p} \\ Z & Z^1_{np} & Z_{n,2p} \end{matrix} \qquad \text{ADS}(S_d): \quad \begin{matrix} X & X_{np} & X_{n,2p} \\ Z & Z^1_{np} & Z^1_{n,2p} \end{matrix}$$

Let us now return to our fault-detection experiment problem, and build a fault-detection experiment E using Method 1 where we now replace X_0 by the ADS associated with each state.

Provided with E and the response of the machine, which is supposed to operate correctly, we will see that we can recover the N states of the machine:

First, in the experiment E, the input sequence $X\ X_{np}\ X_{n,2p-1}$ will figure once with the output sequence $Z\ Z_{np}\ Z_{n,2p-1}$ and once with the output sequence $Z\ Z_{np}\ Z^1_{n,2p-1}$.

From these two different output sequences for the same input sequence, we can identify two states. For the sake of clarity, we will call them S_a and S_b.

Further, we can conclude that the two distinct states S_a and S_b respond to $X\ X_{np}$ by the same output sequence $Z\ Z_{np}$.

By the same reasoning we can say that two distinct states, we call them S_c and S_d respond to $X\ X_{np}$ by the same output sequence $Z\ Z^1_{np}$.

From their different responses to the input sequence $X\ X_{np}$, we then know that S_a and S_b are distinct from S_c and S_d, and, by looking at those responses, that the machine has four distinct states that respond to X by the same output sequence Z.

In the case of fig. 4b, we would first deduce that S^1_a is different from S^1_b, from their different responses to $X^1_{np}\ X^1_{n,2p-1}$, and

then that S^1_c is different from S^1_a and S^1_b, from its different response to X_{np}.

From the nature of the tree, it becomes now obvious that the above reasoning may be repeated as many times asnecessary to finally reach the conclusion that the machine has N states.

In the same way, if during the experiment the same ADS is applied several times, and each time the machine yields the same output sequence, then we can say, that before each application of this input sequence, the machine was in the same state.

From these conclusions, the fact that we can design fault-detection experiments with ADS's instead of PDS should be now clear.

We may then expect these fault detection experiments to be in general much shorter. This is true because ADS's are shorter than PDS's and therefore repeated use of ADS's multiplies this gain. Further, use of shorter sequences in general increases the number of shortcuts. The above analysis example clearly demonstrates this fact.

Another appealing feature of ADS's is that some machines which do not have PDS's, and for which a fault-detection experiment would require the use of characterizing and locating sequences, do have ADS. Consequently, for those machines, a fault-detection experiment can be designed following Method 1, with the use of ADS's.

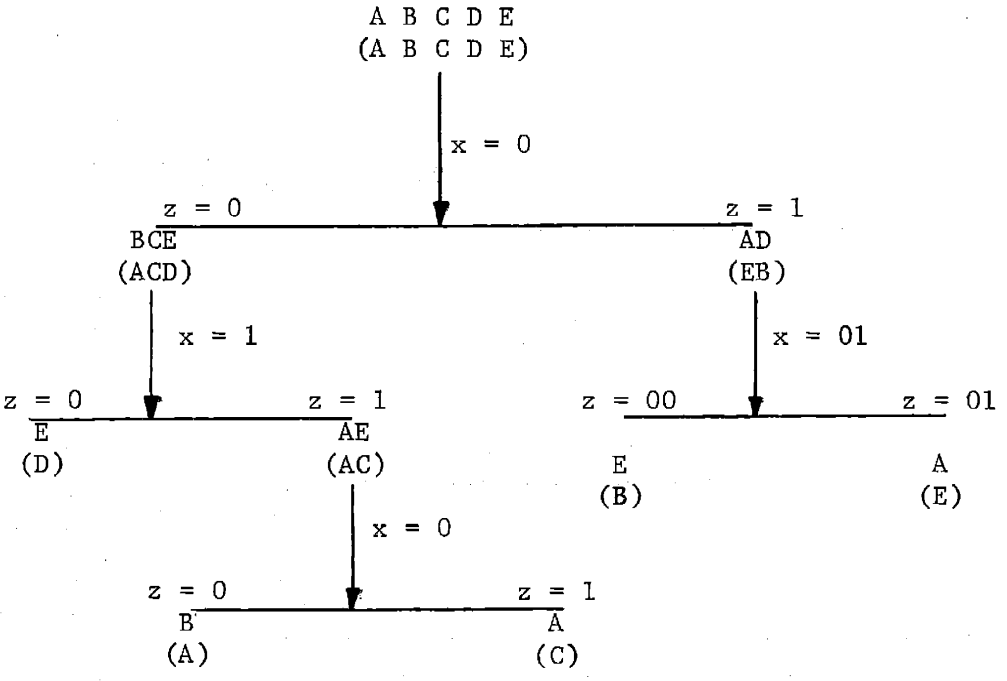
III. 3. A synthesis example

We will design a fault-detection experiment for the machine represented by the state-table of fig. 5:

	0	1
A	B, 0	C, 0
B	D, 1	A, 1
C	C, 0	E, 1
D	E, 0	C, 0
E	A, 1	E, 0

- fig. 5 -

This machine does not possess any PDS. But there exists an adaptive distinguishing experiment represented by the tree of fig. 6.



- fig. 6 -

In this tree, at each step in the course of the distinguishing experiment, the letters in parentheses represent our knowledge of the initial state if this step is actually reached in the experiment. The

letters just above represent the corresponding present state, under the same conditions. For example, if after applying a 0, the output is a 1, then the initial state was B or E and the present state is D or A, respectively.

