010261-8-T

# Theory of Reliable Systems

Final Technical Report covering the period from May, 1971 to June, 1973

J. F. MEYER

July 1973



Prepared under
NASA Grant NGR23-005-463

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SYSTEMS ENGINEERING LABORATORY

THE UNIVERSITY OF MICHIGAN, ANN ARBOR

THE UNIVERSITY OF MICHIGAN

SYSTEMS ENGINEERING LABORATORY

Department of Electrical and Computer Engineering
College of Engineering

SEL Technical Report No. 73

# THEORY OF RELIABLE SYSTEMS

Final Technical Report Covering the Period from
May, 1971 to June, 1973

Project Director

J. F. Meyer

Research Assistants

Z. Aran
L. Hsieh
R. J. Sundstrom
K. Yeh

Programmers

J. G. Bravatto
W. E. Bulley

Prepared under

NASA Grant
NGR23-005-463

Table of Contents

# 1. INTRODUCTION

## 1.1 Objectives

The following is the final report for a research project on the "Theory of Reliable Systems" conducted under NASA Grant NGR23-005-463. The duration of this project was approximately two years, beginning on May 26, 1971 and terminating on June 30, 1973. The purpose of this project was to answer certain fundamental questions relating to the analysis and design of reliable systems, where the systems of primary concern were digital, e.g., digital computers, digital communication systems, digital control systems, etc. The attributes of system reliability to be studied were:

a) Fault tolerance - the ability to maintain error-free input-output behavior in the presence of (temporary and/or permanent) faults in the system

b) Diagnosability - the ability to detect and locate faults in the system

c) Reconfigurability - the ability to reconfigure the system after the occurrence of a fault so as to realize the original behavior or some other (possibly less complex) behavior

with the following objectives:

I. To determine, relative to the above attributes, properties of system structure that are conducive to a particular attribute. Structures so considered range from state-transition functions at one extreme to hardware and software realizations at the

other extreme.

II. To determine methods for obtaining reliable realizations of a given system behavior. This could eventually include realizations which are fault tolerant (relative to the specified behavior) and yet diagnosable (relative to some extended behavior).

III. To determine how properties of system behavior relate to the complexity of fault tolerant (diagnosable, reconfigurable) realizations. Given such relationships, the inherent fault tolerance (diagnosability, reconfigurability) of a given behavior could be measured by the minimum complexity of realizations possessing that reliability attribute.

The above objectives comprise a general statement of the project's direction and goals, as they were conceived when the research was first proposed. Almost all of the investigations conducted during the course of the project had specific goals that were in keeping with one or more of these global objectives. Of the three basic reliability attributes proposed for study, only two, fault tolerance and diagnosability, were investigated in detail during the two year period. Questions of reconfigurability were considered informally in connection with models developed for the study of tolerance and diagnosability. This was done in anticipation of more formal studies of reconfiguration and repair that could be based on such models or on appropriate extensions thereof.

The general approach taken to meet the above stated objectives was system-theoretic in the sense that the study was based on mathematical models and simulation models that represent the structure of a digital system's hardware and/or internal software. Given some class of "real" systems (e.g., switching circuits, computer programs in a given language, etc.), various classes of models at various levels of structural definition can be developed to study this class of real systems. Thus, for example, switching circuits can be represented at a low level by sequential network models or at a high level by sequential machine models. In general, the choice depends on the nature of the questions being asked about the external and internal behavior of the class of real systems under investigation. An advantage of this approach in studying system reliability is that structural changes (due to faults) can be precisely related to their effects on system behavior, thereby permitting the discovery of properties conducive to reliable operation of a system.

## 1.2 Background

The general setting for this research project was the theory of reliable systems that had been developed over the past two decades using the basic approach described above. The first person to use this approach to the study of reliable systems was von Neumann ( [1], 1952, 1956), where the models were networks of switching components (called "organs") and faults in a component were represented by the probability of erroneous component behavior. The next such

effort was the work of Moore and Shannon ( [2], 1956) in which case the models were (formal) relay networks and faults in a relay were represented by two conditional probabilities regarding relay behavior.

Since the time of these initial investigations, this general approach has been used extensively to study various aspects of system reliability and, in particular, computing system reliability (cf. the excellent bibliography by Short [3]). Several books that in some part, at least, are representative of this approach have also appeared during the past decade. These include a collection of papers from the "Symposium on Redundancy Techniques in Computing Systems" ( [4] 1962), a monograph on "Reliable Computation in the Presence of Noise" by Cowan and Winograd ( [5] 1963) and books on "Failure-Tolerant Computer Design" by Pierce ( [6] 1965), "Error Detecting Logic for Digital Computers" by Sellers, Hsiao and Bearnson ( [7] 1968), "Fault Diagnosis of Digital Systems" by Chang, Manning, and Metze ( [8] 1970), and "Fault Detection in Digital Circuits" by Friedman and Menon ( [9] 1971).

Just prior to the initiation of this project, the first International Symposium on Fault Tolerant Computing was held in Pasadena, California, March 1-3, 1971. The papers summarized in the Digest of this conference [10] are representative of the variety of topics presently regarded as relevant to the theory and design of reliable computing systems. Two of these papers, one on the subject of fault location [11] and another on diagnosable machine realizations [12]

were the result of work completed under a prior contract with the Jet Propulsion Laboratory [13] and provided an immediate background for this project at the time of its initiation. The work summarized in [11] also appeared in a paper titled "Locatability of Faults in Combinational Networks" [14]. Another phase of the research completed for J. P. L., which appeared later in a paper titled "Fault Tolerant Sequential Machines" [15], had a strong influence on our present work, not only with regard to research on fault tolerance but also with regard to developing a general framework for the study of systems with faults. Regarding the various specific investigations conducted as part of this project there are, of course, many other references that served as important background. Such references are appropriately indicated in the body of this report and the accompanying technical reports.

## 1.3 Documentation

Concluding with this final report, the research performed under this grant has been extensively documented throughout the duration of the project via semi-annual status reports, technical reports, and publications in the proceedings of technical symposia. Rather than delay the disclosure of this information to a single final report or to its eventual publication in journals, we have attempted to disseminate the methods and results of the work whenever it appeared feasible. The following is a list of the reports and publications

(excluding this report) that comprise the total documentation of this project. Entries are by title and reference number. The reference list should be consulted for information as to author, date, etc.

Semi-annual reports

Semi-annual status report no. 1 [16]

Semi annual status report no. 2 [17]

Semi-annual status report no. 3 [18]

Technical reports

General compound distinguishing sequences [19]

Representation and minimization of diagnostic test sets [20]

A theoretic study of fault detection problems in sequential systems [21]

Checking experiments for sequential machines [22]

A structurally oriented simulation system [23]

Augmentation of machine structure to improve its diagnosability [24]

On-line diagnosis of sequential systems [25]

Publications

On the limits of linearity [26]

Sequential behavior and its inherent tolerance to memory faults [27]

A general model for the study of fault tolerance and diagnosis [28]

Diagnosis of unrestricted faults in sequential machines [29]

Copies of the semi-annual status reports, technical reports [19]-[22], and publications [26]-[28] have been previously submitted as part of the interim reporting process. Copies of technical reports [23]-[25] and the published abstract [29] are included with the submission of this final report.

In the sections that follow, we present an overview of the research performed throughout the two year duration of the project. The discussion is organized according to four general areas of investigation: systems with faults, fault tolerance, fault diagnosis, and simulation. With regard to each of these areas, this report summarizes the various specific topics that were investigated in that area, the motivation for their study, the models on which their study was based,and the results obtained. The primary intent of this summary is to provide a perspective for the various technical reports and publications cited above. It is the latter that comprise a detailed report of the research performed under the grant.

## 2. SYSTEMS WITH FAULTS

The purpose of this investigation was to develop general, formal basis for the study of fault tolerance and diagnosis in systems. This was achieved by developing a theoretical model of a "system with faults." Based on this model and the fundamental concept of a "tolerance relation," the intuitive notions of "fault tolerance" and "diagnosability" were then formulated in precise, yet general terms. Depending on the choice of a representation scheme for the systems in question, the model can represent either the effects of design errors or the effects of physical faults that occur during the life of a system. Also, depending on the representation used, the model permits the study of faults in either the hardware or the software of a computing system.

Beginning with a more restricted notion of a "machine with faults," the model evolved to its present level of generality during the second year of the project. The results of this investigation were presented at the Sixth Hawaii International Conference on System Sciences, Honolulu, January 9-11, 1973 and published in the proceedings of that conference [28].

