



**An Artificial Intelligence Approach to
Analog Systems Diagnosis**

F. J. PIPITONE, K. A. DEJONG, AND W. M. SPEARS

*Navy Center for Applied Research
in Artificial Intelligence*

September 22, 1989

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS															
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.															
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE																		
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NRL Report 9219			5. MONITORING ORGANIZATION REPORT NUMBER(S)															
6a. NAME OF PERFORMING ORGANIZATION Naval Research Laboratory		6b. OFFICE SYMBOL <i>(If applicable)</i>		7a. NAME OF MONITORING ORGANIZATION Naval Air Engineering Center														
6c. ADDRESS (City, State, and ZIP Code) Washington, D.C. 20375-5000			7b. ADDRESS (City, State, and ZIP Code) Lakehurst, NJ 08733															
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Naval Air Engineering Center		8b. OFFICE SYMBOL <i>(If applicable)</i>		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER														
8c. ADDRESS (City, State, and ZIP Code) Lakehurst, NJ 08733			10. SOURCE OF FUNDING NUMBERS <table border="1" style="width:100%; border-collapse: collapse; margin-top: 5px;"> <tr> <td style="width:25%;">PROGRAM ELEMENT NO</td> <td style="width:25%;">PROJECT NO.</td> <td style="width:25%;">TASK NO.</td> <td style="width:25%;">WORK UNIT ACCESSION NO.</td> </tr> <tr> <td>62241N</td> <td></td> <td>WF41-460-000</td> <td>DN155-024</td> </tr> </table>				PROGRAM ELEMENT NO	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.	62241N		WF41-460-000	DN155-024				
PROGRAM ELEMENT NO	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.															
62241N		WF41-460-000	DN155-024															
11. TITLE (Include Security Classification) An Artificial Intelligence Approach to Analog Systems Diagnosis																		
12. PERSONAL AUTHOR(S) Pipitone, F. J., DeJong, K. A. and Spears, W. M.																		
13a. TYPE OF REPORT Interim		13b. TIME COVERED FROM <u>10/85</u> TO <u>9/88</u>		14. DATE OF REPORT (Year, Month, Day) 1989 September 22														
15. PAGE COUNT 29																		
16. SUPPLEMENTARY NOTATION																		
17. COSATI CODES <table border="1" style="width:100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr> <th style="width:33%;">FIELD</th> <th style="width:33%;">GROUP</th> <th style="width:33%;">SUB-GROUP</th> </tr> </thead> <tbody> <tr> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> </tr> </tbody> </table>			FIELD	GROUP	SUB-GROUP										18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Diagnosis Troubleshooting Entropy Analog Probabilistic Causal			
FIELD	GROUP	SUB-GROUP																
19. ABSTRACT (Continue on reverse if necessary and identify by block number) <p style="margin-left: 40px;">In this report techniques are described for the automatic diagnosis of primarily analog systems. These results were obtained after several years of work at NRL in this area, along with a fully implemented research prototype diagnosis system, FIS (Fault Isolation System). Key features are a local qualitative causal model of replaceable module behavior, the absence of the single fault assumption, a rigorous probabilistic treatment of fault probabilities, dynamic best test selection based on heuristics or entropy, and efficient algorithms for computing the probability and the entropy of Boolean expressions.</p>																		
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED															
22a. NAME OF RESPONSIBLE INDIVIDUAL Frank Pipitone			22b. TELEPHONE (Include Area Code) (202)767-2876		22c. OFFICE SYMBOL Code 5510													

CONTENTS

1. INTRODUCTION	1
1.1 The Need for Artificial-Intelligence-Based Diagnostic Systems	1
1.2 A Statement of the Diagnosis Problem	1
1.3 Related Work in AI	3
1.4 Summary of Our Approach	3
2. QUALITATIVE CAUSAL MODELS	4
2.1 Local Causal Rules	5
2.2 The Need for Predefined Tests	6
2.3 Ambiguity Sets	6
3. THE TREATMENT OF FAULT PROBABILITIES	7
3.1 A Priori Probability Model	7
3.2 Calculating Fault Probabilities After Passed Tests	9
3.3 Calculating Fault Probabilities After Failed Tests	10
3.4 Passed Tests and Certification Strategies	13
4. BEST TEST STRATEGIES	15
4.1 Testing and Game Trees	15
4.2 Information Gain vs Cost	16
4.3 Heuristic Strategies	17
4.4 Fast Entropy Algorithm	17
5. FIS: AN IMPLEMENTED DIAGNOSTIC SYSTEM	21
6. CURRENT APPLICATIONS OF FIS	23
7. CONCLUSIONS	24
REFERENCES	24

AN ARTIFICIAL INTELLIGENCE APPROACH TO ANALOG SYSTEMS DIAGNOSIS

1. INTRODUCTION

This report describes some general diagnostic reasoning techniques that exploit recent advances in the field of artificial intelligence (AI). They are applicable to a variety of human-engineered systems, including hydraulic, mechanical and optical ones, but the primary focus has been on electronic systems. These techniques were developed over a period of several years of research in this area at the Naval Research Laboratory. One of the products of this research is a fully implemented diagnostic reasoning system called Fault Isolation System (FIS) that embodies these techniques and is in use at a variety of government and industry laboratories.

The FIS system is also described in considerable detail in other publications [1,2]. The intent of this report is to present the methods in a way in which they can be adapted and used by others.

1.1 The Need for Artificial-Intelligence-Based Diagnostic Systems

The maintenance of electronic equipment has drawn increasing attention during the past decade as a potential artificial intelligence application, particularly in the military. This has been motivated by the increasing complexity of military electronic systems and the resulting high-maintenance costs as well as the scarcity of highly trained technicians.

The Navy has been particularly interested in and supportive of the development of fault-isolation expert systems that can improve the quality of their maintenance and troubleshooting activities. As an example, Navy technicians on aircraft carriers may be responsible for troubleshooting several hundred different (sub)systems for which they have had varying amounts of training; (frequently little or none). To compensate, the Navy has invested and continues to invest heavily in automatic test equipment (ATE) to aid or replace these technicians. The quality of the "test programs" that drive these ATE stations varies dramatically in spite of a uniform high cost to acquire them.

For the past few years, we have had the opportunity to explore the use of artificial intelligence (AI) and expert system technology in this setting. We feel that we have now evolved a set of methods that directly addresses the issues discussed above and we have shown the effectiveness of these methods by implementing a diagnostic system (FIS) that embodies these techniques.

1.2 A Statement of the Diagnosis Problem

Since there are many different kinds of diagnosis problems, it is important to understand more precisely the diagnostic reasoning problem addressed in this report. We will do so by describing what questions must be answered by the diagnostic reasoning system, and what kinds of knowledge about the unit under test (UUT) are assumed to be available to support the diagnostic process.

The questions to be answered by the diagnostic system are intuitively quite simple. We want the system to be able to recommend at any point the next best test to make on the UUT from a set of predefined available tests, and we want it to estimate the probability that a given replaceable component (or Boolean combination of components) is faulty, given some test results. These are usually done cyclically, as shown in Fig. 1.

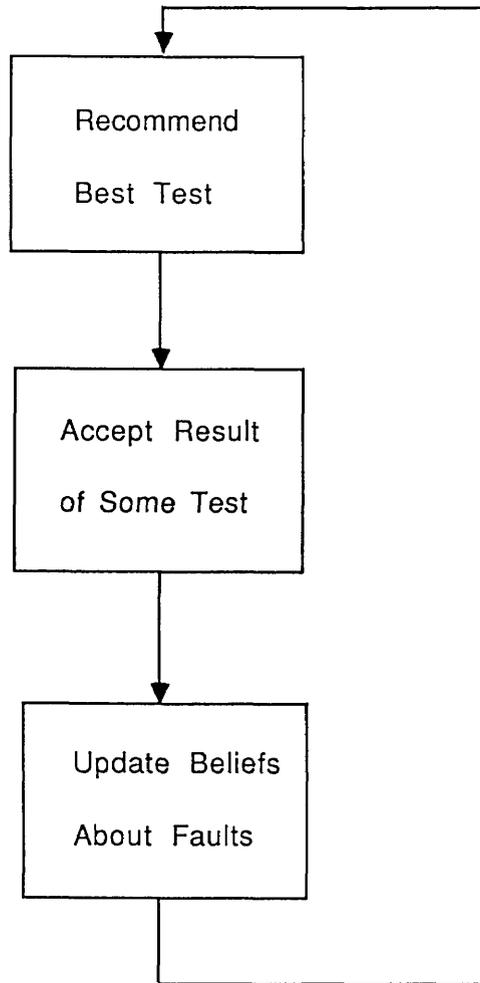


Fig. 1 — Top level activities of a diagnosis system

To decide what kind of knowledge about a UUT must be available and how it should be represented in machine-readable form for effective diagnosis is a more difficult problem. In the field of artificial intelligence this is known, in general terms, as *the knowledge representation problem*. Closely related to this is *the knowledge acquisition problem*: Having chosen a representation, how easy or difficult is it to get the required knowledge about a particular problem into this format. In many AI applications, knowledge acquisition is a crucial consideration because it can involve a high cost in terms of time and money. This is also true of fault diagnosis. To create a computer program that can troubleshoot a piece of equipment with useful accuracy and efficiency, a considerable amount of effort must be invested in knowledge acquisition. Therefore, we wish to represent the UUT with a minimal amount of knowledge that still allows effective diagnostic reasoning.