From the tree, we deduce the ADS associated with each state, and its corresponding output response (fig. 7)

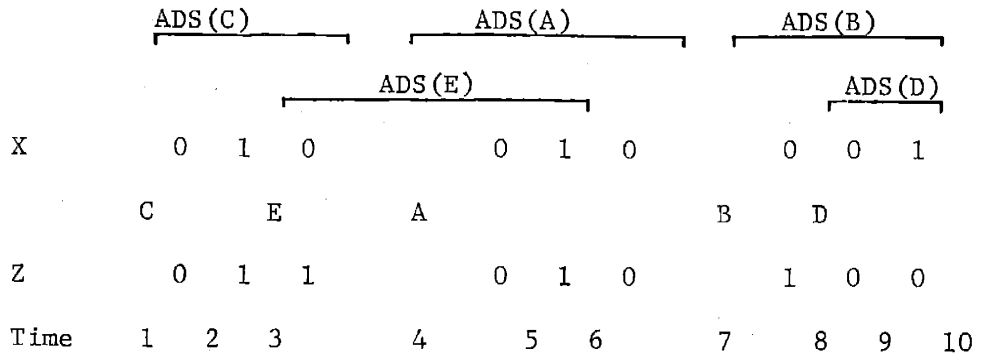
State	ADS	Response
A	010	010
B	001	100
C	010	011
D	01	00
E	001	101

- fig. 7 -

With this set of ADS we can design a fault detection experiment by applying Method 1, where X_0 will stand for these ADS's.

This experiment will begin with the machine in state C (we will see why later). Noting that 111 is a synchronizing sequence which leaves the machine in state E, the sequence 11101 will take the machine to state C, whatever the initial state was.

The first part of the experiment will consist of applying the ADS for each state, so as to identify all the states. This first part is represented in fig. 8:



- fig. 8 -

From this, we see, first, that S(3) and S(7) are different (different responses to 001); second that S(1) and S(4) are different (different responses to 010), and that they are both different from S(8) (different responses to 01); third that S(3) and S(7) respond to a 0 by a 1, while S(1), S(4) and S(8) respond to a zero by a 0. Hence S(1), S(3), S(4), S(7) and S(8) are five different states.

Since the particular name assigned to each of the above states is unimportant, we will name them by the letters that are assigned to the corresponding states in the given state-table (fig. 5).

We have chosen to begin the experiment with the machine in state C, because this leads to an important overlapping of the ADS's and gives rise to free transition checkings.

We may now begin the second part of the experiment. For this, we need to know first which state is S(10). To check that S(10) is state E, we apply the ADS 001 which gives the response 101. S(10), which is then E, followed by a 0 implies that S(11) is A (same situation than from S(3)). So, if at t = 13, we apply a 0, s (14) must be B. (fig. 9)

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14
X		0	1	0	0	1	0	0	0	1	0	0	1	0
State	C		E	A			B	D		E	A			B
Z		0	1	1	0	1	0	1	0	0	1	0	1	0

- fig. 9 -

Before starting checking transitions, it is appropriate to make the following remarks:

Since S(1) is C and S(3) is E, if we check that the 0-transition from C leaves the machine is C, we will consequently know that the 1-transition from C leaves the machine in E (just by looking at fig. 9). In the same way, S(8) being D and S(10) being E, checking that the 0-transition from D leaves the machine in E, will tell us that the 1-transition from E leads the machine to E.

By the same remark, the 0-transition from A will be also useful knowing, because S(4) ≡ A and the state only three units of time later is known to be B. So, if we design our experiment according to these remarks we will gain some free transition checks.

It is generally worth keeping the preceding observations in mind when designing a fault-detection experiment, because they appear to be useful in may cases (at least when some ADS is relatively short).

Let us now return to ***our*** example. Recall that we left the design of ***our*** experiment at t = 14 with S(14) ≡ B. The state-table tells us that S(13) should equal A, but we have not yet been able to deduce it from the experiment; this information would be useful for checking the 0-transition from A, because we know that S(14) ≡ B. In order to do so apply 10 from S(14), this tells us that S(13) ≡ A from the response of

S(13) to 010. This checks the 0-transition from A. From this we deduce that S(12) ≡ B, and then, S(13) being A, that under an input of 1 the machine goes from B to A and produces a 1.

We have now designed the experiment up to t = 16, with S(16) ≡ B. An input of 0 will lead the machine to D, from which we can check the 0-transition by applying a 0, followed by 001 (ADS(E)). The 1-transition from E is then checked, as noted in the above remark.

This leaves the machine in S(21) which is A. We check the 1-transition from this state by applying a 1, followed by 010 (ADS(C)). This leaves the machine in S(25) which is A. At this point the experiment is represented in fig. 10, and the knowledge of the state-table we have gained so far, in fig. 11.

Time	1	2	3	4	5	6	7	8	9	10	11	12
X	0	1	0	0	1	0	0	0	1	0	0	1
State	C		E	A	B	A	B	D	E	E	A	B
Z	0	1	1	0	1	0	1	0	0	1	0	1

Time	13	14	15	16	17	18	19	20	21	22	23	24	25
X	0	1	0	0	0	0	0	1	1	0	1	0	
State	A	B	A	B	D	E			A	C		E	A
Z	0	1	0	1	0	1	0	1	0	0	1	1	

- fig. 10 -

	0	1
A	B, 0	C, 0
B	D, 1	A, 1
C	, 0	
D	E, 0	
E	A, 0	E, 0

- fig. 11 -

From fig. 10, S(2) is a state which responds to a 1 by a 1 and leaves the machine in E. We then deduce from the state-table of fig. 11 that S(2) cannot be A or B or E. Hence, either S(2) ≡ C or S(2) ≡ D. Therefore, if we check that the 1-transition from D leads us to C, and not to E (S(3)), we will then conclude that S(2) ≡ C. This is done from S(25) by applying 00, which brings the machine in D, then a 1, followed by 010 (ADS(C)). After this we know that S(2) ≡ C. Since S(1) is C, and S(3) is E, the last transitions (0- and 1-transitions from C) are checked free.

