012146-3-T

(NASA-CR-142602) ON-LINE DIAGNOSIS OF SEQUENTIAL SYSTEMS, 3 Final Technical Report, 1 Jan. - 31 Dec. 1974 (Michigan Univ.) 186 p HC $7.00 CSCL 09B

N75-21077

Unclas
G3/66 18071

On-Line Diagnosis of Sequential Systems - III

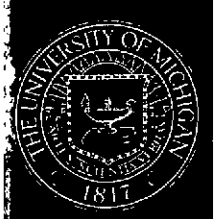
Final Technical Report covering the period
from January 1, 1974 through December 31, 1974

R. J. SUNDSTROM

under the direction of
Professor J. F. MEYER

January 1975

Prepared under
NASA Grant NGR 23-005-622



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. 84

ON-LINE DIAGNOSIS OF SEQUENTIAL SYSTEMS - III

by

Robert J. Sundstrom

Under the direction of
Professor John F. Meyer

Final Technical Report Covering the Period from
January 1 1974 through December 31 1974

January 1975

Prepared under
NASA Grant
NGR 23-005-005

ABSTRACT

ON-LINE DIAGNOSIS OF SEQUENTIAL SYSTEMS

by

Robert Joseph Sundstrom

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, etc.) it is desirable to constantly monitor the performance of the system, as it is being used, to determine whether the actual system is within tolerance of the intended system. Informally, by "on-line diagnosis" we mean a monitoring process of this type.

This study begins with the introduction of a formal model which can serve as the basis for a theoretical investigation of on-line diagnosis. Within this model a fault of a system S is considered to be a transformation of S into another system S' at some time τ. The resulting faulty system is taken to be the system which looks like S up to time τ and like S' thereafter. Notions of fault tolerance and error are defined in terms of the resulting system being able to mimic some desired behavior as specified by a system $\tilde{S}$. A notion of on-line diagnosis is formulated which involves an external detector and a maximum time delay within which every error caused

by a fault in a prescribed set must be detected.

This study focuses on the diagnosis of two important sets of faults: the set of "unrestricted faults" and the set of "unrestricted component faults." The set of unrestricted faults of a system is defined to be simply the set of all possible faults of that system. It is shown that if a system is on-line diagnosable for the unrestricted set of faults then the detector is at least as complex, in terms of state set size, as the specification. Moreover, this is true even if an arbitrarily large delay is allowed in the diagnosis.

One means of diagnosing the set of unrestricted faults of a system is by duplication and comparison. For systems which have (delayed) inverses (i.e., systems which are information lossless) a possible alternative is the use of a loop check. Here, it is established that if an inverse system is information lossless then it can always be used for unrestricted fault diagnosis. Although the lossless condition is sufficient, it is shown further that there exist systems for which a lossy inverse can also be used for unrestricted fault diagnosis. Since not every system has an inverse, let alone one which can be used for unrestricted fault diagnosis, it is not always possible to apply this technique directly. However, it is shown that every system has a realization to which this scheme can be successfully applied.

The on-line diagnosis of systems which are structurally decomposed and represented as a network of smaller systems is also investigated. The fault set considered here is the set of unrestricted component faults; namely, the set of faults which only affect one component of the network. A characterization of networks which can be diagnosed using a combinational detector is obtained. It is further shown that any network can be made diagnosable in the above sense through the addition of one component. In addition, a lower bound is obtained on the complexity of any component, the addition of which is sufficient to make a particular network combinationally diagnosable.

TABLE OF CONTENTS

	Page
LIST OF SYMBOLS	iv
INDEX OF TERMS	vii
LIST OF ILLUSTRATIONS	x
I. INTRODUCTION	1
1.1 Outline of the Problem	1
1.2 Brief Survey of the Literature	8
1.3 Synopsis of the Report	10
II. A MODEL FOR THE STUDY OF ON-LINE DIAGNOSIS	14
2.1 Resettable Discrete-Time Systems	15
2.2 Resettable Systems with Faults	31
2.3 Fault Tolerance and Errors	44
2.4 On-line Diagnosis	57
III. GENERAL PROPERTIES OF DIAGNOSIS	62
IV. DIAGNOSIS OF UNRESTRICTED FAULTS	69
4.1 Unrestricted Faults	71
4.2 Diagnosis via Independent Computation and Comparison	73
4.3 Diagnosis with Zero Delay	78
4.4 Diagnosis with Nonzero Delay	84
V. DIAGNOSIS USING INVERSE MACHINES	90
5.1 Inverses of Machines	92
5.2 Diagnosis Using Lossless Inverses	97
5.3 Applicability of Inverses for Unrestricted Fault Diagnosis	105

	Page
VI. DIAGNOSIS OF NETWORKS OF RESETTABLE SYSTEMS	111
6.1 Networks of Resettable Systems	113
6.2 Unrestricted Component Faults	120
6.3 Characterization of Combinationally Diagnosable Networks	123
6.4 Construction of Combinationally Diagnosable Networks	133
VII. CONCLUSIONS AND OPEN PROBLEMS	145
APPENDIX	156
REFERENCES	168

LIST OF SYMBOLS
(In order of first appearance)

Symbol	Representation	Page
T	Time base	15
S	System	15
$(I, Q, Z, \delta, \lambda)$	Discrete-time system	15
I	Input alphabet	15
Q	State set	15
Z	Output alphabet	15
δ	Transition function	15
λ	Output function	16
I^*		16
Λ	Null sequence	16
I^+		16
β	Behavior	17
$(I, Q, Z, \delta, \lambda, R, \rho)$	Resettable system	18
R	Reset alphabet	18
ρ	Reset function	18
(I, Q, δ, R, ρ)	Resettable state system	19
M	Resettable machine	20
$\mathcal{S}(I, Z, R)$		20
$\mathfrak{M}(I, Z, R)$		20
(I, Z, λ)	Memoryless machine	20
$\hat{\beta}$	Extended behavior	25

Symbol	Representation	Page
P	Reachable part	25
$\equiv$	Equivalence	26
$(\sigma_1, \sigma_2, \sigma_3)$	Realization	26
e	Identity function	29
P_C	Coordinate projection	30
F	Fault set	32
$f, (S', \tau, \theta)$	Fault	33
S^f	Result of fault	33
(S, F)	System with faults	35
$[\,]$	Equivalence class	42
$\tilde{M}$	Specification	44
(r, x, y)	Error	51
$S_1 * S_2$	Cascade connection	57
$[u, v]$		58
D	Detector	59
k	Delay	59
$\mathscr{D}$	Set of detectors	61
U	Unrestricted faults	71
F_o	Permanent output faults	80
M^n	Delay machine	92
$\overline{M}$	Inverse of M	92
$N, (I, R, (S_1, \ldots, S_n), (K_1, \ldots, K_n), Z, \lambda)$	Network	113

Symbol	Representation	Page
K_i	System connection rule	113
S_N	System defined by N	116
M_N	Machine defined by N	117
(N', τ, θ)	Fault of network	120
U_C	Unrestricted component fault	120
C		123
C_i		123
π_C		123
$\#\|J\|$		123
$\tilde{C}$		124
$(\omega_1, \omega_2, \omega_3, \omega_4)$	Isomorphism	163
M_R	Reduction of M	164

INDEX OF TERMS

behavior
    of S for condition (r, t), p. 24
    of S for initial reset r, p. 24
    of S in state q, p. 17

cartesian product, p. 29
cascade connection, p. 57
cause, of error, p. 51
component system, p. 113
connection rules, system, p. 113
coordinate projection, p. 30
cover, p. 123
    singleton, p. 123
cross product function, p. 30

defining a machine by a network, p. 116
diagnosis
    (D, k)-diagnosable, p. 59
    k-self-diagnosable, p. 61

equivalent
    faults, p. 41
    inputs, p. 167
    resettable machines, pp. 26, 156
    states, pp. 26, 156
error, p. 51
    minimal, p. 51
    occurrence of, p. 52

failure, p. 36
fault, p. 33
    improper, p. 32
    of network, p. 120
    permanent, p. 33
    permanent output, p. 80
    proper, p. 32
    result of, pp. 32, 33
    unrestricted, p. 71
    unrestricted component, p. 120
fault-detection signal, p. 60
fault-secure, p. 62

independent of M's input, p. 60
isomorphism, p. 163
    strong, p. 163

logic circuit level, p. 154

machine,
    autonomous, p. 167
    combinational, p. 20
    delay, p. 92
    information lossless, p. 94
    inverse, p. 92
    lossy, p. 94
    memoryless, p. 20
    reachable, pp. 25, 156
    reduced, pp. 26, 157
    resettable, p. 18
    state-assigned, p. 149
    transition distinct, p. 167

network
    of resettable machines, p. 117
    of resettable systems, p. 113
    state, p. 118

reachable, p. 25, 156
    ℓ-reachable, pp. 25, 156
    part, pp. 25, 156
realization
    class, p. 31
    combinational machine, p. 20
    fault-free, p. 32
    faulty, p. 32
    network, p. 117
    output-augmented, p. 105
    resettable machine, p. 27, 157
    sequential machine, p. 26
reduced form, p. 164
reduction, p. 164
redundant, p. 124
    totally, p. 124

representation scheme, p. 31
reset
    alphabet, p. 18
    function, p. 18

specification class, p. 31
synchronization, p. 67
synchronizing function, p. 68
system
    discrete-time, p. 15
    resettable, p. 18
    resettable state, p. 19

time base, p. 15
tolerance
    fault, p. 45
    relation, p. 44
transition distinct, p. 167
translation, of systems, p. 41

LIST OF ILLUSTRATIONS

		Page
Fig. 2.1.	Schematic diagram for $S = (I, Q, Z, \delta, \lambda, R, \rho)$	19
Fig. 2.2.	Circuit for M_1	21
Fig. 2.3.	Transition table for M_1	21
Fig. 2.4.	State graph for M_1	22
Fig. 2.5.	State graph for M_2	22
Fig. 2.6.	A discrete-time system	23
Fig. 2.7.	Resettable machine M_1'	23
Fig. 2.8.	M realizes $\tilde{M}$ under $(\sigma_1, \sigma_2, \sigma_3)$	27
Fig. 2.9.	Resettable machine $\tilde{M}_3$	28
Fig. 2.10.	Resettable machine M_3	28
Fig. 2.11.	A fault $f = (S', \tau, \theta)$ of S	33
Fig. 2.12.	Resettable machine M_1''	35
Fig. 2.13.	Triple modular redundancy with voting and disagreement detecting	47
Fig. 2.14.	Machine M_4	49
Fig. 2.15.	Circuit for M_4	49
Fig. 2.16.	Machine M_4'	50
Fig. 2.17.	Machine M_ℓ	53
Fig. 2.18.	The cascade connection of S_1 and S_2	58
Fig. 2.19.	Diagnosis of (M, F) using the detector D	59
Fig. 4.1.	Diagnosis via duplication in the detector	73

		Page
Fig. 4.2.	A generalization of duplication in the detector	74
Fig. 4.3.	The comparator used in the proof of Theorem 4.2	76
Fig. 4.4.	Machines $\tilde{M}_1$, M_1, and D_1 and σ_3: $Z \rightarrow \tilde{Z}$	82
Fig. 4.5.	Machines M_2 and D_2	88
Fig. 5.1.	Machines M_1 and $\overline{M}_1$	93
Fig. 5.2.	Machine M in series with an inverse $\overline{M}$ of M	94
Fig. 5.3.	On-line diagnosis using inverse machines	97
Fig. 5.4.	A detector which uses a 0-delayed inverse	97
Fig. 5.5.	Machines M_2 and $\overline{M}_2$	99
Fig. 5.6.	Machines M_3 and $\overline{M}_3$	100
Fig. 5.7.	Machine M_3^2	101
Fig. 5.8.	Machine $\overline{M}_3'$	104
Fig. 5.9.	A lossless machine with a lossy inverse	107
Fig. 5.10.	An output-augmented realization of $\overline{M}'$ of Fig. 5.9	107
Fig. 5.11.	Machine $\overline{M}_1'$	110
Fig. 6.1.	Network N_1	115
Fig. 6.2.	Diagram of network N_1	116
Fig. 6.3.	Machine M_{N_1}	118
Fig. 6.4.	Machine $\tilde{M}_1$	118
Fig. 6.5.	Network N_1''	130
Fig. 6.6.	Network N_2	138
Fig. 6.7.	Machine $\tilde{M}_2$	139

		Page
Fig. 6.8.	$\sigma_3 : P_2 \rightarrow \tilde{Q}_2$	139
Fig. 6.9.	Network N_3	142
Fig. 6.10.	Machine $\tilde{M}_3$	143
Fig. 7.1.	State graph of M	151
Fig. 7.2.	3-dimensional view of M	152
Fig. 7.3.	State graph of M'	153

CHAPTER I

Introduction

1.1 Outline of the Problem

For many applications, especially those in which a computer is controlling a real-time process (e.g., telephone switching, flight control of an aircraft or spacecraft, control of traffic in a transportation system, etc.), reliability is a major factor in the design of the system. The need for high reliability arises because of the serious consequences errors may have in terms of danger to human lives, loss of costly equipment, or disruption of business or manufacturing operations. For example, it is economically unsound to shut down a steel mill for even a short time in order to repair a comparatively inexpensive controlling computer. The seriousness of the consequences, of course, depends upon the application and must be weighed against the cost of improving the reliability.

A number of techniques exist for improving computer reliability. One of the more obvious is the use of more reliable components. While the use of reliable components is clearly very important, it has been recognized that this technique alone is not sufficient to meet the requirements for modern ultrareliable computing systems [35].

Another general technique which is useful in some applications is the use of masking redundancy such as Triple Modular Redundancy. The reader is referred to Short [35] for a general survey of masking techniques. One major drawback to masking redundancy is that if failed components are not replaced and the mission time is long, then the reliability of a system which uses masking redundancy can actually be less than that of the corresponding simplex system [25].

A third means of increasing system reliability and availability is through fault diagnosis and subsequent system reconfiguration or repair. For example, a computer designed to control telephone switching, the No. 1 Electronic Switching System (ESS) contains duplicates of each module and fault diagnosis is achieved primarily by dynamically comparing the outputs of both modules [11]. Once a fault is detected, the faulty module is identified and removed from service under program control. The faulty module is then repaired manually with diagnostic help from the fault-free computer. Another ultra-reliable computer, the Jet Propulsion Laboratory Self-Testing and Repairing (STAR) computer, also makes use of modularity and standby sparing [4].

One means of performing fault diagnosis is to continuously monitor the performance of the system, as it is being used, to determine whether its actual behavior is tolerably close to the intended behavior. It is this sort of monitoring which we mean by the term "on-line diag-

nosis." Others have used the term "error detection" to refer to this sort of monitoring ([22], [23]).

Implementation of on-line diagnosis may be external to the system, both internal and external, or completely internal. In the last extreme, on-line diagnosis is sometimes referred to as "self-diagnosis" or "self-checking" ([8], [9]).

The signals generated by a monitoring device can be used in many ways. For example, the IBM System/360 utilizes checking circuits to detect errors [6]. The signals generated by these circuits are used in some models to freeze the computer so that the instruction which was currently executing may be retried if possible, and to assist in the checkout and repair of the computer if automatic retry attempt fails. Ultra-reliable computers typically use the signals generated by the monitoring device to provide the computer system with the information it needs to automatically reconfigure itself so as to avoid using any faulty circuits. One other use for such signals is to simply inform the system user that the system is not operating properly and that there may be errors in his data.

In general, on-line diagnosis is used to signal that the system is operating properly or that it is in need of repair. In most computer systems this task is also performed in some part by "off-line diagnosis." By off-line diagnosis we are referring to the process of removing the system from its normal operation and applying a series of prearranged tests to determine whether any faults are present in

the system. There are major differences between on-line and off-line diagnosis and it is important to be aware of the capabilities and the limitations of each.

One basic difference is that on-line diagnosis is a continuous process whereas off-line diagnosis has a periodic nature. Transient faults are difficult to diagnose with off-line diagnosis because if a fault is transient in nature it may not be in the system when it is tested. On the other hand, since on-line diagnosis is a continuous monitoring process both permanent and transient faults can be diagnosed. It has been recognized by Ball and Hardie [5] and others that intermittents do occur frequently, and that finding an orderly means to diagnose them is an important unsolved problem. Thus the inability of off-line diagnosis to deal satisfactorily with transients is a severe limitation.

Another basic difference is that the delay between the occurrence of a fault and its subsequent detection is generally greater for off-line than on-line diagnosis. Recovery after a fault has been diagnosed may sometimes be achieved by reconfiguration and restarting. However, in a real-time application irrepeatable or nonreversable events may take place if an error occurs and is not immediately detected. In any application, if there is a delay between the occurrence of an error and the subsequent diagnosis of a fault, then contamination of data bases may occur thus making restarting difficult. For these

reasons, the inherent delay associated with off-line diagnosis can be a serious limitation.

One further difference between on-line and off-line diagnosis is that with off-line diagnosis the system must be removed from its normal operation to apply the tests. This also may not be acceptable in a real-time application.

The cost of either form of diagnosis depends on the nature of the system to be diagnosed, the technology to be used in building the system, and the degree of protection against faulty operation that is required. With on-line diagnosis the cost is almost totally in the design, construction, and maintenance of extra hardware. With off-line diagnosis the cost is the initial generation of the tests and in the subsequent storage and running of these tests.

In general, off-line diagnosis is useful for factory testing and for applications where immediate knowledge of any faulty behavior is not essential. Off-line diagnosis is also useful for locating the source of trouble once such trouble is indicated by on-line diagnosis. For example, as stated earlier Bell System's No. 1 ESS uses duplication and comparison as its primary error detection scheme. But once an error has been detected, off-line diagnosis is used to determine which processer exhibited the erroneous behavior and to locate the faulty module in that processer.

In the Design Techniques for Modular Architecture for Reliable Computing Systems (MARCS) study a more integrated use of on-line diagnosis is proposed whereby a number of checking circuits observe the performance of various parts of the computer [8]. With a scheme such as this, information about the location of a fault can be obtained from knowledge of which checking circuit indicated the trouble.

Both on-line and off-line diagnosis have been used to check the operation of electronic computers from the first vacuum tube machines until the present time. In particular, off-line diagnosis procedures were developed for the ENIAC computer, the BINAC system had duplicate processors, and the UNIVAC used a more economical on-line diagnosis scheme involving 35 checking circuits [12]. During the past decade, however, the development of theory and techniques for fault diagnosis in digital systems and circuits have focused mainly on problems of off-line diagnosis (see [9] and [14] for example).

An alternative means of performing diagnosis has been investigated by White [37]. His novel scheme is similar to on-line diagnosis in that it involves redundant processing of information and subsequent checking for consistency. However, with his scheme the redundancy is in time rather than in space. After every operation is performed, a related operation is initiated which uses the

same circuitry but with different signals. The results of these two operations are then checked for consistency.

This scheme is useful for checking machines which were not designed with the additional circuitry required for on-line diagnosis. However, this technique is likely to be very expensive, in terms of both operating speed and microprogram memory requirements. In an example implemented by White, a self-checking microprogram to emulate the PDP-8/I on the Meta 4 ran an estimated 3.9 times slower than a non-checking version of this microprogram and used 5 times as much microprogram memory.

One other approach to diagnosis is simply to have human users or observers of the system watch for obvious misbehavior. Since faults often give rise to behaviors which are clearly erroneous, many faults can be detected in this manner. The effectiveness of this method is highly dependent upon the individual system and program, and is exceedingly difficult to evaluate. It seems reasonable to assume, however, that this method is less effective than any of the methods previously discussed. Certainly, this method is unacceptable for many applications.

1.2 Brief Survey of the Literature

The work that has been done on on-line diagnosis has been mainly concerned with the development of specific diagnosis techniques. One early paper is Kautz's study [19] of fault detection techniques for combinational circuits. In this paper he investigated a number of techniques including the use of codes and the possibility of greater economy if immediate detection of errors was not necessary. Some of the more common on-line diagnosis techniques are discussed in a book by Sellers, Hsiao, and Bearnson [34]. Much of what is in this book and a large portion of the techniques that can be found elsewhere in the literature are concerned with special circuits such as adders and counters. For example, see the work of Avizienis [3], Rao [33], Dorr [10], and Wadia [36].

Relatively little work can be found on the theory of on-line diagnosis. As with the investigation of on-line diagnosis techniques, much of the theory of on-line diagnosis focuses on arithmetic units. In one of the earliest works of a theoretical nature, Peterson [30] showed that an adder can be checked using a completely independent circuit which adds the residue, modulo some base, of the operands. He went on to show that any independent check of this type was a residue class check. Further theoretical work concerning the diagnosis of arithmetic units using residue codes can be found in Massey [24] and Peterson [32].

An early theoretical result of a more general nature was published by Peterson and Rabin [3$_1$]. They showed that combinational circuits can differ greatly in their inherent diagnosability and that in some cases virtual duplication is necessary.

A later and very important paper is that of Carter and Schneider [7]. They propose a model for on-line diagnosis which involves a system and external checker. The input and ouput alphabets of the system are encoded and the checker detects faults by indicating the appearance of a non-code output. A system is self-checking if for every fault in some prescribed set, (i) the system produces a non-code output for at least one code space input, and (ii) the system never produces incorrect code space outputs for code space inputs. Thus, (i) insures that every fault can be detected during normal usage, and (ii) insures that if no fault has been detected then the output can be relied upon to be correct. The checkers that they consider are also self-checking. Using this model they prove that any system can be designed to be self-checking for the set of single faults.

Anderson [1] has named property (i) "self-testing" and property (ii) "fault-secure," and he has investigated these properties for combinational networks. In Chapter III it is shown that the notion of diagnosis considered in this study is a generalization of the fault-secure property.

1.3 Synopsis of the Report

This report describes a formal investigation of the theory and techniques applicable to the on-line diagnosis of sequential systems. The formal approach taken in this report leads to a fuller understanding of current on-line diagnosis practices and suggests generalizations of known techniques. It also provides a framework for evaluating the advantages and limitations of the various on-line diagnosis schemes.

With decreasing cost of logic and the increasing use of computers in real-time applications where erroneous operation can result in the loss of human life and/or large sums of money the use of on-line diagnosis can be expected to increase greatly in the near future. The importance of this area along with the relative lack of theoretical results is our motivation for initiating this study of on-line diagnosis.

Before entering into the actual synopsis it is appropriate to discuss the objectives of this investigation. Let $\tilde{S}$ be a system which serves as a specification of some desired behavior, let F be a set of faults, let $\mathscr{D}$ be a set of possible external detectors, and let k be a maximum time delay within which every error caused by a fault in F must be detected. The basic on-line diagnosis problem can now be stated as follows:

> Given $\tilde{S}$, F, $\mathscr{D}$, and k find an (economical) realization S of $\tilde{S}$ and a detector $D \in \mathscr{D}$ such that D can observe S and signal within k time steps any error caused by a fault in F.

Towards the end of solving this basic problem the following questions have been formulated. These questions serve as more specific objections and their answers will help to solve the basic on-line diagnosis problem.

I. What are good on-line diagnosis techniques? That is, what good means are available for finding appropriate realizations and detectors? When is each technique applicable?

II. Given $\tilde{S}$, S, F, $\mathscr{D}$, and k, does a suitable detector exist in $\mathscr{D}$? That is, when is a given realization diagnosable? If such a detector exists how can it be constructed? A solution to this problem would certainly help to solve the previous one.

III. What time-space tradeoffs are possible between the added complexity needed for diagnosis and the maximum allowable delay? We expect that there will be situations where if the detector is given additional time in which to indicate an error then diagnosis may be simplified.

IV. What relationships exist between faults and errors? Given S and F, what errors are possible? Given $\tilde{S}$ and F, how can one find a realization S of $\tilde{S}$ such that the system with faults (S, F) gives rise only to errors of a given type? These are important questions because given a diagnosis technique or a particular type of detector, it will often be easy to determine just what types of errors are detectable. The faults that are diagnosable will then have to be inferred from this information. Conversely, we will want to find

realizations such that the faults we are concerned with will cause errors that we can detect.

V. What properties of system structure and system behavior are conducive to on-line diagnosability? Structural and behavioral properties are important for it is expected that they will relate directly to diagnosis techniques. Behavioral properties could be used to measure the inherent diagnosability of a given behavior in terms of the minimum added complexity which would be required to obtain a given level of on-line diagnosis.

The first problem considered in this investigation was the formulation of a formal model which could serve as a basis for a theoretical study of on-line diagnosis. This model is developed fully in Chapter II. First an appropriate class of system models is formulated which can represent both the behavior and the structure of fault-free and faulty systems. Then notions of realization, fault, fault-tolerance and diagnosability are formalized which have meaningful interpretations in the context of on-line diagnosis. The following chapters are all concerned with the properties of the notion of diagnosis which is introduced in this chapter.

Chapter III contains some elementary properties of diagnosis which are independent of the particular class of faults under consideration. The result of this chapter help to give a basic understanding of on-line diagnosis and are used in the later chapters.

Chapter IV is concerned with the diagnosis of the set of unrestricted faults. This set of faults is simply the set of all possible faults of the system under consideration. The major result of this chapter gives a lower bound on the complexity of any detector which can be used for unrestricted fault diagnosis of a given system.

In Chapter V, the use of inverse systems for the diagnosis of unrestricted faults is considered. Inverse systems are formally introduced, and a partial characterization of those inverse systems which can be used for unrestricted fault diagnosis is obtained. Since not every system has an inverse system, let alone one which is suitable for unrestricted fault diagnosis, it is not always possible to apply this technique directly. However, it is shown that every system has a realization upon which this technique can be successfully applied.

In Chapter VI, the diagnosis of systems which are structurally decomposed and are represented as a network of smaller systems is studied. The fault set considered here is the set of faults which only affect one component system in the network. A characterization of those networks which can be diagnosed using a purely combinational detector is achieved. A technique is given which can be used to realize any network by a network which is diagnosable in the above sense. Limits are found on the amount of redundancy involved in any such technique.

CHAPTER II

A Model for the Study of On-Line Diagnosis

In this chapter a formal model is developed which is suitable for a theoretical study of on-line diagnosis of sequential systems. The development begins with the introduction of a class of system models, called "resettable discrete-time systems," which will serve as the basis of this study. Within this model a fault of a system S is considered to be a transformation of S into another system S' at some time τ. The resulting faulty system is taken to be the system which looks like S up to time τ and like S' thereafter. Next the companion notions of fault tolerance and error are defined in terms of the resulting system being able to mimic some desired behavior. Finally, a notion of on-line diagnosis is introduced. This notion involves an external detector and a maximum time delay within which every error caused by a fault in some prescribed set must be detected.

2.1 Resettable Discrete-Time Systems

On-line diagnosis is inherently a more complex process than off-line diagnosis because of two complicating factors: i) it has to deal with input over which it has no control and ii) faults can occur as the system is being diagnosed. We would like to build a theory of on-line diagnosis using conventional models of time-invariant (stationary, fixed) systems (e.g., sequential machines, sequential networks, etc.). However, due to the second factor mentioned above these conventional models can no longer be used to represent the dynamics of the system as it is being diagnosed. A system which is designed and built to behave in a time-invariant manner becomes a time-varying system as faults occur while it is in use. Therefore, a more general representation based on time-varying systems is required. Based on this fundamental observation we have developed what we believe to be an appropriate model for the study of on-line diagnosis.

Definition 2.1: Relative to the time-base $T = \{\ldots, -1, 0, 1\ldots\}$, a discrete-time system (with finite input and output alphabets) is a system

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

where I is a finite nonempty set, the input alphabet

Q is a nonempty set, the state set

Z is a finite nonempty set, the output alphabet

$\delta: Q \times I \times T \rightarrow Q$, the transition function

λ: $Q \times I \times T \rightarrow Z$, the output function.

The interpretation of a 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 $\delta(q, a, t)$. In the special case where the functions δ and λ are independent of time (i.e., are time-invariant), the definition reduces to that of a (Mealy) sequential machine. In the discussion that follows it is assumed that S is finite-state (i.e., $|Q| < \infty$).

To describe the behavior of a system, we first extend the transition and output functions to input sequences in the following natural way. If I^* is the set of all finite-length sequences over I (including the null sequence Λ) then:

$$\bar{\delta}: Q \times I^* \times T \rightarrow Q$$

where, for all $q \in Q$, $a \in I$, $t \in T$:

$$\bar{\delta}(q, \Lambda, t) = q$$

$$\bar{\delta}(q, a, t) = \delta(q, a, t)$$

$$\bar{\delta}(q, a_1 a_2 \ldots a_n, t) = \delta(\bar{\delta}(q, a_1 a_2 \ldots a_{n-1}, t), a_n, t + n - 1) .$$

Similarly, if $I^+ = I^* - \{\Lambda\}$:

$$\bar{\lambda}: Q \times I^+ \times T \rightarrow Z$$

where for all $q \in Q$, $a \in I$, $t \in T$:

$$\overline{\lambda}(q, a, t) = \lambda(q, a, t)$$

$$\overline{\lambda}(q, a_1 a_2 \ldots a_n, t) = \lambda(\overline{\delta}(q, a_1 a_2 \ldots a_{n-1}, t), a_n, t + n - 1) .$$

Henceforth $\overline{\delta}$ and $\overline{\lambda}$ will be denoted simply as δ and λ.

Relative to these extended functions, the behavior of S in state q is the function

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

where

$$\beta_q(x, t) = \lambda(q, x, t) .$$

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 $t + |x| - 1$ ($|x|$ = length (x)) .

Many investigations of on-line diagnosis and fault tolerance have studied redundancy schemes such as duplication and triplication. Typically they have not dealt with the problem of starting each copy of a machine in the same state. In this study we will be examining these schemes and others for which the same problem arises. Since many existing systems have reset capabilities, and since this feature solves the above synchronizing problem we will use a special type of system for which the reset capabilities are explicitly specified. This explicit

specification of the reset capability is essential since it is an important part of the total system and it may be subject to failure.

Definition 2.2: A resettable discrete-time system (resettable system) is a system

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

where $(I, Q, Z, \delta, \lambda)$ is a discrete-time system

R is a finite nonempty set, the reset alphabet

ρ: $R \times T \rightarrow Q$, the reset function.

A resettable system 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. This method of specifying reset capability is a matter of convenience. This feature could just as well have been incorporated as a restriction on the transition function relative to a distinguished subset of input symbols called the reset alphabet. Thus a resettable discrete-time system can indeed be regarded as a special type of discrete-time system. If δ, λ, and ρ are all independent of time the definition reduces to that of a resettable sequential machine. Thus a resettable machine can be viewed as a resettable system which is invariant under time-translations.

Given a resettable system we can view it as a system organized as in Fig. 2.1.