Informally, a "system with faults" is a system, along with a set of potential faults of the system and description of what happens to the original system as the result of each fault. The original system and the systems resulting from faults are members of one or two prescribed classes of (formal) systems, a "specification"

class for the original system and a "realization" class for the resulting systems. More precisely, specifications and realizations are represented by a representation scheme $(\mathcal{S}, \mathcal{R}, \rho)$ where

i) $\mathcal{S}$ is a class of systems, the specification class,

ii) $\mathcal{R}$ is a class of systems, the realization class,

iii) $\rho: \mathcal{R} \longrightarrow \mathcal{S}$ where, if $R \in \mathcal{R}$, R realizes $\rho(R)$.

To illustrate this notion, consider the following classes of systems:

$\mathcal{N}$ = all sequential switching networks

$\mathcal{M}$ = all finite state sequential machines

$\mathcal{F}$ = all n-ary, numerical partial recursive functions

$\mathcal{T}$ = all Turing machines

$\mathcal{P}$ = all pairs of n-ary predicates over the integers Z

$\mathcal{P}$ = all programs (in the sense of King [30]) on n variables with values from Z

Relative to the above classes, some examples of a representation scheme are given by the following choices of $\mathcal{S}$, $\mathcal{R}$, and $\rho$.

| | $\mathcal{S}$ | $\mathcal{R}$ | $\rho: \mathcal{R} \longrightarrow \mathcal{S}$ |
|---|---|---|---|
| 1) | $\mathcal{N}$ | $\mathcal{N}$ | identity function |
| 2) | $\mathcal{M}$ | $\mathcal{M}$ | identity function |
| 3) | $\mathcal{M}$ | $\mathcal{N}$ | $\rho(N)$ = the machine M defined by N |
| 4) | $\mathcal{F}$ | $\mathcal{T}$ | $\rho(T)$ = the partial recursive function f realized by T |
| 5) | $\mathcal{Q}$ | $\mathcal{P}$ | $\rho(P)$ = some predicate pair (I, J) such that P is correct for I and J |

In a representation scheme $(\mathcal{S}, \mathcal{R}, \rho)$, a system with faults is a structure $(S, F, \phi)$ where

i) $S \in \mathcal{S}$

ii) F is a set, the faults of S

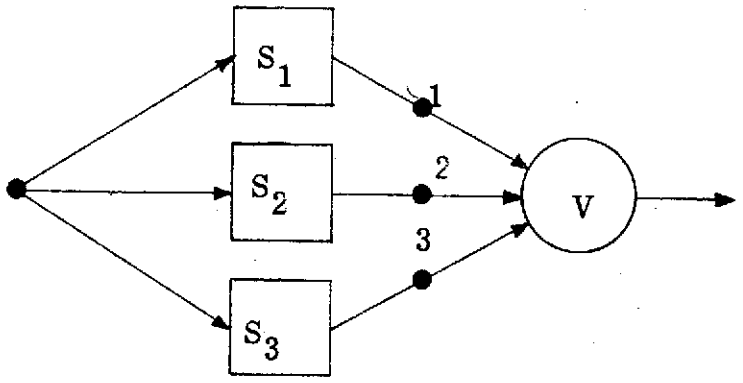
iii) $\phi$: $F \longrightarrow \mathcal{R}$ such that, for some $f \in F$, $\rho(\phi(f)) = S$.

If $f \in F$, the system $S^f = \phi(f)$ is the result of f. If $\rho(S^f) = S$ then f is improper (by iii), F contains at least one improper fault); otherwise it is proper. A realization $S^f$ is fault-free if f is improper; otherwise $S^f$ is faulty.

A system with faults is a formal representation of a (potentially) unreliable system where S represents the original fault-free specification, F represents a set of potential faults that can occur in the process of realizing S, and for a given $f \in F$, $S^f = \phi(f)$ is the realization that results from the occurrence of f. An improper fault f represents a fault-free realization process since, by definition, $\rho(S^f) = S$, i.e., $S^f$ realizes S. A proper fault f represents a faulty realization process in the sense that $\rho(S^f) \neq S$.

To illustrate the notion of a system with faults, consider the representation scheme $(\mathcal{S}, \mathcal{R}, \rho)$ where $\mathcal{S}$ and $\mathcal{R}$ are some class of networks of sequential switching systems ($\mathcal{S} = \mathcal{R}$) and $\rho$ is the identity function. Suppose further that $S \in \mathcal{S}$ is the system:

S:




where S employs triple modular redundancy, that is, $S_1 = S_2 = S_3$ and V is a voter. Suppose further that the potential faults are combinations of stuck at 0 and stuck at 1 faults at nodes 1, 2 and 3. Then, in the representation scheme $(\mathcal{S},\mathcal{R},\rho)$, $(S, F,\phi)$ is a system with faults where

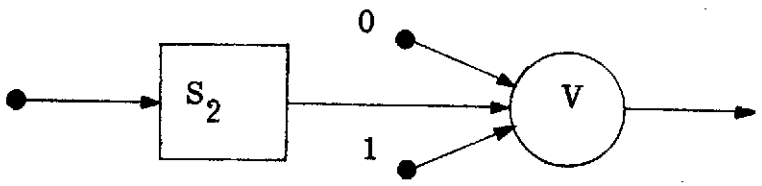
S is as above

$$F = \{(a_1, a_2, a_3) \mid a_i \in \{0, 1, x\}\}$$

$$\text{where } a_i = \begin{cases} 0 \text{ if node i stuck at 0} \\ 1 \text{ if node i stuck at 1} \\ x \text{ if node i is fault-free} \end{cases}$$

$$\phi: F \longrightarrow \mathcal{R}$$

where, for example, $\phi(0, x, 1)$ is the faulty system:




One of the important unifying aspects of the concept of a system with faults is that it permits the representation of either birth defects or life defects. Moreover, what distinguishes these two types of defects is seen clearly via the formalism. In representing faults

that occur in the process of designing a system, a representation scheme $(\mathcal{S},\mathcal{R},\rho)$ is chosen such that the original design specification is represented by a member of the specification class $\mathcal{S}$ and the possible outcomes of the design process by members of the realization class $\mathcal{R}$. Since specifications differ from realizations in this case (or otherwise we would be saying that the outcome of the design process is already known), a study of birth defects requires a representation scheme $(\mathcal{S},\mathcal{R},\rho)$ where $\mathcal{S} \neq \mathcal{R}$. On the other hand, in representing faults that occur in the process of using a system, one begins with a realization that is presumed fault-free and is concerned with faulty realizations that result from life defects. In this case, a specification is a realization and one chooses a representation scheme $\mathcal{R} = (\mathcal{R},\mathcal{R},\rho)$. Thus, a representation scheme required for a study of life defects is simply a special case of a more general representation scheme required for the study of birth defects. Among other things, this suggests that concepts and techniques studied in the context of design reliability (e.g., program verification techniques) should be applicable to questions of reliable use (e.g., diagnosis of hardware faults).

A surprisingly wide variety of unreliable systems, from switching networks to computer programs, can be formally represented as systems with faults. This includes applications where there is no explicit representation of faults, but only an explicit representation of the faulty systems. Moreover, the formulation

permits precise, general definitions of "tolerated" fault and "diagnosable" fault that apply to an arbitrary system with faults. The definitions are made relative to a specified "tolerance relation" $\tau$ which dictates the type and extent of tolerance or diagnosability. By choosing two tolerance relations $\tau$ and $\tau'$, it is possible to consider faults which are both tolerated (with respect to $\tau$) and diagnosable (with respect to $\tau'$). This yields a convenient representation for investigating the diagnosis of tolerated faults.

Much of the research conducted throughout the course of this project was based on special classes of systems with faults or on models which could be equivalently formulated as systems with faults. Likewise, the various concepts of tolerance and diagnosis considered were specializations of the general definitions referred to above. Consequently, a familiarity with the concepts described here and in reference [28] is assumed in the discussion that follows.

## 3. FAULT TOLERANCE

In keeping with the general statement of the goals of this project (objectives I, II and III; section 1.1), the study of fault tolerance in digital systems had three specific objectives:

i) To determine structural properties of sequential machines that are necessary and/or sufficient for the tolerance of permanent memory faults.

ii) To determine methods for obtaining fault tolerant state-assigned machine realizations of a specified sequential behavior that possess minimum or near minimum memory redundancy (relative to all realizations of that behavior which have the same fault tolerance).

iii) To determine properties of sequential behavior that relate to "inherent" fault tolerance where, relative to some tolerance type $\tau$ and tolerance level t, the inherent tolerance of a behavior B is inversely proportional to the minimum memory redundancy required by a $(\tau, t)$-tolerant realization of B.