The complete experiment is thus the following (Recall that 11101 brings the machine in state C):

X	1	1	1	0	1	0	1	0	0	1	0	0	0	1	0	0	1	0	1	
State			E	A	C	C	E	A	B	A	B	D	E	E	A	B	A	B		
Z				1	0	0	1	1	0	1	0	1	0	0	1	0	1	0	1	

X (Cont.)	0	0	0	0	0	1	1	0	1	0	0	0	1	0	1	0				
State (Cont)	A	B	D	E	A	B	A	C	C	E	A	B	D	C	C	A	A			
Z (Cont.)	0	1	0	1	0	1	0	0	1	1	0	1	0	0	1	1				

IV. MACHINES NOT HAVING A SET OF ADAPTIVE DISTINGUISHING SEQUENCES

We now consider the problem of designing checking experiments for machines which do not have a set of ADS's, i.e. those for which we have to apply Method 2 (see part II).

Our goal will be to show how the choice of an appropriate set of characterizing sequences (CS) can influence the length and simplify

the design of checking experiments.

Each of the three following examples will develop several interesting points.

IV. 1. Example 1

Let us consider the following machine:

	0	1
A	B, 0	D, 0
B	A, 0	B, 0
C	D, 1	A, 0
D	D, 1	C, 0

- fig. 12 -

For this machine, Hennie [2] selects the CS's 0 and 10, which are the shortest ones, and deduces the locating sequences (LS):

A
0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 1

B
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0

C
0 1 0 1 0 1 0 1 1 0
1 0 1 0 1 0 1 0 0 0

D
0 0 0 0 1 0
1 1 1 1 0 1

The complete experiment using these LS's requires 152 input symbols.

Another set of CS's for this machine is 010 and 10. The responses of each state to those sequences are:

State	Response to 010	Response to 10
A	000	01
B	001	00
C	101	00
D	101	01

Let us design the LS's from this set of CS's.

First, A is the only state that responds to 010 by 000. It can then be distinguished from the others by the sequence 010 alone. The same reasoning applies for state B. The LS's for A and B, which are then of first order, are:

$$A \quad \begin{matrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix} \qquad\qquad B \quad \begin{matrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

From this we know that at least two states (A and B) respond to a 0 by a 0. Since from the LS's of C and D, we will discover that C and D respond to a 0 by a 1, the input sequence 0 (instead of the complete 010) will be sufficient to distinguish these states from both A and B. The CS 10 will distinguish C from D. From this the LS's for C and D are:

$$C \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}^3 \quad \begin{matrix} 1 & 0 \\ 0 & 0 \end{matrix} \qquad\qquad D \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix}^3 \quad \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$$

A fault detection experiment, using these LS's requires only 35 symbols (see Appendix II)

The reason why we have been able to achieve such a short experiment is the existence of some LS's of first order. In general, such short

sequences are very useful for the following reasons: first, every LS has to be applied at least once, thus, the shorter, the better; second, in the transition checking part of Method 2, we use a LS (named L_k) twice in each transition check, the existence of a short LS will then give a very convenient L_k; third, a transition leading to a state associated with a LS of first order needs only to be checked once.

It follows that the choice of a good set of CS's is a very influential factor in the design and the length of an experiment.

Before searching for a good set of CS's, it is worthwhile to indicate the convenience of utilizing the partitions associated with every input sequence:

Every input sequence can be associated with a partition

$$P = (B_1, B_2, \ldots, B_p)$$

In which the B_i's represent blocks of states, two states being in the same block if they respond to the input sequence by the same output sequence.

In our example, the partitions associated with the CS's 010 and 10 are respectively:

$$P_1 = (A, B, CD)$$

$$P_2 = (AD, BC)$$

The fact that 010 and 10 are a set of CS's means in this notation that*

$$P_1 \cdot P_2 = (A, B, C, D) = 0$$

* Recall that the product of two partitions P_1 and P_2 is a partition P_3 such that two states are in the same block in P_3 if they are in the same block both in P_1 and in P_2.

Looking at the partition P_1 associated with 010, the fact that A and B require LS's of first order is clearly represented by the presence of A and B in a single-state block of P_1.

In the same way, the partitions associated with the CS's 0 and 10 (utilized in Hennie's paper) are respectively:

$$P^1_1 = (AB, CD)$$
$$P^1_2 = (AD, BC)$$

$P^1_1 \cdot P^1_2 = 0$ since 0 and 10 form a set of CS's.

None of these partitions possesses a single-state block; consequently there are no LS's of order one.

We see from this that partitions can represent in a convenient way the properties associated with a set of CS's. We will use them for that reason in the remainder of the discussion.

Remark

Recall that the general formula for a LS of second order is, with the notations of II. 2b:

$$\left[X_1\ T(Q_{i1}, S_i)\right]^{n+1} X_2$$

Of course, the following formula is valid as well:

$$\left[X_2\ T(Q_{i2}, S_i)\right]^{n+1} X_1$$

Using the first (second) formula, Hennie [2] concluded that the state before the application of X_2 (X_1) is identical to the state before some application of X_1 (X_2) in the LS. The deduction process that enables us to reach that conclusion is based solely on the use of X_1 (X_2).