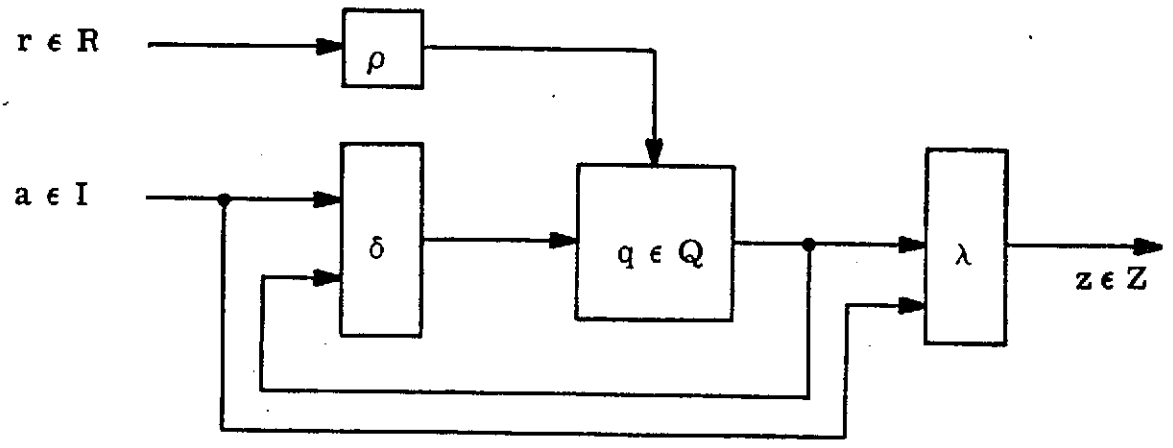


Fig. 2.1. Schematic Diagram for $S = (I, Q, Z, \delta, \lambda, R, \rho)$

In many discussions the output function of a system will not be of direct concern; the focus of attention will be upon the state transitions. This motivates the following definition.

Definition 2.3: A resettable discrete-time system $S = (I, Q, Z, \delta, \lambda, R, \rho)$ is a resettable state system if $Z = Q$ and $\lambda(q, a, t) = q$ for all $q \in Q$, $a \in I$, and $t \in T$.

Since the output alphabet and output function of a resettable state system need not be explicitly specified, a resettable state system $S = (I, Q, Z, \delta, \lambda, R, \rho)$ will be denoted by the 5-tuple (I, Q, δ, R, ρ).

This formulation of resettable state systems as special types of resettable systems allows us to directly apply the following theory of on-line diagnosis to state machines.

Notation: Resettable systems will be denoted by S, S', S_1, S_2, etc., and resettable machines will be denoted by M, M', M_1, M_2, etc. Unless otherwise specified, M will denote the resettable machine $(I, Q, Z, \delta, \lambda, R, \rho)$; M' will denote the resettable machine $(I', Q', Z', \delta', \lambda', R', \rho')$; and so forth. $\mathcal{S}(I, Z, R)$ will denote the set of systems with input alphabet I, output alphabet Z, and reset alphabet R. That is,

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

$\mathfrak{M}(I, Z, R)$ will denote the corresponding set of resettable machines.

Definition 2.4: A resettable sequential machine $M = (I, Q, Z, \delta, \lambda, R, \rho)$ is memoryless or combinational if $|Q| = 1$.

The triple (I, Z, λ) where $\lambda: I \longrightarrow Z$ will be used to denote any memoryless machine with input alphabet I, output alphabet Z, and output function λ. The memoryless machine $M = (I, Z, \lambda)$ is said to realize the function λ from I into Z.

We will represent sequential machines in the usual manner, i.e., via transition tables or state graphs. Resettable machines are represented by minor extensions of these two methods. The transition table of a resettable machine is identical to that of a machine with addition of one column on the right to accommodate the reset function. If $\rho(r) = q$ then r will appear in this additional column in the row corresponding to state q. Similarly, the state graph of a resettable machine is identical to that of a machine with the addition of one short

arrow for each r ϵ R. This arrow will be labeled r and will point to state $\rho(r)$.

Example 2.1: Let M_1 be the sequence generator with reset alphabet $\{0\}$ and input alphabet $\{1\}$ which has been implemented by the circuit in Fig. 2.2.

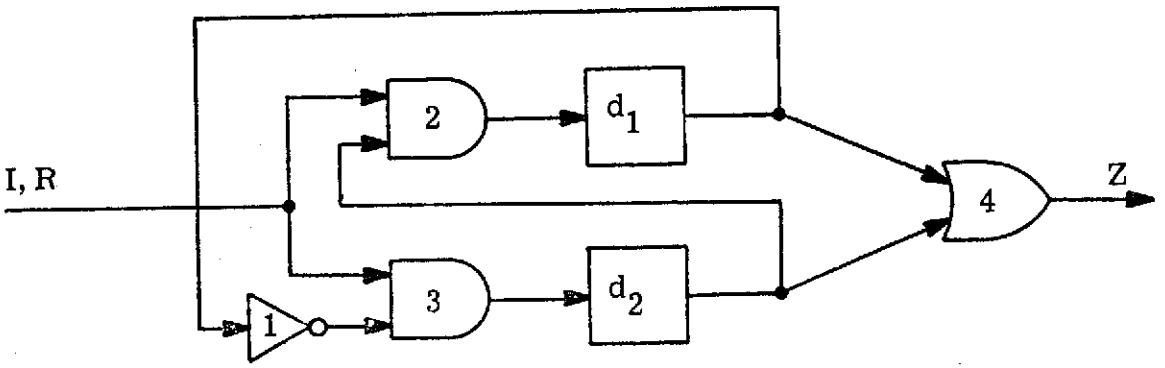


Fig. 2.2. Circuit for M_1

The transition table and the state graph for M_1 are shown in Figs. 2.3 and 2.4.

Q_1 \ I_1	1	R_1
00	01/0	0
01	11/1	
10	00/1	
11	10/1	

Fig. 2.3. Transition Table for M_1

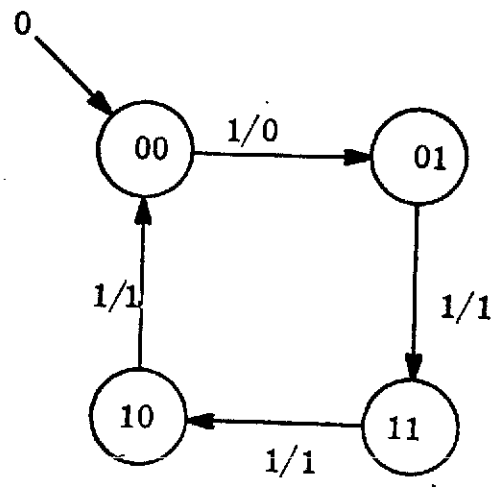


Fig. 2.4. State Graph for M_1

The circuit in Fig. 2.2 is also an implementation of a similar machine M_2 with input alphabet $\{0, 1\}$. The state graph for M_2 is shown in Fig. 2.5.

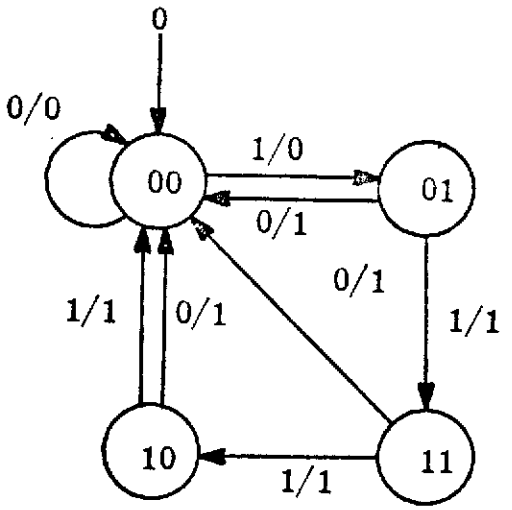


Fig. 2.5. State Graph for M_2

Thus, in M_2 the input symbol "0" can be interpreted as an input or as a reset. In M_2 the outputs for input 0 are explicitly specified whereas in M_1 they may be regarded as classical "don't cares."

We can view a particular discrete-time system as a system which looks like some machine M_i in one time interval, like M_{i+1} in another interval, and so on. This is also a good means of specifying a system.

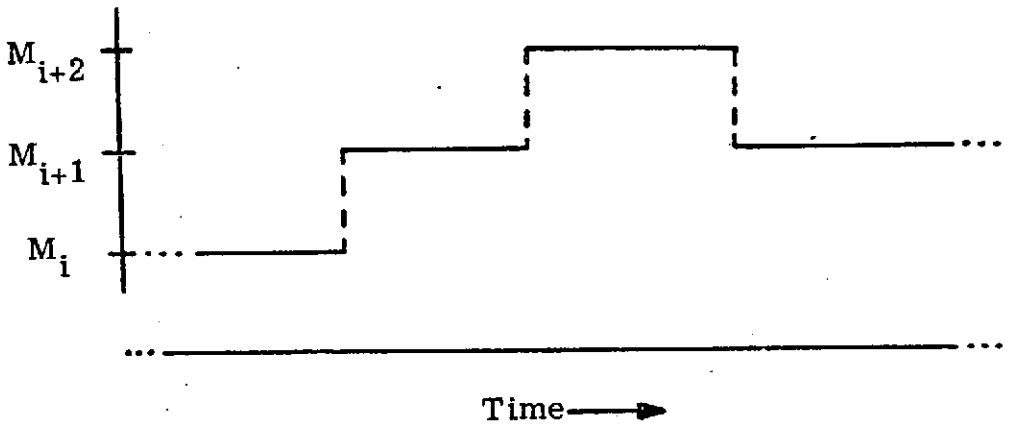


Fig. 2.6. A Discrete-Time System

Example 2.2: Suppose that M_1 was implemented as in Fig. 2.2 and that this circuit operated correctly up to time 100 when gate 2 became stuck-at-0. What actually existed was not a resettable machine but a (time-varying) resettable system S which looks like M_1 up to time 100 and like a different machine, say M_1' thereafter. The graph for M_1' is shown in Fig. 2.7.

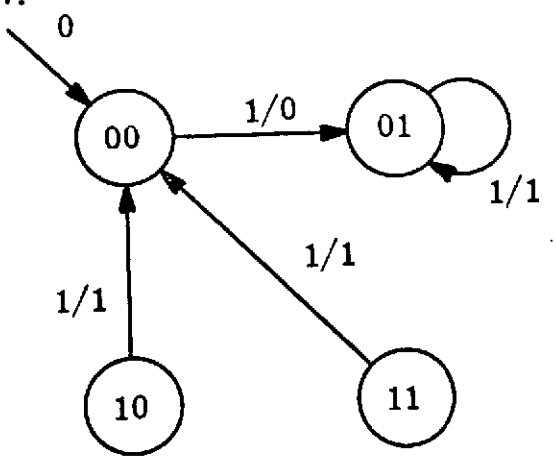


Fig. 2.7. Resettable Machine M_1'

We can represent S as follows:

$$S = \begin{cases} M_1 & \text{for } t < 100 \\ M_1' & \text{for } t \geq 100. \end{cases}$$

By this we mean that $I = I_1 = I_1'$ and likewise for Q, Z, and R, and that

$$\delta(q, a, t) = \begin{cases} \delta_1(q, a) & \text{for } t < 100 \\ \delta_1'(q, a) & \text{for } t \geq 100 \end{cases}$$

and similarly for λ and ρ.

For resettable systems we take the definitions of $\overline{\delta}$, $\overline{\lambda}$, and β_q to be the same as those for systems. It is also convenient in the case of resettable systems 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

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

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 β_r.

It is useful to extend the behavior function $\beta_{r,t}$ in a natural manner to represent the sequence to sequence behavior of S. For $r \in R$ and $t \in T$

$$\hat{\beta}_{r,t}: I^+ \rightarrow Z^+$$

where for all $a_1 \ldots a_n \in I^+$

$$\hat{\beta}_{r,t}(a_1 \ldots a_n) = \beta_{r,t}(a_1) \ldots \beta_{r,t}(a_1 a_2 \ldots a_n) .$$

We will now introduce a few properties of resettable machines which will be important to our developing model of on-line diagnosis. A more complete treatment of the properties of resettable machines can be found in the appendix.

These properties are defined for resettable machines rather than for resettable systems because they will be applied to "fault-free" systems, which in this study are always time-invariant.

We begin with some concepts of "reachability." Let M be a resettable machine. The reachable part of M, denoted by P, is the set

$$P = \{\delta(\rho(r), x) \mid r \in R, \; x \in I^*\} .$$

M is reachable if P = Q. M is ℓ-reachable if

$$P = \{\delta(\rho(r), x) \mid r \in R, \; x \in I^* \text{ and } |x| \leq \ell\} .$$

Note that a machine can be ℓ-reachable but not reachable.

An elementary result of graph theory states that in a directed graph with n points, if a point v can be reached from a point u then there is a path of length n - 1 or less from u to v. An immediate consequence of this is that any machine M is $(|P| - 1)$-reachable.

Let M, M' ϵ $\mathfrak{M}(I, Z, R)$. M is <u>equivalent</u> to M' (written $M \equiv M'$) if $\beta_r = \beta'_r$ for all $r \epsilon R$. Two states $q \epsilon Q$ and $q' \epsilon Q'$ are <u>equivalent</u> ($q \equiv q'$) if $\beta_q = \beta'_{q'}$. It is easily verified that these are both equivalence relations, the first on $\mathfrak{M}(I, Z, R)$ and the second on the states of machines in $\mathfrak{M}(I, Z, R)$.

A resettable machine M is <u>reduced</u> if for all q, $q' \epsilon P$, $q \equiv q'$ implies $q = q'$. A basic result of sequential machine theory states that for every machine there is an equivalent reduced machine and that this machine is unique up to isomorphism. The corresponding result for resettable machines is given in the appendix.

A concept which is central to sequential machine theory is that of a "realization." The corresponding resettable machine concept will be very important to our theory of on-line diagnosis. We will introduce it by first stating Meyer and Zeigler's definition of realization for sequential machines [27].

<u>Definition 2.5</u>: If M and $\tilde{M}$ are sequential machines then M <u>realizes</u> $\tilde{M}$ if there is a triple of functions $(\sigma_1, \sigma_2, \sigma_3)$ where $\sigma_1 : (\tilde{I})^+ \rightarrow I^+$ is a semigroup homomorphism such that $\sigma_1(\tilde{I}) \subseteq I$, $\sigma_2 : \tilde{Q} \rightarrow Q$, $\sigma_3 : Z' \rightarrow \tilde{Z}$ where $Z' \subseteq Z$, such that for all $\tilde{q} \epsilon \tilde{Q}$

$$\tilde{\beta}_{\tilde{q}} = \sigma_3 \circ \beta_{\sigma_2(\tilde{q})} \circ \sigma_1$$

It has been shown by Leake [23] that this strictly behavioral definition of realization is equivalent to the structurally oriented definition of Hartmanis and Stearns [16].

If M and $\tilde{M}$ are resettable machines then our definition of realization is somewhat different. Inherent in this definition is our presupposition that a resettable system will be reset before every use.

Definition 2.6: If M and $\tilde{M}$ are two resettable machines then M realizes $\tilde{M}$ if there is a triple of functions $(\sigma_1, \sigma_2, \sigma_3)$ where $\sigma_1: (\tilde{I})^+ \rightarrow I^+$ is a semigroup homomorphism such that $\sigma_1(\tilde{I}) \subseteq I$, $\sigma_2: \tilde{R} \rightarrow R$, $\sigma_3: Z' \subseteq Z$, such that for all $\tilde{r} \in \tilde{R}$,

$$\tilde{\beta}_{\tilde{r}} = \sigma_3 \circ \beta_{\sigma_2(\tilde{r})} \circ \sigma_1$$

This concept can be viewed pictorally as in Fig. 2.8.

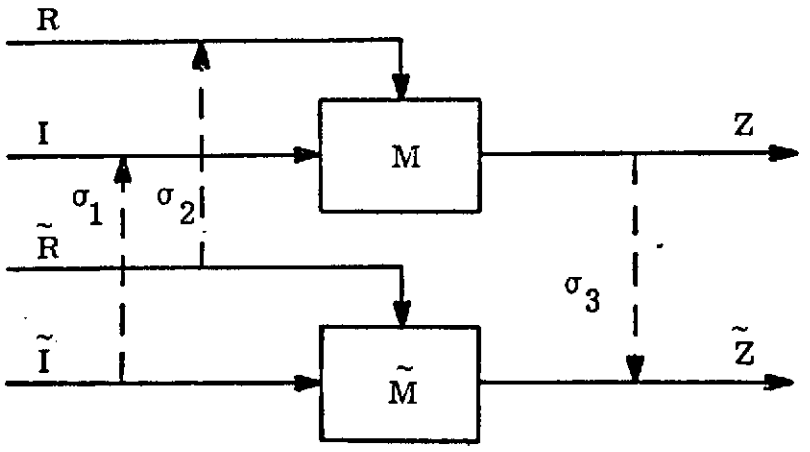


Fig. 2.8. M Realizes $\tilde{M}$ under $(\sigma_1, \sigma_2, \sigma_3)$

Example 2.3: Let $\tilde{M}_3$ and M_3 be the resettable machines shown in Fig. 2.9 and Fig. 2.10.

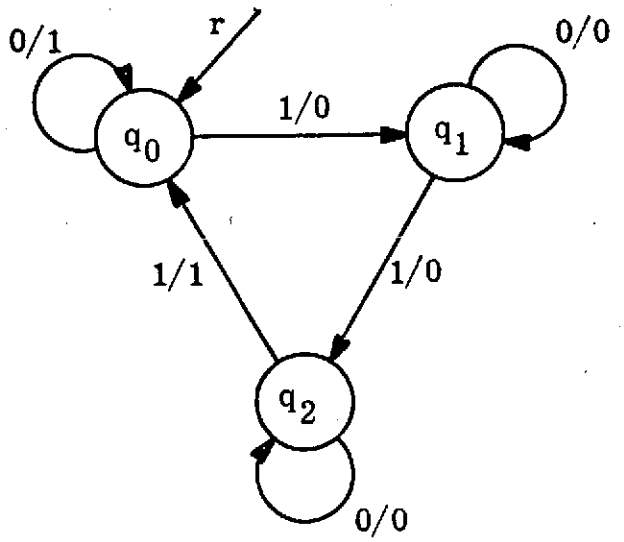


Fig. 2.9. Resettable Machine $\tilde{M}_3$

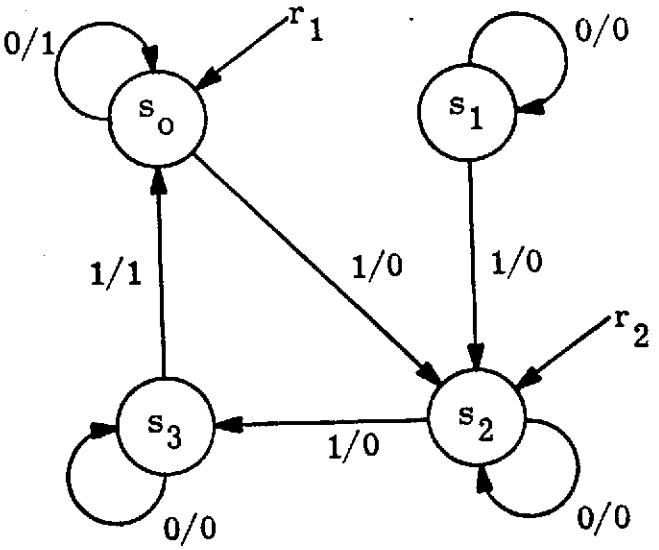


Fig. 2.10. Resettable Machine M_3

Then M_3 realizes $\tilde{M}_3$ under the triple $(\sigma_1, \sigma_2, \sigma_3)$ where $\sigma_1: (\tilde{I}_3)^+ \rightarrow I_3^+$ is the identity, $\sigma_2: \tilde{R}_3 \rightarrow R_3$ is defined by $\sigma_2(r) = r_1$, and $\sigma_3: Z_3 \rightarrow \tilde{Z}_3$ is the identity. To verify this claim we need only observe that $\tilde{\beta}^3_r(x) = \beta^3_{r_1}(x)$ for all $x \in (\tilde{I}_3)^+$.

Notice that the definition of realization for resettable machines is less restrictive than that for sequential machines in the sense that for resettable machines we only require the realizing system to mimic the behavior of the reset states of the realized machine; while in the sequential machine case the realizing system must mimic the behavior of every state of the realized system. On the other hand, the definition in the resettable case is more restrictive in the sense that for each reset state in the realized machine not only does there exist a state in the realizing machine which mimics its behavior, but we also know how to get to that state.

Before proceeding with our model of on-line diagnosis we must introduce a few notational conventions. The identity function on a set A will be denoted by e_A. When it is clearly understood which set is being mapped the subscript will be deleted.

If $A_1, \ldots, A_n$ is a sequence of n sets, its <u>cartesian product</u> is the set $A_1 \times \ldots \times A_n = \underset{i=1}{\overset{n}{\times}} A_i = \{(x_1, \ldots, x_n) \mid x_i \in A_i,\ i = 1, \ldots, n\}$. The cartesian product of an empty sequence of sets is taken to be any singleton set.

Given a cartesian product $A = \overset{n}{\underset{i=1}{\times}} A_i$, a <u>coordinate projection</u> of A is a function $P_i : A \rightarrow A_i$ defined by $P_i\ (x_1, \ldots, x_n) = x_i$.

If $f_1 : A \rightarrow B_1, \ldots, f_n : A \rightarrow B_n$ is a sequence of functions, the <u>cross-product function</u> $\overset{n}{\underset{i=1}{\times}} f_i : A \rightarrow \overset{n}{\underset{i=1}{\times}} B_i$ is defined by $\overset{n}{\underset{x=1}{\times}} f_i(a) = (f_1(a), \ldots, f_n(a))$. The cross-product function can be used to extend coordinate projections to project on to any subset of coordinates: if $C \subseteq \{1, \ldots, n\}$ then $P_C : A \rightarrow \overset{\times}{\underset{i \in C}{}} A_i$ is defined by $P_C = \overset{\times}{\underset{i \in C}{}} P_i$. In particular P_ϕ is a constant function with domain A.

2.2 Resettable Systems with Faults

Our model of a "resettable system with faults" is a specialization of Meyer's general model of a "system with faults" [29].

> 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 of two prescribed classes of (formal) systems, a "specification" class for the original system and a "realization" class for the resulting systems. More precisely, we say that a triple $(\mathcal{S}, \mathcal{R}, \rho)$ is a (system) representation scheme if
>
> i) $\mathcal{S}$ is a class of systems, the specification class,
> ii) $\mathcal{R}$ is a class of systems, the realization class,
> iii) $\rho: \mathcal{R} \rightarrow \mathcal{S}$ where, if $R \in \mathcal{R}$, R realizes $\rho(R)$.
>
> By a class of systems, in this context, we mean a class of formal systems, i.e., a set of formally specified structures of the same type, each having an associated behavior that is determined by the structure [29].

In this study we are concerned with the reliable use of a system. That is, we are concerned with degradations in structure which Meyer calls "life defects." This is contrasted with reliable design in which case we would be concerned with "birth defects." Thus, in our case, a specification is a realization and we choose a representation scheme $\mathcal{R} = (\mathcal{R}, \mathcal{R}, \rho)$ where ρ is the identity function on $\mathcal{R}$.

Assuming that a faulty resettable system has the same input, output, and reset alphabets as the fault-free system S, the following class of resettable systems will suffice as a realization class:

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

In summary, the representation scheme that we are choosing for our study of on-line diagnosis is the scheme $(\mathcal{R}, \mathcal{R}, \rho)$ where $\mathcal{R} = \mathcal{S}(I, Z, R)$ and ρ is the identity function on $\mathcal{R}$.

In such a scheme the seemingly difficult problem of describing faults and their results becomes relatively straightforward. Before we state our particular notion of a fault and its results we will repeat here Meyer's general notion of a "system with faults" [29].

> A system with faults in a representation scheme $(\mathcal{S}, \mathcal{R}, \rho)$ is a structure (S, F, ϕ) where
>
> i) $S \in \mathcal{S}$
> ii) F is a set, the faults of S
> iii) ϕ: $F \rightarrow \mathcal{R}$ 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 [29].

In applying this notion to our study we must first define what we mean by a fault of a resettable system. Given a resettable system $S \in \mathcal{S}(I, Z, R)$, a fault f of S can be regarded as a transformation of S into another system $S' \in \mathcal{S}(I, Z, R)$ at some time τ. Accordingly, the resulting faulty system looks like S up to time τ and like S' thereafter. Since S may be in operation at time τ we must also be concerned with the question of what happens to the state of S as this transformation takes place. We handle this with a function θ from the state set of S to that of S'. The interpretation of θ is that if S is in state q immediately before time τ then S' is in state $\theta(q)$ at time τ. More precisely,

Definition 2.7: 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 θ: $Q \longrightarrow Q'$.

A fault $f = (S', \tau, \theta)$ of S is a permanent fault if S' is time invariant.

We view the occurrence of a fault $f = (S', \tau, \theta)$ of a system S as shown in Fig. 2.11.

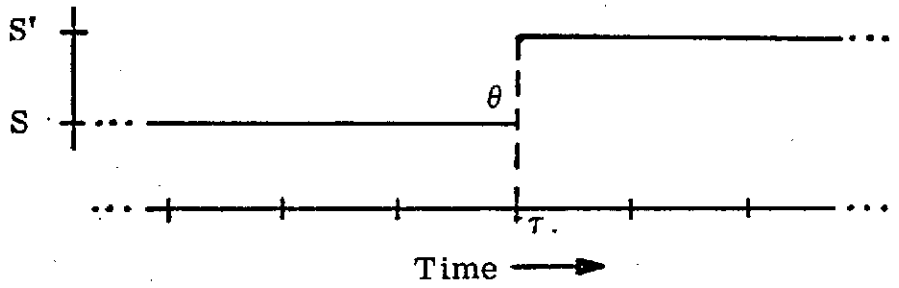


Fig. 2.11. A Fault $f = (S', \tau, \theta)$ of S

Given this formal representation of a fault of S, the resulting faulty system is defined as follows.

Definition 2.8: 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 τ. Note that, for any fault f of S, $S^f \in \mathcal{S}(I, Z, R)$.

Example 2.4: Recall that in Example 2.2 M_1 was transformed into M_1' at time 100. We would say now that $f = (M_1', 100, e)$ is a permanent fault of M_1 and that S is the result of f (i.e., $S = M_1^f$).

Example 2.5: Again consider M_1 as implemented by the circuit in Fig. 2.2 and let g be the fault which is caused by d_1 becoming stuck-at-1 at time 50. Then $g = (M_1'', 50, \theta)$ is a permanent fault of M_1 where M_1'' is the machine shown in Fig. 2.12 and $\theta: Q_1 \rightarrow Q_1''$ is defined by the table

q	$\theta(q)$
00	10
01	11
10	10
11	11

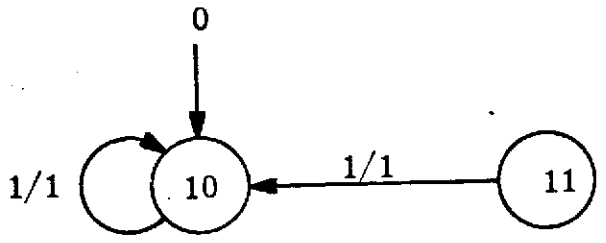


Fig. 2.12. Resettable Machine M_1''

M_1^g will behave as M_1 up to time 50 and thereafter it will produce a constant sequence of 1's.

To complete the model, a resettable 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., $f = (S, 0, e)$), and ϕ: $F \rightarrow \mathcal{S}(I, Z, R)$ where $\phi(f) = S^f$, for all $f \in F$. Given this definition, we can drop the explicit reference to ϕ in denoting a resettable system with faults, i.e., (S, F) will mean (S, F, ϕ) where ϕ is as defined above.

In the remainder of this study we will be dealing exclusively with resettable systems. Thus we will refer to resettable systems simply as systems and to resettable machines as machines.

A word is in order about our definition of faults. The interpretation here is one of effect, not cause, e. g., we don't talk of stuck-at-1 OR gates but rather of the system which is created due to some presumed physical cause. We will refer to these physical causes as component failures or simply as failures. A fault, by our definition, consists of precisely that information which is needed to define the system which results from the fault. This allows us to treat faults in the abstract; independent of specific network realizations of the system and without reference to the technology employed in this realization and the types of failures which are possible with this technology. We are assured, however, that for each fault we have enough information to assess the structural and behavioral effects of the fault; in particular as these effects relate to fault diagnosis and tolerance.

There are limits, however, to how much can be done with a purely effect oriented concept of faults. When a system is sufficiently structured to allow a reasonable notion of what may cause a fault we certainly will want to make use of this notion. When this is the case we may, through an abuse in language, refer to a specific failure at time τ as a fault. What we will mean is that we have stated a cause of fault and that there is a unique fault which is the result of this failure at time τ.

It is interesting to see what the scope of our definition of fault is in terms of the types of failures which will result in faults. Recall that a fault f of a system S is a triple, $f = (S', \tau, \theta)$, where $S' \in \mathcal{S}(I, Z, R)$. Thus S' is a (resettable) system with the same input, output, and reset alphabets as S. The previous sentence contains, implicitly, every restriction that we have put on faults. First of all, S' is a (resettable) system. Thus it remains within our universe of discourse. In particular, its reset inputs still act like reset inputs. That is, they cause S' to go into a particular state regardless of the state it was in when the reset input was applied. The restrictions on the input, output, and reset alphabets are reasonable since after a fault occurs the system presumably will have the same input and output terminals as it had before the fault occurred.

Let $f = (S', \tau, \theta)$ be a fault. Because S' may vary with time we have considerable latitude in the types of failures which we may consider. In particular, we may consider simultaneous permanent failures in one or more components, simultaneous intermittent failures in one or more components, or any combination of the above occurring at the same or varying times. For example, a fault f may be caused by an AND gate becoming stuck-at-1 at time τ_1, followed by an OR gate becoming stuck-at-0 at time τ_2.

Let us now compute the behavior of S^f in state q. Let $x = a_1 \ldots a_n \in I^+$. Then

$$\beta_q^f(x,t) = \lambda^f(q,x,t)$$

$$= \lambda^f(\delta^f(q,a_1\cdots a_{n-1},t),a_n,t+n-1)\,.$$

There are three cases which must be considered.

Case i) $q \in Q$ and $t+n-1 < \tau$. Then

$$\beta_q^f(x,t) = \lambda(\delta(q,a_1\cdots a_{n-1},t),a_n,t+n-1)$$

$$= \beta_q(x,t)\,.$$

Case ii) $q \in Q$, $t+n-1 \geq \tau$, and $t < \tau$. Say $t+n-m = \tau$. Then

$$\beta_q^f(x,t) = \lambda'(\delta'(\theta(\delta(q,a_1\cdots,a_{n-m},t)),a_{n-m+1}\cdots a_{n-1},$$

$$t+n-m),a_n,t+n-1)$$

$$= \beta'_{\theta(\delta(q,a_1\cdots a_{n-m},t))}(a_{n-m+1}\cdots a_n,t+n-m)$$

$$= \beta'_{\theta(\delta(q,y,t))}(z,\tau) \quad \text{where } y = a_1\cdots a_{n-m}$$

$$\text{and } z = a_{n-m+1}\cdots a_n\,.$$

Case iii) $q \in Q'$ and $t \geq \tau$. Then

$$\beta_q^f(x,t) = \lambda'(\delta'(q,a_1\cdots a_{n-1},t),a_n,t+n-1)$$

$$= \beta'_q(x,t)\,.$$

Thus we have proved:

Theorem 2.1: Let S be a system and $f = (S', \tau, \theta)$ a fault of S. Then for each $t \in T$ and $x \in I^+$

$$\beta^f_q(x,t) = \begin{cases} \beta_q(x,t) \text{ if } q \in Q \text{ and } t + |x| \leq \tau \\ \beta'_{\theta(\delta(q,y,t))}(z,\tau) \text{ if } q \in Q,\ t + |x| > \tau, \text{ and} \\ \qquad t < \tau \text{ where } x = yz \text{ and } |y| = \tau - t \\ \beta'_q(x,t) \text{ if } q \in Q' \text{ and } t \geq \tau. \end{cases}$$

(As in the definitions of δ^f and λ^f arguments not specified may be assigned arbitrary values.)