If we let $\mathfrak{M}$ denote the class of all finite-state sequential machines then, relative to the general framework described in the previous section, the representation scheme for this study was $(\mathfrak{M}, \mathfrak{M}, \rho)$ where $\rho$ is the identity function (see example 2), p. 9 ). The faults considered were "memory faults" in the sense of [15], where if $M = (I, Q, Z, \delta, \omega)$ is a machine (i.e., $M \in \mathfrak{M}$, a memory fault of M is

a function $\mu$: $Q \longrightarrow Q$ on the states of M. The result of $\mu$ is the sequential machine $M^{\mu} = (I, Q^{\mu}, Z, \delta^{\mu}, \omega^{\mu})$ where

$$Q^{\mu} = \{\mu(q) \mid q \in Q\}, \text{ the image of } \mu$$

$$\delta^{\mu} = \mu\delta \text{ restricted to } Q^{\mu} \times I$$

$$\omega^{\mu} = \omega \text{ restricted to } Q^{\mu} \times I$$

(Reference [15] should be consulted for justification and illustration of these definitions.) Accordingly, a system with faults, in this context, is a system $(M, F, \phi)$ where M is a sequential machine, F is a set of memory faults containing the identity function (the improper fault), and $\phi$ is the function given by $\phi(\mu) = M^{\mu}$, for all $\mu \in F$.

The real systems so represented by this formalism are discrete-time, time-invariant, finite-alphabet (e.g., binary), finite-state systems that are subject to faults in their memory structure. Thus, for example, these models can be used to study the effects of faulty memory elements (e.g., flip-flops) on the behavior of a digital circuit. The restriction to memory faults was motivated by the fact that it is memory which distinguishes nontrivial finite-state systems from purely combinational (one-state) systems. Moreover, as the purpose here was to study how the structure of a machine relates to the effects of faults on behavior, the restriction to memory faults is not that severe.

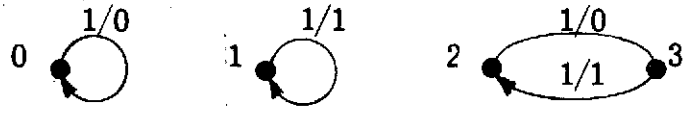
Given this class of systems (machines) with faults, several types of fault tolerance were considered, each corresponding to a specific "tolerance relation" $\tau$ (cf. [27]). These include the notions of "equivalence masking," "inclusion masking," and "R-masking" (where

R is a distinguished set of reset states) [15].

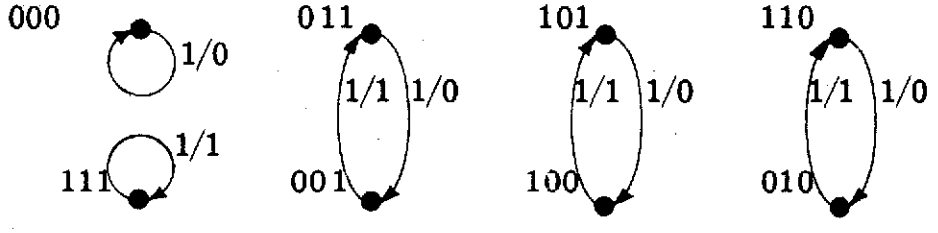
With respect to a specific type of tolerance, the topics of primary concern were "minimally redundant fault tolerant realizations" and the "inherent tolerance of a sequential behavior" (cf., objectives ii) and iii) stated at the beginning of this section). Prior to this study, methods for obtaining fault tolerant realizations of a given behavior have all resulted in "memory redundancy" levels that depend only on the number of states required by a nonredundant realization. Indeed, for certain realization methods (e.g., replicate-and-vote schemes), memory redundancy is invariant. If, however, the method of realization is left unspecified then, relative to all realizations having a specified type and level of fault tolerance, the minimum redundancy required (i.e., the inherent tolerance) may vary from behavior to behavior, even among behaviors requiring the same number of states in their reduced realizations.

One of the primary results of our investigation is that his is indeed the case, that is, there exist behaviors of the same "size" with different inherent tolerance. This and related results were first disclosed at the Fifth Hawaii International Conference on System Sciences, Honolulu, January 11-13, 1972. Precise statements of these results and their proofs are published in the proceedings of that conference [27]. In summary, the main result was established by exhibiting two behaviors, each of which can be realized by a 4 state reduced machine and yet one requires more memory redundancy

than the other to obtain a realization which tolerates single stuck-at faults. The behavior having less inherent tolerance (i. e., requiring greater memory redundancy) was that of a modulo 4 clock. The behavior having greater inherent tolerance was that of a (reduced) machine with the following state graph:




With respect to equivalence-masking of single stuck-at faults, a minimally redundant fault tolerant realization of the latter behavior is given by the following state graph:




A network realization of this machine is shown in Fig. 3. 1. Note that, relative to a minimal realization having no fault tolerance, only one additional delay flip-flop (3 as opposed to 2) is required to tolerate all single flip-flop faults. The memory redundancy of this realization is 1. 5 and, since it is minimally redundant, this number represents the inherent tolerance of the behavior realized. Accordingly, a fault tolerant realization based on triplication and voting, which yields a memory redundancy of 3, is not optimum with regard to this behavior. A network realization using the latter scheme is shown in Fig. 3. 2, where its complexity can be compared to that of Fig. 3. 1.