In other words, a more important role is given to X_1 (X_2) than to X_2 (X_1).

In one example, where $X_1 = 010$, and $X_2 = 10$, the LS's have been obtained from the first formula. Had we chosen the second formula, all the LS's would have been of second order, because there is no single state block in the partition P_2 associated with $X_2 = 10$.

IV. 2. Example 2

The most convenient way to list all the partitions for a given machine is to draw a homing tree. From what we have learned, it is reasonable to draw an adaptive tree rather than a preset one.

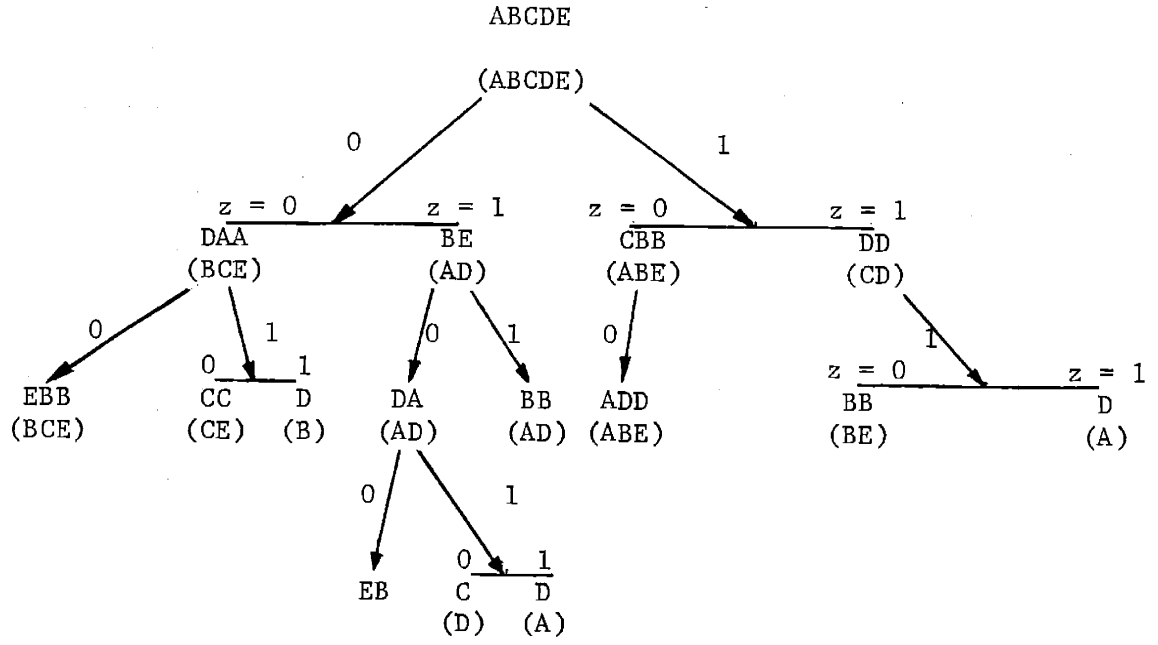
In this example we will see how, from this tree, we can deduce the partitions. Since we derive these partitions from an adaptive tree, they may now be associated with a set of adaptive sequences rather than with a single preset sequence.

We will then show how the LS's are built in this case.

Let us consider the following machine, and draw it adaptive tree:

	0	1
A	B, 1	C, 0
B	D, 0	B, 0
C	A, 0	D, 1
D	E, 1	D, 1
E	A, 0	B, 0

- fig. 13 -



- fig. 14 -

Recall that the letters in parentheses represent the initial state in which the machine was at the beginning of the experiment, and the letters directly above represent the corresponding present state.

In this tree, we are not interested in finding homing sequences. Consequently, a branch associated with an homogeneous uncertainty is not to be stopped, except:

1. When all the states in this uncertaintly are the same (for example: we can stop at DD on the right of the tree, because we will never split C and D from that point), or

2. When this uncertainty will lead to a splitting of the states which has already been met in the tree. (for example, we stop after EBB, on the left of the tree, because at this point we cannot expect to obtain better than the splitting of (BCE) in (CE) and (B) already gotten by the sequence 01).

From this tree we deduce the following six partitions:

P_1 = (AD, BCE)

P_2 = (A, D, BCE)

P_3 = (AD, B, CE)

P_4 = (A, B, CE, D)

P_5 = (ABE, CD)

P_6 = (A, BE, CD)

The partition P_1 is obtained by remarking that there exists a sequence (namely: 0) which distinguishes between the blocks AD and BCE.

P_4 is obtained by remarking that there exists a set of adaptive sequences (namely: 01 and 001) which distinguishes between A, B, CE and D.

The other partitions are obtained in the same way.

It is important at this point not to confuse a set of adaptive sequences, associated with <u>one</u> partition, and a set of CS's, associated with several partitions P_1, ..., P_p such that $P_1 \cdot P_2 \cdot \cdots \cdot P_p = 0$.

The design of LS's using the partitions P_4 and P_6 ($P_4 \cdot P_6 = 0$) will make this point clear:

P_4 is associated with the set of adaptive sequences 01 and 001. The table below, deduced from the tree indicates which sequence is to be used for each state, and gives the corresponding output sequence.

Initial State	Input Sequence	Output Sequence
A	0 0 1	1 0 1
B	0 1	0 1
C	0 1	0 0
D	0 0 1	1 0 0
E	0 1	0 0

The sequence associated with P_5 is 1; the output is 0 if the initial state is A or B or E, 1 if the initial state is C or D.

For designing the LS's we will use the formula $[X_1\ T]^{n+1}\ X_2$, X_1 being either 001 or 01, and X_2 being 1.

