SU-SEL-72-034

Adaptive Design Methods for Checking Sequences

by

Raymond T. Boute

July 1972

Technical Report No. 30

This work was supported by the
National Science Foundation under
Grant GJ-27527

DIGITAL SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES

STANFORD UNIVERSITY · STANFORD, CALIFORNIA

SEL-72-034

ADAF'TIVE DESIGN METHODS FOR CHECKING SEQUENCES

by

Raymond T. Boute

July 1972

Technical Report No. 30

DIGITAL SYSTEMS LABORATORY

Department of Electrical Engineering Department of Computer Science

Stanford University

Stanford, California

Research reported in this paper was supported by the National Science Foundation under grant GJ 27527, and while Mr. Boute was partially supported by the Nationaal Fonds voor Wetenschappelijk Onderzoek of Belgium.

Adaptive design methods for checking sequences

by

Raymond T. Boute

Digital Systems Laboratory
Department of Electrical Engineering
Stanford University

ABSTRACT

The length of checking sequences for sequential machines can be considerably reduced if, instead of preset distinguishing sequences, one uses so-called "distinguishing sets" of sequences, which serve the same purpose but are generally shorter. The design of such a set turns out to be equivalent to the design of an adaptive distinguishing experiment,* though a checking sequence, using a distinguishing set, remains essentially preset. This property also explains the title.

All machines having preset distinguishing sequences also have distinguishing sets. In case no preset distinguishing sequences exist, most of the earlier methods call for the use of locating sequences, which result in long checking experiments. However, in many of these cases, a distinguishing set can be found, thus resulting in even more savings in length.

Finally, the characterizing sequences used in locating sequences can also be adaptively designed, and thus the basic idea presented below is advantageous even when no distinguishing sets exist.

* BY "experiment" we mean the application of sequence(s) to the machine while observing the output. In some instances, the words "experiment" and "sequence" can be used interchangeably.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

The concept of "experiments" on sequential machines [1] has led to methods for checking certain classes of machines against faults [2,4,5]. For strongly-connected (an essential requirement) and reduced (a simplifying condition) machines, checking experiments consist, in principle, of three parts:

(1) A synchronizing [2,3] sequence or, if none exists, a homing sequence followed by an appropriate sequence to bring the machine in a given initial state. This latter sequence depends on the state at the end of the homing sequence, and is thus always adaptive.

(2) Identification of the states by means of distinguishing or, if none exist, locating sequences. This is essentially based on the assumption that no fault can increase the number of states of the machine.*

(3) Checking the transitions out of each state, again including identification of the next states.

Most often (2) and (3) are not really separate parts, but are designed together: state identifications and transition checks are performed together in the checking sequence whenever doing so might shorten the final result.

Usually, the distinguishing or locating sequences used for state identification in a checking experiment are designed as preset [3]

* Otherwise the procedure is more complicated and requires much longer checking experiments.

experiments. Although a checking experiment is essentially preset (i.e. apart from the initialization, in case no synchronizing sequence exists), it is possible to replace the single distinguishing sequence, that is used for the identification of every state, by a well-chosen set of sequences, each of which is "adapted' to the state that is being identified. It turns out that the design of such a distinguishing set is equivalent to the design of an adaptive distinguishing experiment [3].

The sequences in a distinguishing set are nearly always shorter, and never longer, than the shortest preset distinguishing sequences. Since state identifications have to be done very frequently during a checking experiment, this results in considerable savings in the number of input symbols in the checking sequence.

Furthermore, there are many machines that have no preset distinguishing sequence but for which a distinguishing set can be found. For such machines, the use of a distinguishing set eliminates the need for (usually very long) locating sequences and thus results in even more important reductions in length. The possibility of constructing short checking experiments for machines with adaptive distinguishing sequences has been anticipated by I. and Z. Kohavi in [7], although they did not further explore the underlying principles and the practical design aspects.

In section II, we recall the basic ideas regarding preset distinguishing sequences for later comparison with the use of distinguishing sets. We then define the concept of "distinguishing sets" in a rigorous fashion and prove that they can be used instead of preset distinguishing sequences. A simple design procedure is presented and the similarities and differences with the design of adaptive distinguishing experiments are pointed out, as well as the advantages over distinguishing sequences. Finally, in section III, we show by means of examples how to implement these ideas in designing checking experiments. The use of a _transition check status table_ allows additional short cuts in an algorithmic fashion, while in the past short cuts were found in a rather "ad hoc" fashion. The use of a distinguishing set turns out to allow more "telescoping" than distinguishing sequences.

II. PRESET DISTINGUISHING SEQUENCES AND DISTINGUISHING SETS

In this section we explain the basic ideas leading to the replacement of preset distinguishing sequences by distinguishing sets. The reader will soon realize that the design of distinguishing sets is equivalent to the design of adaptive distinguishing experiments, which are discussed in detail by Hennie [3].