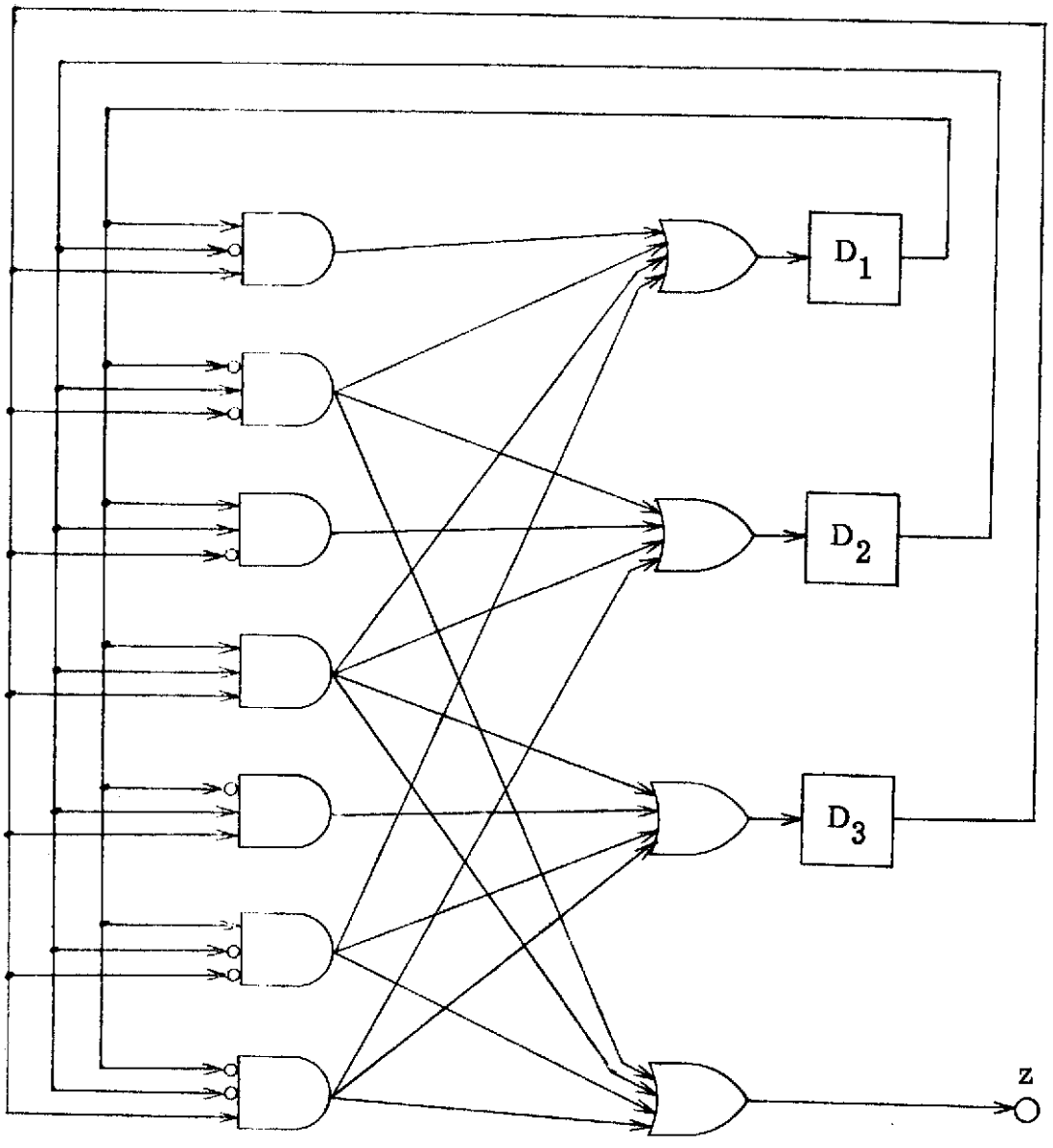



Figure 3.1

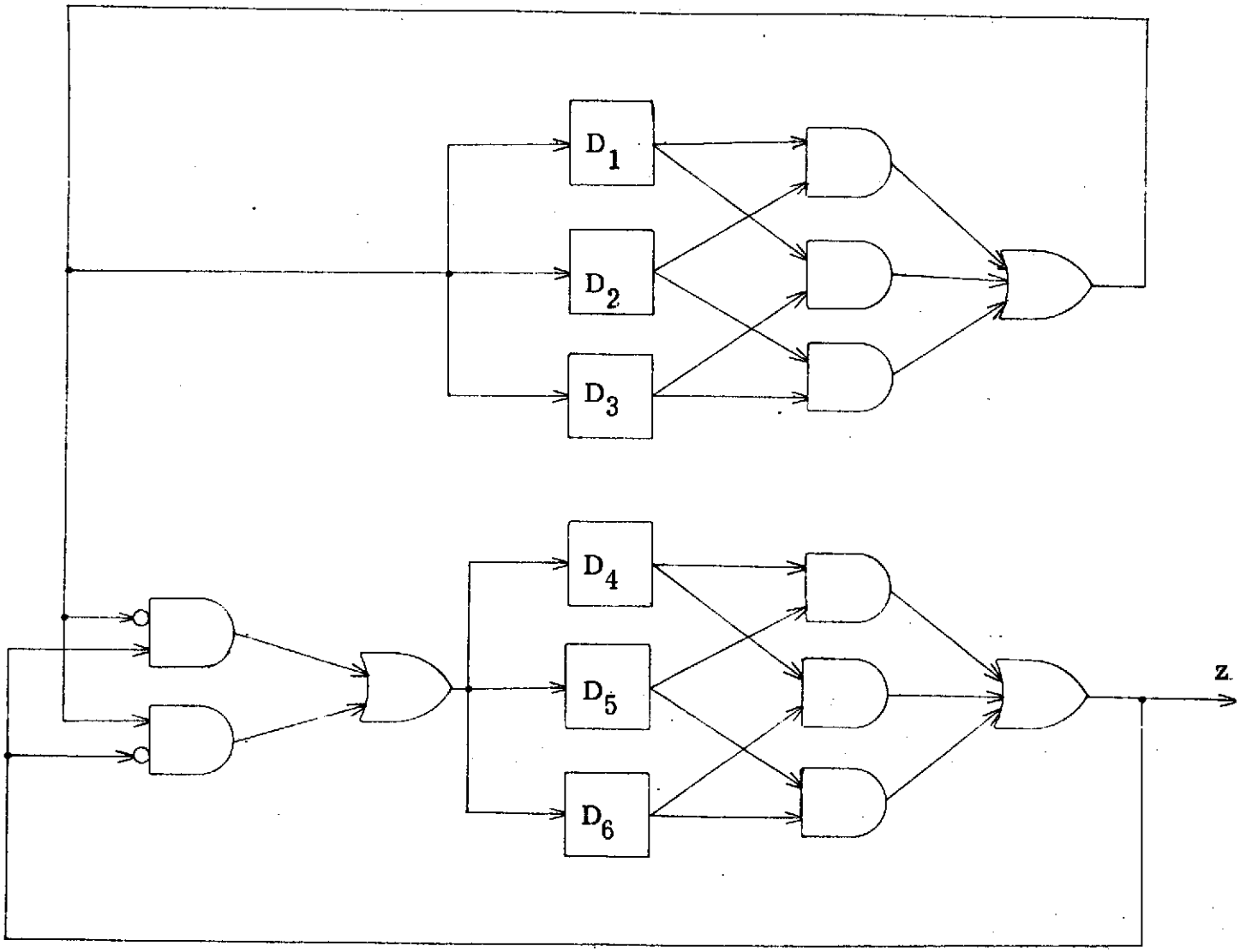



Figure 3. 2

The above results were obtained during the first year of the project. At the beginning of the second year, efforts were made to find additional examples and, more generally, to discover properties of sequential behavior are indicative of inherent tolerance. This task turned out to be much more difficult than originally anticipated and it soon became evident that some experimental data was needed to support further theoretical study. This prompted the development of a Structurally Oriented Simulation System (SOSS) for use as an experimental aid in the study of both fault tolerance and fault diagnosis (cf. section 5 and technical report [23]). As the basic version of SOSS was completed just prior to the termination of the project, its application in the study of inherent tolerance remains a subject of future research.

## 4. FAULT DIAGNOSIS

In keeping with the general statement of the goals of this project, the study of fault diagnosis in digital systems had three specific objectives:

i) To determine properties of machine structure and behavior that are conducive to efficient fault diagnosis.

ii) To determine methods for obtaining diagnosable machine realizations of machines (or behaviors) that are not diagnosable.

iii) To determine methods for obtaining machine and network realizations of a given behavior that are fault tolerant relative to the specified behavior but are diagnosable relative to the "extended" behavior of the realization.

During the course of the project, considerable effort was devoted to meeting objectives i) and ii), producing a variety of significant results which are summarized below. Such work had to precede any specific effort devoted to objective iii), since fault diagnosis, per se, had to be relatively well understood before considering the diagnosis of fault tolerant systems. Consequently, the latter effort was just getting underway when the project terminated.

Our study of fault diagnosis in digital systems was concerned with two basic diagnostic environments: "off-line" and "on-line." By an off-line environment, we mean that the system has been removed from its operating environment prior to the application of diagnostic tests.

Thus, off-line diagnosis permits complete control of input to the system during the test period. This is to be contrasted with "on-line" or "concurrent" diagnosis where the system is in an operating environment and is continually receiving input dictated by this environment. An on-line diagnosis procedure must therefore contend with input over which it has no direct control. Moreover, in many applications, an error produced by a fault must be detected in a relatively short period of time after the error occurs, thereby complicating the problem even further.

During the first year of the project our study of fault diagnosis was concerned exclusively with the off-line environment. During the second year, both off-line and on-line diagnosis were investigated. These efforts are summarized in the subsections that follow.

## 4.1 Off-line diagnosis

When a possibly faulty system is diagnosed in an off-line environment, it is usually assumed that no additional faults occur once the diagnostic procedure is initiated. With this assumption, off-line diagnosis of sequential systems can be studied relative to a representation scheme $(\mathfrak{M}, \mathfrak{M}, \rho)$ where $\mathfrak{M}$ is a class of sequential machines. If $(M, F, \phi)$ is a machine with faults (in the above representation scheme), our research on off-line diagnosis can be further categorized according to the nature of the fault set F. In general, we can regard the faults of $(M, F, \phi)$ as being "specified" by the set F. However, when the elements of F are specified no more explicitly than

the faulty machines they result in, we say that the faults are "unspecified." More precisely, $(M, F, \phi)$ is a machine with unspecified faults if $F \subseteq \mathfrak{M}$ and $\phi(f) = f$, for all $f \in F$. Otherwise, the faults are specified. Thus, in a machine with unspecified faults, no distinction is made between "cause" (fault) and "effect" (result of a fault). In particular, it is impossible for different faults to result in the same faulty machine. This need not be the case, however, when faults are specified. It is for these reasons that machines with unspecified faults have been distinguished from machines with specified faults. We begin with a discussion of the latter.

4.1.1. Specified faults

In the representation scheme $(\mathfrak{M}, \mathfrak{M}, \rho)$ where $\mathfrak{M}$ is the class of all finite-state sequential machines, the systems considered here were machines with memory faults (cf. section 3 of this report), i.e., the same class of systems with faults considered in our investigation of fault tolerance. This choice was motivated by the prospect of eventually combining results concerning tolerance and diagnosis to obtain machine and network realizations that are both fault tolerant and diagnosable.

More specifically, given a sequential machine $(M, F, \phi)$ with memory faults, the problem studied was the representation and minimization of a set of diagnostic "tests" for all faults in F. The type of diagnosis considered was fault "detection" relative to some specified initial state. More precisely, if $\mu \in F$ and q is a state of M,

an input sequence x is a $(\mu, q)$-detection sequence if the resulting output sequence for M started in q differs from that of the faulty machine $M^{\mu}$ started in $\mu(q)$. (If there is at least one $(\mu, q)$-detection sequence, $\mu$ is q-detectable; in other words $\mu$ is q-detectable if and only if it is not $\{q\}$-masked in the sense of [15].) Accordingly, a set X of input sequences is an $(F, q)$-test set if, for all $\mu \in F$, X contains a $(\mu, q)$-detection sequence.