Corollary 2.1.1: Let S be a system and $f = (S', \tau, \theta)$ a fault of S. Then for each $r \in R$, $t \in T$, and $x \in I^+$

$$\beta^f_{r,t}(x) = \begin{cases} \beta_{r,t}(x) \text{ if } t + |x| \leq \tau \\ \beta'_{\theta(\delta(\rho(r,t),y,t))}(z,\tau) \text{ if } t + |x| > \tau \text{ and} \\ \qquad t \leq \tau \text{ where } x = yz \text{ and} \\ \qquad |y| = \tau - t \\ \beta'_{r,t}(x) \text{ if } t > \tau. \end{cases}$$

Proof: By its definition

$$\beta^f_{r,t}(x) = \beta^f_{\rho^f(r,t)}(x,t).$$

Again we have three cases to consider.

Case i) $t + |x| \leq \tau$. Then $t < \tau$ and $\rho^f(r,t) = \rho(r,t) \in Q$. Therefore by Theorem 2.1

$$\beta^f_{\rho^f(r,t)}(x,t) = \beta_{\rho(r,t)}(x,t)$$

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

Case ii) $t + |x| > \tau$ and $t \leq \tau$. If $t < \tau$ then $\rho^f(r,t) = \rho(r,t) \in Q$ and Case ii) of Theorem 2.1 applies with $\rho(r,t)$ in place of q. If $t = \tau$ then $\rho^f(r,t) = \theta(\rho(r,t)) \in Q'$ and case iii) of the theorem applies giving us

$$\beta^f_{\rho^f(r,t)}(x,t) = \beta'_{\theta(\rho(r,t))}(x,t)$$

$$= \beta'_{\theta(\delta(\rho(r,t),\Lambda,t))}(x,t) .$$

Case iii) $t > \tau$. In this case $\rho^f(r,t) = \rho'(r,t) \in Q'$. Therefore

$$\beta^f_{\rho^f(r,t)}(x,t) = \beta'_{\rho'(r,t)}(x,t)$$

$$= \beta'_{r,t}(x) .$$

We have noted that we will often be interested in the physical cause of a fault. For example, in a network realization of a machine we may be interested in faults which are caused by a specific NAND gate becoming stuck-at-1. Since this gate failure results in different faults

as we consider it occurring at different times it seems natural to give a name to this family of faults. More generally, we will define an equivalence relation on a set of faults such that a family of faults such as we have just mentioned will be an equivalence class.

First we must define an equivalence relation on $\mathcal{S}(I, Z, R)$ such that two systems S, $S' \in \mathcal{S}(I, Z, R)$ are equivalent if they are identical except for a shift in time.

Definition 2.9: Let S, $S' \in \mathcal{S}(I, Z, R)$. S' is a n-translation of S if $Q = Q'$ and for all $q \in Q$, $a \in I$, $r \in R$, and $t \in T$

i) $\delta(q, a, t) = \delta'(q, a, t+n)$

ii) $\lambda(q, a, t) = \lambda'(q, a, t+n)$

iii) $\rho(r, t) = \rho'(r, t+n)$.

If S' is a n-translation of S then it can be shown that for all $q \in Q$, $r \in R$, $x \in I^+$, and $t \in T$

$$\beta_q(x, t) = \beta'_q(x, t+n)$$

and

$$\beta_{r,t}(x) = \beta'_{r,t+n}(x) .$$

Definition 2.10: Let (S, F) be a system with faults and let $f_1 = (S_1, \tau_1, \theta_1)$ and $f_2 = (S_2, \tau_2, \theta_2)$ be in F. Then f_1 is equivalent to f_2 ($f_1 \equiv f_2$) if S_1 is a $(n_1 - n_2)$-translation of S_2 and $\theta_1 = \theta_2$.

Theorem 2.2: The above relations are equivalence relations.

Proof: The relation of "n-translation" is an equivalence relation on $\mathcal{S}(I, Z, R)$ because "=" is an equivalence relation. The relation "$\equiv$" on a set of faults of a system is an equivalence relation because "n-translation" and "=" are both equivalence relations.

Notation: We denote then equivalence class of F which contains the fault $f = (S, \tau, \theta)$ by $[f]_F$. When the class of faults is clear we will drop the F. Generally if F is not mentioned we take it to be the set of all possible faults of a system S. We let $f_i = (S_i, i, \theta)$ denote the fault in $[f]$ which occurs at time i. When dealing with behaviors β^{f_i} will denote the behavior of S^{f_i}, and β^i will denote the behavior of S_i.

Let $f_i = (S_i, i, \theta)$ and $f_j = (S_j, j, \theta)$ be equivalent faults of a machine M. Since M is a (i-j)-translation of itself, it can be verified directly from Definition 2.8 that M^{f_i} is a (i-j)-translation of M^{f_j}. Hence,

Theorem 2.3: Let f be a fault of M and let f_i, $f_j \in [f]$. Then for all $q \in Q$, $x \in I^+$, $r \in R$ and $t \in T$

$$\beta_q^{f_i}(x, t+i) = \beta_q^{f_j}(x, t+j)$$

and

$$\beta_{r, t+i}^{f_i}(x) = \beta_{r, t+j}^{f_j}(x) .$$

In this section we have defined and studied the notion of a fault of a system. In the remainder of this study we shall limit our investigations to the case in which the fault-free system is time-invariant. That is, we shall be studying faults of machines. This is not a serious restriction since the behavior of (fault-free) computers and related digital equipment does not vary with time. Nevertheless, the concepts developed in this and the preceding section are necessary since faulty machines (except in the case of improper faults) are time-varying. Given a fault $f = (S', \tau, \theta)$ of a machine M, S' will not be restricted to being time-invariant. This allows us to consider intermittant faults.

2.3 Fault Tolerance and Errors

Given a system with faults (S, F) and a proper fault $f \in F$, an immediate question is whether the faulty system S^f is usable in the sense that its behavior resembles, within acceptable limits, that of the fault-free system S. We will use the general notion of a "tolerance relation" [29] to make more precise what is meant by "acceptable limits." A tolerance relation for a representation scheme $(S, \mathcal{R}, \rho)$ is a relation γ between $\mathcal{R}$ and $\mathcal{S}$ ($\gamma \subseteq \mathcal{R} \times \mathcal{S}$) such that, for all $R \in \mathcal{R}$, $(R, \rho(R)) \in \gamma$ (i.e., $\rho \subseteq \gamma$). In this section we will develop the particular notions of "acceptable limits" that we will be using in this study of on-line diagnosis.

Given a machine M it will be understood that M realizes a specific reduced and reachable machine $\tilde{M}$ under the triple $(\sigma_1, \sigma_2, \sigma_3)$. Under the intended interpretation, $\tilde{M}$ serves as the specification of some desired behavior and M serves as the fault-free realization of this behavior. This relationship between M and $\tilde{M}$ will underlie our basic notions of fault tolerance, error and on-line diagnosis.

In this study we will only be concerned with the behavior of M under those resets and inputs which correspond via σ_1 and σ_2 to resets and inputs of $\tilde{M}$. No requirements will ever be put on $\beta_r(x)$ or $\beta^f_{r,t}(x)$, where f is a fault of M, if $r \notin \sigma_2(\tilde{R})$ or $x \notin \sigma_1(\tilde{I}^+)$ because these are considered to be "non-code space resets" and "non-code space inputs." For this reason we will always assume that σ_1 and σ_2 are onto. In actually dealing with machines for which σ_1 or σ_2 is not onto, occurrences

of "non-code space resets" and "non-code space inputs" could be ignored or they could be treated as errors which must be detected. These two options correspond to Carter and Schneider's [7] Don't Care Assignments 1 and 2.

We will be using two basic notions of fault tolerance. The first, and weaker, corresponds to the preservation of the behavior of M only insofar as its mimicing of $\tilde{M}$ is concerned.

Definition 2.11: Let f be a fault of a machine M. Then f is 1-tolerated by M for resets at time t if for all $\tilde{r} \in \tilde{R}$

$$\beta_{\tilde{r}} = \sigma_3 \circ \beta_{\sigma_2(\tilde{r}),t} \circ \sigma_1$$

Alternatively, since σ_1 and σ_2 are onto and since $\tilde{\beta}_{\tilde{r}} = \sigma_3 \circ \beta_{\sigma_2(\tilde{r})} \circ \sigma_1$, f is 1-tolerated by M for resets at time t if for all $r \in R$

$$\sigma_3 \circ \beta_r = \sigma_3 \circ \beta^f_{r,t}$$

In the special case where f is 1-tolerated by M for resets at time 0, we will simply say that f is 1-tolerated by M.

The second, and stronger, notion of tolerance does not allow for the tolerance of any change in behavior.

Definition 2.12: Let f be a fault of a machine M. Then f is 2-tolerated by M for resets at time t if for all $r \in R$, $\beta_r = \beta^f_{r,t}$.

Again, f is 2-tolerated by M if it is 2-tolerated by M for resets at time 0.

Our definition of 1-tolerated induces a relation γ_1 on $\mathcal{R}$ where $M^f \gamma_1 M$ if and only if f is 1-tolerated by M. If f is improper then $M^f = M$ and thus f is 1-tolerated by M. Hence $M \gamma_1 M$, and therefore γ_1 is a tolerance relation. Likewise 2-tolerated induces a tolerance relation γ_2. If f is 2-tolerated by M then we can see that f is 1-tolerated by M. Hence, as sets, $\gamma_2 \subseteq \gamma_1$. Finally, note that if σ_3 is 1-1 and f is 1-tolerated by M then f is 2-tolerated by M.

Example 2.6: Let M be the realization of $\tilde{M}$ which consists of 3 copies of $\tilde{M}$, a voter, and a disagreement detector as shown in Fig. 2.13. Then any fault f which affects only one copy of $\tilde{M}$ is 1-tolerated but may not be 2-tolerated, and its presence may be detected by the disagreement detector.

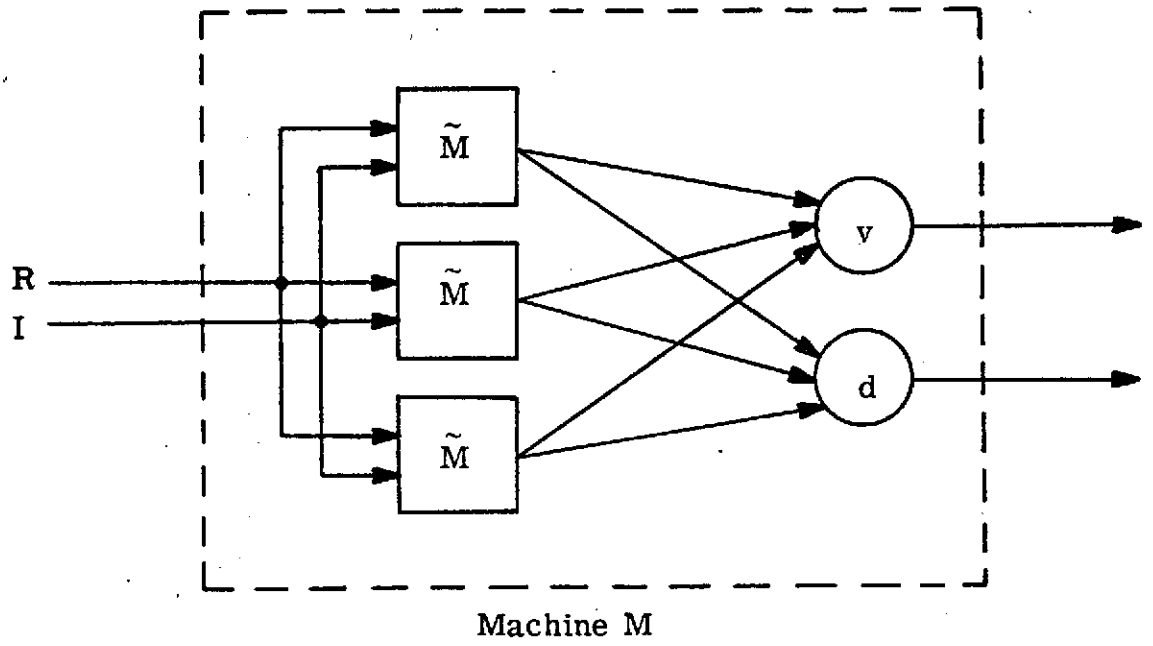


Fig. 2.13. Triple Modular Redundancy with Voting and Disagreement Detecting

Our definitions of 1 and 2-tolerated by M for resets at time t are refined notions of fault tolerance. Coarser notions, and ones more in keeping with the literature, would be behavioral equivalence for resets at any time. We prefer our finer definitions for with them the effects of time can be more naturally analyzed. One question which we will study later is: For resets at how many (and which) times must a fault be tolerated for it to be tolerated for resets at any time?

When a discussion or theorem applies equally well to 1-tolerated and to 2-tolerated we will just use the general term "tolerated." We also do this latter in this section when we discuss "errors."

It would be convenient if, without loss of generality, it was possible to consider the behavior of systems only for resets released at time 0. The following result shows that this can be done by a simple change in the fault set under consideration.

Theorem 2.4: Let $f = (S', \tau, \theta)$ be a fault of machine M. Then f is tolerated by M for resets at time t if and only if $f_{\tau-t}$ is tolerated by M.

Proof: By Theorem 2.3, $\beta^{f_\tau}_{r,t} = \beta^{f_{\tau-t}}_{r,0}$. Hence, $\sigma_3 \circ \beta^{f_\tau}_{r,t} = \sigma_3 \circ \beta^{f_{\tau-t}}_{r,0}$, and $\sigma_3 \circ \beta_r = \sigma_3 \circ \beta^{f_\tau}_{r,t}$ if and only if $\sigma_3 \circ \beta_r = \sigma_3 \circ \beta^{f_{\tau-t}}_{r,0}$. This establishes the result.

Thus a fault f is tolerated by M for resets at any time if and only if the class [f] of faults equivalent to f is tolerated by M. Due to this we will always consider resets to be released at time 0 when dealing with fault tolerance of machines and no generality will be lost. Clearly, due to Theorem 2.3, this same sort of time translation can be applied to any other behavioral attribute.

Example 2.7: Let M_4 be the sequence generator shown in Fig. 2.14. This machine could be implemented by the circuit shown in Fig. 2.15.

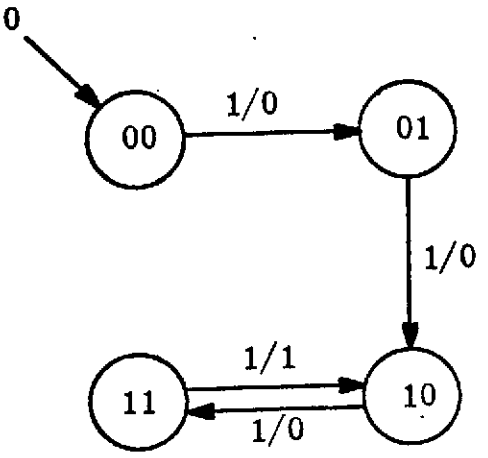


Fig. 2.14. Machine M_4

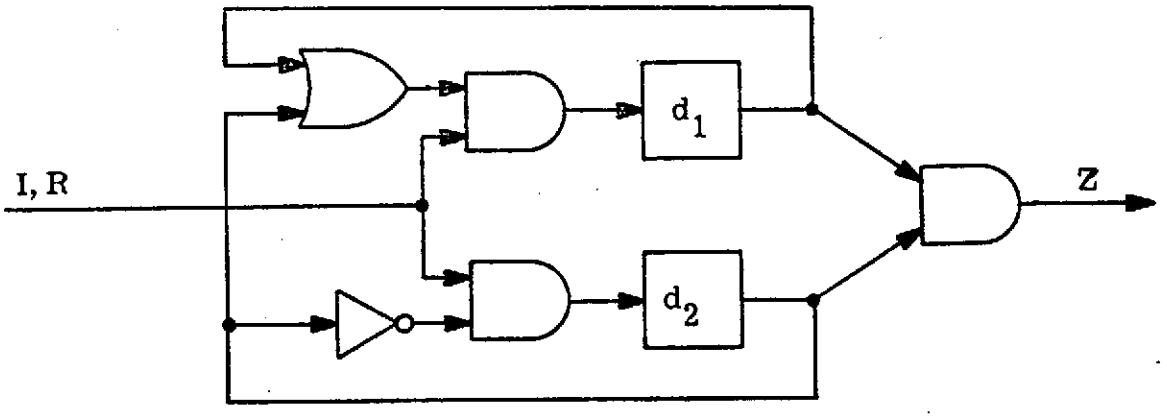


Fig. 2.15. Circuit for M_4

Let f be a fault of M_4 which is caused by d_1 becoming stuck-1 at time τ. Then $f = (M_4', \tau, \theta)$ where M_4' is the machine represented by the graph in Fig. 2.16 and θ is as indicated below.

q	$\theta(q)$
00	10
01	11
10	10
11	11

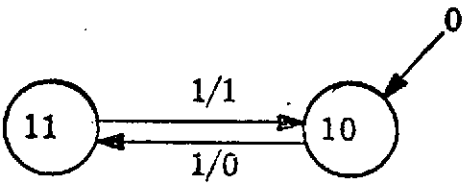


Fig. 2.16. Machine M_4'

Consider f_{-1}, i.e., the fault $(M_4', -1, \theta)$, and note that $\beta_0^{f_{-1}}(11) = 1$ whereas $\beta_0(11) = 0$. Thus f_{-1} is not 2-tolerated by M_4. On the other hand both M_4 and M_4^{-1} will produce the sequence 00010101... when reset at -10. Thus f_{-1} is 2-tolerated by M_4 for resets at -10. By applying Theorem 2.4 we can learn, for example, that f_i is not 2-tolerated by M_4 for resets at time $i+1$ and that f_9 is 2-tolerated by M_4.

Corresponding to our two types of fault tolerance we can define two types of errors.

Definition 2.13: Let M be a machine, $r \in R$, $x \in I^+$, and $y \in Z^+$ where $|x| = |y|$. The triple (r, x, y) is called a 1-error (2-error) of M if $\sigma_3(\hat{\beta}_r(x)) \neq \sigma_3(y)$ $(\hat{\beta}_r(x) \neq y)$.

If (r, x, y) is an error of M and f is a fault of M for which $\hat{\beta}_r^f(x) = y$ then we say that the fault f causes the error (r, x, y). Note that any given error could be caused by many different faults.

The relation between fault tolerance and errors is very simple. A fault f is 1-tolerated (2-tolerated) if and only if it causes no 1-errors (2-errors). The relation between 1-errors and 2-errors is also straightforward. Namely, every 1-error is a 2-error, and if σ_3 is 1-1 then every 2-error is a 1-error. Errors are very important in any study of fault diagnosis because a fault can never be detected until it causes an error. The general goal of on-line diagnoses is protection against undesirable behavioral manifestations of faults, i.e., protection against errors.

Since an error can represent erroneous behavior of any duration, and since we will wish to detect erroneous behavior when it first begins to appear, we introduce the concept of a "minimal error." Informally, an error (r, x, y) is a minimal error if only the last symbol of the output sequence y is out of tolerance. More formally, an error (r, ua, vb) where $a \in I$ and $b \in Z$ is a minimal error if (r, u, v) is not an error. If (r, x, y) is a minimal 1-error then it is a 2-error but not necessarily a minimal 2-error. A minimal error

(r, x, y) is said to occur at time $|x| - 1$. This is the time at which the last symbol in y is emitted.

Often we will be in a situation where we are concerned with a machine M tolerating a set of faults which are all caused by the same phenomenon but which may occur at any time. More specifically, let f be a fault of M. We would like results which assured us that if some finite subset of $[f]$ was tolerated by M then all of $[f]$ was tolerated by M. Later we will be interested in the same problem with regard to diagnosis.

Our first result of this nature hinges on the fact that any reachable state of an ℓ-reachable machine is reachable by time ℓ.

Theorem 2.5: Let f be a fault of an ℓ-reachable machine M and suppose f_i is tolerated by M for $0 \leq i \leq \ell$. Then f_i is tolerated by M for all $i \geq 0$.

Proof: Assume, to the contrary, that f_i is not tolerated by M for some $i > \ell$. Then there exists an error (r, x, y) which is caused by f_i. Hence $\hat{\beta}_r^{f_i}(x) = y$. Let $x = x_1x_2$ and $y = y_1y_2$ where $|x_1| = |y_1| = i$. By Corollary 2.1.1 we know that

$$\hat{\beta}_r^{f_i}(x) = \hat{\beta}_r(x_1)\hat{\beta}^i_{\theta(\delta(\rho(r), x_1))}(x_2, i) = y_1y_2 .$$

Let $q = \delta(\rho(r), x_1)$. Since M is ℓ-reachable, there exists $s \in R$ and $u \in I^+$ such that $|u| = j \leq \ell$ and $\delta(\rho(s), u) = q$. By Theorem 2.3

$\hat{\beta}^{i}_{\theta(q)}(x_2, i) = \hat{\beta}^{j}_{\theta(q)}(x_2, j)$. Therefore if $\hat{\beta}_s(u) = v$ then $\hat{\beta}^{f_j}_{\varsigma}(ux_2) = \hat{\beta}_s(u)\hat{\beta}^{j}_{\theta(\delta(\rho(s),u))}(x_2, j) = v\hat{\beta}_{\theta(q)}(x_2, j) = vy_2$. Clearly, (s, ux_2, vy_2) is an error and it is caused by f_j. Therefore f_j is not tolerated. Contradiction. This establishes the result.

The following general example shows that Theorem 2.5 is the strongest result possible, in the sense that if the hypothesis is at all weakened then there exists a fault f and a machine M for which the conclusion is invalid.

Example 2.8: Consider the ℓ-reachable autonomous machine M_ℓ shown in Fig. 2.17. Let m be an integer between 0 and ℓ inclusive, and let

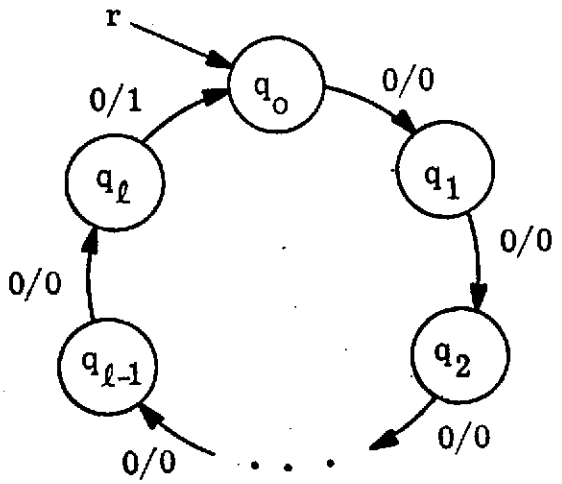


Fig. 2.17. Machine M_ℓ

$f = (M_\ell, \tau, \theta)$ be a fault of M_ℓ where

$$\theta(q_j) = \begin{cases} q_j \text{ if } j \neq m \\ \delta(q_j, 0) \text{ if } j = m \end{cases}$$

Consider M_ℓ to be realizing itself. That is, take $\tilde{M} = M_\ell$.

The occurrence of $f = (M_\ell, \tau, \theta)$ has an effect on the behavior of M_ℓ if and only if M_ℓ could be in state q_m at time τ. Therefore, $f_i = (M_\ell, i, \theta)$ is tolerated by M_ℓ if and only if $i \not\equiv m \pmod{\ell + 1}$. Hence f_i is tolerated by M_ℓ for $i = 0, \ldots, m-1, m+1, \ldots, \ell$ does not imply f_i is tolerated by M_ℓ for all $i \geq 0$. Since both m and ℓ were arbitrarily chosen, this general example shows that the hypothesis of Theorem 2.5 cannot be weakened.

Let us now look at faults which occur before time 0. In the previous result we have not mentioned this case because if f_i and f_j are equivalent faults and i or j is less than 0 then there is, in general, no relation between the behaviors of M^{f_i} and M^{f_j} for resets released at time 0. However, in the important special case where $f = (M', \tau, \theta)$ is a permanent fault, any $f_i \in [f]$ with $i < 0$ will, with respect to resets released at time 0, cause identical behavior.

Lemma 2.1: Let $f = (M', \tau, \theta)$ be a permanent fault of M. Then $\beta_r^{f_i} = \beta_r^{f_j}$ for all $r \in R$ and $i, j < 0$.

Proof: Let $i, j < 0$. Because f is permanent, $f_i = (M', i, \theta)$ and $f_j = (M', j, \theta)$. By Corollary 2.1.1, $\beta_r^{f_i} = \beta_r'$ and $\beta_r^{f_j} = \beta_r'$ for all $r \in R$. This establishes the result.

Theorem 2.6 : . Let f be a permanent fault of an ℓ-reachable machine M. If f_i is tolerated by M for $-1 \leq i \leq \ell$ then f_i is tolerated by M for all $i \in T$.

Proof: By Lemma 2. 1, $\beta_r^{f_i} = \beta_r^{f_{-1}}$ for all $i < 0$. Hence, f_{-1} is tolerated by M implies that f_i is tolerated by M for all $i < 0$. By Theorem 2. 5, f_i is tolerated by M for all $i \geq 0$. This establishes the result.

Before leaving this line of development we will make some final observations. Note that a machine M is 0-reachable if and only if $\rho(R) = P$. In particular, every memoryless machine is 0-reachable. By Theorem 2. 5, if M is 0-reachable and f_o is tolerated by M then f_i is tolerated by M for all $i \geq 0$.

If $f = (M', \tau, \theta)$ is a fault of M we think of f as affecting the reset mechanism of M if $\rho'(r) \neq \theta(\rho(r))$ for some $r \in R$. If this is not the case then a further result, similar to Lemma 2.1 can be obtained.

Lemma 2. 2 : Let $f = (M', \tau, \theta)$ be a permanent fault of M and suppose that $\rho'(r) = \theta(\rho(r))$ for all $r \in R$. Then $\beta_r^{f_i} = \beta_r^{f_j}$ for all $r \in R$ and $i, j \leq 0$.

Proof: Since $\rho'(r) = \theta(\rho(r))$, by Corollary 2. 1. 1, $\beta_r^{f_o} = \beta'_r$ for all $r \in R$. The result now follows just as in the proof of Lemma 2. 1.

Putting the above observations together yields:

Theorem 2.7 : Let $f = (M', \tau, \theta)$ be a permanent fault of M. Suppose that $\rho'(r) = \theta(\rho(r))$ for all $r \in R$ and that $\rho(R) = P$. If f_i is tolerated by M for any $i \leq 0$ then f_i is tolerated by M for all $i \in T$.

Proof: By Lemma 2.2 f_i is tolerated by M for all $i \leq 0$. Since $\rho(R) = P$, M is 0-reachable. Therefore, by Theorem 2.5 f_i is tolerated by M for all $i \geq 0$. This establishes the result.

2.4 On-line Diagnosis

Our notion of on-line diagnosis of a system involves an external detector (assumed to be fault-free) which observes the input and the output of the system and makes a decision as to whether the behavior of the system is within "acceptable limits" as set forth by our notions of fault tolerance. Initial synchronization of the system with its detector is achieved by using the same reset to initialize both systems.

The formal relation between a system and its detector is that of a "cascade connection."

Definition 2.14: The cascade connection of two systems S_1 and S_2 for which $R_1 = R_2$ and $I_2 = Z_1 \times I_1$ is the system

$$S_1 * S_2 = (I_1, Q, Z_2, \delta, \lambda, R_1, \rho)$$

where

$$Q = Q_1 \times Q_2$$

$$\delta((q_1, q_2), a, t) = (\delta_1(q_1, a, t), \delta_2(q_2, (\lambda_1(q_1, a, t), a), t))$$

$$\lambda((q_1, q_2), a, t) = \lambda_2(q_2, (\lambda_1(q_1, a, t), a), t)$$

$$\rho(r, t) = (\rho_1(r, t), \rho_2(r, t)).$$

Schematically, $S_1 * S_2$ can be pictured as in Fig. 2.18.

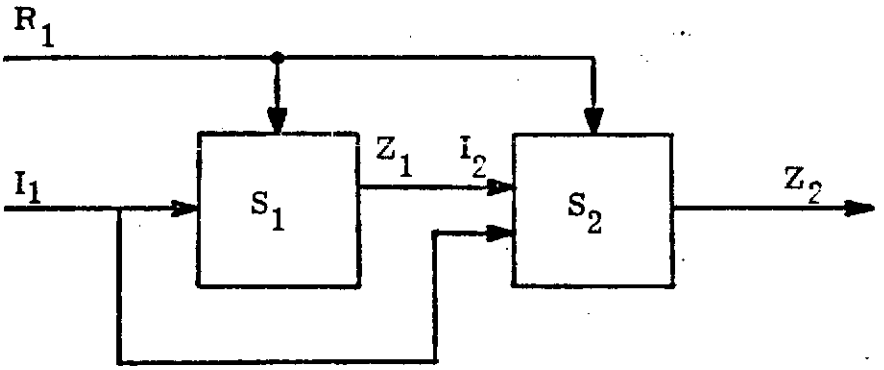


Fig. 2.18. The Cascade Connection of S_1 and S_2

Notation: If $u = z_1 z_2 \ldots z_n \in Z^+$ and $v = a_1 a_2 \ldots a_n \in I^+$ then the pair $[u, v]$ will denote the sequence $(z_1, a_1)(z_2, a_2)\ldots(z_n, a_n) \in (Z \times I)^+$.

Let $S_1 * S_2$ be the cascade connection of S_1 with S_2. Let β^1, β^2, and β^* denote the behavior functions of S_1, S_2, and $S_1 * S_2$ respectively. It can be shown directly from the definition of a cascade connection that for all $x \in I_1^+$, $q_1 \in Q$, $q_2 \in Q_2$, $r \in R_1$, and $t \in T$,

$$\beta^*_{(q_1, q_2)}(x, t) = \beta^2_{q_2}([\hat{\beta}^1_{q_1}(x, t), x], t)$$

and

$$\beta^*_{r,t}(x) = \beta^2_{r,t}([\hat{\beta}^1_{r,t}(x), x]) .$$

We can now formally define our notion of on-line diagnosis.

Definition 2.15: Let (M, F) be a machine with faults, let D be a machine for which M * D is defined, and let k be a nonnegative integer. (M, F) is (D, k)-1-diagnosable (2-diagnosable) if

i) $\beta^*_r = 0$ for all $r \in R$, and

ii) if (r, x, y) is a minimal 1-error (2-error) caused by some $f \in F$ then

$$\hat{\beta}^D_r([\hat{\beta}^f_r(xw), xw]) \neq 0^{|xw|} \text{ for all } w \in I^* \text{ with } |w| = k .$$

Thus, the detector D observes the operation of M^f and must make a decision based on this observation as to whether an error has occurred. Note that the fault-free realization M and the detector are both time-invariant (i.e., machines), and that the detector takes no part in the computation of M's output.

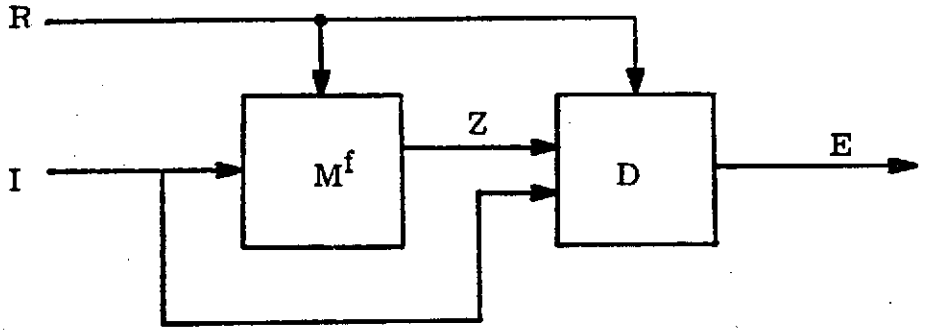


Fig. 2.19. Diagnosis of (M, F) using the Detector D

The two conditions of Definition 2.15 can be paraphrased as:

i) D responds negatively if no fault occurs; i.e., D gives no false alarms, and

ii) for all $f \in F$, D responds positively within k time steps of the occurrence of the first error caused by f.