First we introduce some notation and basic definitions.

Notation

We denote a sequential machine M as follows [6]: $M=<I,O,Q,\delta,\lambda>$ where I, 0, Q are respectively the input, output and state sets, 6: $Q\times I\rightarrow Q$ the next-state function and λ: $Q\times I\rightarrow O$ (or $Q\rightarrow O$ for Moore machines) the output function. Further, $I^*= I^+\cup\{\Lambda\}$, where I^+ is the set of nonempty finite sequences of symbols from I and Λ is the empty sequence. Finally, we extend δ and λ in a natural way to sequences:

δ : $Q\times I^*\rightarrow Q$, where $\delta(q,\overline{x})$ is the final state of the machine, started in q and driven by input sequence $\overline{x}$.

$\overline{\lambda}$: $Q\times I^*\rightarrow O^*$, where $\overline{\lambda}(q,\overline{x})$ is the response of the machine to $\overline{x}$ when started in state q. For Moore machines: $\overline{\lambda}:Q\times I^*\rightarrow O^+$.

Convention: $\delta(q,\Lambda)=q$. Also $\overline{\lambda}(q,\Lambda)=\Lambda$ for Mealy and $\overline{\lambda}(q,\Lambda)=\lambda(q)$ for Moore machines.

A preset distinguishing sequence for a machine M is a sequence $\overline{x}\in I^*$ such that $\overline{\lambda}(q,\overline{x})=\overline{\lambda}(q',\overline{x})$ implies $q=q'$.

In other words, the machine responds differently to $\overline{x}$ for each initial state.

Only reduced machines -- but not all of them -- have distinguishing sequences. However, all reduced machines have characterizing sets [2,3].

A characterizing set for a machine M is a finite subset $\mathcal{C} \subseteq I^*$ such that $\bar{\lambda}(q,\bar{x})=\bar{\lambda}(q',\bar{x})$ for all $\bar{x} \in \mathcal{C}$ implies $q=q'$.

Usually, in case no distinguishing sequence exists, state identification is accomplished by locating sequences, as explained in [2]. A locating sequence (for a given state) is built from characterizing sequences and includes repetitions to ensure that the circuit is in the same state each time a new characterizing sequence is introduced for identifying that state. We will not discuss this subject in detail, since characterizing sets can be redefined (and used) in essentially the same way as distinguishing sets (to be defined later).

A. Preset Distinguishing Sequences for Checking Experiments

Definition: For every $\bar{x} \in I^*$, define a partition $\pi_{\bar{x}}$ on Q as follows: $q \equiv q'$ $(\pi_{\bar{x}})$ iff $\bar{\lambda}(q,\bar{x})=\bar{\lambda}(q',\bar{x})$.

From the preceding definitions we immediately deduce the following lemma:

Lemma: $\bar{x}$ is a (preset) distinguishing sequence iff $\pi_{\bar{x}} = 0$ (i.e. each block is a singleton).

This lemma leads directly to a design procedure [3] which we explain here for later comparison with the design of distinguishing sets.

Design of Preset Distinguishing Sequences

We construct a tree-like directed graph, starting with π_Λ and proceeding level by level. The vertices are partitions of the form $\pi_{\bar{x}}$. Each $\pi_{\bar{x}}$ gets either $|I|$ successors, namely the partitions $\pi_{\bar{x}i}$ as i ranges over I, or none at all: we do not introduce successors for $\pi_{\bar{x}}$ in case a partition equal to $\pi_{\bar{x}}$ has already been encountered before, during the construction of the tree. In this fashion, repetitions are avoided and the procedure terminates after a finite number of steps.

Distinguishing sequences -- if any exist -- are then represented by paths leading from π_Λ to some zero-partition,

For implementation it is easier to represent blocks of partitions $\pi_{\bar{x}}$ by their state transformations under $\bar{x}$, i.e. if $(q_1, q_2,...q_k\}$ is a block of $\pi_{\bar{x}}$, it will be represented by the block $\{\delta(q_1,\bar{x}),...,\delta(q_k,\bar{x})\}$ during the procedure. This representation is adequate for deciding whether or not we reached a partition $\pi_{\bar{x}} = 0$, provided no <u>merges</u> occur for the sequence $\bar{x}$. By a merge we mean that, for some 2 states ql $\neq q_2$, we have $\bar{\lambda}(q_1,\bar{x})=\bar{\lambda}(q_2,\bar{x})$ and $\delta(q_1,\bar{x})=\delta(q_2,\bar{x})$. But since such sequences can never be an initial part of a distinguishing sequence, the corresponding paths in the graph are terminated as soon as a merge is observed. This is very easy to do when using the representation just described, since then two different states, e.g. $\delta(q,\bar{x})$ and $\delta(q',\bar{x})$ are in the same block iff $\bar{\lambda}(q,\bar{x})=\bar{\lambda}(q',\bar{x})$. If some next input (after $\bar{x}$) leads to a merge, this is immediately detected, and for such an input no edge will leave that block.