Although the problem of representing and minimizing an $(F, q)$-test set has been studied from various points of view for the special case of combinational (one-state) systems, the results do not apply, in general, to nontrivial sequential systems. In our study, it was found that similar results could indeed be obtained for sequential machines, although the actual process of determining a minimum cost test set for a machine is, in general, much more difficult. A detailed description of this study and its results are given in the technical report "Representation and Minimization of Diagnostic Test Sets" [20] and in sections 4 and 5 of the technical report "A Theoretic Study of Fault Detection Problems in Sequential Systems" [21].

To summarize, the problem of representing detection sequences was considered first and it was shown that if D is the set of all sequences that distinguish faulty behavior (for a given fault fault) from fault-free behavior (in a given state), then D is a regular set. Moreover, a regular expression that denotes D can be obtained from expressions associated with the fault-free and faulty behaviors in much the same way that test sets

are determined for combinational networks using Boolean differences. The problem of obtaining a minimum cost test set (with cost measured in terms of testing time) was then considered where the primary difficulty lay in the fact that detection sets, although regular, are nevertheless infinite. However, we were able to show that the cost of an optimum (minimum cost) test is bounded from above by a quantity which depends only on the size of the fault-free machine and the number of faults to be detected. From this it follows that the length of longest sequence in any minimum cost test set is also bounded by a known quantity. Thus only a finite subset of each fault detection set needs to be considered in deriving an optimum test set. Employing some rather natural notions of fault equivalence and fault dominance, the problem of finding an optimum test set for a machine with faults was then formulated as a covering problem. This parallels the solution suggested by others for obtaining optimum test sets for combinational networks.

Although implementation of the procedure may be impractical for large networks, these results are nevertheless valuable in that they delimit the complexity of the problem. Such solutions may also indicate simpler, more practical methods of solving the problem at some sacrifice in optimality or completeness of the test set.

4.1.2. Unspecified faults

Relative to a particular sequential system, as represented by some sequential machine $M = (I, Q, Z, \delta, \omega)$, a general and yet tractable

representation scheme for machines with faults (where the machine in each case is M) is given by the set of all machines over I and Z that have at most as many states as M, that is, the set:

$$\mathfrak{M}(M) = \{M' \mid M' = (I, Q', Z, \delta', \lambda') \text{ and } |Q'| \leq |Q|\}.$$

If $(M, F, \phi)$ is a machine with faults in the representation scheme $(\mathfrak{M}(M), \mathfrak{M}(M), \rho)$, the physical restrictions implied by the representation are that i) a faulty system has the same input and output terminals as the fault-free system and ii) if Q represents all possible physical states of the fault-free system, then a permanent physical fault will not cause an increase in the number of physical states. (Note, however, that the number of nonequivalent states can increase since M need not be reduced.) Relative to a given machine M, the representation is therefore quite general. Thus, for example, a machine $(M, F, \phi)$ with memory faults, as previously defined in section 3, is also a machine with faults in the scheme $(\mathfrak{M}(M), \mathfrak{M}(M), \rho)$.

This scheme is also appropriate for the representation of faults in digital systems where nothing is assumed regarding the explicit nature of faults, per se, and relatively little is assumed about the effects of such faults. More precisely, in terms of a machine with faults $(M, F, \phi)$, it is assumed that the faults of M are unspecified (in the sense defined at the outset of section 4.1) and, in addition, are "unrestricted" in the sense that any machine in $\mathfrak{M}(M)$ is a possible faulty machine.

Historically, the diagnosis of machines with unspecified, unrestricted faults was first considered by Moore [31] who established the existence of a certain type of diagnostic "experiment" for strongly connected machines. His paper also posed several interesting open questions which were later investigated by Hennie [32]. The type of diagnosis embodied by Hennie's notion of a "checking sequence" is basically the type of diagnosis considered in our investigation, that is, fault detection as opposed to fault identification.

Fundamental to known methods of designing checking sequences has been the ability to identify which state the machine was in prior to applying the input sequence(s) used to implement the identification procedure. To permit this type of initial-state identification, checking experiments have employed distinguishing sequences [32], variable length distinguishing sequences [33], and compound distinguishing sequences [34]. An initial subject of our investigation in this area was an even more general class of sequences which include each of the above as special cases. To conform with earlier terminology, these sequences (which are actually sets of input sequences) are called general compound distinguishing sequences (GCDS). The detailsof this investigation are described in a technical report by the same name [19]. To summarize, it was shown that the concept of a GCDS is indeed more general than that of a compound distinguishing sequence. It was also shown that one can effectively decide whether a machine has a GCDS through

construction of a "general distinguishing tree." Finally, we have shown that existence of a GCDS for a machine M characterizes the existence of a simple, adaptive initial state identification experiment for M and, more generally, that the existence of a "general characterizing set" corresponds to the existence of a multiple, adaptive initial-state identification experiment.

A second topic of investigation was the "checkability" of a sequential machine, where the point of departure was a formal definition of checking sequence that includes the types of sequences so named by Hennie [32]. The definition applies to an arbitrary machine M (with unspecified, unrestricted faults) in an arbitrarily specified initial state q. In particular, M need not be reduced or strongly connected. Later on in the study, a special type of checking sequence called a detecting sequence was also considered. The difference between a checking sequence and a detecting sequence is that, in the latter case, a positive response to the sequence says that the state of the machine, just before application of the sequence, has the desired behavior; in the former case we can only guarantee that some state of the machine has the desired behavior. Hence a detecting sequence is better than a checking sequence in the sense that the former does not require a search for the desired state. The results of this investigation are described in the technical report "Checking Experiments for Sequential Machines" [22] and in a paper titled "Diagnosis of Unrestricted Faults in Sequential Machines," an ab-

stract of which was published in the Digest of the 1973 International Symposium on Fault-Tolerant Computing [29].

To summarize, the investigation here was primarily concerned with conditions on the structure and behavior of a machine M that are necessary and/or sufficient for M to possess at least one checking (detecting) sequence. In case M is reduced and reachable from some state q, it was shown first that the existence of a checking (detecting) sequence can be characterized directly in terms of the input-output behavior of M in state q. For the general case where M need not be reduced and reachable, several necessary conditions for an input sequence to be a checking sequence and for a machine to be checkable were established. Particular attention was given to the "transition-checking" aspect of checking sequences. Specifically, the necessity of checking all the transitions of a machine was studied and, for the case when a checking sequence does not have to check all transitions, the portion that needs to be checked was delineated. The effects of structural redundancy were then examined relative to various types of initial state behavior and it was found that, in general, the existence of a checking sequence depends only on the initial state behavior of a machine, and not on its particular structure. Said another way, the checkability of an arbitrary machine M in state q depends only on the nature of the "canonical" (reduced and reachable) machine having the behavior of M in q. An actual characterization of checkability in these terms, however, remains an open question.

A third major activity regarding the (off-line) diagnosis of unspecified faults has been an investigation of methods for improving the diagnosability of a machine by "augmenting" its structure. This includes the extreme case where the original machine is not diagnosable. Given a machine M', an augmentation of M' is a machine M that can simulate the behavior of M' in real time through appropriate encoding the decoding of input and output symbols. (An augmentation M where only the input [output, state] set of M is larger than that of M' is called an input [output, state] augmentation.) What was sought, then, were augmentations M of M' that are diagnosable in some prescribed sense. Several types of diagnosability were so considered, including the possession of a checking sequence (i.e., "checkability") and the possession of a repeated symbol distinguishing sequence. The details of this research activity are described in sections II and III of the technical report "A Theoretic Study of Fault Detection Problems in Sequential Systems" [21] and in the technical report "Augmentation of Machine Structure to Improve its Diagnosability" [24].

To summarize, the fundamental question as to whether any noncheckable machine has a checkable augmentation was shown to have a positive answer. In fact, a checkable input-augmentation can be constructed for any machine with one more input symbol than the given machine. It has also been established that any checkable augmentation of a given minimal transition-distinct and nonsimply-

connected machine must have a larger input set than the given machine. Augmentation techniques were also investigated for another type of diagnosability, namely the ability to infer initial state using a repeated symbol distinguishing sequence (RDS). Such sequences are of interest since they can be used in the construction of more efficient (shorter) checking sequences. Here it was shown that input augmentations (with an RDS) exist for any machine but that state augmentations do not exist when the machine to be augmented is reduced and does not have a distinguishing sequence. It was also established that output augmentations always exist but generally correspond to circuit realizations having an excessive increase in the number of output terminals. This is undesirable relative to modern large scale integrated electronic technology, because a limited number of output pins are allowed for each LSI chip.