Let us briefly consider some different approaches to diagnosis in light of the relative ease or difficulty of the knowledge acquisition required to support them. First, consider the approach of directly writing a test decision tree. This corresponds to the conventional practice of writing programs to control ATE in which the burden of diagnostic reasoning lies primarily on the human. What he enters into the computer is essentially a decision tree that will control the ATE autonomously or with minimal human interaction. Although converting such a decision tree into computer code is usually not very labor-intensive, considerable time and effort are usually involved in producing the decision tree itself, because the human needs to think through a large number of diagnostic situations. This decision tree approach, therefore, has high knowledge acquisition cost. It also has the disadvantages of limited fault coverage, inflexibility, lack of explanation capability, and susceptibility to human error and suboptimal decisions.

At another extreme is the approach of entering into the computer a detailed circuit-level model of the UUT sufficient to simulate its behavior. This approach has the possibility of low knowledge acquisition cost since CAD/CAM data could be used to provide the information in computer-readable form, and only the important aspects of behavior (performance specifications) would need to be largely human specified. The difficulty with this approach lies in our inability to use such a low-level description of a UUT efficiently for diagnosis. Even simple circuit-level simulations can take hours of computer time, although this may improve in the future with advances in computing hardware and simulation methods.

Consequently, a primary focus of our research has been to develop a knowledge representation scheme that is easy to acquire and useful (in a practical sense) for developing efficient diagnostic systems. We have achieved that goal by exploiting from the AI community emerging ideas about knowledge representation.

1.3 Related Work in AI

Recently, some notable work has been directed at some of the same problems we address here. For example, DeKleer [3] gives a clear analysis of the problems of computing fault hypothesis probabilities, test result probabilities, and the use of entropy for test recommendation. However, this approach requires a strong UUT model; one must be able to predict module output values from module input values. It is often impractical to provide such a model for analog systems, although digital systems lend themselves well to this approach. Also, computational efficiency is not addressed. For the results to be used widely in applications, one needs fast probability and entropy algorithms and module simulations. Genesereth [4] and Davis [5] have done considerable work in the area of diagnosis based on structural and functional UUT descriptions, but focus has been on digital systems. Other recent work [6] based on Bayesian belief networks [7], provides a powerful mathematical approach to treating the joint statistics of component faults and signal abnormalities.

1.4 Summary of Our Approach

When looking at AI technology, it is tempting to conclude that one could significantly improve this sort of troubleshooting activity with reasonable straightforward applications of current expert system technology. However, several aspects of the problem raise significant technical issues. First, with several hundred different systems to maintain, it is infeasible to think in terms of independently developed expert systems for each one. Rather, one thinks in terms of a more general fault isolation shell that provides a common knowledge acquisition/representation scheme for use with all subsystems. However, even with this level of generality, there are still several hundred knowledge bases to

be built, debugged, and maintained in a context in which there can be considerable overlap and/or similarity in the content of many of the knowledge bases. These observations strongly suggest the development of a sophisticated knowledge acquisition system that can be used to facilitate the construction of a new knowledge base for a specific system in a variety of ways including reusing and/or adapting existing knowledge modules.

Compounding the problem of applying current expert-system technology is the fact that, for many of the subsystems being maintained, little human expertise exists in the traditional sense of finding people who are good at fixing particular subsystems and capturing their knowledge in sets of associative rules. This is particularly true for newly developed systems for which empirical experience is absent. Rather, technicians depend heavily on the structural and functional descriptions contained in the technical manuals of the many subsystems they attempt to maintain. This suggests that simple rule-based architectures are not likely to be sufficient for the task at hand, and that a model-based approach may be appropriate.

At the same time, it is clear (as discussed earlier) that detailed quantitative models are in general too inefficient for effective diagnosis. As a consequence, we have adopted an intermediate approach of providing to the diagnostic system a simplified model of the qualitative behavior of the replaceable modules and the structure (connectivity) of the UUT that is both easy to acquire and can be used efficiently for diagnosis. We refer to these models as qualitative causal models.

By using this form of knowledge representation, we have been able to develop an effective diagnostic system with the following notable features:

- (a) the ability to do accurate diagnosis by using a qualitative behavior model of a complex analog/digital UUT without simulation,
- (b) the possibility of efficient UUT knowledge acquisition,
- (c) an efficient probabilistic reasoning method specialized for device troubleshooting based on Bayesian principles,
- (d) a natural treatment of multiple faults, and
- (e) an efficient method for computing the entropy of a complex system for use in best test selection.

The core set of techniques that provide these features is described in more detail in the sections that follow. The techniques range from very general ones, such as the algorithms for computing the probability and the entropy of an arbitrary Boolean expression, to highly domain-specific ones, such as the heuristic methods for certifying modules after passed tests.

2. QUALITATIVE CAUSAL MODELS

As previously mentioned, a central element in our approach to diagnosis is the capture of the important aspects of the behavior of a UUT for diagnostic purposes in a qualitative causal model. A qualitative causal model of a UUT consists of: (a) a causal description of the set of replaceable modules (determined by the level to which fault isolation is to occur), (b) a description of the connectivity (structure) of the replaceable modules, and (c) a set of possible tests from which diagnostic sequences can be constructed. Also, the model can contain (if available) estimates of a priori failure rates as well as the costs of making tests and replacing modules.

The diagnostic power comes from the requirement that the description of each replaceable module must include a set of local causal rules describing its behavior. The diagnostic system then uses both the structural (connectivity) information and the sets of local causal rules to determine the set of possible global causes (ambiguity set) of any failed test. This information is used in conjunction with a priori fault probabilities (which are assumed to be uniform if not readily available) by an efficient Bayesian algorithm to estimate posterior probabilities of module faults and test outcomes. These are used along with test cost data and module replacement costs (both of which are assumed to be uniform if unavailable) to make the next best test or replacement recommendations. Two strategies are described for computing the next best test; one is based on specialized heuristics and one is based on information theoretic entropy. The latter uses an efficient new algorithm for computing the Shannon entropy of a complex system. An important advantage of our approach to diagnosis is that no single-fault assumptions need to be made.

2.1 Local Causal Rules

The heart of the UUT description is a network of local qualitative causal rules that relate measurable (at least in principle) parameters among various terminals of each module. These are intended to describe the behavior of each module in its context. Thus where loading and other effects of Kirchoff's laws affect behavior, these effects are assumed to have been taken into account. We are not restricted to unidirectional modules with fixed I/O voltage behavior. There are two types of rules—through-rules containing information about the correct behavior of UUT modules and module-rules containing information about fault modes of modules, as discussed below. The forms of the two types are as follows:

Through-rule:

Given precondition, parameter1 abnormality1 at terminal1 can cause parameter2 abnormality2 at terminal2

Module-rule:

Faulty modulename can cause parameter abnormality at terminal.

A precondition is a predicate on the state of the UUT inputs (or input history). Preconditions are optional and are used primarily to model devices such as multiplexers and devices with enable inputs in which a causal path is present or absent depending on some signal conditions. Although we have experimented with elaborate precondition schemes that involve estimating probabilities of control signal values, we have found it quite adequate to simply include in the description of each test the states qualitative value such as *high* or *low*. For example, a through-rule is "Given that power-supply-3 is in the "on" state, dc voltage high at t3 can cause frequency low at t4." This might describe the way a voltage-controlled oscillator propagates a voltage error, given that its power supply voltage is within specification. In the case of a module with two inputs that both affect the same output parameter, such as a summing junction, a problem can arise. Suppose two of the rules are "input1 dc high can cause output dc high" and "input2 dc high can cause output dc high." Then if "output dc high" is suspected, both inputs are to be suspected high. However, one might be very high and the other slightly low, and the latter would be missed. However, if such a case occurs, the system will find a fault leading to the high input, and the ensuing repair may even cure the slightly low input. Otherwise, a second diagnostic pass can search for it. Each causal rule is associated with the module immediately connected to the terminals appearing in the rule.

Through-rules represent the correct behavior of a module in context in the form of qualitative relationships among quantities at its terminals. Module-rules represent limited knowledge about how a given module can fail. Note that even if we had no such fault model knowledge available, we could allow the system to include all conceivable module rules by default and still have a viable diagnosis system. That is, we could assume that any abnormality at a terminal could have been caused by the failure of any directly connected module. This would lead to somewhat larger ambiguity sets (suspect sets) than it would with a more minimal set of module-rules. However, most of the diagnostic power of this kind of model comes from through-rules, since it is largely by chaining them that we determine what portion of the UUT affects a given measurement. When in doubt about the inclusion of any rule, we must include it, else we risk missing some faults.

2.2 The Need for Predefined Tests

It is necessary to predefine a set of tests for a diagnostic system because no matter how complete a model of the structure and input/output behavior of a UUT is used, it cannot know the intended use of the UUT. For example, in a radar receiver, a diagnostic system must know for what frequencies it must operate correctly, and what tested parameters must be correct to within what tolerance.

Our notion of a test is a specified *terminal* to be used as a test point, an electrical (or other) *parameter* to be measured there, and a set of *abnormalities* such as {bad} or {hi, lo} with associated numerical ranges. A range of "ok" is also required. Note that the triple "terminal parameter abnormality" is of the same form as the right-hand side (effect) of a causal rule. In fact each such triple occurring in a test must occur in the right-hand side of at least one rule. Also, the states of all rule preconditions, if any, must be given for each test, as discussed in Section 2.1. Finally, each test should have a prescribed *stimulus setup* state. This is simply a unique symbol for each distinct state of the input stimuli of the UUT used in the tests. It is useful in determining the relevance of a passed test to a suspect module (Section 3.4).