The advantages of this representation become apparent in the following example.

<u>Example</u>

For machine M1, whose state-output table is given by Table l(a), the design graph for preset distinguishing sequences is shown in Fig. 1. The tabular equivalent, which is more practical for computer implementation is given in Table l(b). The three shortest distinguishing sequences are: 100, 101, 110.

Blocks containing only 1 state are not represented. In case all blocks are singletons (i.e. $\pi_{\bar{x}} = 0$ and $\bar{x}$ is a distinguishing sequence) an asterisk is written.

TABLE1

(a) Machine M_1

state q	input x 0	1
A	B/0	D/1
B	C/1	D/0
C	B/0	A/1
D	A/0	B/0

$\delta(q,x)/\lambda(q,x)$

(b) Tabular Equivalent of Fig. 1

input sequences	blocks to be split	input 0	1
Λ	ABCD	---	AD,BD
1	AD,BD	AB	BD
10	AB	*	*
11	BD	*	BD

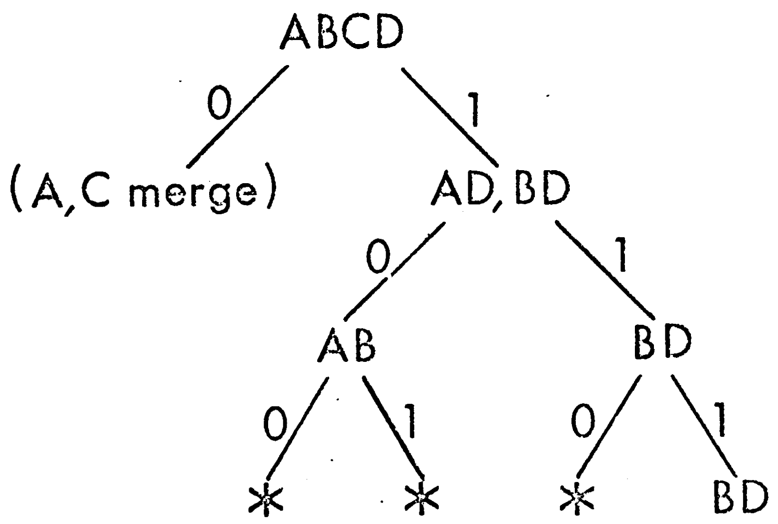


Figure 1. Design of preset distinguishing sequences for M_1.

Application to Checking Sequences

Since the design of checking sequences is discussed by Hennie [2] we will emphasize here only the role of distinguishing sequences. This discussion is necessary in order to justify later on their replacement by distinguishing sets.

We assume here that the good machine has a distinguishing sequence. Let N be the number of states. We also assume that no faulty machine can have more than N states. (property 1)

A checking sequence is always designed in such a way that, after the initialization part, the machine under test would be in a predetermined* state q_0, in case it were the good machine. (property 2)

The checking part itself is constructed in such a way that it would take the good machine, started in q_0, at least once through each of its N states, and identify each of these states by means of the same distinguishing sequence $\bar{x}$ (yielding N different responses, by definition).

This allows to make the following conclusions for the machine under test, depending on its response to the checking sequence:

(1) if the response is incorrect, the machine must be faulty, because of property 2.

(2) if the response is correct, we have obtained N different responses to the same sequence. Together with property 1, this implies that there are exactly N (nonequivalent) states. This further implies that, in case the same response to the

* In this way, only one checking experiment has to be designed. Otherwise, one would need several ones depending on the outcome of the initialization.

distinguishing sequence $\bar{x}$ is obtained at different points of the experiment, the circuit under test must have been in the same state each time.

The last statement in (2) forms the basis for the verification of state transitions, since we can now identify each state unambiguously by means of the distinguishing sequence $\bar{x}$.

B. Distinguishing Sets

Our main purpose is here to replace the distinguishing sequence used to identify the states by a set of sequences that are individually designed for each state to be identified. The following scheme is not the most general solution possible, but seems to be the easiest to design and implement.

The initialization of the checking experiment is done in exactly the same fashion as explained before. Thus point (1) above is still valid. If we also want to obtain conclusions similar to (2), based solely* on observing the appearance of each sequence (possibly different for each state) from the distinguishing set $\mathfrak{D}$, together with a correct response, we can proceed as follows.