Our main effort, therefore, was directed toward the study of state-output augmentations, in which moderate enlargement of the output set is attained at the expense of an enlarged state set. One of the most significant and surprising results of this effort was the following fact: given an arbitrary sequential machine, there exists a state-output augmentation (with an RDS) which has no more than twice as many output symbols as that of the given machine. Equivalently, in circuit terms, there is a state-output augmentation that has at most one more output terminal than the given circuit. In addition, it was established by constructions that, in the worst case,

there exists at least one such state-output augmentation whose state set size is proportional to $n^2$, where n is the size of the state set of the given machine.

4.2 On-line diagnosis

In many applications, especially those in which a computer is being used to control some process in real-time (e.g. telephone switching, flight control of an aircraft or spacecraft, control of traffic in a transportation system, etc.), it is desirable to constantly monitor the performance of the system, as it is being used, to determine whether actual behavior is within tolerance of intended behavior. Informally, by "on-line diagnosis" we mean a monitoring process of this type, where the level or extent of diagnosis can be external to the diagnosed system, both external and internal, or completely internal. In the last extreme, on-line diagnosis is sometimes referred to as "self-diagnosis" or "self-testing" [8].

The motivation for initiating a theoretical study of on-line fault diagnosis is the increasing use of computers in real-time applications where i) erroneous operation can result in the loss of human life and/or large sums of money and ii) interruptions in the operation, for the purpose of off-line diagnosis, are intolerable. In particular, our discussions with NASA-Langley regarding such applications were influential in precipitating this study.

During the past decade, the development of theory and techniques for fault diagnosis in digital circuits and systems have focused mainly on problems of off-line diagnosis. This is due to the fact that on-line diagnosis is inherently a more complex process, the complicating factors being that i) the diagnostic procedure must contend with input over which it has no control and ii) faults can occur as the system is being diagnosed. Because of these factors, conventional time-invariant (stationary, fixed) system models (e.g., sequential networks, sequential machines, etc.) can no longer be used to represent the dynamics of a system as its being diagnosed.

Based on these observations, the initial problem considered in our study was the formulation of an appropriate class of system models (i.e., a class of systems with faults) that could serve as a basis for the theoretical study of on-line diagnosis. Once such systems were defined, the next problem considered was the formulation of notions of fault tolerance, error, diagnosability, realization, etc. that have a meaningful interpretation in the context of on-line diagnosis. After these concepts were made precise, certain fundamental questions were posed and their investigation was initiated.

The research outlined above is fully described in the technical report "On-line Diagnosis of Sequential Systems" [25]. To summarize, the realization class chosen for the representation scheme is a class of discrete-time systems which are not necessarily time-invariant.

More precisely, relative to the time-base $T = \{\ldots -1, 0, 1, \ldots\}$, a resettable discrete-time system (with finite input, output, and reset alphabets) is a system

$$S = (I, Q, Z, \delta, \lambda, R, \rho)$$

where

| | | |
|---|---|---|
| | $I$ | is a finite set, the input alphabet |
| | $Q$ | is a set, the state set |
| | $Z$ | is a finite set, the output alphabet |
| | $\delta$: | $Q \times I \times T \longrightarrow Q$, the transition function |
| | $\lambda$: | $Q \times I \times T \longrightarrow Z$, the output function |
| | $R$ | is a finite nonempty set, the reset alphabet |
| | $\rho$: | $R \times T \longrightarrow Q$, the reset function. |

The interpretation of a resettable discrete-time system is a system which, if at time t is in state q and receives input a, will at time t emit output symbol $\lambda(q, a, t)$ and at time $t + 1$ be in state $\delta(q, a, t)$. It is resettable in the sense that if reset r is applied at time $t - 1$ then $\rho(r, t)$ is the state at time t. Note that, in the special case where the functions $\delta$, $\lambda$, and $\rho$ are independent of time (i.e. are time-invariant), the definition reduces to that of a (resettable) sequential machine. In the discussion that follows we will refer to a resettable discrete-time system S as simply a system and will assume, unless otherwise qualified, that S is finite-state (i.e., $|Q| < \infty$).

The behavior of a system is described by first extending the transition and output functions to the domain $Q \times I^* \times T$, where $I^*$ is the set of all finite length sequences over I, including the null sequence (detailed definitions of the extensions are omitted here). Relative to the extended output function $\bar{\lambda}$, the behavior of S in state q is the function

$$\beta_q: I^+ \times T \longrightarrow Z$$

where $$\beta_q(x, t) = \bar{\lambda}(q, x, t).$$

Thus, if the state of the system is q and it receives input sequence x starting at time t, then $\beta_q(x, t)$ is the output emitted when the last symbol in x is received (i.e., the output at time t + length (x) - 1). It is also convenient to specify behavior relative to a reset input r that is released at time t, that is, the behavior of S for condition (r, t) ($r \in R$, $t \in T$) is the function

$$\beta_{r,t}: I^+ \longrightarrow Z$$

where $$\beta_{r,t}(x) = \beta_{\rho(r,t)}(x, t).$$

If t = 0, $\beta_{r,0}$ is referred to as the behavior of S for initial reset r and is denoted simply as $\beta_r$.

Assuming that a faulty system has the same input, output and reset alphabets as a fault-free system, the following class of systems suffices as a realization class.

$$\mathcal{S}(I, Z, R) = \{S' \mid S' = (I, Q', Z, \delta', \lambda', R, \rho')\}.$$

Accordingly, the representation scheme chosen for our study of on-line diagnosis is the scheme $(\mathfrak{R}, \mathfrak{R}, \rho)$ where $\mathfrak{R} = \mathcal{S}(I, Z, R)$ and $\rho$ is the identity function on $\mathfrak{R}$.

In such a scheme, the seemingly difficult problem of describing faults and their results becomes relatively straightforward. Given a system $S \in \mathcal{S}(I, Z, R)$, a fault f of S can be regarded as a transformation of S into some other system S' at some time $\tau$. Accordingly, the resulting faulty system looks like S up to time $\tau$ and like S' from $\tau$ thereafter. More precisely, if $S \in \mathcal{S}(I, Z, R)$, a fault of S is a triple

$$f = (S', \tau, \theta)$$

where $S' \in \mathcal{S}(I, Z, R)$, $\tau \in T$, and $\theta$: $Q \longrightarrow Q'$. (The function $\theta$ describes what happens to states as the fault occurs.) Given this formal representation of a fault of S, the result of $f = (S', \tau, \theta)$ is the system

$$S^f = (I, Q^f, Z, \delta^f, \lambda^f, R, \rho^f)$$

where $Q^f = Q \cup Q'$

$$\delta^f(q, a, t) = \begin{cases} \delta(q, a, t) & \text{if } q \in Q \text{ and } t < \tau - 1 \\ \theta(\delta(q, a, t)) & \text{if } q \in Q \text{ and } t = \tau - 1 \\ \delta'(q, a, t) & \text{if } q \in Q' \text{ and } t \geq \tau \end{cases}$$

$$\lambda^f(q, a, t) = \begin{cases} \lambda(q, a, t) & \text{if } q \in Q \text{ and } t < \tau \\ \lambda'(q, a, t) & \text{if } q \in Q' \text{ and } t \geq \tau \end{cases}$$

$$\rho^f(r, t) = \begin{cases} \rho(r, t) & \text{if } t < \tau \\ \theta(\rho(r, t)) & \text{if } t = \tau \\ \rho'(r, t) & \text{if } t > \tau. \end{cases}$$

(Arguments not specified in the above definitions may be assigned arbitrary values.)

In justifying this representation of the resulting faulty system one should regard a fault $f = (S', \tau, \theta)$ as actually occurring between time $\tau - 1$ and $\tau$. Note that, for any fault f of S, $S^f \in \mathcal{S}(I, Z, R)$. With these concepts of fault and faulty systems firmly established, a system with faults, in this representation scheme, is a structure

$$(S, F, \phi)$$

where $S \in \mathcal{S}(I, Z, R)$, F is a set of faults of S including at least one improper fault (e.g., $f = (S, 0, \theta)$ where $\theta$ is the identity function), and $\phi$: $F \longrightarrow \mathcal{S}(I, Z, R)$ where $\theta(f) = S^f$, for all $f \in F$. Given this definition, we crop the explicit reference to $\phi$ in denoting a system with faults, i.e., (S, F) means $(S, F, \phi)$ where $\phi$ is as defined above. This then completes the description of the system models on which our study of on-line diagnosis was based.