Condition i) implies $0 \in Z_D$, the output alphabet of D. Each $z \in Z_D$ other than 0 is called a fault-detection signal. The choice of the symbol "0" to indicate that the machine M is operating properly is purely for notational convenience. In general we could let any subset of Z_D indicate proper operation and let the complement of this set in Z_D be the set of fault-detection signals. In a practical application this choice would depend on the design constraints on the detector.

As we have done with fault tolerance and with errors, if a theorem or remark applies to both "1-diagnosable" and "2-diagnosable" we will just state it once using the general term "diagnosable."

Let D be a detector for M. Then $I_D = Z \times I$. There will be times when the observation of M's input by D will be unnecessary or undesired. If for all $z \in Z$ and $a, b \in I$ (z, a) and (z, b) are equivalent inputs of D then we will say that D is independent of M's input. In this case the behavior of D does not depend on the second coordinate of D's input and we will take I_D to be simply Z.

Recall that with this concept of diagnosis that we are only considering faults of M. Faults of D must be analyzed separately. In

finding a realization M of $\tilde{M}$ and a detector D there is some leeway in how much of the added complexity required for diagnosis should go into the detector and how much should go into the realization. If it all goes into the realization then D will serve only to select out certain coordinates of M's output to be used as the output of D. That is, D will be memoryless and realize a projection. In this case we will say that (M, F) is k-self-diagnosable. In general, it is desirable for the detector to be self-diagnosable for some suitable set of faults.

The basic on-line diagnosis problem can now be restated as follows:

> Given a machine $\tilde{M}$, a class of faults F, a class of detectors $\mathscr{D}$ and a delay k find an (economical) realization M of $\tilde{M}$ and a detector $D \in \mathscr{D}$ such that (M, F) is (D, k)-diagnosable.

In this chapter we have developed a model for the study of on-line diagnosis of resettable machines, and we have restated the basic on-line diagnosis problem. In the following chapters results are developed which will help to solve this basic problem.

CHAPTER III

General Properties of Diagnosis

In this short chapter we will present a few results on diagnosis per se. That is, they are general results which tell us some things about diagnosis, independent of the particular fault set being diagnosed or of any particular diagnosis technique. In the following chapters we look at the diagnosis of specific sets of faults and investigate the capabilities and limitations of on-line diagnosis techniques.

It is interesting to see how our concept of on-line diagnosis compares with a similar concept introduced by Carter and Schneider [7] and called "fault-secure" by Anderson [1]. As stated by Anderson, "A circuit is fault-secure if, for every fault in a prescribed set, the circuit never produces incorrect code space outputs for code space inputs."

Before making a formal comparison this notion must be translated into our framework. In doing so we will strive to be faithful to Anderson's intent.

Definition 3.1: A machine with faults, (M, F), is fault-secure if (r, x, ya), where $a \in Z$, is a minimal 2-error caused by some $f \in F$ implies $a \notin \{\beta_r(x) \mid r \in R, x \in I^+\}$.

Thus if (M, F) is fault-secure then a combinational detector which only observes the output of M can detect all minimal 2-errors. More formally,

Theorem 3.1: (M, F) is fault-secure if and only if (M, F) is (D, 0)-2-diagnosable where D is memoryless and independent of M's input.

Proof: (Necessity) Assume that M is fault-secure. Define λ_D: $Z \rightarrow \{0, 1\}$ by

$$\lambda_D(z) = \begin{cases} 0 \text{ if } z \in \{\beta_r(x) \mid r \in R,\ x \in I^+\} \\ 1 \text{ otherwise} \end{cases}$$

Let D be the memoryless detector which realizes λ_D. Then D is independent of M's input and it can easily be verified that (M, F) is (D, 0)-2-diagnosable.

(Sufficiency) Assume that (M, F) is (D, 0)-2-diagnosable where D is memoryless and independent of M's input. Let λ_D: $Z \rightarrow \{0, 1\}$ denote the function realized by D and let $Z' = \{\beta_r(x) \mid r \in R,\ x \in I^+\}$. Then $\lambda_D(z) = 0$ for all $z \in Z'$ for otherwise a false alarm could occur. Let (r, x, ya) where $a \in Z$ be a minimal 2-error. If $a \in Z'$ then $\lambda_D(a) = 0$ and f is not detected without delay. Therefore $a \notin Z'$. Hence (M, F) is fault-secure.

Thus the concept of (D, k)-diagnosable is a generalization of the concept of fault-secure. In particular, (D, k)-diagnosis allows for (i) different tolerance relations, (ii) nonzero delay in diagnosis,

(iii) detectors with memory, and (iv) explicit observation by the detector of the input to the system being monitored.

Our next result shows that "2-diagnosable" is indeed a stronger property than "1-diagnosable." This result is a consequence of the fact that every 1-error is a 2-error but not conversely.

Theorem 3.2: If (M, F) is (D, k)-2-diagnosable then (M, F) is (D, k)-1-diagnosable, but not conversely.

Proof: Let (M, F) be (D, k)-2-diagnosable. Then no false alarms will occur and every minimal 2-error will be detected within k time steps of its occurrence. Let (r, x, y) be a minimal 1-error. Then $\sigma_3(\hat{\beta}_r(x)) \neq \sigma_2(y)$ and hence $\hat{\beta}_r(x) \neq y$. Thus (r, x_1, y_1) is a minimal 2-error for some x_1 and y_1 such that $x = x_1x_2$ and $y = y_1y_2$. Since this minimal 2-error is detected within k time steps of its occurrence the minimal 1-error (r, x, y) must also be detected within k time steps of its occurrence. Hence (M, F) is (D, k)-1-diagnosable.

The counterexample which shows that the converse does not hold is given in the next chapter in the proof of Theorem 4.4.

Although the converse of Theorem 3.2 does not hold in general, the following partial converse can be obtained.

Theorem 3.3: If (M, F) is (D, k)-1-diagnosable and σ_3 is 1-1 then (M, F) is (D, k)-2-diagnosable.

Proof: We observed in Section 2.3 that if σ_3 is 1-1 then every 2-error is a 1-error. The result is an immediate consequence of this fact.

The next result will help us to see the relationship between fault diagnosis and fault tolerance.

Theorem 3.4: Let (M, F) be a machine with faults. If F is tolerated by M then (M, F) is $(D_0, 0)$-diagnosable where D_0 is a trivial memoryless machine which realizes the constant 0 function.

Proof: Condition i) is clearly satisfied, and condition ii) is satisfied because if F is tolerated by M then no $f \in F$ will cause any errors.

The decision in this case can be trivially made since no errors are ever produced. The situation for tolerated faults is not so simple as this result may seem to indicate for it must be remembered that 1-tolerated does not imply 2-tolerated and thus a 1-tolerated fault could be detected through a 2-error (see Example 2.6).

We will now develop some results concerning diagnosis which are analogous to Theorems 2.5, 2.7 and 2.9. Recall that these theorems allowed us to infer the tolerance of an infinite set of equivalent faults from knowledge that a specific finite subset of them is tolerated.

Theorem 3.5: Let M be a machine and let D be a detector for M. Suppose that the cascade connection M * D is ℓ-reachable, and that f is a fault of M. If $(M, \{f_i\})$ is (D, k)-diagnosable for $0 \leq i \leq \ell$ then $(M, \{f_i\})$ is (D, k)-diagnosable for all $i \geq 0$.

Proof: Assume that $(M, \{f_i\})$ is (D, k)-diagnosable for $0 \leq i \leq \ell$. Then condition i) of Definition 2.15 is immediately satisfied. Let (r, x, w) be a minimal error caused by f_i where $i > \ell$, and let $u \in I^+$ with $|u| = k$. To show that $(M, \{f_i\})$ is (D, k)-diagnosable for $0 \leq i$ we need only show that $\hat{\beta}_r^D([\hat{\beta}_r^{f_i}(xu), xu]) \neq 0^{|xu|}$.

Let $x = x_1 z$ where $|x_1| = i$, and let $\delta^*(\rho^*(r), x_1) = (q, q')$. Since M * D is ℓ-reachable there exists $s \in R$ and $y \in I^+$ with $0 \leq |y| \leq \ell$ such that $\delta^*(\rho^*(s), y) = (q, q')$. Say $|y| = j$. Since $(M, \{f_j\})$ is (D, k)-diagnosable, $\hat{\beta}_s^D([\hat{\beta}_s^{f_j}(yzu), yzu]) \neq 0^{|yzu|}$, and since the fault detection signal must occur after the fault occurs,

$$\hat{\beta}_{q'}^D([\hat{\beta}_{\theta(q)}^{f_j}(zu, j), zu]) \neq 0^{|zu|}.$$

Now by Theorem 2.3, $\hat{\beta}_{\theta(q)}^{f_i}(zu, i) = \hat{\beta}_{\theta(q)}^{f_j}(zu, j)$ and hence $\hat{\beta}_{q'}^D([\hat{\beta}_{\theta(q)}^{f_i}(zu, i), zu]) \neq 0^{|zu|}$. Therefore

$$\hat{\beta}_r^D([\hat{\beta}_r^{f_i}(x_1 zu), x_1 zu]) = \hat{\beta}_r^D([\hat{\beta}_r^{f_i}(x_1), x_1])\hat{\beta}_{q'}^D([\hat{\beta}_{\theta(q)}^{f_i}(zu, i), zu])$$
$$\neq 0^{|xu|}.$$

Hence $(M, \{f_i\})$ is (D, k)-diagnosable for all $i \geq 0$.

Example 2.8, which shows that the hypothesis of Theorem 2.5 cannot be weakened, works likewise for Theorem 3.4. This example works for both fault tolerance and fault diagnosis because, as was pointed out by Theorem 3.3, tolerated faults are trivially diagnosable.

Theorem 3.6: Let M be a machine and let D be a detector for M such that M * D is ℓ-reachable. If f is a permanent fault of M and $(M, \{f_i\})$ is (D, k)-diagnosable for $-1 \leq i \leq \ell$ then $(M, \{f_i\})$ is (D, k)-diagnosable for all $i \in T$.

Proof: Assume that f is a permanent fault and that $(M, \{f_i\})$ is (D, k)-diagnosable for $-1 \leq i \leq \ell$. By Theorem 3.4, $(M, \{f_i\})$ is (D, k)-diagnosable for all $i \geq 0$. By Lemma 2.6, $\beta_r^{f_i} = \beta_r^{f_{-1}}$ for all $r \in R$ and $i < 0$. Hence every f_i with $i < 0$ will cause exactly the same errors. Since $(M, \{f_{-1}\})$ is (D, k)-diagnosable it follows that $(M, \{f_i\})$ is (D, k)-diagnosable for all $i < 0$. This establishes the result.

Let D be a detector for a machine M. It will often be the case that the second coordinate of the state of M * D can be uniquely determined from the first coordinate. In particular, this is always the case when $|Q_D| = 1$. More formally, the cascade connection of M_1 with M_2 is synchronized if there exists a function $h: Q_1 \rightarrow Q_2$

such that for each (q_1, q_2) in the reachable part of $M_1 * M_2$, $h(q_1) = q_2$. Such a function is called the synchronizing function of $M_1 * M_2$ and it must satisfy $h(\rho_1(r)) = \rho_2(r)$ for each $r \in R$.

If $M * D$ is synchronized and M is ℓ-reachable then $M * D$ is also ℓ-reachable. We have observed in Chapter II that M is 0-reachable if and only if $\rho(R) = P$, and that, in particular, every memoryless machine is 0-reachable. Hence if $\rho(R) = P$ and $M * D$ is synchronized then $M * D$ is 0-reachable. In this case we know that if f_o is diagnosable then f_i is diagnosable for $0 \leq i$.

We terminate this line of development by stating the strongest result of this nature.

Theorem 3.7: Let M be a machine for which $\rho(R) = P$. Let D be a detector for M such that $M * D$ is synchronized. Let $f = (M', \tau, \theta)$ be a permanent fault for which $\rho'(r) = \theta(\rho(r))$ for all $r \in R$. If $(M, \{f_i\})$ is (D, k)-diagnosable for any $i \leq 0$ then $(M, \{f_i\})$ is (D, k)-diagnosable for all $i \in T$.

Proof: Assume that $(M, \{f_\ell\})$ is (D, k)-diagnosable where $\ell \leq 0$. By Lemma 2.8, $\beta_r^{f_i} = \beta_r^{f_j}$ for all $i, j \leq 0$. Therefore $(M, \{f_i\})$ is (D, k)-diagnosable for all $i \leq 0$. Since $\rho(R) = P$ and $M * D$ is synchronized, $M * D$ is 0-reachable. Thus by Theorem 3.4, $(M, \{f_i\})$ is (D, k)-diagnosable for all $i \geq 0$. This establishes the result.

CHAPTER IV

Diagnosis of Unrestricted Faults

The investigation of this chapter is concerned with the general case in which the set of potential faults is "unrestricted." This set of faults is precisely the set of all faults of the machine being diagnosed, and hence it is truely unrestricted.

Aside from representing a "worst-case" fault environment, there are certain practical reasons for considering unrestricted faults, at least at the outset. In particular, as the scale of integrated circuit technology becomes larger, it becomes more difficult to postulate a suitably restricted class of faults such as the class of all "stuck-at" faults. Moreover, although other failure models such as bridging failures have been proposed and studied (see [15] and [26] for example), little is known about the diagnosis of such failures. In addition, intermittent and multiple failures are also possible and are even more difficult to model. Finally, for a given failure it may be impossible to determine the θ function of the fault caused by this failure. Thus fault sets which do not restrict the fault mapping θ are advantageous.

Unrestricted faults are typically diagnosed using the technique of duplication. One of the aims of this chapter is to take a deeper look at duplication and at a generalization of this scheme. An

alternative to using duplication for the diagnosis of unrestricted faults is investigated in Chapter V.

The main result in this chapter states that to achieve 1-diagnosis of the unrestricted faults of a machine M, the detector must have as many states as $\tilde{M}$, the behavioral specification for M. Furthermore, to achieve 2-diagnosis, the detector must have as many states as M_R, the reduction of M. These bounds on the state set size of the detector are independent of the delay allowed for the diagnosis.

4.1 Unrestricted Faults

As stated above, the set of unrestricted faults of a machine is simply the set of all faults of that machine. More formally,

Definition 4.1: The set of unrestricted faults of machine M, denoted by U_M, is the set $U_M = \{f \mid f \text{ is a fault of } M\}$. That is,

$$U_M = \{(S', \tau, \theta) \mid S' \in \mathcal{S}(I, Z, R),\ \tau \in T,\ \text{and } \theta: Q \rightarrow Q'\}\ .$$

When it is clear what machine is under consideration, the identifying subscript will be dropped.

One important property of the set of unrestricted faults is the relation between this fault set and the set of errors that may be caused by faults in this set. Given any $r \in R$, $x \in I^+$ and $y \in Z^+$ with $|x| = |y|$, there is a fault $f \in U$ such that $\hat{\rho}_r^f(x) = y$. Therefore faults in U can cause any possible erroneous behavior, and for (M, U) to be (D, k)-diagnosable all of these possible erroneous behaviors will have to be detected by D.

Due to the above observation it is clear that the output of M^f (the system actually being observed by the detector) can give no information about what the correct output should be. Therefore, for the diagnosis of unrestricted faults, the ability of D to observe M's input directly is crucial. This observation is made explicit in the following result.

Theorem 4.1: If (M, U) is (D, k)-1-diagnosable, D is independent of M's input, and $\tilde{M}$ is transition distinct then M is autonomous.

Proof: Suppose that (M, U) is (D, k)-1-diagnosable, D is independent of M's input, and $\tilde{M}$ is transition distinct. Assume, to the contrary, that M is not autonomous. Then there exists $r \in R$ and $x, y \in I^+$ such that $|x| = |y|$ and $\sigma_3(\hat{\beta}_r(x)) \neq \sigma_3(\hat{\beta}_r(y))$. Let $v \in I^*$ with $|v| = k$. For no false alarms to occur we must have $\hat{\beta}_r^D(\hat{\beta}_r(xv)) = 0^{|xv|}$ and $\hat{\beta}_r^D(\hat{\beta}_r(yv)) = 0^{|yv|}$. Let $f \in U$ be a fault for which $\hat{\beta}_r^f(xv) = \hat{\beta}_r(yv)$. Since $(r, x, \hat{\beta}_r^f(x))$ is a 1-error it must be detected within k time steps of its occurrence. But $\hat{\beta}_r^D(\hat{\beta}_r^f(xv)) = \hat{\beta}_r^D(\hat{\beta}_r(yv)) = 0^{|yv|}$. Contradiction. Hence M must be autonomous.

4.2 Diagnosis Via Independent Computation and Comparison

It is a well-known and obvious fact that if a system is duplicated and both copies are run in parallel with the same inputs then by dynamically comparing the outputs of the two copies any error which does not appear simultaneously in both copies will be immediately detected.

Our view of duplication is shown in Fig. 4.1. In this figure

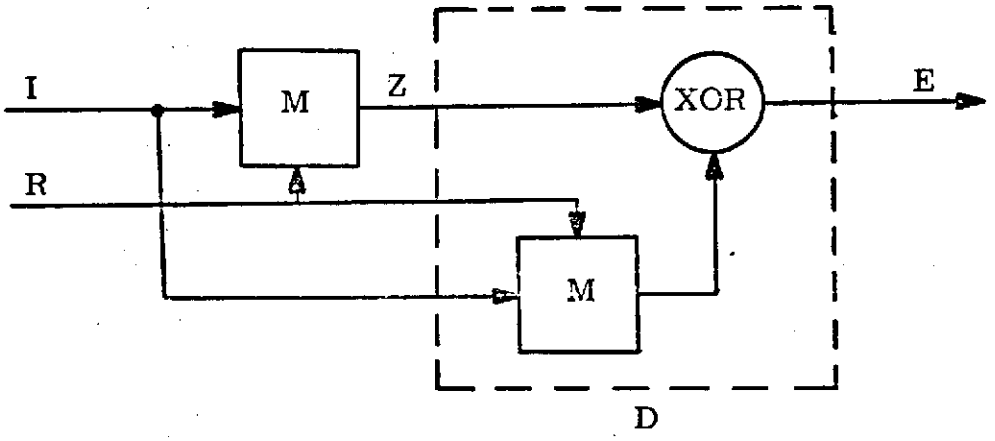


Fig. 4.1. Diagnosis via Duplication in the Detector

the detector D consists of a copy of M along with a generalized Exclusive-OR gate whose output is 0 if and only if its inputs are identical. Given such a detector D, it is immediately clear that (M, U) is (D, 0)-2-diagnosable.

Duplication is an expensive technique, involving somewhat more than twice the circuitry required for the unchecked system alone, but it has a number of positive attributes. In addition to being capable of diagnosing the unrestricted set of faults,

synthesis is easy and self-testing and self-diagnosable comparators are known to exist [1].

The basic configuration shown in Fig. 4. 1 can be generalized to the configuration shown in Fig. 4. 2. In this figure the detector

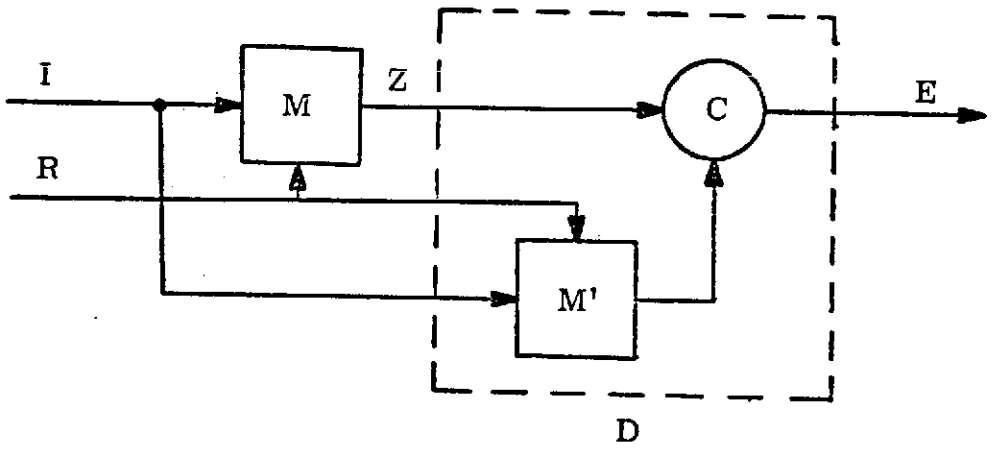


Fig. 4. 2. A Generalization of Duplication in the Detector

consists of a machine M' which runs in parallel with M and a combinational comparator C which dynamically compares the outputs of M and M'. Note that for the cascade connection M * D to be defined we must have I' = I and R' = R.

With this scheme M' may be much less complex than M. However, we will show that there is a relationship between the size of the state set of M' and the level of diagnosis which may be possible using M'.

In the following result we give a necessary and sufficient condition for (M, U) to be $(D, 0)$-diagnosable where D is structured as in Fig. 4.2. The basic intuition for this result is that (M, U) is $(D, 0)$-1-diagnosable if and only if it is possible to perfectly predict the behavior of $\tilde{M}$ from that of M'.

Theorem 4.2: Let M realize $\tilde{M}$ under $(\sigma_1, \sigma_2, \sigma_3)$. Let $[M', C]$ denote a detector for M constructed from M' and C as shown in Fig. 4.2. There exists σ_3' such that M' realizes $\tilde{M}$ under $(\sigma_1, \sigma_2, \sigma_3')$ if and only if there exists C such that (M, U) is $([M', C], 0)$-1-diagnosable. Similarly there exists σ_3' such that M' realizes M under (e, e, σ_3') if and only if there exists a C such that (M, U) is $([M', C], 0)$-2-diagnosable.

Proof: (Necessity) Assume that M' realizes $\tilde{M}$ under $(\sigma_1, \sigma_2, \sigma_3')$. Then $\sigma_3' \circ \beta'_{\sigma_2(\tilde{r})} \circ \sigma_1 = \tilde{\beta}_{\tilde{r}}$ for all $\tilde{r} \in \tilde{R}$. Since M realizes $\tilde{M}$ under $(\sigma_1, \sigma_2, \sigma_3)$, $\sigma_3 \circ \beta_{\sigma_2(\tilde{r})} \circ \sigma_1 = \tilde{\beta}_{\tilde{r}}$ for all $\tilde{r} \in \tilde{R}$. Hence $\sigma_3' \circ \beta'_{\sigma_2(\tilde{r})} \circ \sigma_1 = \sigma_3 \circ \beta_{\sigma_2(\tilde{r})} \circ \sigma_1$. Recall that σ_1 and σ_2 are assumed to be onto. Because of this assumption, it follows that $\sigma_3' \circ \beta'_r = \sigma_3 \circ \beta_r$ for all $r \in R$. Let C be the comparator shown in Fig. 4.3.

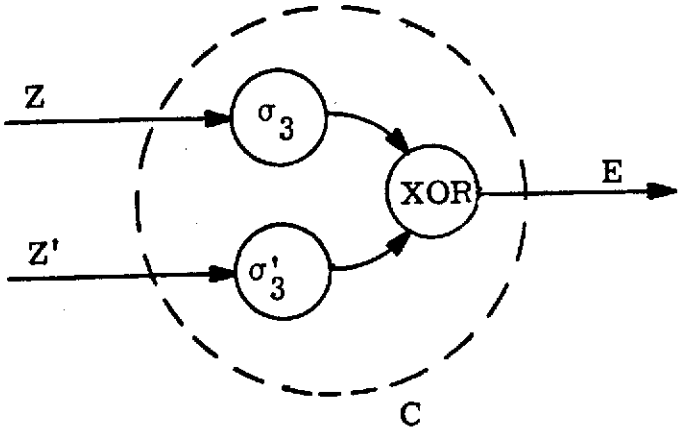


Fig. 4.3. The Comparator Used in the Proof of Theorem 4.2

Since $\sigma'_3 \circ \beta'_r = \sigma_3 \circ \beta_r$ the detector $[M', C]$ will give no false alarms. Let (r, x, y) be a minimal 1-error caused by $f \in U$. Then $\sigma_3(\beta_r(x)) \neq \sigma_3(\beta_r^f(x))$. Hence, $\sigma'_3(\beta'_r(x)) \neq \sigma_3(\beta_r^f(x))$, and this will cause the Exclusive-OR gate to emit a 1. Therefore the minimal 1-error (r, x, y) is detected with no delay. Hence (M, U) is $([M', C], 0)$-1-diagnosable.

Similarly, if M' realizes M under (e, e, σ'_3) then $\beta_r = \sigma'_3 \circ \beta'_r$ and a comparator as shown in Fig. 4.3, but without the σ_3 function, can be used to achieve $([M', C], 0)$-2-diagnosis of (M, U).

(Sufficiency) Assume that (M, U) is $([M', C], 0)$-1-diagnosable. To prove that there exists a σ'_3 such that M' realizes $\tilde{M}$ under $(\sigma_1, \sigma_2, \sigma'_3)$ we must exhibit a function σ'_3 and show that $\sigma_3 \circ \beta_r = \sigma'_3 \circ \beta'_r$. This is sufficient because M realizes $\tilde{M}$ under $(\sigma_1, \sigma_2, \sigma_3)$

and σ_1 and σ_2 are assumed to be onto.

Since no false alarms may occur we know that $C(\beta_r(x), \beta'_r(x)) = 0$ for all $r \in R$ and $x \in I^+$. Define σ'_3 as follows: $\sigma'_3(\beta'_r(x)) = \sigma_3(\beta_r(x))$. Since σ'_3 has the desired property we must simply verify that it is indeed a function.

It is clear that every $z \in \{\beta'_r(x) \mid r \in R,\ x \in I^+\}$ has an image under σ'_3. To see that this image is unique suppose that $\beta'_r(x) = \beta'_s(y)$. We must show that $\sigma_3(\beta_r(x)) = \sigma_3(\beta_s(y))$. Let $\beta'_r(x) = a$, $\sigma_3(\beta_r(x)) = b$, and $\sigma_3(\beta_s(y)) = c$. Then $C(b, a) = C(c, a) = 0$. Assume to the contrary that $b \neq c$. Let $f \in U$ be a fault which causes the output of M to be c at time $|x| - 1$ and which has no other affect. Let $x = uv$ where $v \in I$. Then $(r, x, \hat{\beta}_r(u)c)$ is a minimal 1-error and since $C(c, a) = 0$, it is not detected when it occurs. This contradicts the assumption that (M, U) is $([M', C], 0)$-1-diagnosable. Hence σ'_3 is a function and M' realizes $\tilde{M}$ under $(\sigma_1, \sigma_2, \sigma'_3)$.

The proof that (M, U) is $([M', C], 0)$-2-diagnosable implies that there exists a function σ'_3 such that M' realizes M under (e, e, σ'_3) is essentially the same as the above proof.

From Theorem 4.2 we know that if M realizes M' and M' is reduced and reachable then $|Q| \geq |Q'|$. Hence Theorem 4.2 tells us that if we use the scheme shown in Fig. 4.2 for the diagnosis of unrestricted faults then we must have $|Q'| \geq |\tilde{Q}|$ in order to achieve 1-diagnosis, and $|Q'| \geq |Q_R|$ in order to achieve 2-diagnosis, where M_R is the reduction of M.

4.3 Diagnosis with Zero Delay

The question answered next is whether it is possible to achieve (D, 0)-1-diagnosis of (M, U) with a detector which is less complex, in terms of state set size, than the reduced and reachable specification $\tilde{M}$. One reason to believe that this may be possible is the observation that if $\tilde{M}$ has an inverse then this inverse may have fewer states than $\tilde{M}$, and yet a detector constructed using this inverse may be capable of diagnosing all of U. Examples of such inverses are given in the following chapter.

Theorem 4.3: If (M, U) is (D, 0)-1-diagnosable then $|Q_D| \geq |\tilde{Q}|$.

Proof: Let (M, U) be (D, 0)-1-diagnosable, and assume, to the contrary, that $|Q_D| < |\tilde{Q}|$. Without loss of generality, assume that M is reachable.

Claim: There exists $q, q' \in Q$ and $s \in Q_D$ such that $(q, s), (q', s) \in P^*$, the reachable part of M * D, and $\sigma_3 \circ \beta_q \neq \sigma_3 \circ \beta_{q'}$.

Let $g: Q \rightarrow \mathcal{P}(Q_D) - \phi$ (where $\mathcal{P}(Q_D) = \{X \mid X \subseteq Q_D\}$) be defined by $g(q) = \{s \mid (q, s) \in P^*\}$. Assume that the claim is not true. Then $\sigma_3 \circ \beta_q \neq \sigma_3 \circ \beta_{q'}$ implies $g(q) \cap g(q') = \phi$. We know from the proof of Theorem A.2 that for each $\tilde{q} \in \tilde{Q}$ there is a state $f(\tilde{q})$ for which $\tilde{\beta}_{\tilde{q}} = \sigma_3 \circ \beta_{f(\tilde{q})} \circ \sigma_1$ and that f is necessarily 1-1. Since $\tilde{M}$ is reduced and reachable there must exist $|\tilde{Q}| = \ell$ unique states $\{q_1, \ldots, q_\ell\} \subseteq Q$ such that $i \neq j$ implies $g(q_i) \cap g(q_j) = \phi$,

and therefore $|Q_D| \geq |\tilde{Q}|$. Contradiction. This establishes the claim.

Let $q, q' \in Q$ and $s \in Q_D$ such that $(q, s), (q', s) \in P^*$ and $\sigma_3 \circ \beta_q \neq \sigma_3 \circ \beta_{q'}$. Then there exists a sequence ua where $u \in I^*$ and $a \in I$ such that $\sigma_3(\beta_q(ua)) \neq \sigma_3(\beta_{q'}(ua))$ and if $u \neq \Lambda$ then $\sigma_3(\hat{\beta}_q(u)) = \sigma_3(\hat{\beta}_{q'}(u))$. Since $(q, s) \in P^*$, there exists $r \in R$ and $y \in I^*$ such that $\delta^*(\rho^*(r), y) = (q, s)$.

Recall that given any $r \in R$, $x \in I^+$ and $y \in Z^+$ with $|x| = |y|$, there is a fault $f \in U$ such that $\hat{\beta}^f_r(x) = y$. Let $f \in U$ be a fault for which $\hat{\beta}^f_r(yua) = \hat{\beta}_r(y)\hat{\beta}_{q'}(ua)$. Since it is known that $\sigma_3(\hat{\beta}_q(u)) = \sigma_3(\hat{\beta}_{q'}(u))$, it follows that $(r, yua, \hat{\beta}^f_r(yua))$ is a minimal 1-error. Now (M, U) is $(D, 0)$-1-diagnosable implies $\hat{\beta}^D_r([\hat{\beta}^f_r(yua), yua]) \neq 0^{|yua|}$. Since no false alarms may occur, $\hat{\beta}^D_r([\hat{\beta}_r(y), y]) = 0^{|y|}$. Also, since $(q', s) \in P^*$, $\hat{\beta}^D_s([\hat{\beta}_{q'}(ua), ua]) = 0^{|ua|}$. Thus

$$\begin{aligned}
\hat{\beta}^D_r([\hat{\beta}^f_r(yua), yua]) &= \hat{\beta}^D_r([\hat{\beta}_r(y)\hat{\beta}_{q'}(ua), yua]) \\
&= \hat{\beta}^D_r([\hat{\beta}_r(y), y])\hat{\beta}^D_s([\hat{\beta}_{q'}(ua), ua]) \\
&= 0^{|y|}0^{|ua|} \\
&= 0^{|yua|}
\end{aligned}$$

This contradicts the assumption that (M, U) is $(D, 0)$-1-diagnosable. Therefore $|Q_D| \geq |\tilde{Q}|$.