2.3 Ambiguity Sets

Central to our approach is the notion of an *ambiguity set*. One ambiguity set is associated with each abnormal result of each defined test, and it is defined to be the set of all modules that can fail so as to cause that failed test result. If one can obtain such sets, then one can determine what combinations of good and bad modules are possible after more than one test has failed. In Boolean terms, each ambiguity set is a disjunction of propositions that the included modules are faulty, and two or more ambiguity sets represent a conjunction of such disjunctions. For example, the ambiguity sets {1, 2} and {1, 4} represent the fact that at least one of modules one and two is faulty and that at least one of the modules one and four is faulty. In Boolean terms, $(x \vee x_2) \& (x_1 \vee x_4) = 1$, where x_i is the proposition that module i is faulty. Note that ambiguity sets alone do not allow the representation of an arbitrary Boolean function. For example, they cannot represent the exclusive or of two modules, but we believe that our forms cover the majority of cases occurring in diagnostic applications, and the separability of the ambiguity sets is convenient for *undoing* a test already made.

In diagnostic applications, we find it useful to precompute the ambiguity set for each failed outcome of each test. Also, for each module occurring in an ambiguity set it is useful to precompute *immediate effects*. These are simply the signal abnormalities at the terminals of the module that both

lead causally to the failed test result and which occur as the right-hand side of a rule whose left-hand side is the module. *Immediate effects* are useful as a simple approximation of failure modes (see Section 3.4).

The general probabilistic techniques described in Sections 3 and 4 can be applied no matter how the ambiguity sets are obtained, but if the causal model of Section 2.1 and 2.2 are used, then they can be found by following the network of local causal rules upstream from each failed test result until all modules causally upstream are found. If ambiguity sets are directly produced by humans, they constitute global symptom/cause association rules.

Figure 2 illustrates the diagnostic power of our simple causal rule model. Three failed tests at the same test point are shown, all with different ambiguity sets. The ambiguity sets can be verified by visually chaining from effect to cause through the given list of causal rules, starting at a test result and ending at a set of suspect modules. Earlier model-based diagnostic systems [8,9] are not as specific because they represent only the topology of the UUT. Such systems suspect all upstream modules from any failed test.

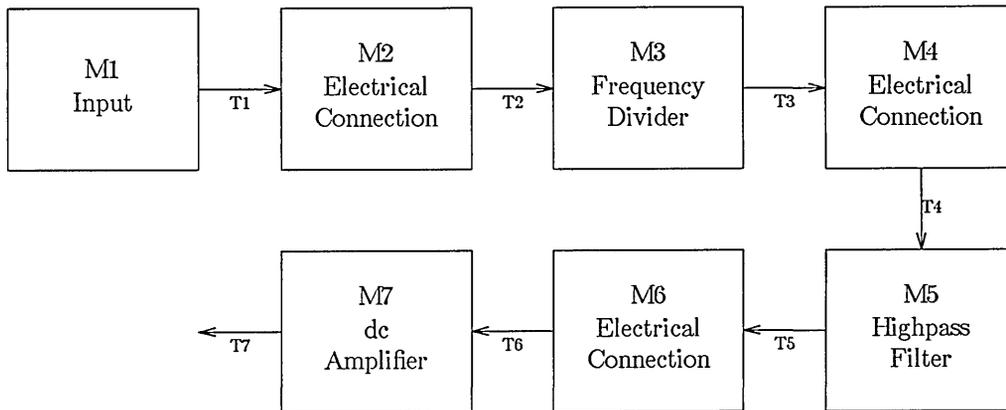
3. THE TREATMENT OF FAULT PROBABILITIES

This section describes an analysis of the joint statistics of UUT faults. A UUT is regarded as having 2^n fault states, since the proposition x_i that module i of the n modules is faulty can be true or false. The probability of an arbitrary fault hypothesis, represented as a Boolean function of the x_i , is developed in stages. First we consider a UUT selected from the entire population of a given UUT type, then a UUT from the population actually undergoing diagnosis, and finally a UUT for which various combinations of passed and failed tests have been made.

In problem domains for which models relating structure and behavior are incomplete and uncertain, such as medical diagnosis, it is often impossible to do probabilistic reasoning in a precise manner, since the relevant joint statistics are unavailable. Therefore, approximate methods have been found to be useful, such as those in MYCIN [10] and PROSPECTOR [11] and the Dempster-Shafer [12] formalism. However, in the domain of diagnosis of human-engineered systems constructed from discrete modules, it is possible to use more precise probabilistic methods for exploiting the known causal and statistical relationships of this domain. For example, the information gleaned from a failed test can often be described as an ambiguity set, a set within which at least one faulty module must lie. Also it is often an acceptable approximation to assume that the replaceable modules fail independently of one another with their own a priori probabilities.

3.1 A Priori Probability Model

We assume that over the entire population of a UUT type the individual module fault propositions x_i are statistically independent of each other with a priori probabilities a_i , so that $a_i a_j$ is the a priori probability of x_i & x_j , for example. We are thus ignoring the case of one component failing first and stressing a second component so that it fails also. This will cause some error in probability calculations, which will affect primarily the selection of tests and therefore the cost (time, money, etc.) of diagnosis rather than its accuracy. Barring this coupling, we can view double faults as independent events, the second of which occurred by chance in the time between the occurrence of the first and the time of testing.



Test Result	Ambiguity Set
T6 freq high	M1, M3
T6 dc low	M5, M6, M7
T6 dc hi	M5

Rules Fired in Propagation of above Test Results

Cause	Effect	Comments
T5 freq high	T6 freq high	M3 not dividing
T4 freq high	T5 freq high	
T3 freq high	T4 freq high	
T2 freq high	T3 freq high	
M3	T3 freq high	
T1 freq high	T2 freq high	
M1	T1 freq high	M1 has high frequency
T5 dc low	T6 dc low	M6 open, M7 pulls low
M6	T6 dc low	
M7	T6 dc low	
M5	T6 dc low	M5 output has offset
M5	T6 dc high	M5 output has offset
T5 dc high	T6 dc high	

Fig. 2 — Propagation of a test result

A diagnostic system does not see the population described above, however. We assume that it sees a population which differs in that a certain number of good UUTs are omitted, so that one is more likely to perform diagnosis on UUTs that have malfunctioned. Specifically, suppose we know empirically that P_0 is the probability that a UUT about to undergo diagnosis is faulty. Now let us refer to the two-module UUT example of Fig. 3, in which area represents probability. Each of the 2^n rectangles represents the probability of one of the fault states (each module good or faulty) of the UUT. The crosshatched area of the upper left represents the good UUTs that are preferentially omitted in the population undergoing diagnosis, so that the remaining area in the "all modules good" rectangle divided by the entire remaining area is $1 - P_0$, the probability that the UUT is good at the start of diagnosis. This analysis extends to arbitrary dimensionality (number of modules) n , but it is not amenable to graphical description.

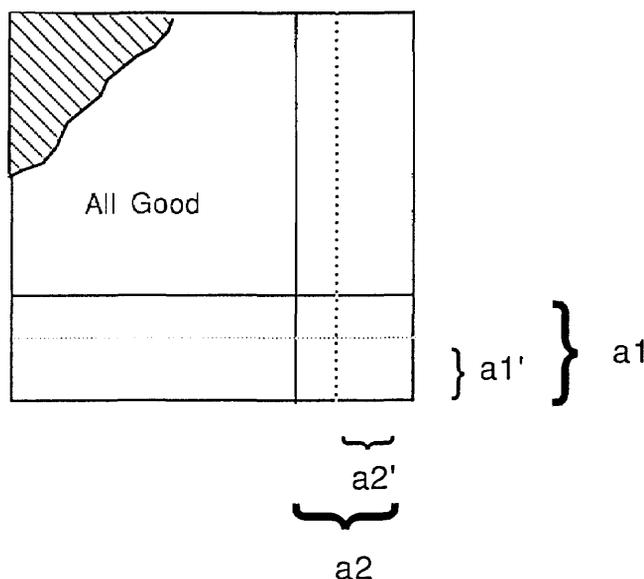


Fig. 3 — Probability analysis before any test fails; a two—module graphical representation

At the start of diagnosis, the probability that module i is faulty is

$$p_i = \frac{P_0 a_i}{1 - \prod_{j=1}^n \bar{a}_j}$$

This is simply the probability P_0 that there is a fault times the fraction of the "faulty area" covered by module i .

3.2 Calculating Fault Probabilities After Passed Tests

We now consider how to update the overall UUT fault probability and the module fault probabilities if one or more tests have been made and passed, and none have failed. We make the simplifying assumption that the information gleaned from passed tests can be represented by a single number c_i for each module i . This is called the *certification factor* and is initially 1.0. It approaches

zero as successive passed tests are made. It represents the factor by which the relative a priori fault probability a_i of module i is reduced by the passing of tests. The manner in which we compute c_i needs careful consideration and is discussed in Section 3.4. Thus, in Fig. 3 the a_i become $a_i' = c_i a_i$ after at least one test passes.