The states A, B, and D being in a single-state block of P_4 possess a LS of first order:

$$A: \begin{matrix} 0 & 0 & 1 \\ 1 & 0 & 1 \end{matrix} \qquad B: \begin{matrix} 0 & 1 \\ 0 & 1 \end{matrix} \qquad D: \begin{matrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{matrix}$$

From this we deduce that at most two states can respond to 01 and 00. Consequently, the LS's for C and E are

$$C: \begin{pmatrix} \overline{0\ \ 1}^{X_1} \\ 0\ \ 0 \end{pmatrix}^3 \quad \overline{1}^{X_2} \quad 1 \qquad \text{that is} \qquad \begin{matrix} 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}$$

$$E: \begin{pmatrix} \overline{0\ \ 1}^{X_1}\ 1\ \ 0 \\ 0\ \ 0\ \ 1\ \ 1 \end{pmatrix}^3 \quad \overline{1}^{X_2} \quad 0 \qquad \text{that is} \qquad \begin{matrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{matrix}$$

Remark

A fault detection experiment taking advantage of these LS's requires 45 symbols. (see Appendix III).

The reader may verify that every other fault-detection experiment which does not use the partition P_4 leads to a much larger number of symbols. Noting then that P_4 cannot be obtained from a preset tree, we show once more the advantage of adaptive sequences over preset ones.

IV. 3. Example 3

Recall (see remark of IV. 1) that to deduce, from the LS

$[X_1\ T]^{n+1}\ X_2$, that the state just before X_2 is the same as the state before some application of X_1 in this LS, we use only X_1.

X_2 is used then to distinguish between all the states that respond to X_1 by the same output sequence, i.e., the states in a same block of the partition P_1 associated with X_1.

In other words, X_1 is used to split the states into blocks, and X_2 is used to split the states within each block. But, in this deduction process, nowhere is it required to use the same X_2 for all the blocks of P_1.

All that we really need is, associated with $P_1 = (B_1, \ldots, B_q)$ a set of partitions $P_2, \ldots, P_p$ such that*:

$$P_1 \cdot P_2 = B_1\ 0$$
$$P_1 \cdot P_3 = B_2\ 0$$
$$\vdots$$
$$P_1 \cdot P_p = B_q\ 0$$

A very important role is then attributed to P_1, we will call this partition a <u>principal partition</u>.

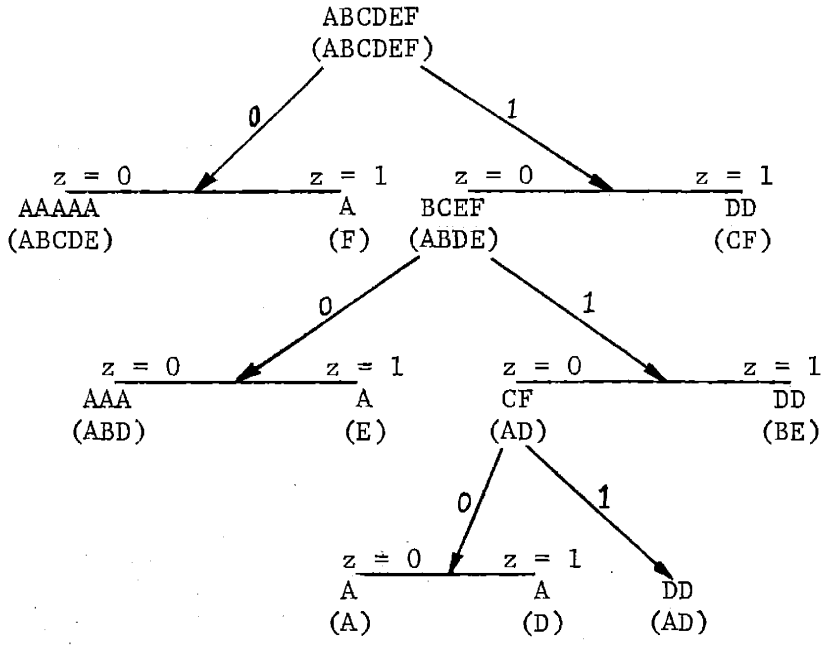
The following example illustrates this point.

Let us draw the adaptive tree and deduce the partitions for the reduced and strongly connected machine of fig. 15:

	0	1
A	A, 0	B, 0
B	A, 0	C, 0
C	A, 0	D, 1
D	A, 0	E, 0
E	A, 0	F, 0
F	A, 1	D, 1

- fig. 15 -

* $P_1 \cdot P_2 = B_1\ 0$ means that no two states of block B_1, belonging to P_1, are in the same block of P_2; in other words, $P_1 \cdot P_2$ is the 0-partition relatively to the states in B_1.



- fig. 16 -

The partitions are:

P_1 = (ABCDE, F)

P_2 = (ABDE, CF)

P_3 = (ABD, CF, E)

P_4 = (AD, BE, CF)

P_5 = (A, BE, CF, D)

For this machine, we see that it does not exist partitions P_i and P_j such that $P_i \cdot P_j = 0$.

Hence, without the above discussion, LS's of order at least three would have been needed.

But we may now notice that, P_5 playing the role of the principal partition:

$P_5 \cdot P_1 = {}_{CF}\, 0$ and $P_5 \cdot P_3 = {}_{BE}\, 0$

A and D being in a single block of P_5, no LS of order higher than two is now needed.