Corollary 4.3.1: If (M, U) is (D, 0)-2-diagnosable then $|Q_D| \geq |Q_R|$, where M_R is the reduction of M.

Proof: Assume that (M, U) is (D, 0)-2-diagnosable, and consider M to be realizing M_R. By Theorem 3.2, (M, U) is (D, 0)-1-diagnosable, and hence, by Theorem 4.3, $|Q_D| \geq |Q_R|$.

Let us now consider the set of faults of M which are caused by the output of M becoming stuck-at-v, where $v \in Z$, at some time τ. More formally, the set of permanent output faults of M is the set

$$F_o = \{f = (M', \tau, e) \mid M' = (I, Q, Z, \delta, \lambda', R, \rho) \text{ where } \lambda'(q, a) = \lambda'(s, b) \text{ for all } q, s \in Q \text{ and } a, b \in I\}$$

Because the set of permanent faults causes the same minimal 2-errors as the set of unrestricted faults, if (M, F_o) is (D, 0)-2-diagnosable then (M, U) is (D, 0)-2-diagnosable. However, U and F_o do not cause the same minimal 1-errors, and in fact, (M, F_o) is (D, 0)-1-diagnosable does not imply that (M, U) is (D, 0)-1-diagnosable. These statements are proved in the following result.

Theorem 4.4: (M, F_o) is (D, 0)-2-diagnosable if and only if (M, U) is (D, 0)-2-diagnosable. However, (M, F_o) is (D, 0)-1-diagnosable does not imply that (M, U) is (D, 0)-1-diagnosable.

Proof: Let (M, F_o) be $(D, 0)$-2-diagnosable. Let (r, ya, w), where $a \in I$, be a minimal 2-error which is caused by $f \in U$. To show that (M, U) is $(D, 0)$-2-diagnosable it suffices to show that $\beta_r^D([\hat{\beta}_r^f(ya), ya]) \neq 0$. Since (r, ya, w) is a minimal error, $\hat{\beta}_r(y) = \hat{\beta}_r^f(y)$ and $\beta_r(ya) \neq \beta_r^f(ya)$. Say $\beta_r^f(ya) = b$, and consider the fault $f' \in F_o$ which is caused by the output of M becoming stuck-at-b at time $|y|$. Then $\hat{\beta}_r^f(ya) = \hat{\beta}_r^{f'}(ya)$, and f' also causes the minimal 2-error (r, ya, w). Since (M, F_o) is $(D, 0)$-2-diagnosable we know that $\beta_r^D([\hat{\beta}_r^{f'}(ya), ya]) \neq 0$. Hence $\beta_r^D([\hat{\beta}_r^f(ya), ya]) \neq 0$ and (M, U) is $(D, 0)$-2-diagnosable.

Now assume that (M, U) is $(D, 0)$-diagnosable. Since $F_o \subseteq U$, it follows immediately that (M, F_o) is $(D, 0)$-diagnosable.

We prove that (M, F_o) is $(D, 0)$-1-diagnosable does not imply (M, U) is $(D, 0)$-1-diagnosable by supplying a counter-example. Let $\tilde{M}_1$, M_1, D_1, and σ_3: $Z \rightarrow \tilde{Z}$ be specified by the tables in Fig. 4.4. Then $\tilde{M}_1$ is reduced and reachable, and M_1 realizes $\tilde{M}_1$ under (e, e, σ_3).

$\tilde{M}_1$:

Q_1 \ $\tilde{I}_1$	0	1	R
a	b/2	c/3	r
b	d/0	d/0	
c	e/0	e/0	
d	d/2	a/3	
e	e/3	a/2	

M_1:

Q_1 \ I_1	0	1	R
a	b/2	c/3	r
b	d/0	d/0	
c	e/1	e/1	
d	d/2	a/3	
e	e/3	a/2	

D_1:

Q_{D_1} \ I_{D_1}	0,0	0,1	1,0	1,1	2,0	2,1	3,0	3,1	R
p	q/1	q/1	q/1	q/1	q/0	q/1	q/1	q/0	r
q	s/0	s/0	t/0	t/0	t/1	t/1	t/1	t/1	
s	s/1	s/1	s/1	s/1	s/0	s/1	s/1	p/0	
t	t/1	t/1	t/1	t/1	t/1	p/0	t/0	t/1	

z	$\sigma_3(z)$
0	0
1	0
2	2
3	3

Fig. 4.4. Machines $\tilde{M}_1$, M_1, and D_1 and σ_3: $Z \rightarrow \tilde{Z}$

Since $|Q_{D_1}| < |\tilde{Q}_1|$ we know from Theorem 4.3 that (M_1, U) is not $(D_1, 0)$-1-diagnosable. To see that (M_1, F_o) is $(D, 0)$-1-diagnosable takes a bit of analysis. Briefly, states p, s, and t duplicate states a, d and e and any error which occurs when M_1 is in one of these states is immediately detected. If M_1 is in b or c then D_1 will be in q and if the output becomes stuck-at 2 or 3 at this time it will be immediately detected. If M_1 is in b or c and a stuck-at-0 or stuck-at-1 fault occurs then it will be tolerated for one time step and detected the next. This establishes the result.

In the above counter-example it is clear that (M_1, F_o) is not $(D_1, 0)$-2-diagnosable because a stuck-at-1 fault which occurs when M_1 is in b causes a 2-error which is not immediately detected. Therefore this example also proves that, in general, (M, F) is (D, k)-1-diagnosable does not imply that (M, F) is (D, k)-2-diagnosable. Also, if (M, F_o) was $(D, 0)$-2-diagnosable for some D then by Theorem 4.4 (M, U) would be (D, 0)-2-diagnosable and from Theorem 4.3 it would follow that $|Q_D| \geq |\tilde{Q}|$. Hence this is also an example of how 1-diagnosis may be achieved with a detector which is less complex than the least complex detector which is sufficient for 2-diagnosis.

4.4 Diagnosis with Nonzero Delay

Suppose now that we allow some arbitrary, but fixed, $k > 0$ in the detection process. Can this additional time be traded off for less detector complexity? Unfortunately, for the unrestricted case, the answer is no. In fact, if (M, U) is (D', k)-1-diagnosable then we can construct a detector D, essentially by eliminating unnecessary states of D', such that (M, U) is $(D, 0)$-1-diagnosable.

Before stating this result formally, we will establish an important lemma.

Lemma 4.1: If (M, U) is (D', k)-1-diagnosable then there exists a detector D such that $|Q_D| \leq |Q_{D'}|$, (M, U) is (D, k)-1-diagnosable, and for each $q \in Q_D$, $\lambda_D(q, (z, a)) = 0$ for some $(z, a) \in Z \times I$.

Proof: Assume that (M, U) is (D', k)-1-diagnosable and construct D from D' as follows:

1) Delete from the state table of D' any row corresponding to a state q for which

$$0 \notin \{\lambda_{D'}(q, (z, a)) \mid (z, a) \in Z \times I\} .$$

2) In the resulting table, replace every reference to the deleted state with a reference to an arbitrary remaining state, and set the corresponding output to 1.

3) Repeat steps 1) and 2) until no further deletions are possible.

Since $|Q_{D'}| < \infty$ the above algorithm will terminate in a finite number of iterations.

From the nature of the above construction it is clear that $|Q_D| \leq |Q_{D'}|$ and for each $q \in Q_{D'}$, $\lambda_D(q, (z, a)) = 0$ for some $(z, a) \in Z \times I$. It only remains to be shown that (M, U) is (D, k)-1-diagnosable.

If the detector D' is in a state q for which $0 \notin \{\lambda_{D'}(q, (z, a)) \mid (z, a) \in Z \times I\}$, then an error must have occurred because if D' is in q then an error detection signal will be emitted regardless of the input to D'. Hence this error could be signaled whenever a transition to q is indicated, and there would be no loss in diagnosis and no possibility for a false alarm. Since all minimal errors which q signaled would then be signaled before D' got to state q , q could be eliminated. This is the essence of what is accomplished in steps 1) and 2). This elimination process is necessarily iterative because step 2) may introduce new states to be deleted.

Since this construction is diagnosis preserving, (M, U) is (D, k)-1-diagnosable.

Theorem 4.5 : If (M, U) is (D', k)-1-diagnosable then there exists a detector D with $|Q_D| \leq |Q_{D'}|$ such that (M, U) is $(D, 0)$-1-diagnosable.

Proof: Assume that (M, U) is (D', k)-1-diagnosable. From Lemma 4.1 there exists a detector D such that $|Q_D| \leq |Q_{D'}|$, (M,U) is (D, k)-1-diagnosable, and for each $q \in Q_{D'}$, $\lambda_D(q, (z, a)) = 0$ for some

$(z, a) \in Z \times I$.

Claim: (M, U) is $(D, 0)$-1-diagnosable.

Assume, to the contrary, that (M, U) is not $(D, 0)$-1-diagnosable. Using induction on the delay of the diagnosis, we will deduce that (M, U) is not (D, m)-1-diagnosable for all $m \geq 0$. This will establish the result for it contradicts the hypothesis that (M, U) is (D, k)-1-diagnosable.

Having assumed that the basis step for our induction is true, we assume that (M, U) is not (D, m)-1-diagnosable for some $m \geq 0$, and we must show that this implies (M, U) is not $(D, m+1)$-1-diagnosable.

Since (M, U) is not (D, m)-1-diagnosable, there exists a minimal 1-error (r, x, y) caused by $f \in U$ and a sequence $v \in I^+$ with $|v| = m$ such that $\hat{\beta}_r^D([\hat{\beta}_r^f(xv), xv]) = 0^{|xv|}$. Let $\delta_D(\rho_D(r), [\hat{\beta}_r^f(xv), xv]) = s$. Let $(z, a) \in Z \times I$ such that $\lambda_D(s, (z, a)) = 0$. By Lemma 4.1 we know that such a (z, a) exists. Let f' be a fault for which $\hat{\beta}_r^{f'}(xva) = \hat{\beta}_r^f(xv)z$. Then $(r, x, \hat{\beta}_r^{f'}(x))$ is a minimal 1-error but $\hat{\beta}_r^D([\hat{\beta}_r^{f'}(xva), xva]) = 0^{|xva|}$. Hence (M, U) is not $(D, m+1)$-1-diagnosable. Therefore, (M, U) is not $(D, 0)$-1-diagnosable implies (M, U) is not (D, m)-1-diagnosable for all $m \geq 0$.

But we know that (M, U) is (D, k)-1-diagnosable. Hence (M, U) is $(D, 0)$-1-diagnosable. This establishes the result.

Corollary 4.5.1: If (M, U) is (D, k)-1-diagnosable then $|Q_D| \geq |\tilde{Q}|$.

Proof: This is an immediate consequence of Theorem 4.5 and Theorem 4.3.

Corollary 4.5.2: If (M, U) is (D, k)-2-diagnosable then $|Q_D| \geq |Q_R|$, where M_R is the reduction of M.

Proof: Assume that (M, U) is (D, k)-2-diagnosable, and consider M to be realizing M_R. From Theorem 3.2, it follows that (M, U) is (D, k)-1-diagnosable. The result now follows immediately from Corollary 4.5.1.

Although Corollaries 4.5.1 and 4.5.2 are results of a negative nature, i.e., they tell what is not possible, in conjunction with what we know is possible with duplication they tell us much about the diagnosis of unrestricted faults. They say that regardless of the specific machine under consideration, the diagnosis scheme used, and the delay allowed, any detector which can diagnose the unrestricted faults of a given machine must be essentially as complex as that machine. In particular, with regard to state set size as our measure of complexity, it is impossible to improve upon duplication. This provides an answer to Question II, page 11. These results also answer Question III; namely, for unrestricted faults no space-time tradeoff is possible, i.e., greater allowable delays in diagnosis cannot be traded off for lessened detector complexity.

We know from Theorem 4.4 and Corollary 4.3.1 that (M, F_o) is $(D, 0)$-2-diagnosable implies $|Q_D| \geq |Q_R|$. Can this result be generalized as was done for unrestricted faults by the previous corollary? The following example shows that the answer is no. This example serves as a good example of when a space-time trade-off is possible.

Example 4.1: Consider machines M_2 and D_2 of Fig. 4.5. Since M_2 is reduced and reachable, $|Q_2| = |Q_{2_R}|$, where M_{2_R} is the reduction of M_2.

M_2:

Q_2 \ I_2	0	1	R
a	b/0	c/0	r
b	a/2	d/2	
c	d/2	a/2	
d	e/0	c/0	
e	d/1	a/1	

D_2:

Q_{D_2} \ I_{D_2}	0	1	2	R
p	p/1	s/0	t/0	
s	p/0	s/1	t/0	r
t	p/0	s/0	t/1	

Fig. 4.5. Machines M_2 and D_2

Note that no output symbol can appear next to itself in any output sequence produced by M_2. Since D_2 will produce an error detection signal precisely when two consecutive inputs to it are identical, it can detect all permanent output faults of M_2 with a delay of at most one. Therefore (M_2, F_o) is $(D_2, 1)$-2-diagnosable, yet $|Q_{2_R}| > |Q_{D_2}|$.

CHAPTER V

Diagnosis Using Inverse Machines

It is well known that many circuits can be diagnosed by what is commonly called a "loop check." This involves regenerating the input to the circuit from the output and then comparing the regnerated input with the actual input. Often the "inverse" circuit is easier to implement than the original circuit, thus providing a savings over duplication. For example, division can be checked using multiplication. It is also possible to have greater confidence in a loop check than in duplication, especially if the checking circuit is less complex than the original circuit.

In this chapter we will investigate the use of "inverse machines" for diagnosis using a loop check. Informally, machine $\overline{M}$ is an inverse of machine M if $\overline{M}$ can reconstruct the input to M from its output with at most a finite delay.

Machines which have inverses can be characterized as being those machines which are "information lossless." Information lossless machines are machines whose behavior functions satisfy a condition which is similar to, but weaker than, the condition which a 1-1 function must satisfy.

Information lossless machines and inverse machines were first introduced by Huffman [18]. Huffman devised a test for information losslessness and for the existence of inverses. It should be pointed

out that our definitions of these notions are slightly less general than Huffman's. The definitions in this paper are directed towards the use of inverse machines for diagnosis. Even [13] later devised a better means of determining information losslessness, and he presented two means for obtaining inverse machines.

Information lossless machines and inverse machines are also discussed in textbooks by Kohavi [21] and Hennie [17]. Kohavi provides a fuller description of Even's techniques for obtaining inverse machines, and Hennie describes a different means of obtaining inverse machines.

The questions about the use of inverse machines for diagnosis which we seek to answer in this chapter are: When can an inverse be used for the diagnosis of unrestricted faults? Given a machine M and an inverse $\overline{M}$ of M, what will be the delay in diagnosis if $\overline{M}$ is used to diagnose M using a loop check? How can an arbitrary machine be realized so that unrestricted fault diagnosis is possible using a loop check?

We concentrate on unrestricted fault diagnosis in this chapter because this is the most natural and important fault class which can be diagnosed using a loop check. Inverse machines can be used for the diagnosis of more restricted sets of faults but synthesis and analysis for more general levels of diagnosis seem to be very difficult.

5.1 Inverses of Machines

Before the inverse of a machine can be formally defined, one preliminary notion must be introduced.

Definition 5.1: An (I, n)-delay machine (delay machine) is a machine $M^n = (I, I^n, I, \delta, \lambda, R, \rho)$ such that if $a_i \in I$, $1 \leq i \leq n+1$, then

$$\delta((a_1, \ldots, a_n), a_{n+1}) = (a_2, \ldots, a_{n+1})$$

and

$$\lambda((a_1, \ldots, a_n), a_{n+1}) = a_1 .$$

An (I, n)-delay machine simply delays its input for n time steps. Stated more precisely, if M^n is an (I, n)-delay machine then

$$\beta^n_{(a_1, \ldots, a_n)}(a_{n+1} \cdots a_{n+m}) = a_m .$$

Definition 5.2: Let M and $\overline{M}$ be two machines such that $R = \overline{R}$ and $Z = \overline{I}$. $\overline{M}$ is an (n-delayed) inverse of M if there exists an (I, n)-delay machine M^n with reset alphabet R such that for all $r \in R$ and $x \in I^+$

$$\overline{\beta}_r(\hat{\beta}_r(x)) = \beta^n_r(x) .$$

Note that if $\overline{M}$ is an inverse of M then $I \subseteq \overline{Z}$. However, it is not necessary to have $I = \overline{Z}$. Symbols which are in $\overline{Z}$ but not in I can be useful for diagnosis. Since they will never appear while $\overline{M}$ is receiving its input from M, the appearance of one immediately

signifies that an error has occurred.

$\overline{M}$ might more properly have been dubbed a "right inverse" of M for if $\overline{M}$ is an inverse of M it is not necessarily true that M is an inverse of $\overline{M}$. This is illustrated in Example 5.1. This example is a counter-example to the claims of Kohavi [21] and Even [13] that if $\overline{M}$ is an inverse of M then M is an inverse of $\overline{M}$.

Example 5.1: Consider machines M_1 and $\overline{M}_1$ of Fig. 5.1. $\overline{M}_1$ is a 0-delayed inverse of M_1 but M_1 is not an inverse of $\overline{M}_1$.

M_1:

Q_1 \ I_1	0	1	R
a	b/0	d/3	
b	c/1	a/0	
c	d/2	b/1	
d	a/3	c/2	

$\overline{M}_1$:

$\overline{Q}_1$ \ $\overline{I}_1$	0	1	2	3	R
p	q/0	q/1	q/0	q/1	r
q	p/1	p/0	p/1	p/0	

Fig. 5.1. Machines M_1 and $\overline{M}_1$

In fact, there is no machine which is an inverse of $\overline{M}_1$. This is because the input symbols 0 and 2 are equivalent and so there is no way in which they can be distinguished once they have been applied.

Intuitively, machines which have inverses lose no information as they transform sequences from I^+ into sequences from Z^+. This intuitive notion is captured in the following definition.

Definition 5.3: A machine M is information lossless of delay n if for all $r \in R$ and $a_1a_2\ldots a_m$, $b_1b_2\ldots b_m \in I^+$ $(a_i, b_i \in I,\ 1 \leq i \leq m)$

$$\hat{\beta}_r(a_1a_2\ldots a_m) = \hat{\beta}_r(b_1b_2\ldots b_m)$$

implies $a_i = b_i$ for $1 \leq i \leq m-n$.

M is said to be lossless if it is information lossless of delay n for some nonnegative integer n. M is lossy if it is not lossless.

Example 5.2: Machine M_1 of Fig. 5.1 is information lossless of delay 0 and machine $\overline{M}_1$ of Fig. 5.1 is lossy.

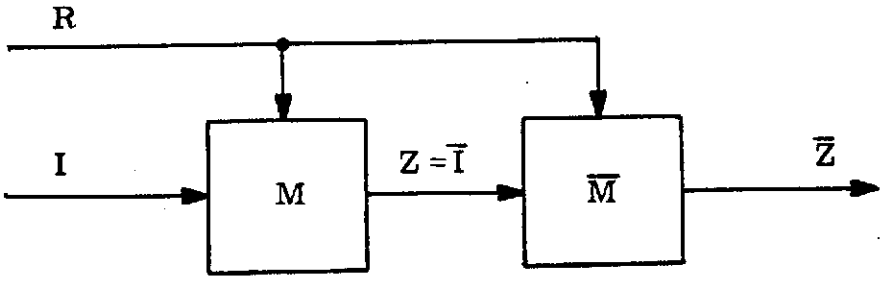


Fig. 5.2. Machine M in Series with an Inverse $\overline{M}$ of M

Referring to Fig. 5.2, if M is lossless and $\overline{M}$ is an inverse of M then intuitively no information is lost as sequences from I^+ are transformed into sequences from Z^+ by M. The same is true for

the entire process which consists of transforming sequences from I^+ into sequences from Z^+ and then back again. Therefore it is somewhat surprising to see, as we have in Example 5.2, that $\overline{M}$ may be lossy. This may occur because while $\overline{M}$ must lose no information in transforming the sequences it observes at the output of M, M may not be capable of producing all possible output sequences. Thus while $\overline{M}$ must be lossless with respect to a subset of Z^+ it may be lossy with respect to all of Z^+.

Even [13] gives an algorithm for determining if a given machine is lossless, and if so, of what delay. It is particularly easy to determine whether a given machine is lossless of delay 0. This is because a machine M is lossless of delay 0 if and only if the output symbols in every row which corresponds to a reachable state are all distinct.

Machines for which inverse machines exist can be characterized as being precisely those machines which are lossless. More precisely,

Theorem 5.1: M has a n-delayed inverse if and only if M is information lossless of delay n.

Proof: (Necessity) Assume that $\overline{M}$ is a n-delayed inverse of M. Let $r \in R$ and $a_1 \ldots a_m$, $b_1 \ldots b_m \in I^+$ ($a_i, b_i \in I$, $1 \leq i \leq m$) such that $\hat{\beta}_r(a_1 \ldots a_m) = \hat{\beta}_r(b_1 \ldots b_m)$. We must show that $a_i = b_i$ for all i, $1 \leq i \leq m-n$.

Since $\overline{M}$ is a n-delayed inverse of M there exists an (I, n)-delay machine M^n such that $\overline{\beta}_r \circ \hat{\beta}_r = \beta_r^n$. In particular, $\overline{\beta}_r(\hat{\beta}_r(a_1 \ldots a_\ell)) = \beta_r^n(a_1 \ldots a_\ell) = a_{\ell-n}$ and $\overline{\beta}_r(\hat{\beta}_r(b_1, \ldots, b_\ell)) = \beta_r^n(b_1 \ldots b_\ell) = b_{\ell-n}$ for all ℓ, $n < \ell \leq m$.

Now $\hat{\beta}_r(a_1 \ldots a_m) = \hat{\beta}_r(b_1 \ldots b_m)$ implies $\overline{\beta}_r(\hat{\beta}_r(a_1 \ldots a_\ell)) = \overline{\beta}_r(\hat{\beta}_r(b_1 \ldots b_\ell))$ for all ℓ, $1 \leq \ell \leq m$. Therefore $a_{\ell-n} = b_{\ell-n}$ for all ℓ, $n < \ell \leq m$. That is, $a_i = b_i$ for all i, $1 \leq i \leq m-n$. Hence, M is lossless of delay n.

(Sufficiency) Given a machine M which is lossless of delay n, we can show that M has a n-delayed inverse by constructing one. Techniques for constructing inverses of lossless sequential machines can be found in Hennie [17] and Kohavi [21]. With minor modifications to insure the existence of suitable starting states, these techniques can be used to construct inverses of lossless resettable machines.

5.2 Diagnosis Using Lossless Inverses

If $\overline{M}$ is an n-delayed inverse of M then, by definition, there exists an (I, n)-delay machine M^n such that $\overline{\beta}_r \circ \hat{\beta}_r = \beta_r^n$. Diagnosis using inverses can be performed by implementing M, $\overline{M}$, and M^n and dynamically checking to see if the above relationship holds. The basic configuration for diagnosis using inverses is shown in Fig. 5.3.

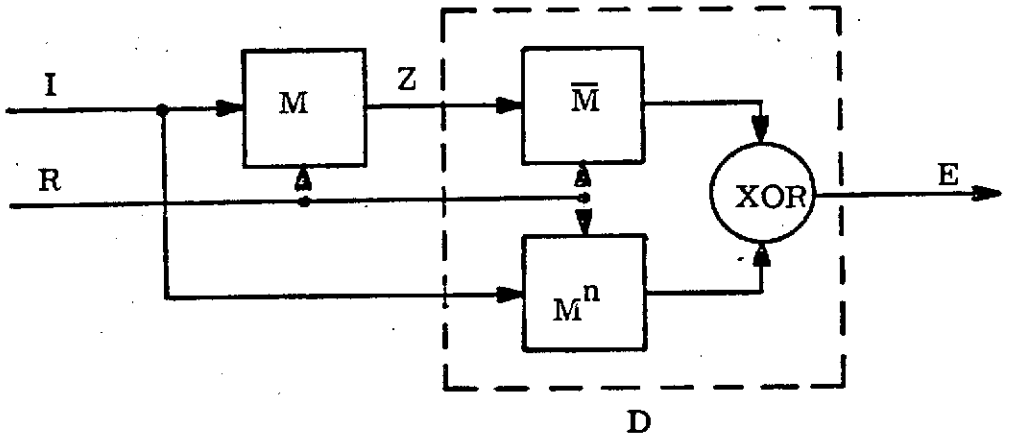


Fig. 5.3. On-line Diagnosis Using Inverse Machines

Since an (I, 0)-delay machine is simply a combinational machine which realizes the identity function on I, a detector which uses a 0-delayed inverse will have the form shown in Fig. 5.4.

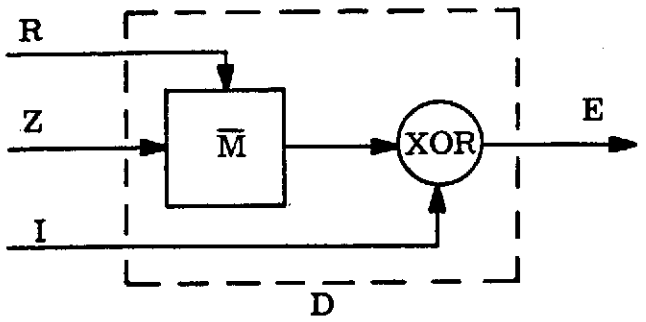


Fig. 5.4. A Detector which Uses a 0-delayed Inverse

We now state the basic result relating the use of lossless inverses with the diagnosis of unrestricted faults.

Theorem 5.2: Let M be a lossless machine and let $\overline{M}$ be an n-delayed inverse of M. Let D be constructed from $\overline{M}$, the (I, n)-delay machine which demonstrates that $\overline{M}$ is an n-delayed inverse of M, and an Exclusive-OR gate as shown in Fig. 5.3. If $\overline{M}$ is lossless of delay d then (M, U) is (D, d)-2-diagnosable.

Proof: Since $\overline{\beta}_r(\hat{\beta}_r(x)) = \beta_r^n(x)$, there will be no false alarms.

Let (r, x, w) be a minimal 2-error caused by a fault $f \in U$. Then $\beta_r^f(x) \neq \beta_r(x)$. Let $y \in I^*$ with $|y| = d$. Since $\overline{M}$ is lossless of delay d, $\hat{\overline{\beta}}_r(\hat{\beta}_r^f(xy)) \neq \hat{\overline{\beta}}_r(\hat{\beta}_r(xy))$. The Exclusive-OR gate will detect this inequality, and hence the minimal 2-error will be detected within d time steps of its occurrence. Therefore (M, U) is (D, d)-2-diagnosable.

This result gives an answer to Question V, page 12; namely, the behavioral property of "having a lossless inverse" is conducive to on-line diagnosis since the unrestricted faults of machines with this property can be diagnosed using a loop check.

It is worth noting that the delay in diagnosis is not the delay of losslessness of M but rather of its inverse $\overline{M}$. Thus an n-delayed inverse can be used to achieve diagnosis without delay if it is lossless of delay 0.

Example 5.3: Consider machines M_2 and $\overline{M}_2$ of Fig. 5.5. M_2 is lossless of delay 2 and $\overline{M}_2$ is a 2-delayed inverse of M_2. Since $\overline{M}_2$ is is lossless of delay 0 it can be used to form a detector D_2 such that (M_2, U) is $(D_2, 0)$-2-diagnosable.

M:

Q_2 \ I_2	0	1	R
a	a/0	b/0	r
b	c/0	d/0	
c	d/1	c/1	
d	b/1	a/1	

$\overline{M}$:

$\overline{Q}_2$ \ $\overline{I}_2$	0	1	R
p	p/0	q/1	r
q	t/1	s/0	
s	t/0	s/1	
t	p/1	q/0	

Fig. 5.5. Machines M_2 and $\overline{M}_2$

Example 5.6, which appears later in this chapter, shows that the converse of Theorem 5.2 does not hold. Namely, it is possible to diagnose the unrestricted fault set of a machine using an inverse which is not lossless. However, not all inverses can be used for the diagnosis of unrestricted faults. Example 5.5 shows how a lossy inverse can be useless for diagnosis. The complete characterization of inverses which can be used for unrestricted fault diagnosis is still an open problem.

Given Theorem 5.2 and the observation that an inverse machine may be lossy, an important question is whether every lossless machine has a lossless inverse. This question is presently unan-

swered. However, it can be shown that if M is lossless of delay 0 then there exists a lossless inverse of M.

The following example shows that it is possible to diagnose the unrestricted fault set of a machine using a lossless inverse which has fewer states than the reduction of the machine being diagnosed.

Example 5.4: Consider machines M_3 and $\overline{M}_3$ of Fig. 5.6. $\overline{M}_3$ is a 2-delayed inverse of M_3, and $\overline{M}_3$ is itself lossless of delay 2.

M_3:

Q_3 \ I_3	0	1	R
a	e/0	f/0	
b	a/1	b/1	
c	a/0	b/0	
d	e/1	f/1	
e	a/0	c/1	
f	d/1	b/0	

$\overline{M}_3$:

$\overline{Q}_3$ \ $\overline{I}_3$	0	1	R
p	s/0	t/1	r
q	t/0	s/1	
s	p/0	q/0	
t	s/1	t/1	

Fig. 5.6. Machines M_3 and $\overline{M}_3$

Therefore a detector D_3 can be constructed from $\overline{M}_3$ and the (I, 2)-delay machine M_3^2 of Fig. 5.7 such that (M_3, U) will be $(D_3, 2)$-2-diagnosable. Notice that M_3 is reduced and reachable and that $|Q_3| > |\overline{Q}_3|$. However, because M_3^2 is also in the detector $|Q_{D_3}| = |\overline{Q}_3||Q_3^2| = 16$. Therefore $|Q_3| < |Q_{D_3}|$. This is in keeping with what we know from Corollary 4.5.2.

Q \ I	0	1	R
00	00/0	01/0	r
01	10/0	11/0	
10	00/1	01/1	
11	10/1	11/1	

Fig. 5.7. Machine M_3^2

It is interesting to note that results established in this and the preceding chapter have something to say about lossless machines, per se. The following result gives a lower bound on the state set size of any lossless inverse of a lossless machine M. This bound is stated in terms of the input alphabet size of M, the delay of losslessness of M, and the state set size of M_R. This result, which deals only with lossless and inverse machines, is proved using Corollary 4.5.1 and Theorem 5.2, which are results dealing with the diagnosis of unrestricted faults.