The effect of this on the current probability P_{UUT} that the UUT is faulty is given by the following equation. It simply reflects a renormalization of the probabilities of the 2^n rectangles of Fig. 3 after removing the area between the ‘‘all good’’ rectangle and the dotted lines. Then P_{UUT} is the sum of the probabilities of all areas other than the ‘‘all good’’ area.

$$P_{\text{UUT}} = 1 - \frac{1}{P_0(1 - \prod_i c_i \bar{a}_i)} \cdot \frac{1}{1 + \frac{1}{(1 - P_0)(1 - \prod_i \bar{a}_i)}}$$

In a similar vein, each module fault probability P_i becomes the sum of the probabilities of the 2^{n-1} rectangles for which that module is faulty, after the renormalization mentioned above.

$$P_i = \frac{P_0 c_i a_i}{(1 - P_0)(1 - \prod_j \bar{a}_j) + P_0(1 - \prod_j c_j \bar{a}_j)}$$

3.3 Calculating Fault Probabilities After Failed Tests

For clarity, we first describe the processing of failed test results only. Then we treat the case of some failed and some passed tests. As described in Section 2.3, each failed test gives rise to an ambiguity set containing all modules that could have caused the abnormal measurement. The set of these ambiguity sets corresponds to a Boolean conjunctive normal form expression in the module fault propositions x_i ; at least one member of the first ambiguity set is faulty, and at least one member of the second ambiguity set is faulty, etc. Let T denote this expression and H denote an arbitrary Boolean function of the x_i and their complements, describing a fault hypothesis whose current probability is desired. A fault hypothesis is defined here as a Boolean combination of assertions that individual modules are good or faulty. For example, $x_1 \& x_2 \& x_3$ is the hypothesis that modules 1 and 3 are bad and module 2 is good, without regard for the other modules.

From Bayes' Rule the current probability of H is given by

$$P(H | T) = \frac{P(H \& T)}{P(T)}, \quad (1)$$

where $P(B)$ is defined for an arbitrary Boolean function B as the *a priori* probability of the proposition B , i.e., the a priori probability that the fault state of the UUT is consistent with B . $P(B)$ is expressible as

$$P(B) = \sum_{B=1} A_i,$$

where

$$A_i \equiv \prod_k f_i(a_k), \text{ and}$$

$$f_i(a_k) \equiv a_k, \text{ if the } k^{\text{th}} \text{ bit of the binary representation of } i = 1.$$

$$\equiv \bar{a}_k, \text{ if the } k^{\text{th}} \text{ bit of the binary representation of } i = 0,$$

where \bar{a}_k denotes $1 - a_k$. The complexity of computing $P(B)$ with the explicit summation is $O(2^n)$. Therefore we introduce a more efficient algorithm for this computation.

Our algorithm computes $P(B)$ for an arbitrary Boolean function of the literals x_i , although B must be given in conjunctive normal form. The strategy is to manipulate B so that all terms of conjunctions are statistically independent (involve no common x_i) and all terms of disjunctions are non-overlapping (mutually exclusive). We then can replace the x_i with the corresponding a priori component failure probabilities a_i and the three Boolean operations with arithmetic operations using the following three laws from probability theory:

- A. If $P(X) = p$ then $P(\bar{X}) = 1 - p$.
- B. If $P(X) = p$ and $P(Y) = q$, then $P(X \vee Y) = p + q$ iff $X \& Y = 0$.
- C. If $P(X) = p$, $P(Y) = q$, and X and Y are statistically independent, then $P(X \& Y) = p \times q$.

We perform the Boolean manipulation with the following algorithm. We assume that B is presented to the algorithm in conjunctive normal form. Complexity will be discussed in terms of the number a of conjuncts at the top level of B and the number n of literals occurring. Note that in the diagnosis application a is the number of ambiguity sets (one per failed test) and n is the number of modules. An example follows the algorithm for clarity.

Boolean Manipulation Algorithm:

Step 1 — Perform Boolean absorption to remove all redundant conjuncts.

Step 2 — Apply DeMorgan's Law to B so that it is in complemented disjunctive normal form.

Step 3 — Order the disjuncts by length (number of conjuncts x_i or \bar{x}_i), longest leftmost.

Step 4 — Correct the disjuncts for overlap with others, from left to right, by ANDing certain terms onto each (to be elaborated later). The entire conjunctive expression thus appended to each disjunct will be called a correction expression. Terminate the algorithm if the conjuncts in this correction expression are all literals.

Step 5 — To each correction expression, apply this algorithm recursively, starting at Step 1.

Steps 1 and 4 of the above Boolean manipulation algorithm need additional explanation. In Step 1, we remove all redundant conjuncts as follows. For each conjunct, compare it with each conjunct to its right. Whenever a pair is encountered, such that all terms of one are present in the other, delete the larger conjunct. For example, $(X \vee Y) \& (X \vee Y \vee Z) = X \vee Y$. The complexity of Step 1 is $O(a^2n)$, assuming that the literals were presorted within each conjunct.

Step 4 (the overlap correction) of the Boolean manipulation algorithm proceeds by:

Step 4a — Working from left to right, take the next disjunct D_N .

Step 4b — For each disjunct D to the right of D_N , perform Step 4c.

Step 4c — Note whether any literal x_i occurs complemented in D and uncomplemented in D_N , or vice-versa. If so, terminate Step 4c; this pair is already mutually exclusive. Otherwise compute $D^- \equiv$ what remains of D after removing from it those conjuncts (x_i or \bar{x}_i) present in D_N , and append (AND) the complement of D^- (expressed as a disjunction, using the DeMorgan's Law) onto D_N . Step 4 has complexity $O(a^2n)$, again assuming that the literals were presorted within each conjunct.

A worst case complexity of the above algorithm is $O(a^an)$. This follows from the fact that the complexity of the first recursion is (a^2n) (from Steps 2 and 4) and each additional recursion (Step 5) introduces an extra factor of a through Step 4. The number of recursions is bounded by a , since Step 4 produces a correction expression with at least one less top level term than its input expression at each recursion. There may exist a smaller upper bound than $O(a^an)$. In practice, most functions B yield few (if any) recursions, so the typical complexity is closer to a^2n . Also, in the diagnosis application, n may be large (~ 100 modules) but a , the number of failed tests, is typically less than 10.

Example:

The following is an example of the application of the above algorithm. In keeping with our diagnostic topic, suppose that three tests have failed and our current ambiguity sets are $\{2,3,4\}$, $\{4,5,6\}$, and $\{6,7\}$. Let us compute the probability of the hypothesis x_7 that module 7 is faulty. The ambiguity sets overlap interestingly (at least two faults are present) and make a good illustration. We define the shorthand notation $\bar{P}(B) \equiv 1 - P(B)$ and x_i is denoted by i . Then the numerator of Eq. (1) becomes $P(H\&T) = P[(2 \vee 3 \vee 4) \& (4 \vee 5 \vee 6) \& (6 \vee 7) \& 7]$. Note that laws A, B, and C cannot be applied at this point to compute the numerical value of $P(H\&T)$. In particular, the four conjuncts are not all statistically independent of one another, since they contain some common literals. Therefore, we will apply the Boolean Manipulation algorithm:

$$\begin{aligned}
 H\&T &= (2 \vee 3 \vee 4) \& (4 \vee 5 \vee 6) \& (6 \vee 7) \& 7 \\
 &= (2 \vee 3 \vee 4) \& (4 \vee 5 \vee 6) \& 7; \text{ Step 1 (Boolean absorption)} \\
 &= \overline{(\bar{2} \& \bar{3} \& \bar{4}) \vee (\bar{4} \& \bar{5} \& \bar{6}) \vee 7}; \text{ Step 2 (DeMorgan's law)} \\
 &= \overline{(\bar{2} \& \bar{3} \& \bar{4} \& (5 \vee 6) \& 7) \vee (\bar{4} \& \bar{5} \& \bar{6} \& 7) \vee 7}; \text{ Step 4 (disjunction overlap removal)}
 \end{aligned}$$

$(5 \vee 6) \& 7$ becomes $\overline{\overline{(5 \& 6)} \vee 7}$; Steps 5 and 2, recursively (DeMorgan's law)

$$= \overline{\overline{(5 \& 6 \& 7)} \vee 7}; \text{ Step 4 (disjunction overlap removal)}$$

Finally, $H \& T = \overline{\overline{(\overline{2 \& 3 \& 4} \& \overline{(5 \& 6 \& 7)} \vee 7)} \vee (\overline{4 \& 5 \& 6} \& 7) \vee 7}$.

Now we can apply laws A, B, and C to all negations, conjunctions, and disjunctions respectively:

$$P(H \& T) = 1 - \overline{P(2)P(3)P(4)P(5)P(6)P(7) + P(7)} - \overline{P(4)P(5)P(6)P(7) - P(7)}.$$

The $P(i)$ are simply the a_i . The numerator of Eq. (1) can be treated similarly.

The method above can be used to compute the current fault probabilities of the individual modules after each test. It can also be invoked by the user to compute the probability of any fault hypothesis, even one including some fault states inconsistent with T .

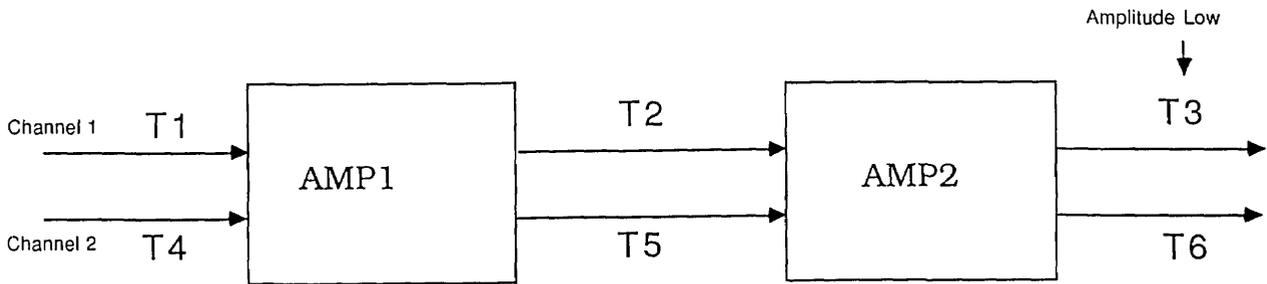
Now we treat the case of having at least one passed test in addition to the failed test(s). As described in Sections 3.2 and 3.4, for each module i we compute a certification factor c_i , which represents the evidence from passed tests. Thus module i is regarded as certified by the amount c_i , and the a priori failure rate a_i is simply replaced with $c_i a_i$ in the probability calculations above.