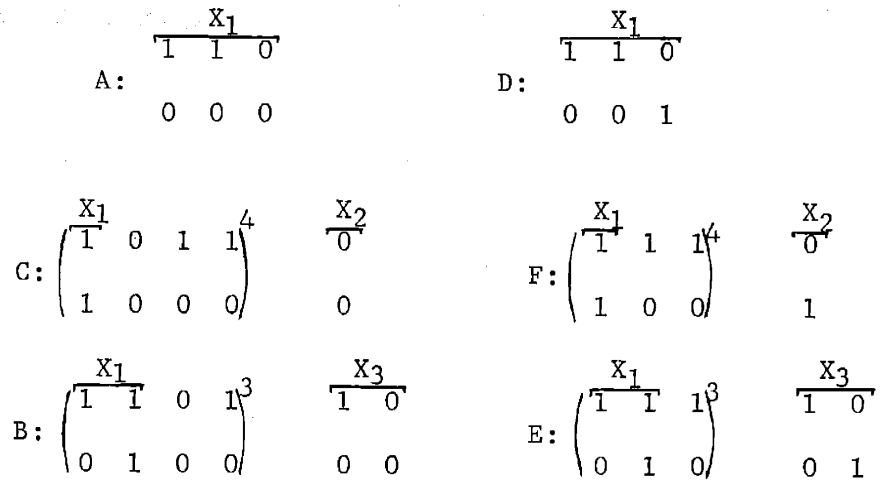
As an illustration we will now build these LS's.

P_5 is associated with the set of sequences 1, 11, 110 (referred as X_1 in fig. 17)

P_1 is associated with the sequence 0 (referred as X_2 in fig. 17)

P_3 is associated with the sequence 10 (referred as X_3 in fig. 17)

The LS's generated from these partitions are the following:



- fig. 17 -

In the LS's for C and F, the "exponent" is 4 (3+1), because from the set of LS's, we deduce that at most 3 states can respond to a 1 by a 1.

When C and F will have been distinguished, we will know that at most two states can respond to 11 by 01; this explains the "exponent": 3 (2+1) in the LS's for B and E.

Let us now be concerned in LS's of order higher than two, and design, for the machine of fig (15) again, a new set of LS's this time using P_1 = (ABCDE, F) as the principal partition. Since we will get

longer LS's than above, this design is not interesting for itself, but it will illustrate another interesting property.

First, we note that F is a single-state block of P_1, associated with the sequence 0. Hence, the LS for F is

$$F: \begin{matrix} 0 \\ 1 \end{matrix}$$

We cannot find any partition P_i such that

$$P_1 \cdot P_i = {}_{ABCDE}\,0 \text{ , however}$$
$$P_1 \cdot P_5 = (A, BE, C, D, F) = Q$$

A, C and D are now each in a single block of Q, and are consequently completely distinguished by X_1 and X_2, associated with P_1 and P_5 respectively; where X_1 is the sequence 0, and X_2 the set of sequences 1, 11, 110. The LS's for A, C and D, which are then of second order, are the following:

$$A: \begin{pmatrix} \overset{X_1}{0} & 6 & \overset{X_2}{1\ 1\ 0} \\ 0 & & 0\ 0\ 0 \end{pmatrix} \qquad C: \begin{pmatrix} \overset{X_1}{0\ 1\ 1} & 6 & \overset{X_2}{1} \\ 0\ 0\ 0 & & 1 \end{pmatrix} \qquad D: \begin{pmatrix} \overset{X_1}{0\ 1\ 1\ 1} & 6 & \overset{X_2}{1\ 1\ 0} \\ 0\ 0\ 0\ 1 & & 0\ 0\ 1 \end{pmatrix}$$

In order to distinguish B from E, we need now a partition P_i such that

$$Q \cdot P_i = {}_{BE}\,0$$

The only such partition is P_3, associated with the sequence $X_3 = 10$

The LS's for B and E can now be built by the Method developed by Hennie [2] for LS's of third order. We then have:

$$B:\left[\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}^{6} \quad \begin{matrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{matrix}^{3} \quad \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}^{6} \quad \begin{matrix} 1 & 0 \\ 0 & 0 \end{matrix}\right]$$

$$E:\left[\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}^{6} \quad \begin{matrix} 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}^{3} \quad \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}^{6} \quad \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}\right]$$

From the LS for F, at most 5 states can respond to a 0 by a 0, this explains the "exponent" 6.

From the LS's for F, A, C and D, only two states can respond to 0 by 0 and to 11 by 01. This explains the "exponent" 3.

Of course, it would be foolish to use this last set of LS's in a fault-detection experiment, because we have previously found a set of much shorter ones; the main intent of this example was to show that even if a block of states from the principal partition requires p partitions to distinguish between all its states, some states from this block may be associated with LS's of order less than p + 1.

More precisely, if in order to distinguish between all the states of a block B_1, belonging to the principal partition P_1, the partitions P_2, P_3, ..., P_q, such that

$P_1 \cdot P_2 \cdot \text{---} \cdot P_q = {}_{B_1}0$, are required, then: the states in a single-state block of the partition $P_1 \cdot P_2$ possess LS's of order two; the states in a single-state block of the partition $P_1 \cdot P_2 \cdot P_3$ possess LS's of order three, and so on.

Remark. From the fact that at most q-1 input sequences are neces-

sary to distinguish between q states, it follows now that the states in a q-state block of the principal partition have locating sequences of order at most q.

IV. 4. The choice of the principal partition

It follows from the above discussion that the choice of the best principal partition is a determining factor in the design of fault detection experiments. A general procedure which would tell us which is the best principal partition would then be very useful.

Unfortunately, it appears that selecting the best principal partition for any one given machine is an intricate problem, whose general solution is made harder by our poor knowledge of finite-state machines with a large number of states.