Definition: A distinguishing set $\mathfrak{D}$ for a machine M with state set Q is a set of input sequences $\bar{x}_i$, one for each $q_i \in Q$, such that for every pair of (different) states $q_i, q_j \in Q$ we can write:

$$\bar{x}_i = \bar{u}_{ij}\,\bar{y}_{ij}$$
$$\bar{x}_j = \bar{u}_{ij}\,\bar{z}_{ij}$$

* This is one of the restrictions that make this scheme not the most general one possible.

where the sequences $\bar{u}_{ij}$, $\bar{y}_{ij}$, $\bar{z}_{ij} \in I^*$ are such that $\bar{\lambda}(q_i,\bar{u}_{ij}) \neq \bar{\lambda}(q_j,\bar{u}_{ij})$.

Remark: The double subscripts ij in the above definition emphasize the fact that the decomposision of $\bar{x}_i$ and $\bar{x}_j$ into $\bar{u}$'s, $\bar{y}$'s, etc. may depend on both q_i and q_j. It is also easy to see that, for Mealy machines, the definition implies that the first input symbol be the same for all $\bar{x}_i \in \mathcal{D}$. This observation is the first step toward the design procedure described below.

Example

For the machine of Table 1, a distinguishing set is given in Table 2(a). Further, in Table 2(b) we show the decomposition into $\bar{u},\bar{y},\bar{z}$'s.

TABLE 2

(a) A Distinguishing Set for Ml and corresponding responses:

state q	input $\bar{x}_q \in \mathcal{D}$	response $\bar{\lambda}(q,\bar{x}_q)$	final state $\delta(q,\bar{x}_q)$
A	11	10	B
B	10	00	A
C	11	11	D
D	10	01	C

(b) Decompositions for the sequences of the distinguishing set in (a):

q_i \ q_j	A	B	C	D
A	---	1,1,0	11, Λ, Λ	1,1,0
B	1,0,1	---	1,0,1	10, Λ, Λ
C	11, Λ, Λ	1,1,0	---	1,1,0
D	1,0,1	10, Λ, Λ	1,0,1	---

$\bar{u}_{ij}$, $\bar{y}_{ij}$, $\bar{z}_{ij}$

Theorem. Let $\mathcal{D}$ be a distinguishing set.

If the machine under test responds to a given application of $\bar{x}_i \in \mathcal{D}$ (at the input) by $\bar{\lambda}(q_i,\bar{x}_i)$ and to $\bar{x}_j \in \mathcal{D}$ by $\bar{\lambda}(q_j,\bar{x}_j)$, then the states before these applications must be different (assuming $q_i \neq q_j$).

Proof. We denote by $\bar{\lambda}_T$ the %-function for the machine under test. Let the states of the machine under test be q resp. q' before application of $\bar{x}_i$ resp. X_j. Then $\bar{\lambda}_T(q,\bar{x}_i)=\bar{\lambda}(q_i,\bar{x}_i)$ and $\bar{\lambda}_T(q',\bar{x}_j)=\bar{\lambda}(q_j,\bar{x}_j)$. By definition, there exists a common initial part $\bar{u}_{ij}$ of $\bar{x}_i$ and $\bar{x}_j$ such that $\bar{\lambda}(q_i,\bar{u}_{ij}) \neq \bar{\lambda}(q_j,\bar{u}_{ij})$. Therefore $\bar{\lambda}_T(q,\bar{u}_{ij}) \neq \bar{\lambda}_T(q',\bar{u}_{ij})$ implying $q \neq q'$.

Q.E.D.

Corollary

If an experiment is constructed using the same algorithm as for designing checking experiments, except for the fact that each state q_i is identified by the corresponding $\bar{x}_i \in \mathcal{D}$ rather than by a fixed distinguishing sequence, then the result is a checking experiment.

Proof. In case the machine under test responds incorrectly to the experiment, it is known to be faulty. In case it responds correctly, each $\bar{x}_i \in \mathfrak{D}$ has appeared at least once (with its correct response). Therefore we know by the preceding theorem that there must be at least N different states. Since, by assumption, we have at most N states, the arguments given for the checking experiments using a distinguishing sequence carry through.

Q.E.D.

Remark

A distinguishing set should not be confused with a compound distinguishing sequence [5] which is defined as a set of input sequences $\mathcal{C} = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_r\}$ having a common prefix $\bar{y}$ but otherwise unrestricted, with the following properties:

(1) For each block B of $\pi_{\bar{Y}}$ there is an $\bar{x}_i \in \mathcal{C}$ that distinguishes the states in B (i.e. given $q, q' \in B$ then $\bar{\lambda}(q, \bar{x}_i) = \bar{\lambda}(q', \bar{x}_i)$ implies $q = q'$).

(2) The deletion of any sequence from $\mathcal{C}$ will leave some states in some block B in $\pi_{\bar{Y}}$ not distinguished by any $\bar{x}_i \in \mathcal{C}$.