3.4 Passed Tests and Certification Strategies

Now we focus on how to compute the certification factors c_i used in Sections 3.2 and 3.3. Since we assume the unavailability of failure mode statistics here, we take a somewhat heuristic approach. We motivate our discussion of certification with the example of Fig. 4, in which a diagnosis system currently suspects that at least one of two modules amp1 and amp2 contains a fault. In particular, an RMS Amplitude test fails at T3, leading us to suspect a gain problem in the upper of two amplifier channels in both modules. Then we might be able to determine which module is likely to be faulty by making a test dependent on amp1 and not on amp2. If the test fails, amp1 is faulty; if the test passes, amp1 is less suspect than it was, and amp2 is more suspect. A human interpreting this passed test would think about what failure mode of amp1 tested ok. If it were the same failure mode that made amp1 initially suspect, amp1 would be exonerated (reduction of c_i to nearly zero). If the passed test depended on a completely independent failure mode of amp1, then amp1 would be only slightly certified (small reduction of c_i).

The desiderata for the c_i are as follows:

- (1) Each c_i should be 1.0 initially and never less than zero.
- (2) Each passed test fed by module i should reduce c_i .
- (3) The different ways a module can fail are often not mutually exclusive. Thus successive passed tests fed by module i should have progressively less effect on c_i , all other factors being equal.



Consider Three Tests:
 (1) Amplitude OK at T2
 (2) Frequency OK at T2
 (3) Amplitude OK at T5

Fig. 4 — Difficulty of interpreting passed tests

- (4) If a module is currently suspected of being faulty because one or more tests depending on it failed, then it should be certified more strongly by some passed tests than by others. In particular, a passed test dependent on immediate effects (an approximation of failure modes; see Section 2.3) which are currently suspected should yield a larger reduction of c_i than other passed tests depending on module i .
- (5) A passed test that satisfies (4) above, which also was made under the same UUT stimulus conditions, should yield an even larger reduction of c_i .

In earlier versions of the implemented FIS system, we tried a simple linear scheme. For example, if a module has 10 module-rules and four of the rules have right-hand sides on the causal path to some test that passed, then the certification factor for the module is $c_i = 1 - 4/10 = .6$. Thus the a priori fault probability of the module would be multiplied by 0.6 before being used in the probability calculations of Section 3.3. However, this violates the above desiderata except (1).

One way to satisfy the desiderata while still avoiding requiring joint statistics data for the failure modes of a module is by the following scheme. Keep track of how many of the tests that depend on each immediate effect of the given module have been made and passed. Then take the sum s of these and compute e^{-ks} as the certification factor for the module, with the constant k chosen empirically to optimize performance. This addresses desiderata (1), (2), and (3). Now if the module in question is contained in any ambiguity set, i.e., if it feeds any failed test, then $e^{-ks - k_a s_a}$ is taken to be the certification factor. Here s_a is the number of passed tests that depend on those particular immediate effects of the module that are responsible for that module being in an ambiguity set. That is, for each passed test we note for each module on which the test depends how many of the abnormalities it can immediately cause (at its terminals) lie on the causal path to the failed (e.g. low or high) outcomes of the passed test. The causal connections between the various immediate effects and the test outcomes can be found by direct lookup of precompiled ambiguity sets of tests (see Section 2.3). The constant k_a is chosen empirically to optimize performance. This satisfies desideratum (4). Finally, to cover (5), one can extend the above exponential expression to $e^{-ks - k_a s_a - k_s s_s}$, where s_s is the number of passed tests that depend on those particular immediate effects of the module that are responsible for that

module being in an ambiguity set and which share the same setup as a failed test implicating such an immediate effect. Many other strategies are possible for the approximate certification of modules in the absence of failure mode joint statistics. The above strategies may suggest other ideas.

4. BEST TEST STRATEGIES

We define the *optimal* best test strategy, as the one that minimizes the average total cost of the tests required to achieve some specified degree of certainty about which modules in the UUT are faulty and which ones are not. Optimality is an unrealistic goal in most applications, but there are various ways to achieve adequate suboptimal performance. In the following sections we cast the best test problem in terms of game trees and then argue for the direct evaluation of the available tests without tree search. Two practical forms of evaluation are described; heuristic strategies, which lend themselves to automated explanation, and a more rigorous information theoretic approach. For the latter we introduce an algorithm for computing the entropy of a set of statistically independent propositions constrained by an arbitrary Boolean function.

4.1 Testing and Game Trees

The problem of finding the *best* test to make in diagnostic reasoning is analogous to the problem of playing a game against an opponent who responds randomly. The test result is the opponent's move and the test selection is our move. The object of our side is to minimize the total cost of finding the fault(s). We could think of the opponent as confounding our efforts by giving test results that sometimes further our goal, but sometimes they don't.

One method that is optimal in the sense of the above paragraph is the miniaverage algorithm (Fig. 5). The strategy is to propagate the total test costs given at the terminal nodes back to the root node. This is done by recursively computing the backed-up cost of a node from the costs of its offspring. There are two cases. The backed-up cost of a test node is the weighted average of the backed-up costs of the offspring result nodes, where the weights are the probabilities of the results. The backed-up cost of a result node is the minimum of the backed-up costs of its offspring test nodes. When the propagation reaches the root node, a result node, the minimizing test node is noted. This node corresponds to the best test.

Although this is an optimal solution, it is infeasible for large problems. The branching factor at the result nodes is then large, since the number of available tests is typically large. The tree size is roughly this branching factor raised to the power d , the average depth of the tree, and d increases with the size of the problem.

This problem can be mitigated by using the gamma miniaverage algorithm [8,13]. This method performs the miniaverage calculation more efficiently by computing the backed-up costs in an order that allows considerable pruning of the tree. Also, Ref. 8 reduces the depth of the tree by starting the propagation not at the terminal nodes of the tree, but at some fixed depth, and estimating the backed-up costs at these nodes with evaluation functions.

We regard a depth of one to be the most practical choice, since the number of available tests is often several hundred. Then the miniaverage method degenerates to simply iterating once over the list of available tests. For each, a weighted average is taken of the evaluation function applied to the UUT states resulting from the several possible test results, where the weights are the estimated test-result probabilities. The minimum valued test is then recommended.

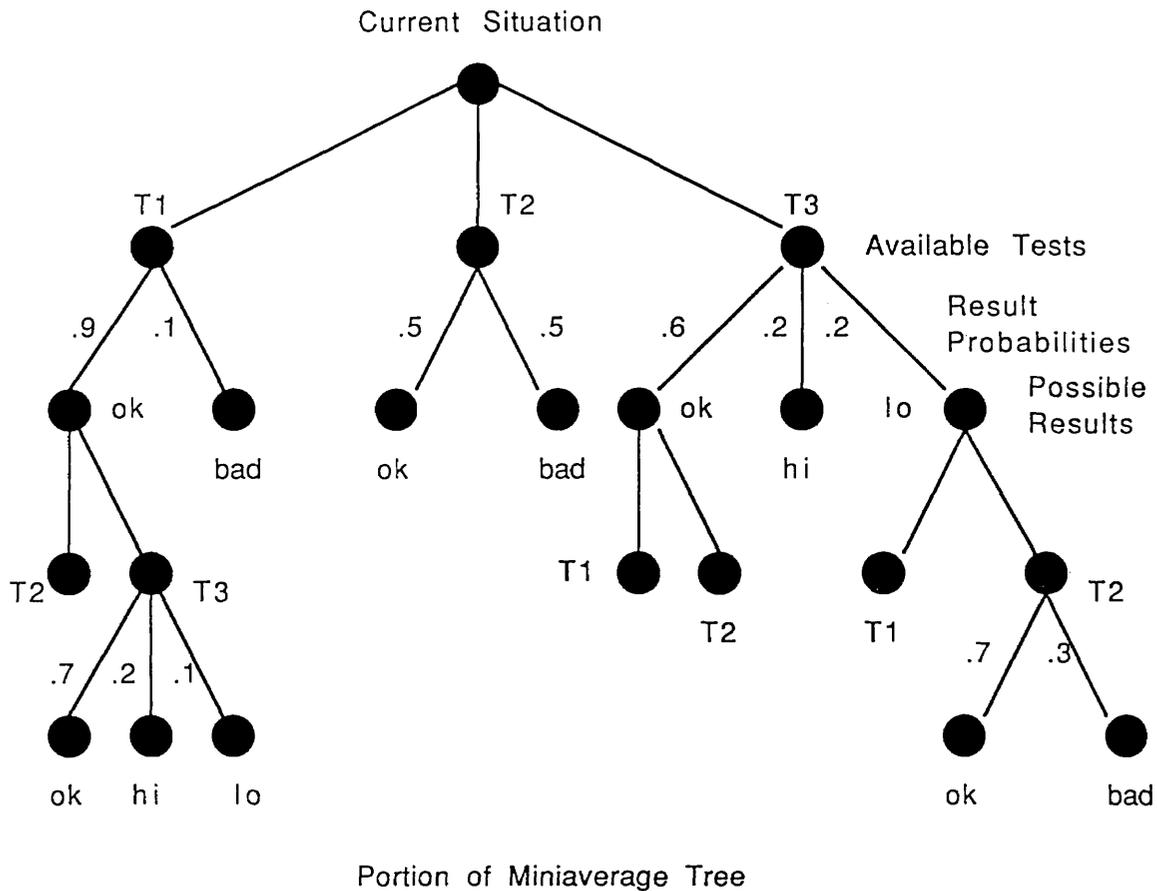


Fig. 5 — Miniaverage methods: optimal but costly

4.2 Information Gain vs Cost

It is difficult to find an evaluation function that directly computes an estimate of the backed up cost of a UUT state without performing the tree search. However, instead of trying to pick a test leading to a minimum expected remaining cost of testing, one can pick a test that maximizes the ratio of the expected information gain for that test to its cost (a predefined test cost). This expected information gain is a weighted average of the reduction in the Shannon entropy of the UUT over the several possible test results, where the weights are the estimated test-result probabilities. The reasoning here is that since the entropy of the UUT is a measure of how much information is required to know its failure state exactly (each module good or bad) and the entropy monotonically decreases at each test, there is no risk of retrogressing from our goal of certainty. Therefore, it is reasonable to seek maximal progress in one step.