The procedure given below is a first attempt to solve this problem. We may expect from it to give the principal partition which leads to the shortest experiment for many machines, even if it does not for every one.

Before explaining this procedure, we will define the order of a partition:

"A partition P is said to be of order p if p is the order of the LS of highest order, generated by P, when P is chosen as the principal partition".

Procedure

Step 1: From the state-table, draw the adaptive tree, and list all the partitions.

Step 2: Determine the order of each partition.

Step 3: Select the partitions of the lowest order r. They form the set S_1. (For obvious reasons: the length of a LS increases at a very high rate, with its order; further, a transition leading to a state associated with a LS of order p has to be checked p times).

If S_1 contains only one partition, take it as a principal partition. Otherwise, go to step 4.

Step 4: Among S_1, select the partitions associated with the LS's of the lowest order m. They form the set S_2. (reason: recall that to check the transitions, a particular LS (called L_k in II) is needed twice for each checking, it is then essential to possess at least one very short LS).

If S_2 contains only one partition, take it as principal partition. Otherwise go to step 5.

Step 5: Among S_2, select the partitions which generate the least number of LS's of the highest order r. They form the set S_3 (same reasons as in step 3).

If S_3 contains only one partition, take it as principal partition. Otherwise go to step 6.

Step 6:

$r \neq m$: go to step 5, replacing S_2 by S_3 and r by r-1.

$r = m$: If only one partition remains take it as principal partition. If we still have to choose among several partitions, then compute the total length of the LS's and take as principal partition, the partition for which this length

is the smallest. (reason: In this last case, all the partitions will have the same structure, consequently the length of the experiment is influenced only by the length of the LS's).

V. C O N C L U S I O N

Several ways of improving the design of fault detection experiments have been presented.

First, we have seen that the systematic use, for all machines, of adaptive sequences, leads to shorter experiments; and that for every machine which possesses a set of adaptive distinguishing sequences, fault-detection experiments can be designed following Method 1.

Second, for machines which do not have adaptive distinguishing sequences, we have derived the concept of principal partition, associated with a set of adaptive characterizing sequences. We have seen how the design of locating sequences is influenced by the structure of the principal partition, and then come to the conclusion that this principal partition is a determining factor of the length and the complexity of a fault-detection experiment. A first attempt to a general procedure to select the best principal partition has been presented. This procedure has been deduced from the properties of the partitions illustrated in this thesis.

It is believed that some work is needed in order to acquire a deeper understanding of the relationship between the structure of the principal partition and the design of locating sequences. This would help to design a better procedure and achieve still shorter experiments.

A P P E N D I X I

(related to III. 1.)

In III. 1., we left the experiment with a knowledge of the states as shown by the following pattern:

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
X	1	1	1	0	1	0	1	1	1	1	0	0	1	0	1	1	0	1	0	1	0	1	1
S(t)	C	A	B		D		C	A		D			B			B		D			A		
Z	0	0	1	1	1	0	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0	0	1

At this point our knowledge of the state table is the following:

	0	1
A		B, 0
B		, 1
C		A, 0
D		, 1

From this state-table, we now deduce that S(9) ≡ S(23) ≡ B, this tells us that an input of 1 causes a transition from B to D (because S(10) is known to be D). This in turn tells us that S(4) ≡ S(14) ≡ S(17) ≡ D, and that an input of 0 causes a transition from D to D with an output of 1 (because S(5) is known to be D).

From this, S(15) ≡ D, and S(16) being B, we deduce that an input of 1 causes a transition from B to D.

At this point, our knowledge of the states in the course of the experiment, and of the state table are:

Time:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
X:	1	1	1	0	1	0	1	1	1	1	0	0	1	0	1	1	0	1	0	1	0	1	1
S(t):	C	A	B	D	C	A	D			B				B		D				A			
Z	0	0	1	1	1	0	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0	0	1

	0	1
A		B, 0
B		D, 1
C		A, 0
D	D, 1	B, 1

From this, S(6) ≡ S(11) ≡ S(19) ≡ B, so an input of 0 causes a transition from B to C with an output of 0 (because S(7) is known to be C).

Then, S(12) ≡ S(20). Thus, an input of 0 causes a transition from C to B with an output of 0.

S(20) being state C, it follows that S(21) ≡ A. And, from S(22) ≡ A, we deduce that an input of 0 causes a transition from A to A with an output of 0.

Our knowledge of the state table is now complete:

	0	1
A	A, 0	B, 0
B	C, 0	D, 1
C	B, 0	A, 0
D	D, 1	B, 1

A P P E N D I X II

(related to IV. 1)

The experiment for the machine of fig. 12 is to be started with the machine in state A, and is the following:

								LS(C)						LS(B)			
				LS(D)							LS(A)						
X	1	0	0	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1
Z	0	1	1	1	0	1	0	1	0	1	0	0	0	0	0	0	1	0
Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

		LS(A)			LS(A)				LS(B)								
X	1	0	1	0	0	1	0	0	0	1	0	1	1	1	1	0	0
Z	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1
Time	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

We will now explain how, from this experiment, we can recover the state table.

a) First from LS(A), LS(B), LS(C) and LS(D) we recover the four states of the machine. We shall name them by the same name assigned to the corresponding states in the state-table of fig. 12:

$$S(13) \equiv S(20) \equiv S(23) \equiv A$$

$$S(15) \equiv S(27) \equiv B$$

$$S(12) \equiv C$$

$$S(5) \equiv D$$

From this*: $C \xrightarrow[0]{1} A$

b) $S(20) \equiv S(23) \equiv A$ implies $A \xrightarrow[000]{010} A$.

