Property 8:

$$\lim_{T \to L_I} w(T) = N_r$$

where N_r is the number of distinct recurrent pages in W_I is as yet not verified by our arguments based only on the automata model B.

DISCUSSION

This note has introduced two finite automata that model two important classes of demand paging algorithms with constant or variable amount of allocated space. The adopted replacement policy and the state-string updating procedure are imbedded in the recursive definition for states. The LRU stack updating can be accomplished without relying on stack distances. The inclusion property of a stack algorithm [6] and the anomalous behavior of the algorithm with the FIFO replacement policy can be treated more simply as shown in [11]. Previous results [12] concerning the properties of the working set model applicable for infinite reference strings are updated to fit the finiteness assumption. As shown in [11] the page-turning rate algorithm also has the capability of revealing memory demands.

Since automata theory is very well established, we might be able to borrow more concepts such as the regular set [9] and the regular expression [9] from this area for the study of program behaviors and for stimulating new ideas and techniques to solve problems related to resource management under various environments.

ACKNOWLEDGMENT

The author wishes to thank B. Brawn (now B. Mattson) for her encouragement throughout the author's year at the IBM San Jose Research Laboratory.

REFERENCES

[1] L. A. Belady, "A study of replacement algorithms for a virtual-storage computer," *IBM Syst. J.*, vol. 5, pp. 78–101, 1966.
[2] B. Randel and C. J. Kuehner, "Dynamic storage allocation system," *Commun. Ass. Comput. Mach.*, vol. 11, pp. 297–305, May 1968.
[3] P. J. Denning, "Virtual memory," *Comput. Surveys*, vol. 2, pp. 153–189, Sept. 1970.
[4] A. V. Aho, P. J. Denning, and J. D. Ullman, "Principles of optimal page replacement," *J. Ass. Comput. Mach.*, vol. 18, pp. 80–93, Jan. 1971.
[5] P. J. Denning, "The working set model for program behavior," *Commun. Ass. Comput. Mach.*, vol. 11, pp. 323–333, May 1968.
[6] R. L. Mattson, J. Gecsei, D. R. Slutz, and I. L. Traiger, "Evaluation techniques for storage hierarchies," *IBM Syst. J.*, vol. 9, pp. 78–117, 1970.
[7] G. S. Shedler and C. Tung, "Locality in page reference strings," IBM Res. Rep. RJ932, pp. 1–48, Oct. 29, 1971.
[8] L. A. Belady, R. A. Nelson, and G. S. Shedler, "An anomaly in space-time characteristics of certain programs running in a paging machine," *Commun. Ass. Comput. Mach.*, vol. 12, pp. 349–353, June 1969.
[9] T. L. Booth, *Sequential Machines and Automata Theory*. New York: Wiley, 1967.
[10] E. Gelenbe, "A unified approach to the evaluation of a class of replacement algorithms," *IEEE Trans. Comput.*, vol. C-22, June 1973.
[11] C. C. Yang, "On the modeling of paging algorithms by finite automata," Dep. Inform. Sci., Univ. of Alabama in Birmingham, Tech. Rep. 73-2, Dec. 1973.
[12] P. Denning and S. C. Schwartz, "Properties of the working-set model," *Commun. Ass. Comput. Mach.*, vol. 15, pp. 191–198, Mar. 1972.

Distinguishing Sets for Optimal State Identification in Checking Experiments

RAYMOND T. BOUTE

***Abstract*—A new concept, called *distinguishing set* or *D-set* is presented. Its use yields a considerable reduction in the length of checking sequences. Arbitrarily chosen examples have indicated a reduction of 30–50 percent. It is shown that the sequences of a distinguishing set actually constitute the optimum, i.e., minimum length, for state identification through input–output observations only.**

Other features, such as *telescoping*[1] and the use of a *transition check status table*, account for further reduction in length. Also, since a distinguishing set may exist in cases where a distinguishing sequence does not, it can often replace the (usually long) locating sequences.

Finally, the principle can be extended to the design of characterizing sequences, and thus yield shorter locating sequences.

***Index Terms*—Checking experiments, distinguishing sequences, distinguishing sets, fault detection, machine identification, sequential machines, state identification, transition check status table, transition verification.**

I. INTRODUCTION

The concept of *experiments* on sequential machines [1] has led to methods for checking classes of machines against faults [2], [4], [5]. For strongly connected and reduced machines, checking experiments consist, in principle, of three parts: initialization [1]–[3], state identification [2], and transition verification [2]. Since the third part involves a large number of state identifications also, these last two parts do not have to be realized separately. Combining them will usually shorten the resulting checking sequence.