Estimates of the probabilities of test results are needed for use in the best-test calculations. The probability of an abnormal (such as high or low) result can be rapidly estimated by looking up the ambiguity set for that result and noting the associated immediate effects of each module. Then an estimate is made of the fraction f_i of the current probability P_i that module i is faulty that is causally related to these immediate effects, by using the certification information for module i . Then the estimated test result probability is $1 - \prod_i (1 - f_i P_i)$. Simply, we estimate the amount of the probability mass of various possible faults that causally lies upstream of the hypothesized test result.

4.3 Heuristic Strategies

The information theoretic approach described in Section 4.2 has the advantage of theoretical rigor, but despite the efficient entropy algorithm elaborated later in Section 4.4 it still can be too slow for real-time applications. This is because one has to process the results of many hypothetical tests to obtain the UUT states whose entropy is to be computed. Therefore, we have experimented with a more heuristic approach. This approach has the advantages of speed and transparency. This is, it allows the development of explanation software. This is of great practical importance for technician's aide and training aide applications. It has the disadvantage that it is difficult to prove anything about its performance.

First we discuss the case in which no test has yet failed. In this case, the objective is to find a test that fails quickly (i.e., at low cost) if there is any fault. If there is not any fault, then it doesn't matter in what order the tests are done; we simply exhaust a given list of tests and declare the UUT not faulty. A reasonable heuristic is then to select a test that maximizes the estimated probability of test failure divided by the test cost. One simple and efficient way to estimate test result probabilities is given in Section 4.2.

Next we discuss the case in which at least one test has failed so far. In this case the objective is to isolate the fault(s) at minimal cost. One of the most useful heuristics is that tests dependent on some but not all of a suspect set (defined below) of modules are powerful. If such a test fails, the suspect set tends to become smaller, essentially by a process of intersection. If it passes, it tends to become smaller by a process of elimination; the certification process reduces the c_i values of the modules on which the test depends. A suspect set can be defined in various ways; for example, as a set of modules whose current fault probabilities satisfy some criterion, or better, as a non-null intersection of some or all of the ambiguity sets of failed tests. Further refining this heuristic, tests are better which not only depend on a subset of the suspect set, but which depend on suspect immediate effects. Better still are the tests that share the same stimulus setup with failed tests that depend on suspect immediate effects. The latter two are directed toward achieving strong certification if the test passes. Note the relevance of Section 3.4. Finally, tests that have low cost are preferred to tests of equal merit but with higher cost. Many variants of these ideas are possible; empirical experience should be used as a guide in specific applications.

4.4 Fast Entropy Algorithm

The UUT entropy discussed in Section 4.2 can be expressed as follows;

$$H(T) = - \sum_{i=1}^{2^n} p_i \log p_i, \text{ where } \log \equiv \log_2. \quad (2)$$

Here p_i is the current probability of the i th complete fault hypothesis (all modules hypothesized good or bad), and n is the number of modules in the UUT. Equation (2) is not in a suitable form for efficient computation because its complexity increases exponentially with the number n of modules. For only 20 modules, $2^{20} = 1048576$ terms would need to be computed. Therefore, we compute this quantity more efficiently by the drastic automatic simplification of Eq. (2) after manipulating the Boolean function T representing the failed test results to date by using the algorithm in Section 3.3. This procedure introduces no approximations but greatly improves running time. It is also generally applicable to the problem of efficiently computing the entropy of a set of statistically independent propositions constrained by an arbitrary Boolean function.

We assume that the description of the state probabilities p_i is that provided by the Boolean algorithm of Section 3.3. That is, the Boolean function B describing the feasible states is represented in a form such that a negation appears at the top level, with a disjunction below that and a conjunction below that. Each conjunct of such a conjunction is either a literal (complemented or uncomplemented) or an expression of the form described in the preceding sentence, recursively. Finally, all disjunctions are of mutually exclusive terms, and all conjunctions are of statistically independent terms (no common literals).

The strategy is to decompose $H(B)$ from the top down by using several mathematical identities about entropy expressions, until it is explicitly computable in terms of arithmetic and transcendental functions of the a priori probabilities a_i of the Boolean literals. Thus, our efficient entropy algorithm is in the same spirit as the efficient probability algorithm of Section 3.3, although the details are different. Gallager [14] provides a good background for the remainder of Section 4.4.

Definitions:

$$A_i \equiv \prod_k f_i(a_k),$$

where

$$f_i(a_k) \equiv a_k, \text{ if the } k\text{th bit of the binary representation of } i = 0$$

$$\equiv \bar{a}_k, \text{ if the } k\text{th bit of the binary representation of } i = 1,$$

where \bar{a}_k denotes $1 - a_k$. Thus the A_i are the 2^n state probabilities with no Boolean constraint.

$A_{S_i} \equiv \prod_{k \in S} f_i(a_k)$, as above, except only a subspace of size 2^l is covered, where l is the size of the given subset S of literals.

$H(B) \equiv - \sum_{B=1} c A_i \log c A_i$, where $1/c = \sum_{B=1} A_i$. This is the Shannon entropy of the probabilities A_i normalized to unity over those states i for which $B = 1$.

$H_n(B) \equiv - \sum_{B=1} A_i \log A_i$. This nonnormalized entropy is not a true entropy, since the A_i are not normalized to unity over the states i for which $B = 1$.

$H_S(B) \equiv - \sum_{B=1} c A_{S_i} \log c A_{S_i}$. This is the entropy over the subspace s spanned by those variables occurring in B . S in A_{S_i} is the set of those variables.

$H_S \equiv - \sum_{i=1}^{2^l} A_{S_i} \log A_{S_i}$, where S is a given subset of l Boolean literals of the n Boolean literals occurring in the complete problem. Note that H_S can be computed in $O(l)$ steps by using the identity *unconstrained independent literals* below.

Entropy identities needed:

negation : $H_n(\bar{B}) = H_n(1) - H_n(B)$.

normalization : $H(B) = cH_n(B) - \log c$, where $l/c = \sum_{B=1} A_i = P(B)$.

independent conjunction : $H_S(B\&C) = H_S(B) + H_S(C)$, if B and C contain no common literals, and are thus statistically independent.

nonoverlapping disjunction : $H_n(B\vee C) = H_n(B) + H_n(C)$, if B and C are mutually exclusive.

unconstrained independent literals : $H_S = - \sum_{i \in S} (a_i \log a_i + \bar{a}_i \log \bar{a}_i)$.

Algorithm:

The following is a statement of the efficient entropy algorithm:

- (1) Express $H(B)$ in terms of $H_n(B)$ using *normalization*.
- (2) Express $H_n(B)$ in terms of $H_n(\bar{B})$ by using *negation*.
- (3) Express $H_n(\bar{B})$ as a sum of terms of the form $H_n(D)$, where D denotes each disjunct of B .
- (4) Express each $H_n(D)$ in terms of $H(D)$ by using *normalization*.
- (5) Express each $H(D)$ as $H_S(D) + H_S$, where S is the set of Boolean literals not occurring in D but occurring in the complete problem.
- (6) Express each $H_S(D)$ as a sum of the $H_S(C)$ over the various conjuncts C of D , using *independent conjunction*.
- (7) For each literal C , set $H_S(C) = 0$.
- (8) For each nonliteral C , apply the whole algorithm recursively to $H_S(C)$, since it has the form of B .
- (9) Expand any H_S terms produced by Step (5), using *unconstrained independent literals*.

Example:

Suppose we wish to compute the entropy $H(\overline{1\&2\&3 \vee 3})$ of the set of 16 states of the four propositions x_1, x_2, x_3, x_4 with a priori probabilities a_1, a_2, a_3, a_4 , respectively, constrained by the Boolean relation $\bar{x}_1 \& \bar{x}_2 \& x_3 \vee \bar{x}_3 = 1$ (Note the shorthand; i for x_i , and $\bar{a} \equiv 1 - a$ for numerical expressions a only).