Although the definitions are completely different, we have observed that for machines with only a few (up to 4) states a distinguishing set $\mathfrak{D}$ satisfies at least property (1) in the above definition. This phenomenon is purely coincidental, due to the small size of the partition blocks (this can be verified by trying a couple of examples). However, consider now a machine with more states, such as in Table 3(a). A distinguishing set is shown in Table 3(b).

The longest[*] common prefix is 1 and $\pi_1 = \overline{BF}, \overline{ACDE}$ and the shortest sequence to distinguish all states of $\overline{ACDE}$ is 1111 (or 1100). Thus no sequence $\bar{x}_i \in \mathfrak{D}$ can split up this block and $\mathfrak{D}$ is therefore not a compound distinguishing sequence.

TABLE 3

To illustrate the difference between a distinguishing set and a compound distinguishing sequence

(a) State-output table

state q	input 0	input 1
A	D/0	B/1
B	B/1	A/0
C	B/1	D/1
D	C/0	F/1
E	B/1	C/1
F	A/0	E/0

(b) Distinguishing set $\mathfrak{D}$

state q	$\bar{x}_q$	response
A	110	100
B	10	00
C	111	110
D	110	101
E	111	111
F	10	01

Construction of a distinguishing set

Now we show that the construction of a distinguishing set is similar to the design of an adaptive distinguishing experiment [3].

[*] It is easily seen that, the longer $\bar{y}$, the smaller the blocks of $\pi_{\bar{y}}$ and the more possibility to satisfy property (1) of a C-set.

(a) <u>For Mealy Machines</u>

As explained before, the first input x is the same for every sequence in $\mathfrak{D}$, thus independent of the state to be identified. In choosing this input, the only condition, apart from optimization criteria explained later on, is that there be no merge under x, i.e. there should be no 2 states $q_1 \neq q_2$ such that $\lambda(q_1,x)=\lambda(q_2,x)$ and $\delta(q_1,x)=\delta(q_2,x)$.

This first input splits Q in equivalence classes under the relation: $q \equiv q'$ iff $\lambda(q,x)=\lambda(q',x)$. Since states in different classes respond <u>differently</u> to the <u>same</u> first input, they are distinguished from each other, and the second symbol of the corresponding sequences may be different (in fact, for singleton classes no second symbol is needed). Thus we can now treat each class separately, in exactly the same fashion as we originally started out with Q.

Distinguishing sets can thus be constructed in the same way, using a tree-like graph, as adaptive distinguishing experiments (see Hennie [3] or the example below). However, there are two differences:

(1) In an A.D.E., the tree is used during the experiment itself in order to decide on the next input at each moment. In our case, however, the checking experiment is essentially preset, and the tree is used only during the design.

(2) During an A.D.E. we may choose arbitrarily the next input among equally acceptable (qua optimality) candidates. While designing a distinguishing set, we have to choose the same next-input for each state of a given block (at that stage). It is indeed not allowed to apply different inputs for different states before a different output has been observed. This follows directly from the definition. This procedure allows us to decide, by looking <u>only</u> at the corresponding

sequences from $\mathfrak{D}$ and their responses, whether the states are, in fact, different.

(b) For Moore Machines

The procedure is completely analogous. The only difference is that before the application of the first input (i.e. we apply A) we can already split Q into equivalence classes, since λ is a map $Q \to O$.

Example

Fig. 2 represents the design graph for machine Ml (see Table 1), and Table 4 its tabular equivalent.

As before, we represent the blocks by the successors of the initial states as we trace our path along the graph. Singletons are not represented. The symbols between parentheses represent the output corresponding to each block, except for single blocks. Table 2(a) represents the distinguishing set having the shortest sequences.

Optimal Choice of a Distinguishing Set

This can be done in exactly the same way as choosing the optimal next-input during an adaptive distinguishing experiment [3]. To recall briefly: assign a number to each block in the graph, starting with zero's for singletons. Assign the predecessor numbers according to the following formulae:

$$n = \min(m_1, m_2, \ldots m_k)$$

$$m_j = \max(n_{j1}, \ldots, n_{ji_j}) + 1$$

The symbols are explained in Fig. 3. Clearly those numbers give, for each block, the length of the shortest sequence for distinguishing a given state in that block, assuming the worst choise regarding that state. For more details we refer to Hennie [3].