Over the years, many methods have been developed to improve the efficiency of the state identification procedure; to name only a few, resolving sequences, I/O sequences [4], [5], and variable-length distinguishing sequences [6].

We introduce the concept of *distinguishing set* (D-set) with a similar purpose (Section II). In fact, we will show how this concept arises in a natural fashion when establishing the minimal requirements that are imposed on the state identification sequences of any checking experiment by the condition that the states must be distinguished by knowing *only* the responses to those sequences. For machines that have a D-set, this approach presents a solution to the shortest checking sequence problem in the sense described by Moore [1] and Hennie [2], that is, based on the state identification and transition verification principle. Since the design of D-sets turns out to be closely related to the design of an adaptive distinguishing experiment, the sequences can be optimized in a similar fashion [3].

The possibility of constructing shorter checking experiments in case there exists an adaptive distinguishing experiment for the machine has been anticipated by an example in [6], but no design procedure or generalization was given. Further, it was not realized that an optimal set of state identifying sequences could be constructed for those machines.

In Section III we illustrate the practical implementation of these ideas in the design of checking experiments. The *telescoping* feature, which is illustrated below, and the use of a *transition check status table* provide a method for making shortcuts in a systematic, rather than the usual ad hoc fashion.

II. DISTINGUISHING SETS

A. Notation

Let M be a sequential machine [7]. We denote by I, O, Q the input, output, and state sets, respectively. Further, $\delta(q,i)$ denotes the next state corresponding to present state q and input i, and $\bar{\lambda}(q,\bar{x})$ the sequence of output symbols when the machine is started in state q and the input sequence $\bar{x}$ is applied. The bar ($^-$) will be used whenever dealing with sequences of symbols. By I^* we denote the set of finite input sequences together with the empty sequence Λ. These symbols, without subscripts, refer to the good machine.

Manuscript received October 16, 1972; revised June 1, 1973. This work was supported by the National Science Foundation under Grant GJ-27527 and by the Nationaal Fonds voor Wetenschappelijk Onderzoek (NFWO), Belgium.

The author was with the Digital Systems Laboratory, Stanford University, Stanford, Calif. 94305. He is now with the Technical Staff, Bell Telephone Manufacturing Company, Antwerp, Belgium.

[1] By *telescoping* we mean the following. If two sequences have to be applied during an experiment, and the final part (suffix) of one sequence is the same as the initial part (prefix) of the other sequence, then we can arrange the experiment in such a way that this common part has to be applied only once. This can be done, provided that such an arrangement is compatible with the conditions (if any) on the state of the machine before the application of the second sequence.

With the subscript T, they refer to the machine under test, which may or may not be faulty.

B. State Identification

In what follows, we implicitly assume that the machine is reduced and strongly connected. The strongly connectedness is an essential condition for a *complete* machine verification. The requirement that the machine be reduced is less stringent, since one is only interested in verifying the reduced behavior. However, for the design of checking experiments, based on the principles discussed by Moore [1] and Hennie [2], equivalent states are not allowed, since otherwise the states cannot be completely identified.

We will deal essentially with the state identification problem. The reader is assumed to be familiar with the basic assumptions and design methods used by Hennie [2].

Let us only mention that, in Hennie's approach, all states are identified either by a single[2] sequence (a *distinguishing sequence*) or by a single[2] set of sequences (*characterizing sequences*). In the latter case, these sequences are combined to form *locating sequences*. This is necessary in order to insure that the machine under test is in the same state whenever a new characterizing sequence is used to identify that state. It should be noted that the use of locating sequences for distinguishing between the states relies on some additional knowledge about the machine under test, namely the assumption that no fault increases the number of states. More precisely, let $\bar{x}_q$ denote[3] the sequence used to identify or locate state q. Then the following statement,

for all q,q',q_T,q_T' that satisfy

$$\bar{\lambda}_T(q_T,\bar{x}_q) = \bar{\lambda}(q,\bar{x}_q),$$

$$\bar{\lambda}_T(q_T',\bar{x}_{q'}) = \bar{\lambda}(q',\bar{x}_{q'}) \text{ and } q \neq q'$$

we have that $q_T \neq q_T'$ (statement A)

is always true for distinguishing sequences, while for locating sequences, it is based on the assumption, stated earlier, about the number of states.