Theorem 5.3: Let M be lossless of delay n, let M_R be the reduction of M, and let $\overline{M}$ be a lossless n-delayed inverse of M. Then

$$|\overline{Q}| \geq \frac{|Q_R|}{|I|^n} .$$

Proof: Consider M to be realizing its reduction M_R, and consider M and $\overline{M}$ in the configuration used for diagnosis shown in Fig. 5.3. Since $\overline{M}$ is lossless, by Theorem 5.2 (M, U) is (D, d)-2-diagnosable where d is the delay of losslessness of $\overline{M}$. Now by Corollary 4.5.1 $|Q_D| \geq$

$|Q_R|$. Since $Q_D = \bar{Q} \times I^n$, $|Q_D| = |\bar{Q}||I|^n$. Thus $|\bar{Q}||I|^n \geq |Q_R|$, or

$$|\bar{Q}| \geq \frac{|Q_R|}{|I|^n} .$$

If one has a lossless machine M of unknown delay and an inverse $\bar{M}$ of M then a lower bound on the delay n of M can be found using the following inequality:

$$n \geq \frac{\log|Q_R| - \log|\bar{Q}|}{\log|I|} .$$

This inequality was obtained directly from the one in Theorem 5.3.

Given a machine $M = (I, Q, Z, \delta, \lambda, R, \rho)$ let Z' denote the subset of Z which may actually appear in an output sequence of M. That is, let $Z' = \{\beta_r(x) | r \in R, x \in I^+\}$.

The following result gives a very simple necessary condition which all lossless machines must satisfy.

Theorem 5.4: If M is lossless then $|I| \leq |Z'|$.

Proof: Assume that M is lossless of order n. Let $f_r: I^+ \rightarrow Z^+ \times Q$ be defined by $f_r(x) = (\hat{\beta}_r(x), \delta(\rho(r), x))$.

Claim: f_r is 1-1.

Let $x, y \in I^+$ where $x \neq y$. If $|x| \neq |y|$ then $|\hat{\beta}_r(x)| \neq |\hat{\beta}_r(y)|$ and hence $f_r(x) \neq f_r(y)$. Thus it suffices to show that f_r restricted to inputs of the same length is 1-1. Let $|x| = |y|$ and assume, to the

contrary, that $f_r(x) = f_r(y)$. Then $\hat{\beta}_r(x) = \hat{\beta}_r(y)$ and $\sigma(\rho(r), x) = \delta(\rho(r), y)$. This implies that $\hat{\beta}_r(xz) = \hat{\beta}_r(yz)$ for all $z \in I^*$, and, in particular, for some z of length n. Since M is lossless of delay n this implies that $x = y$. Contradiction. Hence if $|x| = |y|$ and $x \neq y$ then $f_r(x) \neq f_r(y)$. This establishes the claim.

Since $f_r: I^+ \rightarrow Z^+ \times Q$ is 1-1 and $|x| = |\hat{\beta}_r(x)|$ it follows that $|I|^m \leq |Z'|^m |Q|$ for all $m > 0$. Hence $|I|^m / |Z'|^m |Q| \leq 1$ for all $m > 0$. Since $|Q|$ is a fixed positive integer, this implies that $|I| / |Z'| \leq 1$, or $|I| \leq |Z'|$.

This result has some immediate corollaries concerning inverses of lossless machines.

Corollary 5.4.1: Let M be a lossless machine with $|I| < |Z'|$. Then any inverse $\overline{M}$ of M with $\overline{Z}' = I$ is lossy.

Proof: Let $\overline{M}$ be an inverse of M with $\overline{Z}' = I$. Since $\overline{M}$ is an inverse of M, $Z' \subseteq \overline{I}$, and we know that $|I| < |Z'|$. Hence $|\overline{Z}'| = |I| < |Z'| \leq |\overline{I}|$. By Theorem 5.4, $\overline{M}$ must be lossy.

This corollary says that if M is lossless and $|I| < |Z'|$ then for an inverse $\overline{M}$ of M to be lossless $\overline{M}$ must have output symbols which would never appear while $\overline{M}$ is receiving its input from M. However, if a fault occurs to M and causes an error then $\overline{M}$ could emit one of these symbols. The appearance of one of these symbols in $\overline{M}$'s output would immediately cause an error detection signal

because this same symbol cannot appear in the output of an (I, n)-delay machine.

Corollary 5.4.2: Let M be a lossless machine with a lossless inverse $\overline{M}$. If $\overline{Z}' = I$ then $|I| = |Z'|$.

Proof: This follows immediately from Corollary 5.4.1.

Given the above result, an immediate question is whether M is lossless and $|I| = |Z'|$ implies that any inverse $\overline{M}$ of M is lossless. As Example 5.5 shows, the answer is no.

Example 5.5: Consider machine $\overline{M}_3'$ of Fig. 5.8. $\overline{M}_3'$ is an inverse of machine M_3 of Fig. 5.6 and $I_3 = Z_3$, but $\overline{M}_3'$ is not lossless.

$\overline{Q}_3'$ \ $\overline{I}_3'$	0	1	R
p	q/0	q/0	r
q	s/0	t/0	
s	u/0	v/1	
t	v/0	u/1	
u	s/0	t/0	
v	u/1	v/1	

Fig. 5.8. Machine $\overline{M}_3'$

5.3 Applicability of Inverses for Unrestricted Fault Diagnosis

The use of inverses as a technique for performing diagnosis applies directly only to those machines which have suitable inverses. In the following development it is shown that given an arbitrary machine M', one can always construct a realization M of M' such that M has an inverse which can be used for diagnosis. These lossless realizations are obtained simply by augmenting the output of the original machine. Thus it is shown that diagnosis using inverses is a universally applicable technique, and a part of Question I, page 11 is answered.

Definition 5.4: M is an output-augmented realization of M' if $M = (I', Q', Z' \times A, \delta', \lambda, R', \rho')$ and $\lambda = \lambda' \times \lambda_A$ for some $\lambda_A: Q' \times I' \rightarrow A$.

If M is an output-augmented realization of M' then M realizes M' under $(e, e, P_{Z'})$ where $P_{Z'}$ is the projection of $Z' \times A$ onto Z'.

Kohavi and Lavallèe [20] have given a construction which proves the following results.

Theorem 5.5: Given any machine M', there exists an output-augmented realization M of M' which is lossless of delay n for some n, and in particular, for n = 0.

Theorem 5.6: If M' is lossless of delay n, then for every m, $0 \leq m \leq n$, there exists an output-augmented realization M of M' which is lossless of delay m.

The method that Kohavi and Lavallee use to achieve the above results employs a "testing graph" which is used to determine if the given machine M' is lossless, and if so of what delay. Output augmentation which will yield the desired property is determined by a method of cutting branches in this graph. Minimal augmentation for losslessness of a desired delay is not guaranteed.

A lower bound on the amount of output-augmentation necessary to make a particular machine lossless is given by Theorem 5.4. This result tells us that for the output-augmented realization to be lossless, then the size of its output alphabet must be at least as great as the size of its input alphabet.

Any machine can be made lossless of delay 0 simply by augmenting its output with a copy of the input. This gives an upper bound on the amount of output augmentation which is necessary to make a given machine lossless of delay 0.

It is tempting to use the Kohavi and Lavallee technique to augment an inverse of a machine in the hope of achieving a lossless inverse. However, this is impossible because an output-augmented realization of an inverse $\overline{M}$ of M is not necessarily an inverse of M.

Example 5.6: Consider the configuration shown in Fig. 5.9. Here M' is any machine, and M is the output-augmented realization of M

which was formed simply by augmenting the output of M' with a copy of its input. The inverse $\overline{M}'$ of M shown in this figure is

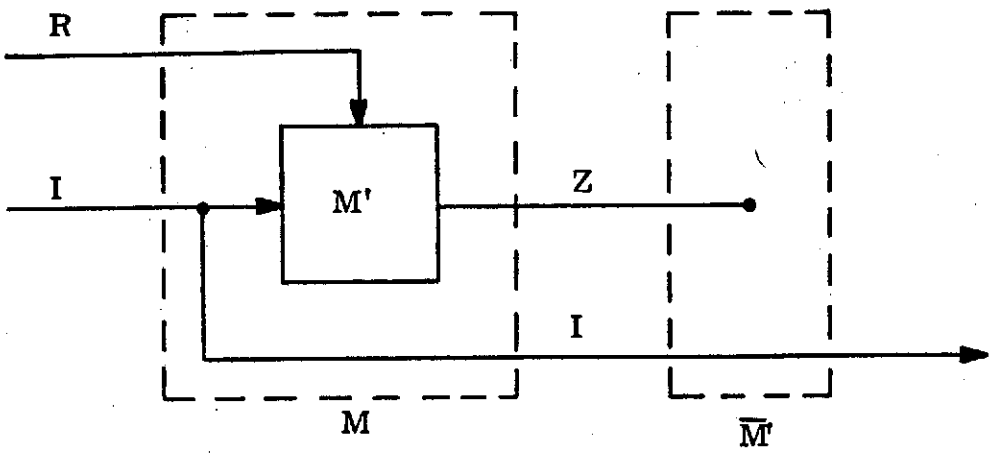


Fig. 5.9. A Lossless Machine with a Lossy Inverse

simply the combinational machine which realizes the projection of $Z \times I$ onto I. This inverse is lossy and is clearly useless for diagnosis.

Now augment the output of $\overline{M}'$ to form the machine $\overline{M}$ shown in Fig. 5.10. This machine is lossless but it is not an inverse of

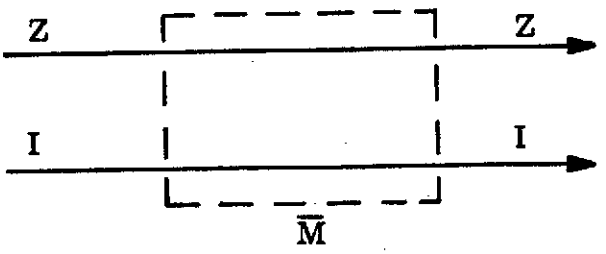


Fig. 5.10. An Output-augmented Realization of $\overline{M}'$ of Fig. 5.9

M and it too is useless for diagnosis.

Although Kohavi and Lavallee's technique cannot be used to construct lossless inverses, it is an important technique because it can be used to construct lossless of delay 0 realizations of any given machine. The following result shows that given a machine which is lossless of delay 0, an inverse of that machine can be constructed which can be used for the diagnosis of unrestricted faults.

Theorem 5.7: Let M be lossless of delay 0. Then there exists an inverse $\overline{M}$ of M such that (M, U) is (D, 0)-2-diagnosable where D is formed from $\overline{M}$ and an Exclusive-OR gate as shown in Fig. 5.4.

Proof: Let $\overline{M} = (Z, P, I \cup \{e\}, \overline{\delta}, \overline{\lambda}, R, \rho)$ where $e \notin I$ and for all $q \in P$ and $a \in Z$

$$\overline{\delta}(q, a) = \begin{cases} \delta(q, b) \text{ if } b \in I \text{ and } \lambda(q, b) = a \\ \text{arbitrary if } a \notin \lambda(q, I) \end{cases}$$

$$\overline{\lambda}(q, a) = \begin{cases} b \text{ if } b \in I \text{ and } \lambda(q, b) = a \\ e \text{ if } a \notin \lambda(q, I) \end{cases}$$

Thus $\overline{M}$ is basically the same as M but with the roles of the input and output interchanged.

The functions $\overline{\delta}$ and $\overline{\lambda}$ are well-defined for if M is lossless of delay 0 and $q \in P$ then $\lambda(q, a) = \lambda(q, b)$ implies $a = b$.

If $|I| < |Z|$ then every symbol in Z cannot appear in every row of the state table of M. This is what gives rise to the transitions of $\overline{M}$ which may be arbitrarily specified.

Consider M and $\overline{M}$ to be operating in series as shown in Fig. 5.2. Since M and $\overline{M}$ have the same reset function, they will initially be in the same state. Now if M and $\overline{M}$ are both in some state $q \in P$ and the input symbol $b \in I$ is applied to M then M will emit $\lambda(q, b)$ and go to state $\delta(q, b)$. $\overline{M}$ will emit $\overline{\lambda}(q, \lambda(q, b)) = b$ and will go to state $\overline{\delta}(q, \lambda(q, b)) = \delta(q, b)$. Thus M and $\overline{M}$ will make the same state transitions and the present output of $\overline{M}$ will always be the present input to M. Hence $\overline{M}$ is a 0-delayed inverse of M.

It remains to be shown that (M, U) is (D, 0)-2-diagnosable. This must be shown directly because $\overline{M}$ is not necessarily lossless.

Since $\overline{M}$ is a 0-delayed inverse of M there will be no false alarms. Let (r, xa, wb) where $a \in I$ and $b \in Z$ be a minimal 2-error. Since any input sequence applied to M will cause M and $\overline{M}$ to experience the same state trajectories, $\delta(\rho(r), x) = \overline{\delta}(\rho(r), w)$. Say $\delta(\rho(r), x) = q$. Since (r, xa, wb) is a minimal 2-error, $\beta_r(xa) \neq b$. Now $\overline{\lambda}(q, \beta_r(xa)) = a$ and therefore $\overline{\lambda}(q, b) \neq a$. This inequality will be detected by the Exclusive-OR gate which will emit a fault detection signal. Hence (M, U) is (D, 0)-2-diagnosable.

It should be noted that the inverse constructed in the proof of the above theorem is not necessarily lossless. By using $|Z| - |I|$ new symbols, instead of just one, $\bar{M}$ could have been constructed to be lossless of delay 0.

Example 5.7: Consider machine $\bar{M}'_1$ of Fig. 5.11. This machine is an inverse of machine M_1 of Fig. 5.1. It was constructed as described in the proof of Theorem 5.7. The transitions of $\bar{M}'_1$ which

Q_1 \ I_1	0	1	2	3	R
a	b/0	-/e	-/e	d/1	r
b	a/1	c/0	-/e	-/e	
c	-/e	b/1	d/0	-/e	
d	-/e	-/e	c/1	a/0	

Fig. 5.11. Machine $\bar{M}'_1$

may be arbitrarily chosen are indicated by a "-". This inverse of M_1 is not lossless, but it can be used for the diagnosis of unrestricted faults of M_1.

A lossless inverse $\bar{M}''_1$ of M_1 can be obtained from $\bar{M}'_1$ simply by changing one of the "e" outputs in each row of the state table of $\bar{M}'_1$ to e'. $\bar{M}''_1$ so constructed would be lossless of delay 0 because the output symbols would be distinct in every row of the state table of $\bar{M}''_1$.

CHAPTER VI

Diagnosis of Networks of Resettable Systems

This chapter considers the problem of diagnosing a machine which has been structurally decomposed and is represented as a network of resettable state machines. The networks considered here are very general and they allow for work within a wide range of structural detail.

The fault set is applied to these networks is the set of "unrestricted component faults." Informally, an unrestricted component fault is a fault which only affects one component machine but which may affect that component in an unrestricted manner. This fault set is a natural restriction of the set of unrestricted faults. We will show that it is possible to diagnose the set of unrestricted component faults of a network with relatively little redundancy.

This chapter focuses on the diagnosis of "state networks." A state network is simply a network in which the external output is the state of the network, i.e., a vector consisting of the state of each component machine in the network. Since the state of a state network is directly observable at its output, state networks are easier to diagnose than arbitrary networks.

The results in this chapter characterize state networks which are diagnosable using combinational detectors. A general construction is given which can be used to augment a given state network such that the resulting state network is diagnosable in the above sense. Upper and lower bounds on the amount of redundancy required by such an augmentation are derived.

6.1 Networks of Resettable Systems

The field of study known as "algebraic structure theory of sequential machines" is concerned with the synthesis and decomposition of sequential machines into networks of smaller component machines. Good discussions of this theory can be found in [2], [16] and [39]. The networks considered in this chapter are very similar to the "abstract networks" introduced by Hartmanis and Stearns [16]. The major differences are in our use of resettable state systems for the components and in our system connection rules which force all computation to be done in the component systems or in the external output function. Hartmanis and Stearns use sequential state machines for their components and they allow for a combinational function f_i from $(\times Q_i) \times I$ into I_i to proceed each component.

Definition 6.1: A network of resettable systems is a 6-tuple $N = (I, R, (S_1, \ldots, S_n), (K_1, \ldots, K_n), Z, \lambda)$ where

- I is a finite nonempty set, the external input alphabet
- R is a finite nonempty set, the external reset alphabet
- $S_i = (I_i, Q_i, \delta_i, R, \rho_i)$ for each i, $1 \leq i \leq n$, is a resettable state system, a component system
- K_i for each i, $1 \leq i \leq n$, is a subset of $\{Q_1, \ldots, Q_n, I\}$, a system connection rule

Z is a finite nonempty set, the external output alphabet

$\lambda: (\underset{i=1}{\overset{n}{\times}} Q_i) \times I \times T \rightarrow Z$, the external output function

such that for each i, $1 \leq i \leq n$, if

$K_i = \{A_1, \ldots, A_\ell\}$ then $I_i = \underset{j=1}{\overset{\ell}{\times}} A_j$.

Under the intended interpretation, the system connection rule K_i specifies from which parts of the network component i receives its input. By the convention introduced in Section 2.1, if $K_i = \phi$ then I_i is any singleton set. Therefore if M_i has no connections then it is an autonomous machine.

Example 6.1: The 6-tuple described in Fig. 6.1 specifies network N_1. This network has two component machines M_1 and M_2 with state sets $\{p_1, p_2\}$ and $\{q_1, q_2\}$ respectively. M_1 is connected to the external input and the output (state) of M_2 and M_2 is connected to the external input and the output (state) of M_1. Network N_1 can be viewed pictorally as shown in Fig. 6.2.

$N_1 = (I,R,(M_1,M_2),(K_1,K_2)Z,\lambda)$

$I = Z = \{0,1\}$, $R = \{r\}$

$(K_1,K_2) = (\{Q_2,I\},\{Q_1,I\})$

M_1:

Q_1 \ I_1	$(q_1,0)$	$(q_1,1)$	$(q_2,0)$	$(q_2,1)$	R
p_1	p_1	p_1	p_1	p_2	r
p_2	p_2	p_1	p_2	p_2	

M_2:

Q_2 \ I_2	$(p_1,0)$	$(p_1,1)$	$(p_2,0)$	$(p_2,1)$	R
q_1	q_1	q_2	q_1	q_1	r
q_2	q_2	q_2	q_2	q_1	

(p,q,a)	$\lambda(p,q,a)$
$p_1\ q_1\ 0$	1
$p_1\ q_1\ 1$	0
$p_1\ q_2\ 0$	0
$p_1\ q_2\ 1$	0
$p_2\ q_1\ 0$	0
$p_2\ q_1\ 1$	1
$p_2\ q_2\ 0$	0
$p_2\ q_2\ 1$	0

Fig. 6.1. Network N_1

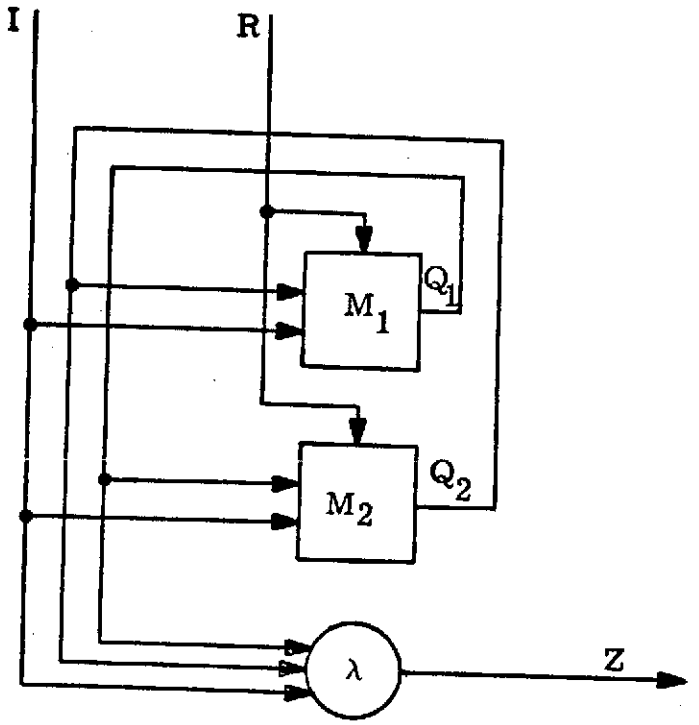


Fig. 6.2. Diagram of Network N_1

Since any machine may be viewed as a one component network a network may convey little or no structural information. On the other hand the structural description given by the network may be very detailed. For example, each component may be a two-state state machine which represents only one flip-flop and one coordinate of the global transition function.

Definition 6.2: A network $N = (I, R, (S_1, \ldots, S_n), (K_1, \ldots, K_n), Z, \lambda)$ defines the system $S_N = (I, Q, Z, \delta, \lambda, R, \rho)$ where

$$Q = \bigtimes_{i=1}^{n} Q_i$$

$$\delta(q, a, t) = \delta((q_1, \ldots, q_n), a, t)$$

$$= \bigtimes_{i=1}^{n} \delta_i[q_i, P_{K_i}(q_1, \ldots, q_n, a), t]$$

$$\rho(r, t) = \bigtimes_{i=1}^{n} \rho_i(r, t)$$

A network of resettable machines is a network in which the component systems and the external output function are all time-invariant. For example, network N_1 of Fig. 6.1 is a network of machines. The system defined by a network of machines N is also time-invariant, and it will be denoted by M_N. A network of machines N realizes a machine M if M_N realizes M. Likewise the definitions of reduced machines, reachable machines, and so forth can be extended to apply to networks of machines.

Example 6.2: Consider network N_1 of Fig. 6.1. This network defines machine M_{N_1} of Fig. 6.3 and it realizes $\tilde{M}_1$ of Fig. 6.4 because M_{N_1} realizes $\tilde{M}_1$.

M_{N_1}:

Q \ I	0	1	R
(p_1, q_1)	$(p_1, q_1)/1$	$(p_1, q_2)/0$	r
(p_1, q_2)	$(p_1, q_2)/0$	$(p_2, q_2)/0$	
(p_2, q_1)	$(p_2, q_1)/0$	$(p_1, q_1)/0$	
(p_2, q_2)	$(p_2, q_2)/0$	$(p_2, q_1)/1$	

Fig. 6.3. Machine M_{N_1}

$\tilde{M}_1$:

Q \ I	0	1	R
a	a/1	b/0	r
b	b/0	c/0	
c	c/0	d/0	
d	d/0	a/1	

Fig. 6.4. Machine $\tilde{M}_1$

A network $N = (I, R, (S_1, \ldots, S_n), (K_1, \ldots, K_n), \lambda, Z)$ is a <u>state network</u> if $Z = \underset{i=1}{\overset{n}{\times}} Q_i$ and $\lambda(q, a) = q$ for all $q \in \underset{i=1}{\overset{n}{\times}} Q_i$ and $a \in I$. If N is a state network then S_N is a state system. For state networks it is unnecessary to explicitly specify the external output alphabet and the external output function.

Since the fault set considered in this chapter does not allow for faults which affect the external output function, we will focus on

the diagnosis of state networks which realize state machines. The diagnosis of the output function will be taken care of separately, possibly by duplication.

Performing diagnosis on state networks is easier, in general, than for arbitrary networks because with state networks the output function does not mask the internal operation of the network.

Decomposing a network into a state network and an output function and then diagnosing each separately has the effect of applying a tighter tolerance relation to the diagnosis of the original network. This is also due to the lack of any masking of the state by the output function.

6.2 Unrestricted Component Faults

Suppose that N and N' are networks. Then $f = (N', \tau, \theta)$ is a fault of N if $f' = (S_{N'}, \tau, \theta)$ is a fault of S_N. Thus a fault of N can be considered to be a transformation of N into another network N' at some time τ. The notions of fault tolerance, error, and diagnosis are extended in a similar manner to apply to networks.

Given a network N, a natural set of faults to consider are those which are caused by failures in one component of N. If $f = (N', \tau, \theta)$ is caused by failures which are restricted to one component of N then N' will differ from N only in that one component. Likewise the function θ from $\times Q_i$ into $\times Q_i'$ will act as the identity on each coordinate except possibly the one affected by f. These faults are described formally in the following definition.

Definition 6.3: Let $N = (I, R, (M_1, \ldots, M_n), (K_1, \ldots, K_n), Z, \lambda)$ be a network of machines. A fault $f = (N', \tau, \theta)$ of N is an unrestricted component fault if for some j, $1 \leq j \leq n$

i) $N' = (I, R, (M_1, \ldots, S_j', \ldots, M_n), (K_1, \ldots, K_n), Z, \lambda)$ where $S_j' \in \mathcal{S}(I_j, Q_j, R)$ and

ii) for all $(q_1, \ldots, q_n) \in \overset{n}{\underset{i=1}{\times}} Q_i$, $\theta(q_1, \ldots, q_n) = (q_1', \ldots, q_n')$ implies $q_i = q_i'$ for all $i \neq j$.

The set of all unrestricted component faults of a network will be denoted by U_C.

Note that since N' is a network, S'_j is required to be a state system. Because the output alphabets of M_j and S'_j are identical and they are both state systems their state sets must also be identical. Thus, unrestricted component faults do not permit state blowup or collapse.

The fault set U_C is sufficiently restricted to make possible its diagnosis with relatively little redundancy. On the other hand, U_C is not unduly restricted for it allows for any number and type of physical failures to occur to any one component; subject, of course, to the general restrictions on faults outlined in Section 2.3. Thus using U_C as the fault class greatly reduces the amount of failure analysis which is necessary within the components.

The relationship between the set of unrestricted component faults of a network and the set of errors that these faults can cause is not as simple as the corresponding relationship for unrestricted faults. It is clear that since an unrestricted component fault can affect at most one component directly, if (r, ua, vb) is a minimal 2-error caused by $f \in U_C$ then b will be out of tolerance in only one coordinate. However, because the failed component may be connected to any other component, minimal 1-errors do not have this property. Nevertheless, a useful property of minimal 1-errors is brought out later in the proof of Theorem 6.1; namely, if (r, x, y) is a minimal 1-error of a "totally redundant" network N caused by an unrestricted component fault then under

reset r and input sequence x the faulty network will, at some time, enter an unreachable state of N.

A natural extension of U_C, the set of unrestricted component faults, would be the set of all faults caused by failures in up to m components, where m is some positive integer. Since it is very likely that any single failure which occurs will be detected before a second failure in a different component occurs, the set of unrestricted component faults is the most important special case of the more general set of faults. It is also, notationally, the easiest to discuss. For these reasons the following development is restricted to this case. However, the characterization of combinationally diagnosable networks given in the following section generalizes easily to multiple component faults. This generalization is discussed at the end of that section. The general approach to the construction of combinationally diagnosable networks used in Section 6..4 also generalizes to the multiple component fault case, although this approach is not felt to be a good approach to the more general problem.

6.3 Characterization of Combinationally Diagnosable Networks

How can state networks for which a combinational detector can diagnose the set of unrestricted component faults be characterized? In this section it is shown that this can be done in terms of the amount of redundancy in the network.

Given a state network of machines N it will be assumed that N realizes some reachable state machine $\tilde{M}$ under the triple $(\sigma_1, \sigma_2, \sigma_3)$. (Since all state machines are reduced, $\tilde{M}$ is automatically reduced.) It will be assumed, as before, that σ_1 and σ_2 are onto. The reachable part of N will be denoted by P.

Notation: Given a state network N let $C \subseteq \{1, \ldots, n\}$ denote a subset of the set of components. Let C_i denote the particular subset $\{1, \ldots, i-1, i+1, \ldots, n\}$. Let $q = (q_1, \ldots, q_n)$ and $s = (s_1, \ldots, s_n)$ be states of N.

Each C induces a partition π_C on $Q = \times Q_i$ where $q \equiv s(\pi_C)$ if and only if $q_i = s_i$ for all $i \in C$.

A cover of a set L is a set of subsets of L whose union is L. Thus every partition of L is also a cover of L. A cover J of L is a singleton cover if $B \in L$ implies $|B| \leq 1$. If J is a cover let $\#|J|$ denote the cardinality of the largest element in J.

The definition of a cover introduced here is more general than the usual notion of a cover (or "set system") as introduced by

Hartmanis and Stearns [16]. They employed set systems to obtain series-parallel decompositions. The notion introduced here is not used to obtain decompositions but rather to analyze any given decomposition.

Let $C \subseteq \{1, \ldots, n\}$ and let $\pi_C = \{B_1, \ldots, B_\ell\}$. C induces the cover

$$\tilde{C} = \{\sigma_3 (B_1 \cap P), \ldots, \sigma_3 (B_\ell \cap P)\} \text{ of } \tilde{Q}$$

where if $B \subseteq P$ then $\sigma_3 (B) = \{ \sigma_3 (q) | q \in B\}$. In particular, $\sigma_3 (\phi) = \phi$.

Each set of states which the components in C can take on corresponds directly to a block of the partition π_C. Thus π_C represents the information about the current state of N which is given by the current states of components in C. $\tilde{C}$ represents the corresponding information as to the state of $\tilde{M}$ which N is currently mimicing. If $\tilde{C}$ is a singleton cover then the current state of each component in C completely determines the corresponding state of $\tilde{M}$. Note that $\{1, \ldots, n\}$ is always a singleton cover.

Definition 6.4: Component M_i of a network N is redundant if $\tilde{C}_i$ is a singleton cover. N is totally redundant if every component of N is redundant.

"Redundant components" are essentially the same as "dependent coordinates" as discussed by Zeigler [39]. The basic difference

is that the concept of a "redundant component" is defined in terms of covers rather than partitions (as is the case with "dependent coordinates"), and hence is a more general concept which allows for state splitting.

If N is totally redundant then knowledge of the state of any n-1 components is sufficient to determine the corresponding state of $\tilde{M}$ although it may not be sufficient to determine the state of the remaining component.

Example 6.3: Consider network N_1 of Example 6.1. Let N_1' be the associated state network which is obtained from N_1 by changing the external output function and alphabet. Let $\tilde{M}_1'$ be the state machine corresponding to machine $\tilde{M}_1$ of Fig. 6.4. Then N_1' realizes $\tilde{M}_1'$ under (e, e, σ_3) where σ_3: $P_1' \rightarrow \tilde{Q}_1'$ is given by the following table:

p	q	$\sigma_3(p, q)$
p_1	q_1	a
p_1	q_2	b
p_2	q_1	d
p_2	q_2	c

Now $\pi_{C_1} = \pi_{\{2\}} = \{\overline{(p_1, q_1), (p_2, q_1)}; \overline{(p_1, q_2), (p_2, q_2)}\}$ and so

$$\tilde{C}_1 = \{\sigma_3\{(p_1, q_1), (p_2, q_1)\}, \sigma_3\{(p_1, q_2), (p_2, q_2)\}\}$$
$$= \{\{a, d\}, \{b, c\}\} .$$

Therefore $\tilde{C}_1$ is not a singleton cover, M_1 is not a redundant component, and N_1 is not totally redundant.

Lemma 6.1: Let N be a totally redundant state network of machines, and let $q = (q_1, \ldots, q_i, \ldots, q_n)$ and $q' = (q_1, \ldots, q'_i, \ldots, q_n)$ be states of N. If $q, q' \in P$ then $\sigma_3(q) = \sigma_3(q')$.

Proof: Let $q, q' \in P$. Since q and q' differ only in their i^{th} coordinate they are in the same block of π_{C_i}. Say that $\pi_{C_i} = \{B_1, \ldots, B_\ell\}$ and that $q, q' \in B_j$. Since $q, q' \in P$, $q, q' \in B_j \cap P$. Since N is totally redundant, $\tilde{C}_i$ is a singleton cover, and thus we must have $\sigma_3(q) = \sigma_3(q')$.