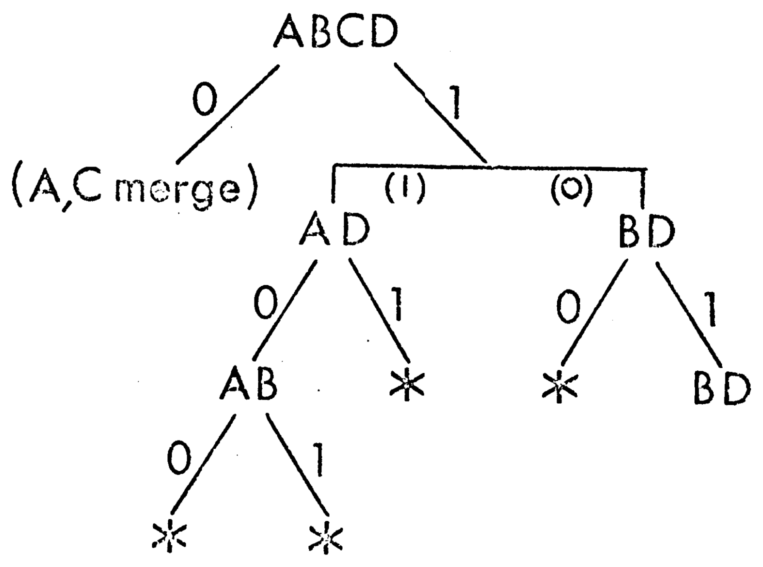
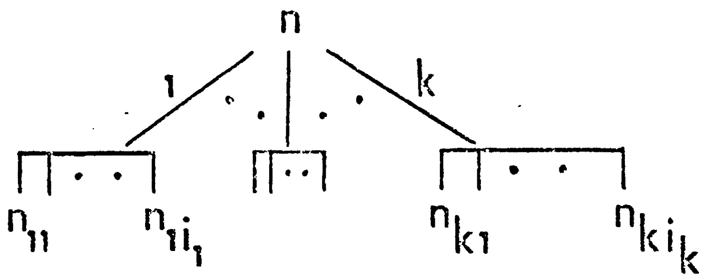


Figure 2 Design of a distinguishing set for M_1

TABLE 4

Tabular Equivalent of Fig. 2

input sequences	blocks to be split	input 0	1
Λ	ABCD	---	AD,BD
1	AD	AB	*
1	BD	*	BD
10	AB	*	*



$$k \leqslant |I|$$

$$i_j \leqslant |O| \;, \; j = 1, 2, \ldots, k$$

Figure 3 Optimal choice for a distinguishing set.

Comparison with (Preset) Distinguishing Sequences

In designing distinguishing sequences, we have to refine partitions, while the design of distinguishing sets consists of splitting up the blocks of these partitions *separately*. The significance of this observation can be appreciated by comparing the construction of the corresponding graphs. The main consequences are:

-- the longest sequence in an optimal distinguishing set is at most as long as the shortest preset distinguishing sequence.

-- since it may be the case that certain blocks cannot be split up simultaneously by any single sequence while allowing refinement by separate sequences, distinguishing sets may exist where no distinguishing sequences can be found.

III. EXAMPLES

(a) Checking Experiment for M_1

Initialization: the shortest synchronizing sequence for M_1 is 0101 and brings the machine in state D.

Checking part. We first illustrate the design of the checking part with separate state identification and transition checking. State identification is done as follows (use Table 2(a)).

- Identify starting state D by $\bar{x}_D = 10$

  This brings us in state C.
- Identify C by 11. New state: D.
- Go from D to A by input 0.
- Identify A by 11. New state: B.
- Identify B by 10. New state: A.

The resulting sequence, together with its response, is given in Table 5(a). If the machine under test responds correctly, we know the states q_1, q_2, q_3, q_4 are different (see central theorem, section II).

TABLE 5(a): State identification for M_1.

	1		0		1		1		0		1		1		1		0		input
q_1				q_2						q_3				q_4					state
	0		1		1		1		0		1		0		0		0		output

We denote these states by the corresponding names borrowed from the good machine (resp. D, C, A, B). Since we assume that the machine under test has at most 4 states, it must now be in one of the states already encountered. If it were the good machine it would be state A. By means of

$x_A = 11$ we verify for the machine under test that it is indeed in the state we decided to call A. The new situation is shown in Table 5(b).

TABLE 5(b): Situation before transition verification.

```
 1 0 1 1 0 1 1 1 0 1 1      (input)
D   C     A   B   A         (state)
 0 1 1 1 0 1 0 0 0 1 0      (output)
                (2)
```

Now we are ready for the transition verification. From Table 5(b) it is clear that the machine under test is in state B since $\delta(A,11) = B$ has been verified.

We check now the transition out of this state under input 1 (on the left side of Table 5(c)) by means of $\bar{x}_D = 10$ (we know we should use $\bar{x}_D$ by referring to the good machine).