C. Distinguishing Sets

Let us now look for the optimal method for distinguishing between the states, only by observing their responses to sequences. Thus no other knowledge is assumed. Henceforth, the only restriction will be that the same state will always be identified in the same fashion. However, in contrast with the preceding discussion, this may be done differently for different states.

In the most general case, the states can be distinguished from each other either by using a single sequence for each state or, if such is not possible, by using a set of sequences for each state. These are generalizations of distinguishing and characterizing sequences, respectively. For the same reasons as before, the implementation of the second case requires that the sequences, associated with a given state, be combined into a single sequence.[4] Since this combination requires some form of additional knowledge or assumptions about the machine under test, this second case will not be discussed. However, it will soon be realized that the concept of (sets of) characterizing sequences can be generalized, following the same basic reasoning as will be used next to introduce the concept of distinguishing sets.

Assume that we want to design a set of sequences (one for each state) such that any two different states can be distinguished by knowing *only* their responses to the two corresponding sequences of the set. Then a different response must occur for some common initial part of these two input sequences because different responses to different sequences cannot imply by themselves that the initial states were different. This condition, subsequently referred to as condition A, will be formalized by the following definition.

Definition: A *distinguishing set* (D-set) $\mathfrak{D}$ for a machine M is a set of input sequences $\bar{x}_i$, one for each state $q_i \in Q$, such that for every pair q_i,q_j of different states we can write $\bar{x}_i = \bar{u}_{ij}\bar{y}_{ij}$ and $\bar{x}_j = \bar{u}_{ij}\bar{z}_{ij}$ where the sequences $\bar{u}_{ij},\bar{y}_{ij},\bar{z}_{ij} \in I^*$ are such that $\bar{\lambda}(q_i,\bar{u}_{ij}) \neq \bar{\lambda}(q_j,\bar{u}_{ij})$.

Example: For the machine of Table I(a), a D-set is given in Table I(b). It is easy to verify that these sequences can be decomposed in the fashion implied by the definition, e.g., if $q_i = A$ and $q_j = B$, then $\bar{u}_{ij} = 1$, $\bar{y}_{ij} = 1$, and $\bar{z}_{ij} = 0$, while for $q_i = C$ and $q_j = A$, we have $\bar{u}_{ij} = 11$, $\bar{y}_{ij} = \Lambda$, $\bar{z}_{ij} = \Lambda$. This is an *a posteriori* decomposition, the proper method for designing $\mathfrak{D}$ being described later on.

TABLE I
MACHINE M_1 AND A DISTINGUISHING SET FOR M_1

(a) State - Output Table of M_1.

present state	next state / output (input 0)	(input 1)
A	B / 0	D / 1
B	C / 1	D / 0
C	B / 0	A / 1
D	A / 0	B / 0

(b) A Distinguishing Set for M_1.

state q	input $\bar{x}_q \in \mathfrak{D}$	response to $\bar{x}_q$	final state
A	11	10	B
B	10	00	A
C	11	11	D
D	10	01	C

Since the definition of D-sets refers to the faultless machine, we need the following theorem to establish the implications for the machine under test. The proof follows directly from the definition.

Theorem: Let $\mathfrak{D}$ be a D-set. If the machine under test responds to $\bar{x}_i \in \mathfrak{D}$ by $\bar{\lambda}(q_i,\bar{x}_i)$ and to $\bar{x}_j \in \mathfrak{D}$ by $\bar{\lambda}(q_j,\bar{x}_j)$ where $q_i \neq q_j$, then the states of the machine under test before applying these sequences will be different. In other words, statement A holds for the sequences of a D-set.

Corollary 1: Let M be an N-state machine. If an experiment is constructed using Hennie's algorithm for designing checking experiments,[5] except for the fact that each state q_i is identified by the corresponding $\bar{x}_i \in \mathfrak{D}$ (instead of by a fixed distinguishing sequence), then this experiment is a checking experiment for all faults that do not increase the number of states.

Proof: In case the machine under test responds incorrectly to the experiment, it is known to be faulty. If it responds correctly, each $\bar{x}_i \in \mathfrak{D}$ has appeared at least once, and with a correct response. Therefore, the preceding theorem implies that the machine under test has at least N states, and thus, because of the assumption regarding the class of faults, exactly N states. Since the algorithm further insures that all transitions are checked, the result is indeed a complete checking experiment, following similar arguments as discussed, for example, by Hennie [2].

Corollary 2: An optimal distinguishing set constitutes the *theoretical optimum* for a set of sequences such that any two different states can be distinguished by knowing only their responses to the two corresponding sequences of the set.