Suppose that an unrestricted component fault f occurs to a totally redundant network of machines N and causes a minimal 2-error (r, x, y). Say that $\beta_r(x) = q = (q_1, \ldots, q_n)$. Due to the nature of f, namely that it affects only one component, $\beta_r^f(x) = q' = (q_1, \ldots, q'_i, \ldots, q_n)$. If $q' \in P$ then Lemma 6.1 tells us that this 2-error is not a 1-error because $\sigma_3(q) = \sigma_3(q')$. If $q \notin P$ then this 2-error could be detected by a combinational detector which flags the unreachable states of N. By using the above lemma and Theorem A.2 the following characterization of combinationally diagnosable detectors can be obtained.

Theorem 6.1: Let N be a state network of machines which realizes a state machine $\tilde{M}$ under $(\sigma_1, \sigma_2, \sigma_3)$. Then (N, U_C) is (D, 0)-1-diagnosable for some combinational detector D if and only if N is totally redundant.

Proof: (Necessity) Suppose that (N, U_C) is (D, 0)-1-diagnosable where D is combinational, and let D realize the function λ_D. Assume, to the contrary, that N is not totally redundant. Then for some i, $\tilde{C}_i$ is not a singleton cover. Hence there exists $q = (q_1, \ldots, q_i, \ldots, q_n)$ and $q = (q_i, \ldots, q_i', \ldots, q_n)$ such that $q, q' \in P$ and $\sigma_3(q) \neq \sigma_3(q')$. Since $q, q' \in P$, $\lambda_D(q) = \lambda_D(q') = 0$ for otherwise a false alarm could occur. Let $f \in U_C$ be a fault caused by the output of M_i becoming stuck-at-q_i' at a time when M could be in q. This fault can cause a 1-error which is not (D, 0)-1-diagnosable. Contradiction. Therefore if (N, U_C) is (D, 0)-1-diagnosable where D is combinational then N must be totally redundant.

(Sufficiency) Assume that N is totally redundant. Let D be the detector which realizes the function $\lambda_D: Q \rightarrow \{0, 1\}$ where

$$\lambda_D(q) = \begin{cases} 0 & \text{if } q \in P \\ 1 & \text{if } q \notin P \end{cases}$$

Clearly, D will give no false alarms.

Let (r, x, y) be a minimal 1-error caused by $f \in U_C$. Let x = uab where $a, b \in I$.

Then $\sigma_3(\beta_r(ua)) = \sigma_3(\beta_r^f(ua))$ and $\sigma_3(\beta_r(uab)) \neq \sigma_3(\beta_r^f(uab))$. Say $\beta_r^f(ua) = q$. Then $\beta_r^f(uab) = \delta^f(q, a, t)$ where $t = |u|$. Because $f \in U_C$, f can affect at most one component of N. Therefore $\delta(q, a)$ will differ in at most one coordinate from $\delta^f(q, a, t)$. Let $\delta(q, a) = s = (s_1, \ldots, s_j, \ldots, s_n)$ and let $\delta^f(q, a, t) = s' = (s_1, \ldots, s_j', \ldots, s_n)$. Since $\sigma_3(q) = \sigma_3(\beta_r(ua))$ and $\beta_r(ua) = \delta(\rho(r), u)$, by Theorem A. 2 $\sigma_3(\delta(q, a)) = \sigma_3(\delta(\rho(r), ua)) = \sigma_3(\beta_r(uab))$. Thus

$$\begin{aligned} \sigma_3(s) &= \sigma_3(\beta_r(uab)) \\ &\neq \sigma_3(\beta_r^f(uab)) \\ &= \sigma_3(s') \end{aligned}$$

If $q \in P$ then $s = \delta(q, a) \in P$ and applying Lemma 6. 1 we deduce that $s' \notin P$. Therefore $\lambda_D(s') = 1$ and the 1-error (r, x, y) is detected without delay.

Alternatively, if $q \notin P$ then $\lambda_D(q) = 1$ and the 1-error (r, x, y) is detected one time step before it occurs. Since in either case the error is detected by the time of its occurrence it follows that (N, U_C) is (D, 0)-1-diagnosable.

This characterization of combinationally diagnosable networks provides an answer to Questions II and V, page 11 ; namely, totally redundant realizations are diagnosable with a combinational detector

and with zero delay, and the structural property of "total redundancy" is conducive to on-line diagnosability.

Given $C \subseteq \{1, \ldots, n\}$, let $\pi_C = \{B_1, \ldots, B_\ell\}$. Then C induces a partition $\overline{\pi}_C$ on P where $\overline{\pi}_C = \{B_1 \cap P, \ldots, B_\ell \cap P\} - \phi$.

If a partition π of a set L is a singleton cover then we will denote this by writing $\pi = 0$. This notation is derived from the observation that this partition is the least element of the lattice of all partitions of L.

Corollary 6.1.1: Let N be a state network of machines. Then (N, U_C) is (D, 0)-2-diagnosable for some combinational detector D if and only if $\overline{\pi}_{C_i} = 0$ for all i, $1 \leq i \leq n$.

Proof: Consider N to be realizing the reduction of M_N. Then σ_3 is 1-1. By Theorems 3.2 and 3.3 (N, U_C) is (D, 0)-2-diagnosable for some combinational D if and only if (N, U_C) is (D, 0)-1-diagnosable for some combinational D.

Now since σ_3 is 1-1, $\tilde{C}_i$ is a singleton cover if and only if $\overline{\pi}_{C_i} = 0$. Hence N is totally redundant if and only if $\overline{\pi}_{C_i} = 0$ for all i, $1 \leq i \leq n$.

The result now follows immediately from Theorem 6.1.

Example 6.4: Consider state network N_1' of Example 6.3. Since N_1' is not totally redundant, from Theorem 6.1 we know

that (N_1', U_C) is not $(D, 0)$-1-diagnosable for any combinational detector D.

Now construct a new network N_1'' from N_1' by adding a new component M_3 as shown in Fig. 6.5.

$$N_1' = (I, R, (M_1, M_2, M_3), (K_1, K_2, K_3))$$

I, R, M_1, M_2, K_1 and K_2 are identical to those of network N_1 of Fig. 6.2.

$$K_3 = \{I\}$$

M_3:

Q_3 \ I_3	0	1	R
s_1	s_1	s_2	r
s_2	s_2	s_1	

Fig. 6.5. Network N_1''

Network N_1'' realizes machine $\tilde{M}_1'$ of this example under (e, e, σ_3') where σ_3': $P_1'' \rightarrow \tilde{Q}_1'$ is given by the following table:

p	q	s	$\sigma_3'(p, q, s)$
p_1	q_1	s_1	a
p_1	q_1	s_2	d
p_2	q_2	s_1	c
p_2	q_2	s_2	b

For network N_1''

$$\pi_{C_1} = \pi_{\{2,3\}} = \{\overline{(p_1, q_1, s_1), (p_2, q_1, s_1)}; \overline{(p_1, q_1, s_2), (p_2 q_1, s_2)};$$

$$\overline{(p_1, q_2, s_1), (p_2, q_2, s_1)}; \overline{(p_1, q_2, s_2), (p_2, q_2, s_2)}\}$$

and $\tilde{C}_1 = \{\{a\}, \{d\}, \{c\}, \{b\}\}$. Thus $\tilde{C}_1$ is a singleton cover and component M_1 is redundant. Similarly one can show that M_2 and M_3 are redundant. Hence N_1'' is totally redundant, and (N_1'', U_C) is $(D, 0)$-1-diagnosable for some combination of detector D.

It is enlightening to consider Corollary 6. 1. 1 from the point of view of error detecting codes. Let N be a state network realization of a reachable state machine $\tilde{M}$. Then each of the reachable states of N can be viewed as a code word of an encoding of $\tilde{Q}$. Two such code words are said to be adjacent if they differ in only one coordinate. Clearly, an encoding can be used to detect all errors in single coordinates if and only if no two code words are adjacent. In addition, it is clear that two code words are adjacent

if and only if $\overline{\pi}_{C_i} \neq 0$ for some i. Thus Corollary 6.1.1 tells us that single error detecting state assignments are necessary and sufficient to insure combinational and delayless 2-diagnosis of unrestricted component faults.

The generalization of the characterization given by Theorem 6.1 and Corollary 6.1.1 to faults caused by multiple failures is straightforward. For example, if failures in two components are being considered then a totally redundant network would be one in which the corresponding state of $\tilde{M}$ could be deduced from the states of any n-2 components of N. With this altered definition the statement of Theorem 6.1 could then remain unchanged. By considering the two failed components as one larger component the proof could also remain virtually unchanged.

6.4 Construction of Combinationally Diagnosable Networks

The basic problem approached in this section is a constrained decomposition problem; namely, given a state machine $\tilde{M}$, find a totally redundant network realization N of $\tilde{M}$. From Theorem 6.1 we know that such a network would be combinationally diagnosable, and thus a solution to this problem would be an answer to Question I, page 11.

The approach to this problem taken here is to find a network realization by conventional decomposition techniques and then make this network totally redundant through the addition of one component.

Example 6.4 showed that a totally redundant network could be constructed from network N_1' through the addition of one component machine. In this section it is shown that this can be done for any network. In addition, upper and lower bounds are derived on the minimum number of states that such an additional component must have.

Theorem 6.2: Let N be a state network of machines. Let $m_i = |Q_i|$, and let $m = \max_{1 \leq i \leq n} m_i$. A network N' where N' realizes N and (N', U_C) is $(D, 0)$-2-diagnosable for some combinational detector D can be constructed from N by the addition of an m state component.

Proof: Without loss of generality take $Q_i = \{0, \ldots, m_i - 1\}$. Let $N = (I, R, (M_1, \ldots, M_n), (K_1, \ldots, K_n))$ and let $N' = (I, R, (M_1, \ldots, M_n,$

M_{n+1}), $(K_1, \ldots, K_n, K_{n+1})$) where $K_{n+1} = \{Q_1, \ldots, Q_n, I\}$ and where M_{n+1} is constructed such that for all $q = (q_1, \ldots, q_{n+1}) \in P'$, the reachable part of N', $\sum_{i=1}^{n+1} q_i \equiv 0 \pmod m$. A machine M_{n+1} with m states which satisfies the above property is described below:

$$M_{n+1} = (I_{n+1}, Q_{n+1}, \delta_{n+1}, R, \rho_{n+1})$$

where

$$I_{n+1} = \bigtimes_{i=1}^{n} Q_i \times I$$

$$Q_{n+1} = \{0, \ldots, m-1\}$$

$$\rho_{n+1}(r) \equiv -\sum_{i=1}^{n} \rho_i(r) \pmod m \text{ for all } r \in R$$

$$\delta_{n+1}(q_{n+1}, (q_1, \ldots, q_n, a)) \equiv -\sum_{i=1}^{n} q_i' \pmod m \text{ for all}$$

$q_i \in Q_i$, $1 \leq i \leq n+1$, and all $a \in I$ where

$(q_1', \ldots, q_n') = \delta((q_1, \ldots, q_n), a)$.

It is clear that N' realizes N. Therefore, it remains only to be shown that (N', U_C) is $(D, 0)$-2-diagnosable for some combinational D.

Let D be the combinational machine which realizes the function $\lambda_D: \bigtimes_{i=1}^{n+1} Q_i \rightarrow \{0, 1\}$ where

$$\lambda_D(q_1, \ldots, q_{n+1}) = \begin{cases} 0 \text{ if } \sum_{i=1}^{n+1} q_i \equiv 0 \pmod{m} \\ 1 \text{ otherwise} \end{cases}$$

Since $(q_1, \ldots, q_{n+1}) \in P'$ implies $\sum_{i=1}^{n+1} q_i \equiv 0 \pmod{m}$ no false alarms will occur.

Let (r, x, y) be a minimal 2-error caused by $f \in U_C$. Since (r, x, y) is a minimal error and f only affects one component of N', $\beta_r(x)$ and $\beta_r^f(x)$ will differ in exactly one coordinate. Say $\beta_r(x) = (q_1, \ldots, q_{n+1})$ and $\beta_r^f(x) = (q_1, \ldots, q_i', \ldots, q_{n+1})$. Now $(q_1, \ldots, q_{n+1}) \in P$ implies $\sum_{i=1}^{n+1} q_i \equiv 0 \pmod{m}$. Since $q_i \neq q_i'$ and $|Q_i| \leq m$, $q_i \not\equiv q_i' \pmod{m}$. Therefore $q_1 + \ldots + q_i' + \ldots + q_{n+1} \not\equiv 0 \pmod{m}$. Hence, the error (r, x, y) is detected without delay, and (N', U_C) is $(D, 0)$-2-diagnosable.

In the proof of Theorem 6.2 a construction is given which can be used to form a totally redundant network from any network of machines. This construction simply involves the addition of one component to N. This theorem also gives an upper bound on the amount of additional redundancy required to make a given network totally redundant. This upper bound is stated in terms of the size of the state set of the additional component.

The detector used in the proof of Theorem 6.2 simply checked to see if the states of the components always summed to 0 (mod m).

By using a different and possibly more complex detector, namely one which can determine if the present state is in the reachable part, the number of states which the additional component must have can be reduced.

Let m_i' be the number of states that M_i, $1 \leq i \leq n$, can actually enter while M_i is a component of network N, and let $m' = \max_{1 \leq i \leq n} m_i'$. That is, let $m' = \max_{1 \leq i \leq n} |P_i(P)|$, where $P_i(P)$ is the projection onto coordinate i of the reachable part of N. Then $m' \leq m$ because $P_i(P) \subseteq Q_i$, $1 \leq i \leq n$, and Theorem 6.2 holds with m replaced by m'. This claim is established in the following theorem.

Theorem 6.3: Let N be a state network of machines. Let $m_i' = |P_i(P)|$, and let $m' = \max_{1 \leq i \leq n} m_i'$. A network N' can be constructed from N by the addition of an m' state component such that N' realizes N and (N', U_C) is (D, 0)-2-diagnosable.

Proof: Without loss of generality take $P_i(P) = \{0, \ldots, m_i' - 1\}$ and $Q_i = \{0, \ldots, m_i - 1\}$. Construct N' by adding component M_{n+1} where N' and M_{n+1} are exactly as in the proof of Theorem 6.2 except for m being replaced by m'.

We will show that (N', U_C) is (D, 0)-2-diagnosable by showing that $\bar{\pi}_{C_i} = 0$ for all i, $1 \leq i \leq n$, and then appealing to Corollary 6.1.1.

Assume, to the contrary, that $\bar{\pi}_{C_i} \neq 0$ for some i, say for i = 1. Let $\pi_{C_1} = \{B_1, \ldots, B_\ell\}$. Then for some j, $1 \leq j \leq \ell$, $|B_j \cap P| > 1$. This implies the existence of two states $q = (q_1, q_2, \ldots, q_n)$ and $q' = (q_1', q_2, \ldots, q_n)$ such that q, q' ϵ P' and $q_1 \neq q_1'$. Now q, q' ϵ P' implies $q_1 + q_2 + \ldots + q_n \equiv 0 \pmod{m'}$ and $q_1' + q_2 + \ldots + q_n \equiv 0 \pmod{m'}$. Hence, $q_1 \equiv q_1' \pmod{m'}$ and since $0 \leq q_1, q_1' < m'$, $q_1 = q_1'$. Contradiction. Therefore $\bar{\pi}_{C_i} = 0$ for all i, $1 \leq i \leq n$, and the result follows immediately from Corollary 6.1.1.

A technique similar to the one used in the proof of Theorem 6.2 could be used for the diagnosis of n Mealy machines which operate in parallel with the same inputs and resets. In this case one additional Mealy machine would be required which had as many output symbols as the machine with the largest output alphabet. There is no guarantee, however, that this technique will result in a savings over duplication because the additional machine may need as many states as the product of the number of states of the original n machines.

We have shown that given a network N, a totally redundant network N' can be constructed thru the addition of a component with no more than m' states where $m' = \max |P_i(P)|$. This amount of additional redundancy is not always necessary for N may already be totally redundant. The following example shows that this amount of additional redundancy is not necessary even if no component of the network is redundant.

Example 6.5: Consider state network N_2 of Fig. 6.6.

$$N_2 = (I, R, (M_1, M_2), (K_1, K_2))$$

$$I = \{0, 1, 2, 3, 4\}, \quad R = \{r\}$$

$$(K_1, K_2) = (\{I\}, \{I\})$$

M_1:

Q_1 \ I_1	0	1	2	3	4	R
p_1	p_2	p_1	p_3	p_3	p_2	r
p_2	p_1	p_2	p_4	p_4	p_1	
p_3	p_3	p_4	p_2	p_1	p_4	
p_4	p_4	p_3	p_1	p_2	p_3	

M_2:

Q_2 \ I_2	0	1	2	3	4	R
q_1	q_1	q_1	q_3	q_4	q_2	r
q_2	q_2	q_2	q_3	q_3	q_1	
q_3	q_3	q_3	q_2	q_2	q_4	
q_4	q_4	q_4	q_2	q_1	q_3	

Fig. 6.6. Network N_2

N_2 realizes state machine $\tilde{M}_2$ of Fig. 6.7 under (e, e, σ_3) where σ_3: $P_2 \rightarrow \tilde{Q}_2$ is given by the table in Fig. 6.8.

Q \ I	0	1	2	3	4	R
a	b	a	e	h	c	r
b	a	b	f	g	d	
c	d	c	f	f	a	
d	c	d	e	e	b	
e	e	f	c	d	g	
f	f	e	d	c	h	
g	g	h	d	b	e	
h	h	g	c	a	f	

Fig. 6.7. Machine $\tilde{M}_2$

p	q	$\sigma_3(p, q)$
p_1	q_1	a
p_1	q_2	d
p_2	q_1	b
p_2	q_2	c
p_3	q_3	e
p_3	q_4	h
p_4	q_3	f
p_4	q_4	g

Fig. 6.8. $\sigma_3: P_2 \longrightarrow \tilde{Q}_2$

Since $|\tilde{Q}_2| = 8$ and $|Q_1 \times Q_2| = 16$ it should be clear that while N_2 is not totally redundant there is some redundancy in this network realization of $\tilde{M}_2$. Thus if we were to add a component M_3 to N_2 in an attempt to form a totally redundant network N_2' we should not be too surprised if we succeeded with a component M_3 with fewer than m' states, where for network N_2 $m' = 4$. In fact, if the 2-state machine $M_3 = (Q_1 \times Q_2 \times I, \{s_1, s_2\}, \delta_3)$ were added to N_2 where δ_3 is such that M_3 is in s_1 whenever M_1 and M_2 are in (p_1, q_1), (p_2, q_2), (p_3, q_3) or (p_4, q_4) and in s_2 whenever M_1 and M_2 are in (p_1, q_2), (p_2, q_1), (p_3, q_4) or (p_4, q_3) then the network N_2' so formed would be totally redundant.

An intuitively satisfying means to verify this claim is as follows. Component M_i computes the information $\tilde{C}_{\{i\}}$ about the corresponding state of $\tilde{M}$. In this case the $\tilde{C}_{\{i\}}$ are the following partitions of $\tilde{Q}_2$.

$$\tilde{C}_{\{1\}} = \{ \overline{a, d}; \overline{b, c}; \overline{e, h}; \overline{f, g} \}$$

$$\tilde{C}_{\{2\}} = \{ \overline{a, b}; \overline{c, d}; \overline{e, f}; \overline{g, h} \}$$

$$\tilde{C}_{\{3\}} = \{ \overline{a, c, e, g}; \overline{b, d, f, h} \}$$

Since $\tilde{C}_{\{1\}} \cdot \tilde{C}_{\{2\}} = \tilde{C}_{\{2\}} \cdot \tilde{C}_{\{3\}} = \tilde{C}_{\{1\}} \cdot \tilde{C}_{\{3\}} = 0$ any two components taken together provide total information as to the corresponding state of $\tilde{Q}_2$. Hence the remaining one will always be redundant.

The following result gives a lower bound on the number of states that an additional component must have in order for the resulting augmented network to be totally redundant. If the network under consideration is already totally redundant then the lower bound given by this result is one. Since the behavior of a state machine with one state is always a constant function, the actual addition of such a component is unnecessary.

Theorem 6. 4: Let N be an n component state network and let N' be the state network formed from N by the addition of a component with ℓ states. If N' is totally redundant then $\ell \geq \max_{1 \leq i \leq n} \#|\tilde{C}_i|$.

Proof: Without loss of generality take $\#|\tilde{C}_1| = \max_{1 \leq i \leq n} \#|\tilde{C}_i|$, and let $d = \#|\tilde{C}_1|$. Then for some $B \in \pi_{C_1}$ and $q = (q_1, \ldots, q_n) \in B$, $|\sigma_3(B \cap P)| = d$. That is, if it is known that M_2 is in q_2, that M_3 is in q_3, and so forth up to M_n being in q_n then there is still a d state uncertainty as to which state of $\tilde{M}$ the state of M currently corresponds. It is necessary for M_{n+1} to have at least d states to resolve this uncertainty.

The above result provides a good lower bound on the amount of additional redundancy required to form a totally redundant network, and it does so by taking into account the redundancy which already exists in the network. This level of redundancy, however, is not

always sufficient because it may be impossible to find a component with d states which will simultaneously resolve the uncertainties represented by $\tilde{C}_1, \tilde{C}_2, \ldots,$ and $\tilde{C}_n$. The following describes just such a situation.

Example 6.6: Consider the state network N_3 of Fig. 6.9.

$$N_3 = (I, R, (M_1, M_2, M_3), (K_1, K_2, K_3))$$

$$I = \{0,1,2\}, \quad R = \{r\}$$

$$(K_1, K_2, K_3) = (\{I\}, \{I\}, \{Q_1, Q_2, I\})$$

M_1:

Q_1 \ I_1	0	1	2	R
p_1	p_1	p_2	p_1	r
p_2	p_2	p_3	p_2	
p_3	p_2	p_1	p_2	

M_2:

Q_1 \ I_2	0	1	2	R
q_1	q_2	q_1	q_2	r
q_2	q_2	q_1	q_1	

M_3:

Q_3 \ I_3	$(p_1,q_1,0)$	$(p_1,q_1,1)$	$(p_1,q_1,2)$	$(p_1,q_2,0)$	$(p_1,q_2,1)$	$(p_1,q_2,2)$	$(p_2,q_1,0)$	$(p_2,q_1,1)$	$(p_2,q_1,2)$	$(p_2,q_2,0)$	$(p_2,q_2,1)$	$(p_2,q_2,2)$	$(p_3,q_1,0)$	$(p_3,q_1,1)$	$(p_3,q_1,2)$	$(p_3,q_2,0)$	$(p_3,q_2,1)$	$(p_3,q_2,2)$	R
s_1	s_1	s_2	s_2	s_1	s_2	s_1	s_1	s_1	s_2	s_1	s_1	s_2	s_1	s_1	s_1	s_1	s_1	s_1	r
s_2	s_1	s_2	s_2	s_2	s_2	s_1	s_1	s_1	s_2	s_2	s_2	s_2	s_2	s_2	s_2	s_2	s_2	s_2	

Fig. 6.9. Network N_3

This network realizes machine $\tilde{M}_3$ of Fig. 6.10.

$\tilde{M}_3$:

Q \ I	0	1	2	R
a	e	b	f	r
b	g	c	h	
c	g	c	g	
d	h	d	h	
e	e	b	a	
f	f	b	a	
g	g	c	b	
h	h	d	b	

Fig. 6.10. Machine $\tilde{M}_3$

For N_3 realizing $\tilde{M}_3$ we have

$$\tilde{C}_1 = \{ \{a,c\},\{b,d\},\{e,g\},\{f,h\} \}$$
$$\tilde{C}_2 = \{ \{a,e\},\{b,h\},\{c\},\{d\},\{f\},\{g\} \}$$
$$\tilde{C}_3 = \{ \{a\},\{b\},\{c,d\},\{e,f\},\{g,h\} \}$$

Therefore $m = \max_{1 \le i \le 3} |Q_i| = 3$ and $d = \max_{1 \le i \le 3} \#|\tilde{C}_i| = 2$.

Suppose that it is desired to add a component M_4 to N_3 in order to form a totally redundant network. Theorem 6.4 tells us that M_4 must have at least 2 states, and Theorem 6.2 tells us that there is a 3-state component which will work. We will show that in this case it is not sufficient for M_4 to have 2 states.

Let M_4 be a 2-state component which when added to N_3 forms N_3'. Let $\tilde{C}_{\{4\}} = \{B_1, B_2\}$. Since $\tilde{C}_{\{4\}}$ is a cover of $\tilde{Q}_3$, $B_1 \cup B_2 = \tilde{Q}_3$. If $|B_1| \geq 5$ or $|B_2| \geq 5$ then $\tilde{C}_1$ would not be a singleton cover because M_2 and M_3 have only 2 states each and together they could not resolve a 5-state uncertainty. Therefore if N_2' is to be totally redundant we must have $|B_1|, |B_2| \leq 4$ and thus $\tilde{C}_{\{4\}}$ will be a partition of $\tilde{Q}_3$.

For N_3' to be totally redundant M_4 must resolve the following pairs of states: $\{a, e\}, \{b, d\}, \{e, g\}, \{f, h\}, \{b, h\}, \{c, d\}, \{e, f\}$, and $\{g, h\}$. It can resolve a pair only if the pair is split between B_1 and B_2. But it is easy to verify that these eight pairs cannot all be simultaneously split by any two-block partition. Therefore there is no 2-state component which when added to N_3 will form a totally redundant network.

CHAPTER VII

Conclusions and Open Problems

In this report a fresh look at on-line diagnosis was taken from a system theoretic point of view. The approach used in this investigation was system theoretic in the sense that resettable discrete-time systems were used as a basis for a well-developed formal model of on-line diagnosis, and formal methods were used to investigate this model. As evidenced by the results in Chapters III through VI this approach has proved to be very fruitful. One advantage of this approach is that the results developed in this report are independent of any particular technology and may be applied to any system which can be modeled as a resettable machine.

In Chapter I, a number of fundamental questions concerning on-line diagnosis were stated, and in Chapter II a complete model for the study of on-line diagnosis was developed. Subsequent chapters provided some answers to these questions for the unrestricted fault case and the unrestricted component fault case. At this point it is appropriate to review these questions to see just what has been accomplished and what remains to be done. These five questions are paraphrased below, and each question is followed immediately by a discussion of it.

I. What good on-line diagnosis techniques are available and when is each applicable?

For unrestricted faults the techniques investigated have been duplication and loop checking. Duplication is very easy to implement, and it was shown in Corollaries 4. 5. 1 and 4. 5. 2 that, in terms of the state set size of the detector, it is impossible to do any better than duplication. Thus duplication is a very good technique. However, with other measures of complexity it may be possible to beat duplication. In addition, duplication suffers from the observation that both copies could have the same failures or built in weaknesses from birth. For these reasons the use of inverses for unrestricted fault diagnosis was also studied, and this technique was shown to be applicable regardless of the specified behavior.

For unrestricted component faults the basic technique studied was the construction of totally redundant networks from arbitrary networks through the addition of one component. This technique was also shown to apply to any specified machine.

Certainly, other techniques for the diagnosis of these sets of faults exist and their investigation is an open problem. One fruitful direction might be to pursue a more general approach to the constrained decomposition problem discussed in Section 6. 4.

II. When is a given realization diagnosable?

Answers to this question depend, of course, on what constraints on allowable detectors and delays are given by a particular meaning of the word "diagnosable." If no restrictions are placed on the set of possible detectors then every realization is diagnosable for any set of faults since the realization could be duplicated in the detector.

For unrestricted faults, if detectors are only allowed to perform a loop check then Theorem 5.2 tells us that realizations with lossless inverses are diagnosable. However, the characterization of all realizations which are diagnosable in this sense is still an open problem.

For unrestricted component faults, we know from Theorem 6.1 that a realization is diagnosable if and only if it is totally redundant.

III. What time-space tradeoffs are possible between the added complexity needed for diagnosis and the maximum allowable delay?

By Corollaries 4.5.1 and 4.5.2 we know that no time-space tradeoff is possible for unrestricted faults. However, Example 4.1 shows that a tradeoff is possible for permanent output faults. For unrestricted component faults the question remains unanswered.

While no generally useful time-space tradeoffs have been found, specific tradeoffs are possible for suitably restricted sets of faults and certain specific behaviors. In addition to Example 4.1, this is evidenced by Example 7.1 which appears later in this chapter.

IV. What is the relationship between a given fault set and the set of errors which can be caused by faults in that set?

This relationship was discussed in Section 4.1 for unrestricted faults, Section 4.3 for permanent output faults, and Section 6.2 for unrestricted component faults. Briefly, unrestricted faults can cause any possible erroneous behavior; permanent output faults cause the same minimal 2-errors as unrestricted faults but not the same minimal 1-errors because the output becomes constant once a permanent output fault occurs; unrestricted component faults cause minimal 2-errors which are out of tolerance in only one coordinate, and if the network under consideration is totally redundant then minimal 1-errors caused by unrestricted component faults always result in an unreachable state of N being entered. As expected, this relationship is very important and was used in results concerning each of these sets of faults.

V. What properties of system structure and behavior are conducive to on-line diagnosis?

For unrestricted component faults the structural property of total redundancy was seen to be quite important. The behavioral property of "having a lossless inverse" was also seen to be useful since the unrestricted faults of such systems could be diagnosed via a loop check.

A potentially fruitful area for further work would be to look at special subclasses of machines (e. g. , definite machines, linear machines, etc.) to see what diagnosis qualities they possess which are not possessed by machines in general.

Since this study focused on the diagnosis of unrestricted faults and unrestricted component faults, one large open area for further research is to answer these questions for other important sets of faults. A possible direction for such research is outlined below.

In this report, abstract (i. e. , totally unstructured) systems have been considered with the exception of some of the examples and the networks considered in Chapter VI. Such an approach is good for developing formally the concepts involved in our theory and for studying the diagnosis of unrestricted faults, but some of the questions raised can best be studied in a more structured environment. One reason for this is that with a structured system we can consider the causes of faults. For example, given an abstract system it makes no sense to speak of the set of faults caused by component failures of a certain type or by bridging failures. However, given a structured representation of a system (e. g. , a circuit diagram) we can discuss these and other types of failures and determine the corresponding faults.

There are many different structural levels that could prove useful to a further investigation into the theory of on-line diagnosis. Two levels which we believe will be important are: the binary

state-assigned level and the logical circuit level. These levels and the basis for their potential usefulness are explained below.

A machine M is said to be binary state-assigned if $Q = \{0, 1\}^n$ for some positive integer n. Given such a machine, various types of memory failures such as stuck-at-0, stuck-at-1, and more general types can be considered. The faults corresponding to these failures can be enumerated and comparisons can be made between various schemes for diagnosing these faults. Memory faults have been studied before in the context of fault tolerance and off-line diagnosis by Meyer [28] and Yeh [38] respectively, and they are an important class of faults for a number of reasons. For example, only a limited amount of structure is needed to discuss them. Thus memory faults can be analyzed before the circuit design of the machine is complete. Also, it is memory which distinguishes truly sequential systems from purely combinational (one-state) systems. Combinational systems are inherently easier than sequential systems to analyze and a number of techniques for the on-line diagnosis of such systems are known (see [19] and [34] for example).

Time-space tradeoffs are also possible in the diagnosis of memory faults. Let F_m denote the set of single memory stuck-at faults, that is, the set of faults caused by a stuck-at failure in one memory element. It can easily be verified that if (M, F_m) is $(D, 0)$-2-diagnosable where D is combinational then the reachable states of M must be encoded into a single error detecting code. However, as

the following example shows, this is not necessarily true if nonzero delay is allowed.

Example 7.1:

Consider the binary state-assigned state machine M whose state graph is shown in Fig. 7.1. Since M is an autonomous state machine the labels on the transitions convey no information and hence are not shown.