Step 1:

$$H(\overline{1\&2\&3 \vee \overline{3}}) = cH_n(\overline{1\&2\&3 \vee \overline{3}}) - \log c,$$

$$\text{where } 1/c = P(\overline{1\&2\&3 \vee \overline{3}}) = 1 - \overline{a_1} \overline{a_2} \overline{a_3} - \overline{a_3}.$$

Step 2:

$$H_n(\overline{1\&2\&3 \vee \overline{3}}) = H_n(1) - H_n(\overline{1\&2\&3 \vee \overline{3}}).$$

Step 3:

$$H_n(\overline{1\&2\&3 \vee \overline{3}}) = H_n(\overline{1\&2\&3}) + H_n(\overline{3}).$$

Step 4:

$$H_n(\overline{1\&2\&3}) = \frac{H(\overline{1\&2\&3}) + \log c'}{c'}, \text{ where } 1/c' = P(\overline{1\&2\&3}) = \overline{a_1} \overline{a_2} \overline{a_3}, \text{ and}$$

$$H_n(\overline{3}) = \frac{H(\overline{3}) + \log 1/c''}{c''}, \text{ where } 1/c'' = P(\overline{3}) = \overline{a_3}.$$

Step 5:

$$H(\overline{1\&2\&3}) = H_S(\overline{1\&2\&3}) + H_{\{4\}} \text{ and } H(\overline{3}) = H_S(\overline{3}) + H_{\{1,2,4\}}.$$

Step 6:

$$H(\overline{1\&2\&3}) = H_S(\overline{1}) + H_S(\overline{2}) + H_S(3) + H_{\{4\}}.$$

Step 7 and 9:

$$H(\overline{1\&2\&3}) = H_{\{4\}} = - [a_4 \log a_4 + \overline{a_4} \log \overline{a_4}], \text{ and}$$

$$H(\overline{3}) = H_{\{1,2,4\}} = - [a_1 \log a_1 + \overline{a_1} \log \overline{a_1} + a_2 \log a_2 + \overline{a_2} \log \overline{a_2} + a_4 \log a_4 + \overline{a_4} \log \overline{a_4}].$$

Now, given the a_i values, one can substitute backwards through the above equations to obtain $H(\overline{1\&2\&3 \vee \overline{3}})$. For example, if $a_1 = a_2 = a_3 = a_4 = .5$, we obtain $H(\overline{1\&2\&3 \vee \overline{3}}) = 1 + \log 3$. We can readily verify this special case by noting that this problem has six states of equal nonzero probability $1/6$. Thus its entropy is $6(1/6 \log(6)) = \log(6) = 1 + \log 3$.

The complexity of the entropy algorithm is closely related to the complexity of the probability algorithm of Section 3.3, since both use the same Boolean manipulation algorithm as their first stage.

Whereas the probability algorithm introduces no significant complexity beyond that of the Boolean algorithm, the entropy algorithm does so in Step 9. Every disjunct encountered in the output of the Boolean algorithm invokes an application of *unconstrained independent literals*, which requires only $2l$ operations, where l is the number of literals occurring in the problem but not in the disjunct. Therefore, if there are d disjuncts, $O(dl_{\max})$ computation is introduced beyond the Boolean computation, where l_{\max} is the maximum l over all the disjuncts. Since l_{\max} is bounded by the total number n of literals, this complexity becomes $O(dn)$. But $d = O(a^n)$, the number of terms generated in Step 4 of the Boolean algorithm in Section 3.3. Therefore, the total complexity added by the numerical part of the entropy algorithm is $O(a^n n)$, which is the same as that of the Boolean algorithm. Thus a total worst-case complexity of the entire entropy algorithm is $O(a^n n)$, although, as mentioned in Section 3.3, the typical case is far better than this, often approximately $a^2 n$.

5. FIS: AN IMPLEMENTED DIAGNOSTIC SYSTEM

As mentioned earlier, these ideas have been incorporated into a working diagnostic system that we have named FIS. This section gives a brief overview of FIS from a user's point of view.

The major components of FIS, as well as its two principal users, are illustrated in Fig. 6. We describe the components chronologically as they are used. First, the knowledge engineer, whose principal expertise is a detailed understanding of the type of equipment to be diagnosed, describes the UUT to the computer. This is done from a computer terminal via the knowledge acquisition interface (KAI), producing the UUT description, which is then stored in a file. A principal concern in the design of the KAI is maximizing the ease with which a user with a good electronics background but little familiarity with computer science can generate a description of the UUT that will yield acceptable diagnostic performance in FIS. This description contains information such as a priori rates of failure of the replaceable modules, costs (primarily in time) of setup and test operations, connectivity and qualitative functional descriptions, a set of allowed tests, and a graphics description for displaying a block diagram of the UUT.

The Electronics Library is primarily intended to store partial or complete qualitative functional descriptions of modules that occur frequently in the UUTs being described. The Electronics Library is to be referred to whenever possible while using the KAI to enhance speed. The resulting approximate module descriptions can then be edited by using the KAI. Also, the KAI can be used to add, delete or modify items in the library. After its completion, the UUT description is processed by the knowledge compiler. This produces a file containing the compiled UUT knowledge.

The knowledge compiler transforms knowledge from a form suitable for editing to a form suitable for efficient diagnostic computation. The knowledge compiler is run after the UUT description is completely finished and stores the compiled UUT description in a file.

Once we have compiled UUT descriptions available, the diagnostic reasoning subsystem can be invoked to perform a variety of diagnostic tasks. The diagnostic reasoning subsystem uses a compiled UUT description to dynamically construct and maintain a belief model about what is properly and improperly functioning as test results become known. This belief model in turn is used to find the best test to make next or to recommend the replacement of some module of the UUT.

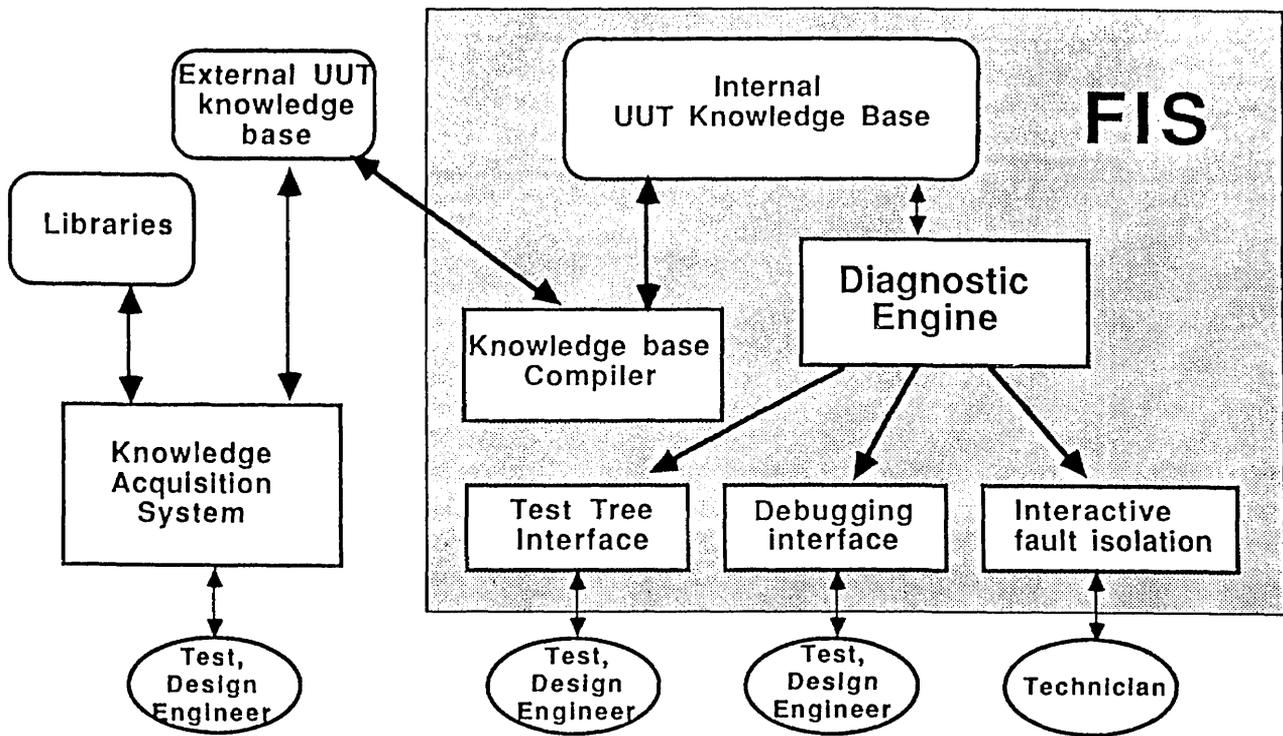


Fig. 6 — A global view of the implemented FIS system

We currently provide two user interfaces to the diagnostic subsystem. The first is a “mixed initiative” interface intended to be used by a technician as an aid during a diagnostic troubleshooting session. In this mode, FIS can

- Update the current beliefs about the UUT based on a technician’s entry of test results;
- Respond to a technician’s query regarding the probability of a fault hypothesis, the merit of a test, the UUT description or the belief state of FIS;
- Make suggestions and recommendations about the next best action to take (further tests or replacements).

The second mode uses the diagnostic subsystem to produce a traditional test tree by invoking the next best action generator, following its advice (hypothetically), and recursively invoking the next best action generator for each of the possible outcomes of the previously recommended action. In this way, large complex test trees are generated automatically with no human intervention.

We have discussed, but have not currently implemented, several other possible user interfaces including a tutorial interface for use in teaching troubleshooting, and a testability interface for evaluating proposed systems early in the design phase.