Proof (outline): Because of condition A, such a set of sequences has to be a D-set (necessary condition). The above theorem shows that a D-set indeed constitutes a set with the required features (sufficient conditon). Thus an optimal D-set is the optimum set satisfying both necessary and sufficient conditions.

[2] That is, the same for all states.
[3] Obviously, the subscript is redundant for a distinguishing sequence.
[4] In fact, all state identification methods that have been proposed thus far can be reduced to this second case.
[5] More precisely, the algorithm described in [2], where it is assumed that no fault can increase the number of states.

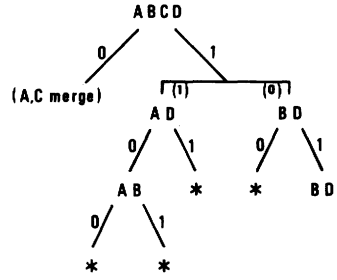

Fig. 1. Design of a distinguishing set for machine M_1.

Remark: It has been found that D-sets can be easily confused with earlier well-known concepts, especially compound distinguishing sequences [5].

For the sake of conciseness, we do not elaborate here on this subject, but refer to a different discussion [8] where this matter is clarified by means of an example.

D. Construction of an Optimal Distinguishing Set

In principle, a D-set can be designed directly from the definition. Since this would require keeping track of the set of all pairs of states, such a method would be rather cumbersome. However, the same result can be obtained by keeping track of the partitions induced on the set of states by the input sequences that are being constructed. Two states are considered as equivalent (or belong to the same partition block) under a given input sequence iff the machine, when started in either one of these states, gives the same response to this input sequence. The design of a D-set then amounts to a tree search (as shown by Fig. 1), which is similar to the design of adaptive distinguishing experiments (ADE). This design, and its optimization, are explained in [3, ch. 3]. We only point out two essential differences between the design of an ADE and of a distinguishing set. First, in an ADE, the tree is used during the experiment itself, while for D-sets, it is used only during the design. Second, during an ADE, one may choose arbitrarily among equally acceptable (qua optimality) next inputs, while during the design of a D-set, it is not allowed to choose different inputs for different states, unless these states have already been distinguished earlier. This is a direct consequence of the definition.

Comparison with (Preset) Distinguishing Sequences: From the preceding design method, it follows that the sequences of an optimal D-set are usually much shorter, and never longer, than the shortest preset distinguishing sequence. The resulting reduction in the length of checking experiments is obvious. However, in the next section it will become clear that the aforementioned telescoping feature of the sequences of a D-set, together with the use of a transition check status table, yields a systematic method for further reduction.

III. APPLICATIONS

In this section, the application of D-sets in checking experiments is illustrated by means of examples. The transition check status table will be introduced informally during the development of these examples.

A. Design of a Checking Experiment for M_1

The shortest synchronizing [3] sequence is 0101 and brings the faultless machine into state D. Using the distinguishing set of Table I(b), the states are then identified in the most convenient order [Table II(a)]. Subsequently, the transition verifications can proceed in the usual fashion [2], as shown in Table II(b). For each transition check, an entry is made into the *transition check status table* [Table II(c)], which is self explanatory and keeps track of the transitions that are verified.

More Efficient Design of Checking Part: Because of the strongly connectedness, every state appears at least once in some next-state entry of the state table. Consequently, every state will be automatically identified during the transition checks, and the initial state identification sequence [Table II(a)] is redundant. Only at the very beginning, some explicit state identification may be necessary in order to identify the state of the machine after the initialization.

TABLE II
CHECKING EXPERIMENT FOR M_1

(a) State Identification

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

(2)

(b) Transition Verification

	1	1	1	1	0	1	1	1	0	1	1	0	1	1	0	0	1	0	0	1	0	1	1	1	0	
A		B	D		C	A		B	C			A	D		C	B		A	B		A	D	B			
	1	0	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)				

(c) Transition Check Status Table

state	(input 0)	(input 1)
A	7	5
B	4	1
C	6	3
D	2	8

The use of distinguishing sets rather than distinguishing sequences frequently makes it possible to *telescope* the 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 input symbol for $\bar{x}_A = 11$. The fashion, in which the partition blocks are split up during the design of a D-set, shows that such a possibility will arise very often. Further, since we always know the state of the faultless machine during the design of the checking part, we can look for this possibility each time a new input symbol is added to the sequence. The information gained from past transitions, stored in the transition check status table, is always at hand to bring the machine into a known state more quickly.