The fundamental notion of fault tolerance considered was behavioral equivalence with respect to a specified starting time t (i.e., the time the system is reset). More precisely, if (S, F) is a system with faults, a fault $f \in F$ is <u>tolerated for resets at time t</u> if

$$\beta_{r,t}(x) = \beta^f_{r,t}(x), \text{ for all } r \in R \text{ and all } x \in I^+.$$

If $\beta_{r,t}(x) \neq \beta^f_{r,t}(x)$ for some r, x, and t then we say that an <u>error</u> has occurred and is <u>caused by</u> f. The basic concept of diagnosability

considered involves a detector D (assumed to be fault-free) which operates in series with S, and a delay time k within which any error caused by a fault must be detected. More specifically, a system with faults (S, F) is (D, k)-diagnosable if, for all $f \in F$,

i) D responds negatively until the first occurence of an error caused by f,

and ii) D responds positively within k time steps of the first occurence of an error caused by f.

Given (S, F), a fundamental question is whether there exists a detector D and a delay k such that (S, F) is (D, k)-diagnosable. If the detector can observe the input to S as well as its output the question has a positive answer; simply let D be a copy of S (i.e., duplicate S). Then S is easily shown to be (D, 0)-diagnosable. Although duplication is an obvious solution to the problem of on-line diagnosis (and the one most frequently employed), it is also a costly solution. Consequently, one of the primary tasks undertaken was the investigation of detectors that are less complex than the systems they diagnose. Also sought were the possible tradeoffs between the complexity of a detector D and the magnitude of the time delay k.

Another fundamental question is how to alter the design of a system in order to improve its on-line diagnosability. More precisely, if (S', F') is (D', k')-detectable, we want to discover methods

for designing an augmentation (realization) S of S' such that (S, F) is (D, k)-detectable where D is less complex than D' and/or $k < k'$. A number of preliminary results have been obtained which begin to answer the questions posed above. Prospects for further research in this area are excellent, the outcome of which should substantially increase our basic understanding of on-line diagnosis.

## 5. SIMULATION

During the second year of the project, we initiated the development of a Structurally Oriented Simulation System (SOSS), a computer program to be used as an experimental aid in the study of reliable systems. Basically, SOSS is a program which can simulate the structure and behavior of a discrete-time, time-invariant, finite-state system. Structure of the simulated system is specified as a network of sequential machines with the ability to further specify local changes in structure that correspond to faults in the original system. This ability to "insert" faults and observe their effects on behavior (through simulation) is the distinguishing feature of SOSS in its intended application to the study of reliable systems. The object of such application is to obtain experimental results regarding fault tolerance, diagnosability, and reconfigurability that can lend insight to both the theory and design of reliable systems. A basic version of SOSS was completed just prior to the termination of the project. A detailed description of the system, with instructions as to its use, is documented in the technical report "A Structurally Oriented Simulation System" [23].

To summarize the development, SOSS was designed to run on-line on the Michigan Terminal System (MTS), i.e., in a conversational interactive mode, via a terminal. The command language was designed to enable the user to employ a simple, yet powerful set of commands in order to specify the structure of a system, alter

structure (insert faults), and simulate the behavior of the original or altered system. Since structure is specified as a network of sequential machines, the user may describe the system at various levels of structural refinement. He may choose to describe the detailed structure of a combinational or sequential switching network, he may describe a system as a composition of several subsystems, or he may describe only the state-transition and output functions of a system by regarding it as a one-component network. As for fault insertion, any permanent fault, beginning with simple "stuck at" faults through functional changes in combinational and machine components may be simulated via alterations in the original structure.

A general description of the systems that can be simulated by SOSS is given in Figure 5.1. The component machines $M_1, M_2, \ldots, M_\ell$ are state machines (i.e., sequential machines having an output function equal to the identity function). The combinational network is a finite acyclic network, each node of which realizes a (general) combinational function of n variables ($n \geq 1$), that is, a function from an n-fold cartesian product of finite sets into a finite set. Such a function can be a simple one to two variable switching functions, or a complicated function having as many as 255 variables each of which can assume up to 255 values.

In order to save storage, only the "value column" of each function table is stored. The order of the table is the natural order defined

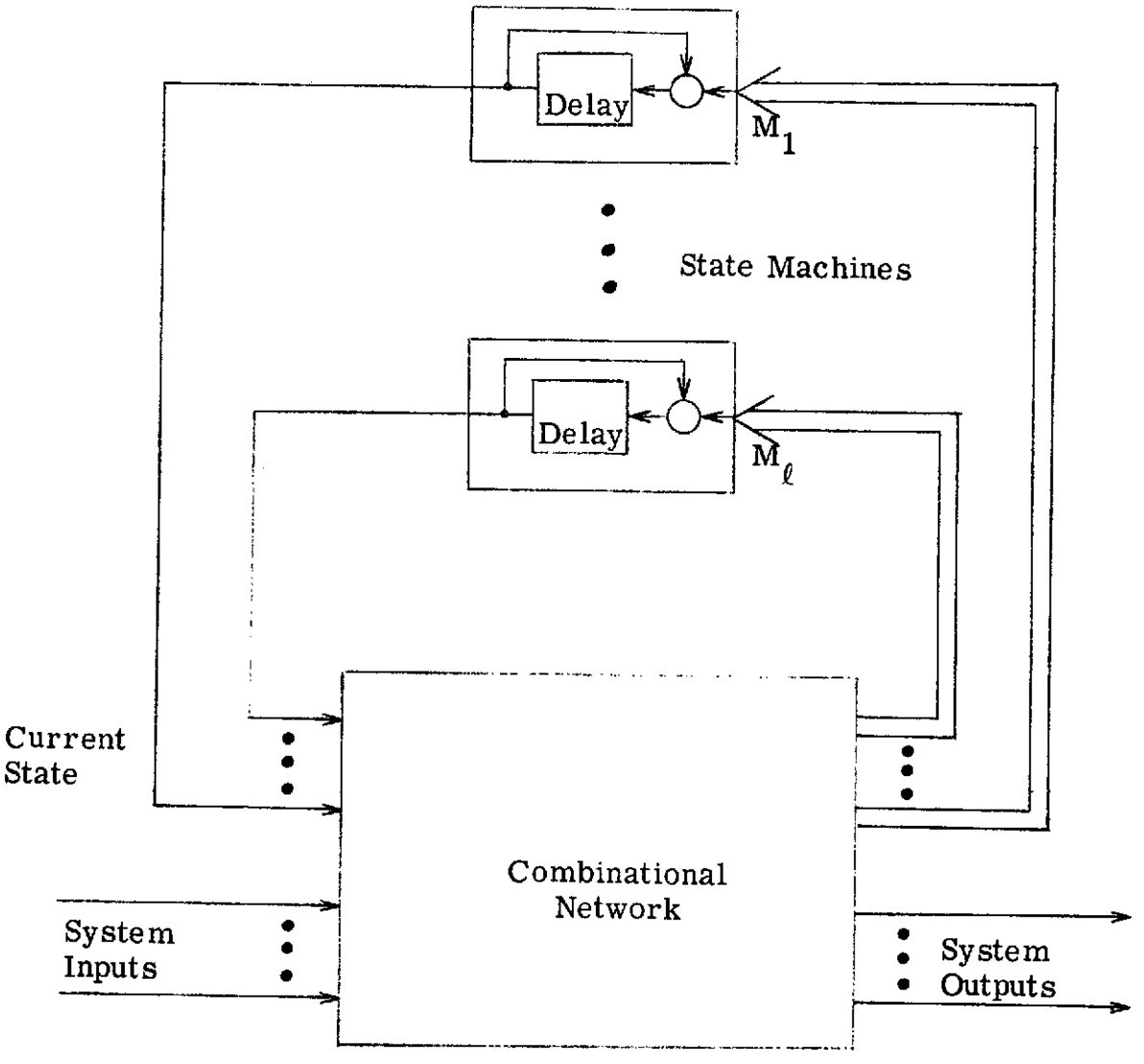

Delay
$M_1$
State Machines
Delay
$M_\ell$
Current
State
Combinational
Network
System
Inputs
System
Outputs

Figure 5.1

by the order in which the input connections to the node are specified. The inputs to the combinational network are the system inputs and the "current state" of the system provided by the state machines. The system outputs can be connected to any point in the simulated system, including the state machines, in order to monitor its behavior. The inputs and outputs may have as many as 255 symbols in their alphabet set.