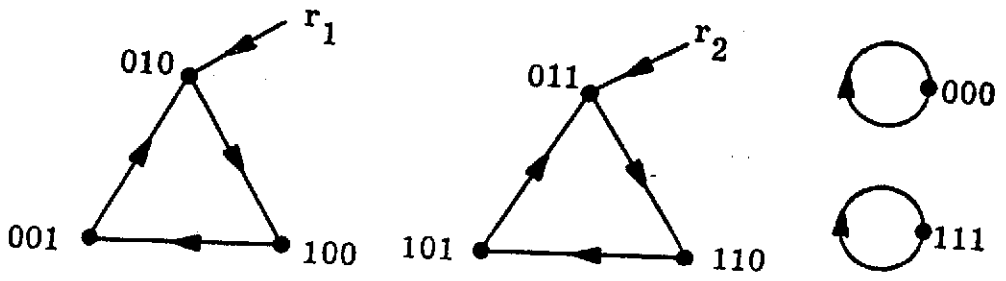


Fig. 7.1. State Graph of M

Claim: (M, F_m) is (D, 2)-2-diagnosable for some combinational D.

Let D be the combinational detector which realizes the function specified by the following table:

z	$\lambda_D(z)$
000	1
001	0
010	0
011	0
100	0
101	0
110	0
111	1

Thus a fault is indicated if and only if the detector observes that the system it is monitoring has entered one of the two unreachable states 000 and 111.

It is instructive to view the action of M in 3-dimensions as shown in Fig. 7.2. In this figure the action of the unreachable (or "error indicating") states have been omitted for clarity.

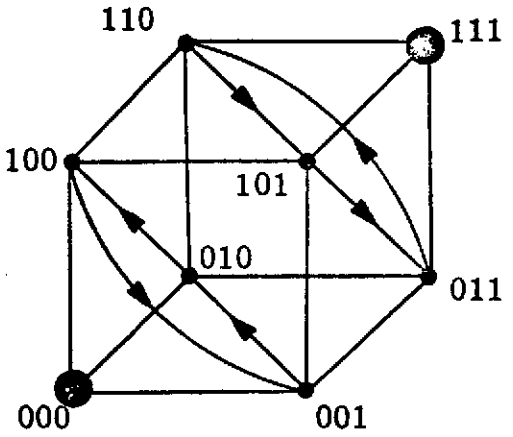


Fig. 7.2. 3-Dimensional View of M

Note that any single memory stuck-at-0 fault will cause the resulting faulty system to enter state 000 within 2 time steps of its occurrence. Similarly, the state 111 will be entered within 2 time steps of the occurrence of a stuck-at-1 fault. Hence (M, F_m) is (D, 2)-2-diagnosable. This homing action after a fault occurs is illustrated below in Fig. 7.3. This figure shows the state graph of M' where $f = (M', \tau, \theta)$ is a fault of M caused by the memory element corresponding to the second coordinate of the state-assignment becoming stuck-at-0.

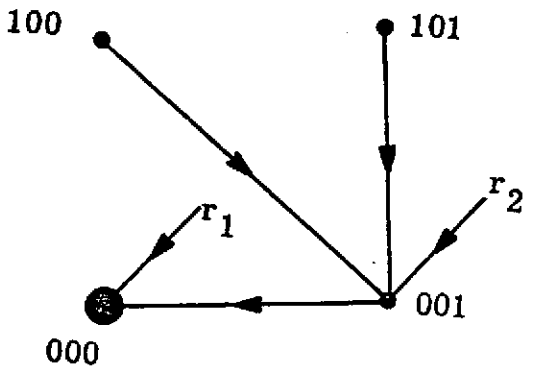


Fig. 7.3. State Graph of M'

The essence of the technique used in this example is to find a state-assigned realization with the property that any single memory stuck-at-fault will cause the resulting faulty machine to enter into a normally unreachable state. This is a generalization of the basic mechanism for diagnosis used by any scheme which involves encoding

the reachable part of the state set into a single error detecting code.

Having looked at the binary state-assigned level of structural detail, let us now turn briefly to the logic circuit level. A system possesses structure at the logical circuit level if a representation of the system is given in terms of a logical circuit composed of primitive logical elements. These may be of the AND-OR variety, threshold elements, or any similar elements of a "building block" nature depending upon the technology being considered. This level is useful for investigating failures in the primitive components. The circuit in Fig. 2.2 is an example of a structural representation at this level and the failure of this circuit discussed in Example 2.2 is a simple example of the analysis that can be conducted at this level.

Further work could also be performed at the network level of structural detail which was introduced in Chapter VI. At this level one could study the problem of implementing on-line diagnosis on a whole computer whereas with the other levels the emphasis would be on diagnosing one module. Note that in our definition of diagnosis the detector is not constrained to give simply a yes-no response. It could also provide extra information for use in automatic fault location. Thus, at this level, the problem of which subsystems must be explicitly observed by the detector to achieve some desired fault location property could be studied.

One problem that requires extension of our present model (at any structural level) is the problem of automatic reconfiguration of the system under the control of the detector. To study this problem, the model used would have to allow for feedback from the detector to the system it is observing. The question of how such an extension should be made is an interesting one and, if answered satisfactorily, could serve as a basis for a systematic investigation of reconfiguration techniques.

APPENDIX

Resettable Machine Theory

The goal of this appendix is not to study the theory of resettable machines per se but rather to cover that part of it which is used in this study of on-line diagnosis. The theory of resettable machines follows closely the theory of sequential machines. The main differences in the definitions stem from the presupposition that a resettable machine is reset before every use. One consequence of this is that the "unreachable" states of a resettable machine are always ignored.

We begin by repeating here the basic machine notions introduced in Chapter II.

Let M be a resettable machine. The reachable part of M, denoted by P, is the set

$$P = \{\delta(\rho(r), x) \mid r \in R,\ x \in I^*\} .$$

M is reachable if P = Q. M is ℓ-reachable if

$$P = \{\delta(\rho(r), x) \mid r \in R,\ x \in I^* \text{ and } |x| \leq \ell\} .$$

Let M, M' $\in \mathfrak{M}(I, Z, R)$. M is equivalent to M' (written $M \equiv M'$) if $\beta_r = \beta'_r$ for all $r \in R$. Two states $q \in Q$ and $q' \in Q'$ are equivalent ($q \equiv q'$) if $\beta_q = \beta'_{q'}$. It is easily verified that these are both equivalence relations, the first on $\mathfrak{M}(I, Z, R)$ and the second on

the states of machines in $\mathfrak{M}(I, Z, R)$. M is reduced if for all $q, q' \in P$, $q \equiv q'$ implies $q = q'$.

If M and M' are two resettable machines then M realizes M' if there is a triple of functions $(\sigma_1, \sigma_2, \sigma_3)$ where σ_1: $(I')^+ \rightarrow I^+$ is a semigroup homomorphism such that $\sigma_1(I') \subseteq I$, σ_2: $R' \rightarrow R$, σ_3: $Z'' \rightarrow Z'$ where $Z'' \subseteq Z$, such that for all $r' \in R'$ $\beta'_{r'} = \sigma_3 \circ \beta_{\sigma_2(r')} \circ \sigma_1$.

The following result is analogous to the result due to Leake [23] which was cited in Section 2.2. It supplies an alternative, and structurally oriented, definition of realization.

Theorem A.1: Let M and M' be two resettable machines with reachable parts P and P'. M realizes M' if and only if there exists a 4-tuple of functions $(\eta_1, \eta_2, \eta_3, \eta_4)$ where

$$\eta_1: I' \rightarrow I$$

$$\eta_2: R' \rightarrow R$$

$$\eta_3: Z \rightarrow Z'$$

$$\eta_4: P' \rightarrow \mathscr{P}(P) - \phi \quad (\mathscr{P}(P) = \{X \mid X \subseteq P\})$$

such that

i) $\delta(\eta_4(p'), \eta_1(a)) \subseteq \eta_4(\delta'(p', a))$ for all $p' \in P'$ and $a \in I'$

ii) $\eta_3(\lambda(p, \eta_1(a))) = \lambda'(p', a)$ for all $p' \in P'$, $a \in I'$, and $p \in \eta_4(p')$

iii) $\rho(\eta_2(r')) \in \eta_4(\rho'(r'))$ for all $r' \in R'$.

Proof: (Necessity) Assume that M realizes M'. Then there exists an appropriate triple of functions $(\sigma_1, \sigma_2, \sigma_3)$ such that $\beta'_{r'}(x) = \sigma_3(\beta_{\sigma_2(r')}(\sigma_1(x)))$. Therefore

$$\beta'_{\rho'(r')}(uv) = \sigma_3(\beta_{\rho(\sigma_2(r'))}(\sigma_1(uv)))$$

for each $r' \in R'$, $u \in (I')^*$ and $v \in (I')^+$. Hence,

$$\beta'_{\delta'(\rho'(r'),u)}(v) = \sigma_3(\beta_{\delta(\rho(\sigma_2(r')),\sigma_1(u))}(\sigma_1(v))) .$$

Thus for each $p' \in P'$ there is a $p \in P$ such that

$$\beta'_{p'}(v) = \sigma_3(\beta_p(\sigma_1(v))) .$$

Consider η_4: $P' \to \mathscr{P}(P) - \phi$ defined by

$$\eta_4(p') = \{p \in P \mid \beta'_{p'} = \sigma_3 \circ \beta_p \circ \sigma_1\}$$

and consider η_1: $\tilde{I} \to I$ defined by

$$\eta_1(a) = \sigma_1(a) .$$

Claim: The 4-tuple $(\eta_1, \sigma_2, \sigma'_3, \eta_4)$ where σ'_3 is an arbitrary extension of σ_3 to Z satisfies i), ii), and iii).

i) Let $p \in \eta_4(p')$. We must show $\delta(p, \eta_1(a)) \in \eta_4(\delta'(p', a))$.

$$\beta'_{\delta'(p', a)}(x) = \beta'_{p'}(xa)$$

$$= \sigma_3(\beta_p(\sigma_1(xa)))$$

$$= \sigma_3(\beta_{\delta(p, \sigma_1(a))}(\sigma_1(x)))$$

$$= \sigma_3(\beta_{\delta(p, \eta_1(a))}(\sigma_1(x))) .$$

Hence, $\delta(p, \eta_1(a)) \in \eta_4(\delta'(p', a))$.

ii) Let $p \in \eta_4(p')$. We must show

$$\sigma_3(\lambda(p, \eta_1(a))) = \lambda'(p', a) .$$

$$\lambda'(p', a) = \beta'_{p'}(a)$$

$$= \sigma_3(\beta_p(\eta_1(a)))$$

$$= \sigma_3(\lambda(p, \eta_1(a))) .$$

iii) Let $r' \in R'$. We must show $\rho(\sigma_2(r')) \in \eta_4(\rho'(r'))$.

$$\beta'_{r'}(x) = \sigma_3(\beta_{\sigma_2(r')}(\sigma_1(x)))$$

implies

$$\rho(\sigma_2(r')) \in \eta_4(\rho'(r')) .$$

(Sufficiency) Suppose there exists functions $(\eta_1, \eta_2, \eta_3, \eta_4)$ as in the statement of the theorem. Let σ_1: $(\tilde{I})^+ \to I^+$ be the natural extension of η_1 to sequences. That is, $\sigma_1(a_1 \ldots a_n) = \eta_1(a_1) \ldots \eta_1(a_n)$. Claim: M realizes M' under $(\sigma_1, \eta_2, \eta_3)$. Consider ζ: $P' \to P$ where

$$\zeta(p') = \text{some } p \in \eta_4(p') \text{ such that}$$
$$\rho(\eta_2(r')) = \zeta(\rho'(r')) \text{ for all } r' \in R'.$$

Let $x = ya$ where $a \in I$. Then

$$\begin{aligned}
\eta_3(\beta_{\eta_2(r')}(\sigma_1(x))) &= \eta_3(\beta_{\rho(\eta_2(r'))}(\sigma_1(x))) \\
&= \eta_3(\beta_{\zeta(\rho'(r'))}(\sigma_1(x))) \\
&= \eta_3(\lambda(\delta(\zeta(\rho'(r')), \sigma_1(y)), \sigma_1(a))) \\
&= \eta_3(\lambda(p, \sigma_1(a))) \text{ where } p \in \eta_4(\delta'(\rho'(r'), y)) \\
&= \lambda'(\delta'(\rho'(r'), y), a) \\
&= \beta'_{\rho'(r')}(ya) \\
&= \beta'_{r'}(x)
\end{aligned}$$

This completes the proof of Theorem A. 1.

The next theorem states that if two reachable states of a state machine M mimic the same state of another state machine M', then for any given input the states that they go to under the transition function δ also mimic the same state of M'.

Theorem A. 2: Let M be a state machine which realizes a state machine M' under $(\sigma_1, \sigma_2, \sigma_3)$ where σ_1 is onto. Then for all q_1, $q_2 \in P$ and $a \in I$, $\sigma_3(q_1) = \sigma_3(q_2)$ implies $\sigma_3(\delta(q_1, a)) = \sigma_3(\delta(q_2, a))$.

Proof: Let q_1, $q_2 \in P$ and assume that $\sigma_3(q_1) = \sigma_3(q_2)$. Say that $q_1 = \delta(\rho(r_1), u_1)$ and $q_2 = \delta(\rho(r_2), u_2)$. Since M realizes M', for all $r' \in R'$, $\sigma_3 \circ \beta_{\sigma_2(r')} \circ \sigma_1 = \beta'_{r'}$. Since M and M' are state machines, for all $r' \in R'$ and $x' \in (I')^*$,

$$\sigma_3(\delta(\rho(\sigma_2(r')), \sigma_1(x'))) = \delta'(\rho'(r'), x') .$$

Let $a \in I$ and denote $\sigma_1(x')$ by x and $\sigma_2(r')$ by r. Then

$$\begin{aligned}
\sigma_3(\delta(q_1, a)) &= \sigma_3(\delta(\rho(r_1), u_1 a)) \\
&= \delta'(\rho'(r'_1), u'_1 a') \\
&= \delta'(\delta'(\rho'(r'_1), u'_1), a') \\
&= \delta'(\sigma_3(\delta(\rho(r_1), u_1)), a') \\
&= \delta'(\sigma_3(q_1), a')
\end{aligned}$$

Likewise, $\sigma_3(\delta(q_2, a) = \delta'(\sigma_3(q_2), a')$. Since $\sigma_3(q_1) = \sigma_3(q_2)$ it now follows immediately that $\sigma_3(\delta(q_1, a)) = \sigma_3(\delta(q_2, a))$.

Theorem A.3: If M realizes M' and M' is reduced and reachable then $|Q| \geq |Q'|$.

Proof: Assume that M realizes M' under $(\sigma_1, \sigma_2, \sigma_3)$ and that M' is reduced and reachable. Then $\beta'_r = \sigma_3 \circ \beta_{\sigma_2(r)} \circ \sigma_1$ for all $r \in R'$. Let $q' \in Q'$. Then there exists $r \in R'$ and $x \in (I')^*$ such that $q' = \delta'(\rho'(r), x)$. Now

$$\beta'_{q'}(y) = \beta'_{\delta'(\rho'(r),x)}(y)$$

$$= \beta'_r(xy)$$

$$= \sigma_3(\beta_{\sigma_2(r)}(\sigma_1(xy)))$$

$$= \sigma_3(\beta_{\delta(\rho(\sigma_2(r)),\sigma_1(x))}(\sigma_1(y)))$$

Hence there exists a function f from Q' into Q such that for each $q' \in Q$, $\beta'_{q'} = \sigma_3 \circ \beta_{f(q')} \circ \sigma_1$.

To prove that $|Q| \geq |Q'|$, it suffices to show that f is 1-1. Let $q_1, q_2 \in Q'$ and assume that $f(q_1) = f(q_2)$. Then $\beta'_{q_1} = \sigma_3 \circ \beta_{f(q_1)} \circ \sigma_1 = \sigma_3 \circ \beta_{f(q_2)} \circ \sigma_1 = \beta'_{q_2}$. Since M' is reduced and reachable this implies that $q_1 = q_2$. Hence f is 1-1. This establishes the result.

Theorem A.4: The relation "realizes" is transitive. That is, M realizes M' and M' realizes M'' implies M realizes M''.

Proof: (Sketch) Assume that M realizes M' under $(\sigma_1, \sigma_2, \sigma_3)$ and that M' realizes M'' under $(\sigma'_1, \sigma'_2, \sigma'_3)$. Then $\beta'_{r'} = \sigma_3 \circ \beta_{\sigma_2(r')} \circ \sigma_1$ for all $r' \in R'$ and $\beta''_{r''} = \sigma'_3 \circ \beta'_{\sigma'_2(r'')} \circ \sigma'_1$ for all $r'' \in R''$. It follows that $\beta''_{r''} = \sigma'_3 \circ \sigma_3 \circ \beta_{\sigma_2(\sigma'_2(r''))} \circ \sigma_1 \circ \sigma'_1$. That is, M realizes M'' under $(\sigma_1 \circ \sigma'_1, \sigma_2 \circ \sigma'_2, \sigma'_3 \circ \sigma_3)$.

If M and M' are resettable machines then M is isomorphic to M' if there exist four 1-1 and onto functions

$$\omega_1 : I \rightarrow I'$$

$$\omega_2 : R \rightarrow R'$$

$$\omega_3 : Z \rightarrow Z'$$

$$\omega_4 : P \rightarrow P'$$

such that for all $r \in R$, $a \in I$, and $q \in P$

i) $\omega_4(\delta(q, a)) = \delta'(\omega_4(q), \omega_1(a))$

ii) $\omega_3(\lambda(q, a)) = \lambda'(\omega_4(q), \omega_1(a))$

iii) $\omega_4(\rho(r)) = \rho'(\omega_2(r))$.

The 4-tuple $(\omega_1, \omega_2, \omega_3, \omega_4)$ is called an isomorphism of M onto M'. If $M, M' \in \mathfrak{M}(I, Z, R)$ and (e, e, e, ω_4) is an isomorphism of M onto M', then M is strongly isomorphic to M'. A basic result of sequential machine theory states that for every machine there is an equivalent reduced machine and that this machine is unique up to strong

isomorphism. The corresponding result for resettable machines is given by Theorem A. 5 and Corollary A. 6. 1.

Theorem A.5 : For every resettable machine M there is a reduced and reachable machine M_R equivalent to M.

Proof: Let $M = (I, Q, Z, \delta, \lambda, R, \rho)$ and let $M_R = (I, Q_R, Z, \delta_R, \lambda_R, R, \rho_R)$ where

$$Q_R = \{[q] \mid q \in P\} \qquad ([q] = \{q' \mid q' \equiv q\})$$

$$\delta_R([q], a) = [\delta(q, a)]$$

$$\lambda_R([q], a) = \lambda(q, a)$$

$$\rho_R(r) = [\rho(r)]$$

To prove this result we must verify (1) that δ_R and λ_R are well-defined, (2) that M_R is reduced and reachable, and (3) that $M \equiv M_R$.

The details of this proof are very similar to the details of the corresponding result in sequential machine theory. They may be found in many textbooks which cover this theory (e. g., see Arbib [2]).

M_R as defined above is called the reduction of M. M' is a reduced form of M if M' is reduced and $M \equiv M'$.

Lemma A. 1: $M \equiv M'$ implies $\beta_{\delta(\rho(r), x)} = \beta'_{\delta'(\rho'(r), x)}$ for all $r \in R$ and $x \in I^*$.

Proof: Let $a \in I, x, y \in I^*$ and $r \in R$. Then

$$M \equiv M' \Rightarrow \beta_r(xya) = \beta'_r(xya)$$

$$\Rightarrow \lambda(\delta(\rho(r), xy), a) = \lambda'(\delta'(\rho'(r), xy), a)$$

$$\Rightarrow \lambda(\delta(\delta(\rho(r), x), y), a) = \lambda'(\delta'(\delta'(\rho'(r), x), y), a)$$

$$\Rightarrow \beta_{\delta(\rho(r), x)}(ya) = \beta'_{\delta'(\rho'(r), x)}(ya) .$$

Theorem A.6 : If M and M' are both reduced and $M \equiv M'$ then M is strongly isomorphic to M'.

Proof: Assume that M and M' are reduced and that $M \equiv M'$. We know that each $q \in P$ is representable in the form $\delta(\rho(r), x)$. Define $\omega_4 : P \rightarrow P'$ by

$$\omega_4(\delta(\rho(r), x)) = \delta'(\rho'(r), x) .$$

Claim: M is strongly isomorphic to M' under (e, e, e, ω_4). We must show that ω_4 is well-defined, 1-1 and onto and that for all $r \in R$, $a \in I$ and $q \in P$

i) $\omega_4(\delta(q, a)) = \delta'(\omega_4(q), a)$

ii) $\lambda(q, a) = \lambda'(\omega_4(q), a)$

iii) $\omega_4(\rho(r)) = \rho'(r) .$

In the following $\omega_4(q)$ is denoted by q'.

Well-defined: Let $p = \delta(\rho(r), x)$ and $q = \delta(\rho(s), y)$, and suppose that $p = q$. Then $\beta_{\delta(\rho(r),x)} = \beta_{\delta(\rho(s),y)}$ and thus by Lemma A.1, $\beta'_{\delta'(\rho'(r),x)} = \beta'_{\delta'(\rho'(s),y)}$. That is, $\beta'_{p'} = \beta'_{q'}$. Since M' is reduced and $p', q' \in P'$ it follows that $p' = q'$. Hence ω_4 is well-defined.

1-1: Again let $p = \delta(\rho(r), x)$ and $q = \delta(\rho(s), y)$ but now suppose that $p \neq q$. Then by reapplying the above arguement $p' \neq q'$. Hence, ω_4 is 1-1.

Onto: Since every $q' \in P'$ is representable in the form $\delta'(\rho'(r), x)$ ω_4 is onto.

That i), ii), and iii) are satisfied is straightforward to verify.

Corollary A.6.1: The reduced form of M is unique up to strong isomorphism. That is, if M' and M'' are reduced forms of M then M' is strongly isomorphic to M''.

Proof: If M' and M'' are reduced forms of M then $M \equiv M'$ and $M \equiv M''$. Hence $M' \equiv M''$. Since M' and M'' are both reduced, by Theorem A.6, M' is strongly isomorphic to M''.

Theorem A.7: If $M \equiv M'$ then M realizes M'.

Proof: $M \equiv M'$ implies $\beta_r = \beta'_r$ for all $r \in R$. Hence M realizes M' under (e, e, e).

A resettable machine M is autonomous if $|I| = 1$.

Given a resettable machine M, two input symbols $a, b \in I$ are equivalent ($a \equiv b$) if $\lambda(q, a) = \lambda(q, b)$ and $\delta(q, a) \equiv \delta(q, b)$ for all $q \in P$. M is transition distinct if no two of its input symbols are equivalent. Any machine which has equivalent inputs is redundant in the sense that the inputs in an equivalence class can be represented by any one of its members without affecting the capabilities of the machine. The following result gives an alternative characterization of equivalent inputs.

Theorem A. 8: Let M be a resettable machine, and let $a, b \in I$. Then $a \equiv b$ if and only if for all $x, y \in I^*$ and $r \in R$, $\beta_r(xay) = \beta_r(xby)$.

Proof: (Necessity) Suppose $a \equiv b$ and assume, to the contrary, that $\beta_r(xay) \neq \beta_r(xby)$ for some $r \in R$ and $x, y \in I^*$. Let $q = \delta(\rho(r), x)$. Now, $\beta_r(xay) \neq \beta_r(xby)$ implies $\beta_q(ay) \neq \beta_q(by)$. If $y = \Lambda$ then $\lambda(q, a) \neq \lambda(q, b)$. If $y \in I^*$ then $\beta_{\delta(q, a)}(y) \neq \beta_{\delta(q, b)}(y)$ and hence $\delta(q, a) \neq \delta(q, b)$. Therefore $a \not\equiv b$. Contradiction. Hence $a \equiv b$ implies $\beta_r(xay) = \beta_r(xby)$ for all $x, y \in I^*$ and $r \in R$.

(Sufficiency) Assume that $a \not\equiv b$. Then for some $q \in P$, $\lambda(q, a) \neq \lambda(q, b)$ or $\delta(q, a) \not\equiv \delta(q, b)$. Let $q = \delta(\rho(r), x)$. Then $\lambda(\delta(\rho(r), x), a) \neq \lambda(\delta(\rho(r), x), b)$ or $\delta(\rho(r), xa) \not\equiv \delta(\rho(r), xb)$. Hence $\beta_r(xa) \neq \beta_r(xb)$ or for some $y \in I^+$, $\beta_r(xay) \neq \beta_r(xby)$. Therefore if $\beta_r(xay) = \beta_r(xby)$ for all $r \in R$, and $x, y \in I^*$ then $a \equiv b$.

REFERENCES

[1] Anderson, D. A., "Design of Self-checking Digital Networks Using Coding Techniques," Coordinated Science Lab, University of Illinois, Urbana, Report R-527, Sept. 1971.

[2] Arbib, M. A., Theories of Abstract Automata, Prentice-Hall, Englewood Cliffs, New Jersey, 1969.

[3] Avizienis, A., "Concurrent Diagnosis of Arithmetic Processors," Digest of the First Annual IEEE Computer Conference, Chicago, Illinois, Sept. 1967, pp. 34-37.

[4] Avizienis, A., G. C. Gilley, F. P. Mathur, D. A. Rennels, J. A. Rohr, and D. K. Rubin, "The STAR (Self-Testing and Repairing) Computer: An Investigation of the Theory and Practice of Fault-Tolerant Computer Design," IEEE Trans. on Computers, Vol. C-20, Nov. 1971, pp. 1312-1321.

[5] Ball, M. and F. Hardie, "Effects and Detection of Intermittent Faults in Digital Systems," in 1969 Fall Joint Comput. Conf., AFIPS Conf. Proc., Vol. 35, Montvale, New Jersey, AFIPS Press, 1969, pp. 329-336.

[6] Carter, W. C., H. C. Montgomery, R. J. Preiss, and H. J. Reinheimer, "Design of Serviceability Features for the IBM System/360," IBM Journal, Vol. 8, April 1964, pp. 115-126.

[7] Carter, W. C., and P. R. Schneider, "Design of Dynamically Checked Computers," Proc. of the IFIPS, Edinburgh, Scotland, August 1968, pp. 878-883.

[8] Carter, W. C., D. C. Jessep, W. G. Bouricius, A. B. Wadia, C. E. McCarthy, and F. G. Milligan, "Design Techniques for Modular Architecture for Reliable Computer Systems," IBM Res. Report RA 12, Yorktown Heights, New York, March 1970.

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

[10] Dorr, R. C., "Self-Checking Combinational Logic Binary Counters," IEEE Trans. on Computers, Vol. C-21, Dec. 1972, pp. 1426-1430.

[11] Downing, R. W., J. S. Nowak, and L. S. Tuomenoksa, "No. 1 ESS Maintenance Plan," Bell System Technical Journal, Vol. 43, Sept. 1964, pp. 1961-2019.

[12] Eckert, J. P., "Checking Circuits and Diagnostic Routines," Instruments and Automation, Vol. 30, Aug. 1957, pp. 1491-1493.

[13] Even, S., "On Information Lossless Automata of Finite Order," IEEE Trans. on Computers, Vol. EC-14, Aug. 1965, pp. 561-569.

[14] Friedman, A. D., and P. R. Menon, Fault Detection in Digital Circuits, Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

[15] Friedman, A. D., "Diagnosis of Short Faults in Combinational Circuits," Dig. 1973 Int. Symp. Fault-Tolerant Computing, June 1973, pp. 95-99.

[16] Hartmanis, J. and R. E. Stearns, Algebraic Structure Theory of Sequential Machines, Prentice-Hall, Englewood Cliffs, New Jersey, 1966.

[17] Hennie, F. C., Finite-State Models for Logical Machines, John Wiley and Sons, Inc., New York, 1968.

[18] Huffman, D. A., "Canonical Forms for Information-Lossless Finite-State Logical Machines," IRE Trans. on Circuit Theory, Vol. CT-6, Special Supplement, May 1959, pp. 41-59.

[19] Kautz, W. H., "Automatic Fault Detection in Combinational Switching Networks," Stanford Research Institute Project No. 3196, Technical Report No. 1, Menlo Park, California, April 1961.

[20] Kohavi, Z. and P. Lavallee, "Design of Sequential Machines with Fault-Detection Capabilities," IEEE Trans. on Computers, Vol. EC-16, Aug. 1967, pp. 473-484.

[21] Kohavi, Z., Switching and Finite Automata Theory, McGraw-Hill, New York, 1970.

[22] Langdon, G. G. and C. K. Tang, "Concurrent Error Detection for Group Look-Ahead Binary Adders," IBM Journal, Vol. 14, Sept. 1970, pp. 563-573.

[23] Leake, R. J., "Realization of Sequential Machines," IEEE Trans. on Computers (correspondence), Vol. C-17, Dec. 1968, p. 1177.

[24] Massey, J. L., "Survey of Residue Coding for Arithmetic Errors," ICC Bulletin, Vol. 3, Rome, Italy, Oct. 1964, pp. 195-209.

[25] Mathur, F. P., "On Reliability Modeling and Analysis of Ultrareliable Fault-Tolerant Digital Systems," IEEE Trans. on Computers, Vol. C-20, Nov. 1971, pp. 1376-1382.

[26] Mei, K. C. Y., "Bridging and Stuck-at Faults," Dig. 1973 Int. Symp. Fault-Tolerant Computing, June 1973, pp. 91-94.

[27] Meyer, J. F., and B. P. Zeigler, "On the Limits of Linearity," Theory of Machines and Computations (Edited by Z. Kohavi and A. Paz), Academic Press, New York, 1971, pp. 229-242.

[28] Meyer, J. F., "Fault Tolerant Sequential Machines," IEEE Trans. on Computers, Vol. C-20, October 1971, pp. 1167-1177.

[29] Meyer, J. F., "A General Model for the Study of Fault Tolerance and Diagnosis," Proc. of the 6th Hawaii International Conference on System Sciences, Jan. 1973, pp. 163-165.

[30] Peterson, W. W., "On Checking an Adder," IBM Journal, Vol. 2, April 1958, pp. 166-168.

[31] Peterson, W. W. and M. O. Rabin, "On Codes for Checking Logical Operations," IBM Journal, Vol. 3, April 1959, pp. 163-168.

[32] Peterson, W. W., Error-Correcting Codes, MIT Press, Cambridge, Mass., 1961.

[33] Rao, T. R. N., "Error-Checking Logic for Arithmetic-Type Operations of a Processor," IEEE Trans. on Computers, Vol. C-17, Sept. 1968, pp. 845-849.

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

[35] Short, R. A., "The Attainment of Reliable Digital Systems through the Use of Redundancy--A Survey," Computer Group News, Vol. 2, March 1968, pp. 2-17.

[36] Wadia, A. B., "Investigation into the Design of Dynamically Checked Arithmetic Units," IBM Res. Report RC 2787, Yorktown Heights, New York, Feb. 1970.

[37] White, J. C. C., "Programmed Concurrent Error-Detection in an Unchecked Computer," Ph. D. dissertation, Electrical Engineering and Computer Science, University of California, Berkeley, 1973.

[38] Yeh, K., "A Theoretic Study of Fault Detection Problems in Sequential Systems," Systems Engineering Laboratory Technical Report No. 64, The University of Michigan, Ann Arbor, 1972.

[39] Zeigler, B. P., "Toward a Formal Theory of Modeling and Simulation: Structure Preserving Morphisms," J. ACM, Vol. 19, Oct. 1972, pp. 742-764.