These observations are illustrated in Table III(a),(b). Each time a new transition is verified, we go back and see whether new known states can be filled in for the machine under test. If such is the case, this may imply that other transitions are automatically verified (e.g., in Table III(a) and (b), the numbers indicate the order in which the transitions are checked).

Note also that the transition check status table can be extended to store knowledge about past transition sequences, instead of single transitions only. This should be done at least for the sequences $\in \mathfrak{D}$. For example, in Table III(a), the state q is known because $\delta(D,10) = C$ has been verified at the start.

Conclusion: Comparing the checking sequence of Table III(a) with the checking sequence of Table III(c) where a distinguishing sequence is used shows that, for this particular example, a 50 percent

TABLE III
Improved Checking Experiment for M_1 and Comparison with Hennie's Method

(a) A More Efficient Checking Experiment for M_1

(input)		1	0	1	1	1	0	0	1	0	1	1	1	0	0	1	0	0	1	1	0
(state)	D		C	A	D		q	B		A	D	B		A	B		A	B	D		
(output)	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
		(5)	(1)	(2)			(3)		(8)		(4)			(6)				(7)			

(b) Transition Check Status Table

state	(input 0)	(input 1)
A	6	2
B	5	7
C	3	1
D	8	4

(c) Checking Experiment Using a Distinguishing Sequence

1	$(100)^2$	0	$(100)^2$	1	$(100)^2$	1	100	0	0	100	1	0	100	1	1	100	1	0	1	100	
	B C		C A		B D		C B			D A			D B			A D					

TABLE IV
Checking Experiment for a Machine that has No Distinguishing Sequence

(a) State - Output Table

present state	next state / output (input 0)	(input 1)
A	D / 1	B / 0
B	A / 0	C / 0
C	B / 1	B / 0
D	C / 0	B / 1

(b) Checking Experiment

Synchronizing sequence: 1001

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

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

saving in length has been obtained. For other examples, chosen in an arbitrary fashion from various sources in the literature, the observed reduction wa always better than 30 percent.

B. Example of a Machine with a Distinguishing Set but no Distinguishing Sequence

Such a machine is represented in Table IV(a). An optimal D-set is given by 01,001,01,001 for states A,B,C,D, respectively. The resulting checking experiment with length 31 is shown in Table IV(b). A set of locating sequences based on the set of characterizing sequences $\{1,01\}$ is given by $(10)^*01, (10)^*01, (11)^*01,1$ for states A,B,C,D, respectively. A single appearance of each locating sequence already requires 31 symbols, even though some ad hoc methods have been used in making the locating sequences shorter (e.g., taking $(10)^N01$ instead of $(10)^{N+1}01$). In this kind of a situation, the use of distinghishing sets is obviously very advantageous.

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.

The ideas leading to the definition of distinguishing sets can be applied to characterizing sequences, and thus yield shorter locating sequences (e.g., in case no distinguishing set exists).

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 given partition block in the design tree. 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," in *Automata Studies* (Annals of Mathematical Studies, no. 34), C. E. Shannon and J. McCarthy, Ed. Princeton, N. J.: Princeton Univ. Press, 1956, pp. 129–153.

[2] F. C. Hennie, "Fault detecting experiments for sequential circuits," in *Proc. 5th. Ann. Symp. Switching Theory and Logic Design*. Princeton, N. J.: Princeton Univ. Press, 1964, pp. 95–110.

[3] ——, *Finite State Models for Logical Machines*. New York: Wiley, 1968.

[4] E. P. Hsieh, "Optimal checking experiments for sequential machines," Ph.D. dissertation, Columbia Univ., New York, N. Y., Sept. 1969.

[5] ——, "Checking experiments for sequential machines," *IEEE Trans. Comput.*, vol. C-20, pp. 1153–1166, Oct. 1971.

[6] I. Kohavi and Z. Kohavi, "Variable-length distinguishing sequences and their application to fault-detection experiments," *IEEE Trans. Comput.*, vol. C-17, pp. 792–795, Aug. 1968.

[7] J. Hartmanis and R. E. Stearns, *Algebraic Structure Theory of Sequential Machines*. Englewood Cliffs, N. J.: Prentice-Hall, 1966.

[8] R. T. Boute, "Adaptive design methods for checking experiments," Digital Syst. Lab., Stanford Electron. Lab., Stanford, Calif., Tech. Rep. 30, July 1972.