In using SOSS, there are three basic modes of operation. These are CREATE, SIMULATE and ALTER. A fourth mode is the COMMAND mode which enables the user to transfer from one mode into another. Entering data to SOSS is done in free format statements . In these statements the letters A through O, R through V, and Y and Z are assigned to combinational network components. The letters P and Q are reserved for component machines and the letters W and X are reserved for system inputs. Each letter is followed by a number in the range 0-255. The number of possible system inputs is 512. This is also the number of possible machine nodes. The number of possible nodes in the combinational network is 5632. The following describes the basic modes of operation.

a) CREATE mode.

In this mode the user creates the system to be simulated. SOSS is initialized upon entry to CREATE and is ready to create a new system. There are three types of information that the user need supply to

SOSS. The first is the alphabet size (AS) of the different components, inputs, and component machines. He then provides the functions of the combinational network components and the transition functions of the state machine. (Recall that this is done by giving an ordered list of the "value" column of the function tables.) The user must also supply the interconnection list for the whole system. Specifying the AS actually creates the node and must be done prior to assigning a function to a node or specifying its interconnections. SOSS assembles the data provided into a complete system. While data is entered, SOSS monitors the created system and issues warnings if specification errors are detected. Such errors could occur in specifying the values of a function, improper connections, repetitions, etc.

The system can be displayed by a display feature whenever the user wishes to inspect it while in CREATE, SIMULATE, or ALTER.

b) SIMULATE mode.

This is the mode in which the system, created by the user, is simulated. When a SIMULATE command is issued, SOSS first enters into a test phase. In

this phase it checks the created system for unspecified connections or functions and warnings are issued if any such failure is found. Upon completion of the test phase, SOSS enters into the SIMULATE mode and proceeds to simulate the system. Simulation is done sequentially. First, the "current state" of the system is initialized either to an initial state given by the user or to the "next state" that resulted from the last simulation. SOSS simulates the system one clock period at a time. The user may specify an input string of any length (for each input variable) or a single symbol at a time. If, during simulation, errors in referencing the function tables are detected, simulation stops.

The output information of a simulation can be obtained by employing the display provision. All information concerning a simulation can be printed out, i.e., current state, inputs, and values of the nodes of the combinational network. Printout can be done for each clock period or following a given length of the input sequence.

c) ALTER mode.

The ability to change a given system and insert faults is one of the most important features of SOSS. This is done in the ALTER mode. This mode is actually a subset of the CREATE mode in that an identical syntax is used. The difference is that the system is not initialized, but changes are made in the original system. To alter an existing system the user merely restates the required information which is to be changed.

The user may store his original system in an MTS file and make alterations on an identical copy. Both can then be run, separately, with the same input strings. In particular, in the intended application where alterations are interpreted as faults, the behavior of the faulty (altered) system can thus be compared with that of the original (fault-free) system. However, due to SOSS's recent completion, not enough experience has been gained as yet to evaluate its utilization.

## 6. REFERENCES

[1] J. von Neumann, "Probabilistic logics," California Institute of Technology, 1952. Also published in Automata Studies (Edited by C. E. Shannon and J. McCarthy), Princeton Univ. Press, Princeton, N.J., 1956.

[2] E. F. Moore, and C. E. Shannon, "Reliable circuits using less reliable relays," Journal of the Franklin Institute, vol. 261, nos. 9 and 10, Sept., Oct. 1956, pp. 191-208 and 281-297.

[3] R. A. Short, "The attainment of reliable digital systems through the use of redundancy - a survey," Computer Group News, March 1968, pp. 2-17.

[4] R. H. Wilcox, and W. C. Mann, Redundancy Techniques for Computing Systems, Spartan Books, Washington, D.C., 1962 .

[5] S. Winograd and J. D. Cowan, Reliable Computation in the Presence of Noise, The M.I.T. Press, Cambridge, Mass., 1963.

[6] W. H. Pierce, Failure Tolerant Computer Design, Academic Press, New York, New York, 1965 .

[7] F. F. Sellers, M. Hsiao, and L. W. Bearnson, Error Detection Logic for Digital Computers, McGraw-Hill, Inc., 1968.

[8] H. Y. Chang, E. G. Manning and G. Metze, Fault Diagnosis of Digital Systems, Wiley, New York, 1970.

[9] A. D. Friedman and P. R. Menon, Fault Detection in Digital Circuits, Prentice-Hall, Englewood Cliffs, N.J., 1971.

[10] Digest 1971 International Symposium on Fault Tolerant Computing, IEEE (Computer Society Publication), March 1971.

[11] F. G. Gray and J. F. Meyer, "Locatability of faults in abstract networks," Digest 1971 Int'l Symp. Fault Tolerant Comp. (op. cit.), pp. 30-33.

[12] J. F. Meyer and K. Yeh, "Diagnosable machine realizations of sequential behavior," Digest 1971 Int'l Symp. Fault Tolerant Comp. (op. cit.), pp. 22-25.

[13] J. F. Meyer, et al, "Theory and design of reliable spacecraft data systems," Final Report, Contract 952492, Jet Propulsion Laboratory, Pasadena, California, Sept. 1970.

[14] F. G. Gray and J. F. Meyer, "Locatability of faults in combinational networks," IEEE Trans. on Computers, vol. C-20, Nov. 1971, pp. 1407-1412.

[15] J. F. Meyer, "Fault tolerant sequential machines," IEEE Trans. on Computers, vol. C-20, Oct. 1971, pp. 1167-1177.

[16] J. F. Meyer, "Semiannual status report no. 1; Theory of reliable systems," NASA Grant NGR23-005-463, Feb. 1972.

[17] J. F. Meyer, "Semiannual status report no. 2; Theory of reliable systems," NASA Grant NGR23-005-463, June 1972.

[18] J. F. Meyer, "Semiannual status report no. 3; Theory of reliable systems," NASA Grant NGR23-005-463, Jan. 1973.

[19] L. Hsieh and J. F. Meyer, "General compound distinguishing sequences," Systems Engineering Laboratory Technical Report No. 62, The University of Michigan, Ann Arbor, 1972.

[20] J. F. Meyer and K. Yeh, "Representation and minimization of diagnostic test sets," Systems Engineering Laboratory Technical Report No. 63, The University of Michigan, Ann Arbor, 1972.

[21] K. Yeh, "A theoretic study of fault detection problems in sequential systems," Systems Engineering Laboratory Technical Report No. 64, The University of Michigan, Ann Arbor, 1972.

[22] L. Hsieh, "Checking experiments for sequential machines," Systems Engineering Laboratory Technical Report No. 65, The University of Michigan, Ann Arbor, 1972.

[23] Z. Aran, "A structurally oriented simulation system," Systems Engineering Laboratory Technical Report No. 70, The University of Michigan, Ann Arbor, 1973.

[24] L. Hsieh, "Augmentation of machine structure to improve its diagnosability," Systems Engineering Laboratory Technical Report No. 71, The University of Michigan, Ann Arbor, 1973.

[25] R. J. Sundstrom, "On-line diagnosis of sequential systems," Systems Engineering Laboratory Technical Report No. 72, The University of Michigan, Ann Arbor, 1973.

[26] J. F. Meyer and B. P. Zeigler, "On the limits of linearity," Theory of Machines and Computations (Edited by Z. Kohavi and Z. Paz), Academic Press, New York, 1971, pp. 229-241.

[27] J. F. Meyer, "Sequential behavior and its inherent tolerance to memory faults," Proceedings of the 5th Hawaii International Conference on System Sciences, Jan. 1972, pp. 476-478.

[28] J. F. Meyer, "A general model for the study of fault tolerance and diagnosis," Proceedings of the 6th Hawaii International Conference on System Sciences, January 1973, pp. 163-165.

[29] L. Hsieh and J. F. Meyer, "Diagnosis of unrestricted faults in sequential machines" (Abstract), Digest of the 1973 International Symposium on Fault Tolerant Computing, June 1973, p. 175.

[30] J. C. King, "Proving programs to be correct," IEEE Trans. on Computers, vol. C-20, Nov. 1971, pp. 1331-1336.

[31] E. F. Moore, "Gedanken-experiments on sequential machines," in Automata Studies (Annuals of Mathematical Studies), Princeton, N.J., Princeton Univ. Press, 1956, 129-153.

[32] F. C. Hennie, "Fault detecting experiments for sequential circuits," in Proc. 5th Ann. Symp. Switching Theory and Logical Design, Nov. 1964, 95-110.

[33] I. Kohavi and Z. Kohavi, "Variable-length distinguishing sequences and their application to the design of fault-detection experiments," IEEE Trans. Comp. (Short Notes), Vol. C-17, August 1968, 792-795.

[34] E. P. Hsieh, "Checking experiments for sequential machines," IEEE Trans. Comp., Vol. EC-20, Oct. 1971, 1152-1166.