Thus $S(16) \equiv S(26) \equiv A$

We deduce from this that

$$B \xrightarrow[0]{0} A \quad \text{and} \quad A \xrightarrow[0]{0} B.$$

This implies that $S(14) \equiv B$ from which we deduce that $B \xrightarrow[0]{1} B$.

At this point we have deduced the following state-table:

	0	1
A	B, 0	
B	A, 0	B, 0
C	, 1	A, 0
D	, 1	, 0

c) We see from S(15) that $B \xrightarrow[00100]{01011} A$. Then, since $S(27) \equiv B$, $S(32) \equiv A$.

Now $S(16) \equiv S(32) \equiv A$ imply that S(17) and S(33) are the same state. This state responds to 0 by 1, and to 10 by 01. It can only be D:

$$A \xrightarrow[0]{1} D.$$

d) We see that $S(11) \xrightarrow[0]{1} C$, for we know that $S(12) \equiv C$. But A, B and C under an input of 1 lead to D, B and A respectively. Hence, S(11) must be D:

$$D \xrightarrow[0]{1} C$$

e) By the same reasoning $S(19) \xrightarrow[0]{1} A$. Hence $S(19) \equiv C$. From this,

* In this appendix and in the following, the notation $S_i \xrightarrow{x} S_j$ means that an input x causes a transition from state S_i to state S_j with an output z.

the same reasoning again tells us that $S(18) \equiv D$. $S(17)$ being known to be D, we deduce that

$$D \xrightarrow[1]{0} D$$

f) We know that $S(5) \equiv S(33) \equiv D$. It follows that $S(6) \equiv S(34) \equiv C$, and that $S(7)$ and $S(35)$ are the same state. This state responds to 0 by 1 and to 10 by 01. It can only be D. Hence $C \xrightarrow[1]{0} D$.

Our knowledge of the state-table is now complete.

A P P E N D I X III

(related to IV. 2)

The experiment for the machine of fig. 13 is to be started in state A, and is the following:

	LS(A)														
		LS(B)		LS(D)				LS(C)					LS(D)		
X	0	0	1	0	0	1	0	1	0	1	0	1	1	0	
Z	1	0	1	1	0	0	0	0	0	0	0	0	1	1	
Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

	LS(E)														
				LS(D)				LS(D)						LS(B)	
X	0	1	1	0	0	1	1	0	0	1	1	0	1	0	1	0
Z	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
Time	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	

		LS(A)						LS(B)					LS(D)		
X	0	0	0	1	0	1	1	0	1	0	1	0	0	0	1	
Z	0	1	0	1	1	0	0	0	1	1	0	0	1	0	0	
Time	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	

We shall now explain how from this experiment we can recover the state table.

a) From the LS's we can recover the five states of the machine. We will name them by the name assigned to the corresponding state in the state table:

$S(1) \equiv S(32) \equiv A$

$S(2) \equiv S(28) \equiv S(38) \equiv B$

$S(13) \equiv C$

$S(4) \equiv S(14) \equiv S(18) \equiv S(22) \equiv S(43) \equiv D$

$S(27) \equiv E$

From this we know three transitions:

$$A \xrightarrow[1]{0} B;\quad C \xrightarrow[1]{1} D;\quad E \xrightarrow[0]{1} B.$$

b) We see from S(18) that $D \xrightarrow[1001]{0011} D$, hence $S(26) \equiv D$

So $D \xrightarrow[1]{0} E$. From this, $S(15) \equiv S(19) \equiv S(23) \equiv E$.

c) We see from S(2) that $B \xrightarrow[01]{01} D$, hence $S(30) = D$, which implies that $S(31) \equiv E$, from which we deduce that $E \xrightarrow[0]{0} A$.

From this, $S(16) \equiv S(20) \equiv S(24) \equiv A$

d) From S(1), $A \xrightarrow[101]{001} D$, hence $S(35) \equiv D$, so that $S(36) \equiv E$, which implies $S(37) \equiv B$. S(38) being B, we deduce: $B \xrightarrow[0]{1} B$.

e) $S(38) = B$ implies $S(40) \equiv D$, which implies $S(41) \equiv E$, which in turn implies that $S(42) \equiv B$. S(43) being D, we deduce: $B \xrightarrow[0]{0} D$.

f) From this, $S(3) \equiv D$. S(4) being D, this implies that $D \xrightarrow[1]{1} D$.

g) We see from the experiment that $S(12) \xrightarrow[0]{1} C$. We know now that B, C, D, and E under an input of 1 go to B, D, D, and B, respectively. Hence S(12) must be A. So $A \xrightarrow[0]{1} C$.

h) $S(11) \xrightarrow[0]{0} A$. Hence S(11) can only be either C or E. Now $S(10) \xrightarrow[0]{1} S(11)$. But there is no transition which leads to state E. Hence $S(11) \equiv C$. Consequently, S(12) being A, $C \xrightarrow[0]{0} A$.

This completes the reconstitution of the state-table.

REFERENCES

1. Moore, E.F., "Gedanken - Experiments on Sequential Machines", Automata Studies, Princeton University Press, Princeton, New Jersey, 1956.

2. Hennie, F.C., "Fault Detection Experiments for Sequential Circuits", Proc. 5th Annual Symposium on Switching Theory and Logical Design, Princeton University, November 1964.

3. Kohavi, Z., and Lavallee, P., "Design of Sequential Machines with Fault-Detection Capabilities", IEEE Transactions on Electronic Computers, August 1967.

4. Gill, A., Introduction to the Theory of Finite-State Machines, McGraw Hill Book Co., 1962.

5. Hennie, F.C., Finite-State Models for Logical Machines, John Wiley & Sons, 1968.