TABLE 5(c): Transition verifications.

```
 1 1 0 1 1 1 0 1 1 0 1 1 0 0 1 0 0 1 0 1 1 1 0
B D   C A   B C     A D   C B   A B   A D B
 0 0 1 1 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0
(1)   (3)   (4)     (5)   (6)   (7)     (8)
```

Note here that $\delta(B, 10) = A$ and $\delta(B,1) = D$ (just verified) now automatically verifies $\delta(D,0) = A$ (see (2), Table 5(b)).

We proceed in this manner until all transitions have been verified. Table 5(c) completes the sequence. The numbers correspond to the order in which the transitions are verified, while Table 6 allows us to keep track of these verifications (and see which check to do next). In practice one does not put numbers in such a "transition check status table" but only check marks, as the design goes along.

TABLE 6

Transition check status table corresponding to Tables 5(b) and (c).

	0	1
A	7	5
B	4	1
C	6	3
D	2	8

More efficient design of the checking part

By now the reader will have realized that separate state identification and transition checking is not the most efficient way for constructing a checking experiment. Indeed, in our general treatment of checking experiments, be it with distinguishing sequences or g-sets, we pointed out that all that is needed for state identification is the appearance of an identification sequence for each state at least once _somewhere_ in the experiment. Because of the strongly-connectedness, every state appears at least once in the next-state columns of the state table, and thus all state identifications will automatically take place during the transition checks. Of course, in the beginning _some_ state identification might be needed in order to know in what state we are just before the first transition check. Also, if we use distinguishing sets rather than distinguishing sequences, it will very often be possible to <u>telescope</u> sequences, for example: state C is identified by $\bar{x}_C = 11$, but since $\delta(C,1) = A$, the second "1" of $\bar{x}_C$ can be used as the first "1" for $\bar{x}_A = 10$. Since we always know the state of the good machine during the design of the checking part, it is possible to check the possibility for telescoping sequences from the distinguishing set each time a new input symbol is to be added. Also, the information gained from past transitions (stored in the transition check status table) is always at hand to get the machine faster into a known state.

We illustrate these observations in Table 7(a). The transition check table (Table 7(b)) and the numbers between parentheses in Table 7(a) show how, every time a new transition is checked, we go back and see whether new known states can be filled in for the machine under test, and maybe some other transitions are checked in this fashion.

TABLE 7(a): More efficient checking experiment for M_1.

```
  1   0   1   1   1   0   0   1   0   1   1   1   0   0   1   0   0   1   1   0
D       C → A → D       q→B       A   D→B         A→B         A   B→D
  0   1   1   1   0   1   0   0   0   1   0   0   0   0   0   0   0   0   0   1
       (1)(2) (5)   (3)   (8)   (4)         (6)             (7)
```

TABLE 7(b)

Transition check status table corresponding to Table 7(a).

	0	1
A	6	2
B	5	7
C	3	1
D	8	4

For what concerns the use of past information, we are not restricted to single-input transition check status tables, but past transitions caused by sequences can also be incorporated: for example, the state marked q in Table 7(a) is known to be C because we verified $\delta(D,10) = C$ at the very start.

The arrows in Table 7(a) indicate transitions that are checked for the first time.

Conclusion. Using the distinguishing set of Table 2(a), the length of the checking sequence for Ml is 24. If we use the shortest distinguishing sequence, we obtain Table 7(c), a checking sequence of length 50 (including initialization). Thus we obtain about 50% saving in length with the use of distinguishing sets (for machine M_1).

TABLE 7(c)

Checking experiment for Ml using a distinguishing sequence.

$$1\ (100)^2\ \underset{\substack{\rightarrow \\ B\ C}}{0}\ (100)^2\ \underset{\substack{\rightarrow \\ C\ A}}{1}\ (100)^2\ \underset{\substack{\rightarrow \\ B\ D}}{1}\ 100\ 0\ \underset{\substack{\rightarrow \\ C\ B}}{0}\ 100\ 1\ \underset{\substack{\rightarrow \\ D\ A}}{0}\ 100\ 1\ \underset{\substack{\rightarrow \\ D\ B}}{1}\ 100\ 1\ 0\ \underset{\substack{\rightarrow \\ A\ D}}{1}\ 100$$

(b) Example of a Machine with a Distinguishing Set but no Distinguishing Sequence

Table 8(a) gives the state and output functions for M_2, a maching having exactly this property, while Table 8(b) represents a distinguishing set (optimal) for M_2. The reader can verify that no preset distinguishing sequence exists.

The graph for designing distinguishing sets is given in Fig. 4.