6. CURRENT APPLICATIONS OF FIS

We have tested and refined FIS on real analog UUTs to assess its performance and practicality and to provide ideas for improvements. We have modelled a radar receiver/exciter subsystem to generate test trees (called diagnostic and functional flowcharts) that are comparable in quality to those written by test programmers. As a second application, we have applied FIS to a Navy sonar system as a technician's aide.

In the radar application, the goal is to demonstrate that FIS can relieve the human test programmer of part of the labor in producing a test program set (TPS), which is a program that controls ATE gear in the automatic execution of a test tree. The automatic test tree generation capability of FIS is used in this application. The part of the human effort that is relieved is the sequencing of the possible tests and the determination of when some module(s) warrant replacement and which ones. In exchange, FIS places a modest knowledge acquisition burden on the user, primarily in the form of causal rules and test cost information. Other parts of this task that we leave under human control are the choice of test equipment and the determination of what constitutes a sufficient set of tests to certify that the UUT is performing correctly, and to do fault isolation. Figure 7 shows a hardcopy version of the more detailed CRT color graphics display of the radar unit. This is useful during the development of the UUT description, when the technician's aide mode is invoked to interactively test the diagnostic performance before test trees are generated.

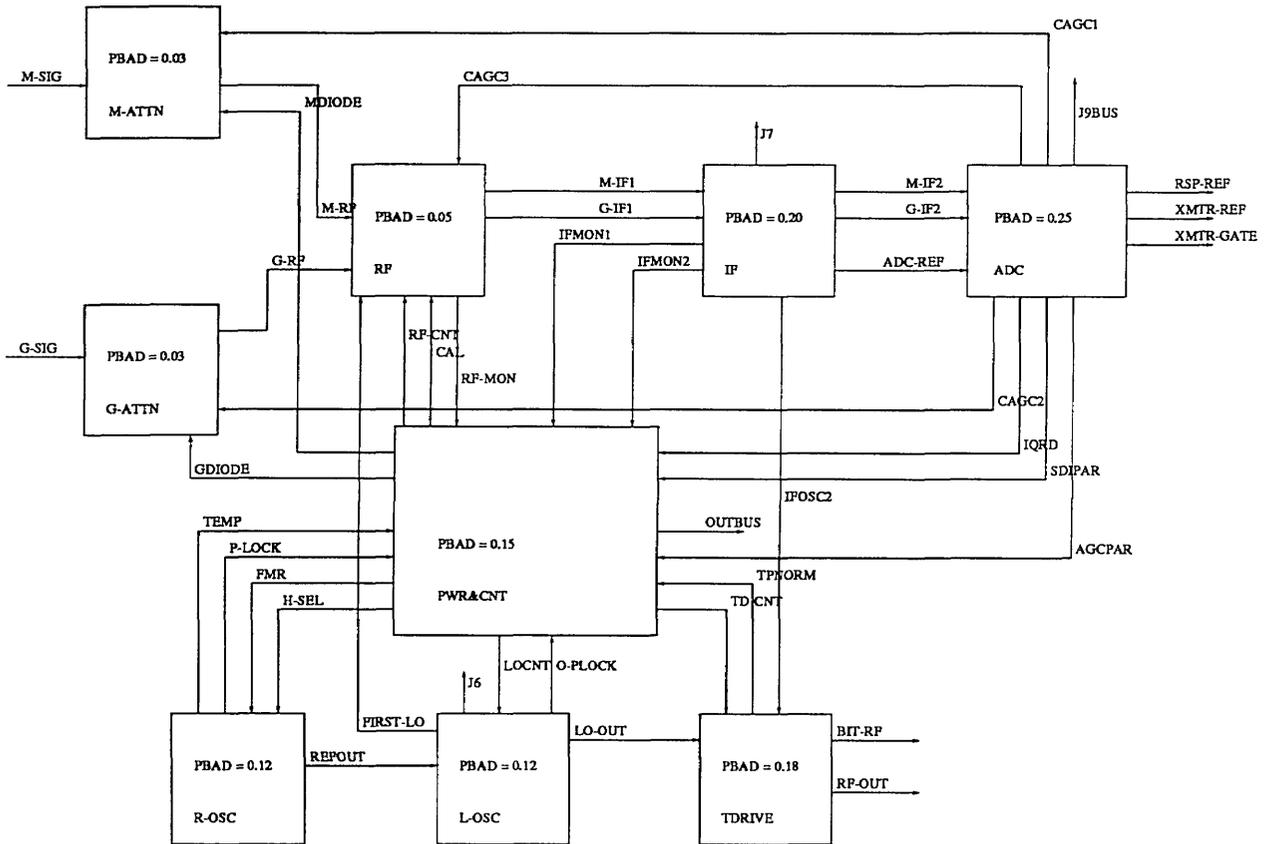


Fig. 7 — An approximation of the color FIS UUT display

This application has suggested some refinements in FIS to make it practical for TPS applications. By improving the speed and accuracy of the best test computation and implementing a more intelligent handling of passed tests, FIS now generates fault isolation trees that are as good or better than the existing ones generated manually.

The second application involves the use of FIS as a maintenance advisor to a technician. It is installed in a rugged portable computer and directs or assists a technician in troubleshooting a complex, primarily analog sonar subsystem consisting of 105 replaceable modules, using manual test equipment. The primary functions are the recommendation of a next best test and the reporting of FIS beliefs after a test is made.

In addition to these activities in which we are directly involved, we have also initiated a technology transfer program in which we provide current versions of FIS to approximately 25 sites in other government laboratories and private industry. Our goal is to make people aware of the potential of AI-based diagnostic systems like FIS and to receive feedback concerning their perceptions of the usefulness of FIS.

7. CONCLUSIONS

The goal of this research effort has been to exploit ideas from the area of AI to build effective diagnostic systems. We have achieved this goal by developing a knowledge representation technique called qualitative causal modeling, which has the property that sufficient behavioral knowledge of a UUT can be captured without high knowledge acquisition costs to allow generation of efficient diagnostic sequences. This representation technique is complemented by a set of efficient algorithms for computing fault probabilities, recommending tests, and making module replacements.

This approach provides two features that are not part of most current diagnostic systems: (a) no single fault assumptions, and (b) the ability to dynamically decide (during fault isolation) what the next best test should be.

In addition, this research has introduced efficient new algorithms for two general problems; (a) computing the probability that a given conjunctive normal form Boolean expression is true, given statistically independent literals, and (b) computing the Shannon entropy of such a set of literals, given that such a Boolean function is true. These were motivated by the goal of avoiding computational bottlenecks everywhere in the FIS system when scaling up to large systems. This has been achieved.

Our current efforts are primarily focused on making minor improvements to FIS based on user feedback. However, we have targeted FIS as an opportunity to exploit some of the recent advances in machine learning. It is clear that any UUT model needs to be continually refined as failure rates change, systems age, etc. Our goal is to use machine-learning techniques to automate this refinement without the need for significant human intervention.

REFERENCES

1. F. Pipitone, "The FIS Electronics Troubleshooting System." *IEEE Computer Magazine*, 68-76 (1986).
2. F. Pipitone, K. DeJong, W. Spears and M. Marrone, "The FIS Electronics Troubleshooting Project," in *Expert System Applications to Telecommunications* J. Liebowitz, ed., (Wiley Co., New York, 1988).

3. J. DeKleer and B.C. Williams, "Diagnosing Multiple Faults." To appear in Artificial Intelligence, North Holland, 1987.
4. M.R. Genesereth, "Diagnosis Using Hierarchical Design Models." Proc. of the Nat. Conf. on AI, pp. 278-283, Pittsburgh, PA 1982.
5. R. Davis, H. Schrobe, W. Hamscher, K. Wieckert, M. Shirley and S. Polit, "Diagnosis Based on Description of Structure and Function." Proc. of the Nat. Conf. on AI pp. 137-142 Pittsburgh, PA, 1982.
6. H. Geffner and J. Pearl, "Distributed Diagnosis of Systems with Multiple Faults," Tech. Rep. R-66 CSD-8600, UCLA Computer Science Dep., Los Angeles, CA 90024, 1986.
7. J. Pearl, "Distributed Revision of Belief Commitment in Multi-Hypotheses Interpretation," Proc. 2nd AAAI Workshop on Uncertainty in AI, Philadelphia, PA, Aug. 1986, pp. 201-209.
8. R.R. Cantone, F.J. Pipitone, W.B. Lander and M.P. Marrone, "Model-based Probabilistic Reasoning for Electronics Troubleshooting," Proc. of the Eighth Internat. Joint Conf. on AI pp. 207-211, Karlsruhe, West Germany, 1983.
9. W.R. Simpson, and H.S. Balaban, "The ARINC Research System Testability and Maintenance Program (STAMP)." Proc. of the 1982 IEEE Autotestcon Conf., Dayton, OH, 1982.
10. E.H. Shortliffe, *Computer-Based Medical Consultations: MYCIN*. (American Elsevier, New York, NY: 1976).
11. R.O. Duda, P.E. Hart, K. Konolige, and R. Reboh, "A Computer-Based Consultant for Mineral Exploration." AI Center, SRI Internat. Menlo Park, CA, 1979.
12. G. Shafer, *A Mathematical Theory of Evidence*. (Princeton University Press, Princeton, NJ 1976).
13. J.R. Slagle and R.C.T. Lee "Applications of Game Tree Searching Techniques to Sequential Pattern Recognition," Communications ACM, **14**(2) 103-110 (1971).
14. R.G. Gallager, *Information Theory and Reliable Communication* (Wiley Co., New York, 1968).