TABLE 8

(a) State-output table for machine M_2.

state q	input x 0	1
A	D/1	B/0
B	A/0	C/0
C	B/1	B/0
D	C/0	B/1

Next state/Output

(b) Distinguishing set for machine M_2.

state q	input $\bar{x}_q$	response $\bar{\lambda}(q,\bar{x}_q)$	final state $\delta(q,\bar{x}_q)$
A	01	11	B
B	001	011	B
C	01	10	C
D	001	010	C

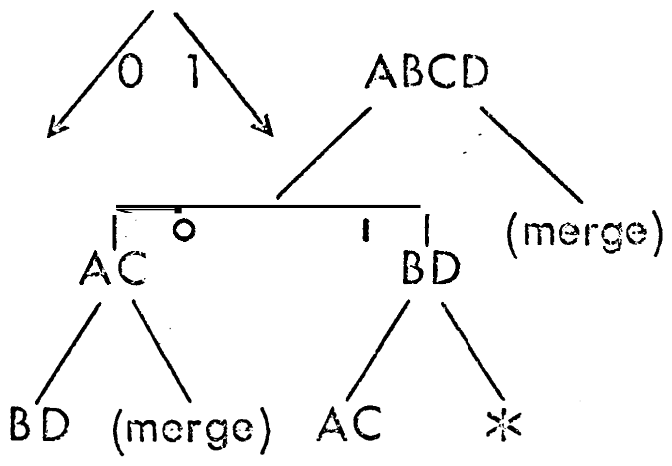


Figure 4. Design of a distinguishing set for M_2

Design of a Checking Sequence Using $\mathfrak{D}$

Initialization: synchronizing sequence 1001, leads to state B.

Checking part: the checking sequence is given in Table 9.

Again we use arrows to indicate where transitions are checked for the first time. Other transitions are implicitly checked (e.g. $\delta(A,01) = B$ and $\delta(A,0) = D$ checks $\delta(D,1)$ etc.). The length of this checking sequence is 31.

TABLE 9

Checking experiment for M_2 using a distinguishing set.

```
  0  0  1  0  0  1  0  0  0  0  1  0  1  0  0  0  1         (input)
B→A       B  A     B A→D→C        C    C→B          B        (state)
  0  1  1  0  1  1  0  1  0  1  0  1  0  1  0  1  1         (output)
(1)    (3)             (2) 4)   (6)     (5)

             ..  0  1  0  0  1  1  1  0  0  1      (input)
             . . B A→B          B C→B              (state)
             ..  0  1  0  1  1  0  0  0  1  1      (output)
                   (7)            (8)
```

With locating sequences, the checking experiment becomes much longer. Table 10 gives the sequences needed for the design. The synchronizing sequence and the appearance of each locating sequence already require 35 symbols. All transition verifications still have to be performed.

TABLE 10

Construction of locating sequences for M_2

state	characterizing set				locating sequence
	input 1		input 01		
	response	final state	response	final state	
A	0	B	11	B	$(10)^4 01$
B	0	C	00	B	$(10)^4 01$
C	0	B	10	C	$(11)^4 01$
D	1	B	00	B	1

CONCLUSION

We have shown that checking experiments can be made considerably shorter by using distinguishing sets. This is a consequence not only of the shorter length of the sequences used for state identification but also of the increased possibility for "telescoping" sequences and implicit transition verification, as shown in the examples.

Further, in case no preset distinguishing sequence exists, a distinguishing set, if one exists, can be used to replace the rather cumbersome locating sequences.

As pointed out during the discussion of distinguishing sets, some even more general approach might be possible. Indeed, if better use could be made of all information given by a checking experiment, we might be able to waive the restriction that the next input be the same for all elements of a block (in the design graph). However, this would require a different design philosophy. It is hoped that other simple methods will be found to further reduce the length of checking experiments.

REFERENCES

[1] E. F. Moore, "Gedanken Experiments on Sequential Machines", _Automata Studies_ No. 34, pp. 129-153, Princeton University Press, Princeton, N.J. (1956).

[2] F. C. Hennie, "Fault Detecting Experiments for Sequential Circuits", _Proc. 5th Ann. Symposium on Switching Theory and Logic Design_, pp. 95-110, Princeton, N.J. (Nov. 1964).

[3] F. C. Hennie, _Finite State Models for Logical Machines_, Wiley, New York (1968).

[4] E. P. Hsieh, _Optimal Checking Experiments for Sequential Machines_, Ph.D. dissertation, Columbia University, N.Y. (September 1969).

[5] E. P. Hsieh, "Checking experiments for Sequential Machines," _IEEE Transactions on Computers_, Vol. C-20, No. 10, pp. 1153-1166 (October 1971).

[6] J. Hartmanis, R. E. Stearns, _Algebraic Structure Theory of Sequential Machines_, Prentice-Hall (1966).

[7] I. Kohavi, Z. Kohavi, "Variable-Length Distinguishing Sequences and their application to Fault-Detection Experiments," _IEEE Transactions on Computers_, vol. C-17, No. 8, pp. 792-795 (August 1968).