Model-Based Probabilistic Reasoning for Electronics Troubleshooting

Richard R Cantone, Frank J. Pipitone, W. Brent Lander,
and Michael P. Marrone

Navy Center for Applied Research
in Artificial Intelligence
U.S. Naval Research Laboratory - Code 7510
Washington, DC 2037b

ABSTRACT

1N-ATE is an on-going project aimed at developing expert consultant systems for guiding a novice technician through each step of an electronics troubleshooting session. One goal of the project is to automatically produce, given a set of initial symptoms, a binary *(pass/fail)* decision tree of testpoints to be checked by the technician. This paper discusses our initial approach using a modified game tree search technique, the *gamma miniaverage method.* One of the parameters which guides this search technique - the cost of each test - is stored *a priori* The two other parameters that guide it - the conditional probability of test outcomes and the proximity to a solution - are provided by a dynamic model of an expert troubleshooter's beliefs about what in the device is good and what is bad. This model of beliefs is updated using probabilistic *"tp.st-resuLt ► plausible-consequences"* rules These rules are either provided by an expert technician, or approximated by a model-guided *Rule Generator* The model that guides the generation of rules is a simple block diagram of the Unit Under Test (UUT) augmented with component failure rates.

1. Introduction

IN-ATE is an on-going project aimed at developing expert consultant systems for guiding novice technicians through each step of an electronics trouble-shooting session This paper describes the current, Mark I implementation of the IN-ATE system. The basic design of this implementation grew out of a series of preliminary conversations with our domain expert - a master technician. Using a Tektronix Model 465 oscilloscope as a potential Unit Under Test (UUT) for particular examples, these discussions focused on the general *types* of information used in troubleshooting electronic equipment. To achieve an implementation in a reasonable amount of time, we chose for inclusion a proper yet important subset, namely:

circuit topology
cost of individual tests
component failure rates
conditional probability of test results
proximity to a solution

In the remainder of this introduction (Section 1) we describe their importance in electronics troubleshooting, and briefly how they have been incorporated in the current implementation. The final section (Section 4) of this paper briefly discusses how other important types of information (e.g., component function) could be added to the current system. The body cf this paper (Sections *2* and 3) describes in detail the current system - a novel application of heuristic search guided by a rule-based reasoning component that makes use of both traditional, expert-supplied rules and rules approximated by a novel, model-guided Rule Generator.

In principle, a technician can isolate the faulty components of an electronic device with an exhaustive search by testing every discrete, low level component (e.g., resistor or transistor.) This simple linear search can find all faults and in fact is often done when there are relatively few components to test. When there are a great many discrete components, the technician can take advantage of the hierarchical organization that designers generally impose upon complex and sophisticated circuits In this tree-like organization the top-most branches represent the major subsystems, such as the power supply, while the bottom leaves represent the discrete components, such as the individual resistors and transistors. This organization of the search space permits the technician to consider, at each level of abstraction, relatively few components. The IN-ATE system, as well as others (e.g., [Davis, 1962b] and [Genesereth, 1982]), attempts to mimic the troubleshooter by starting the search for faults at the top of the component hierarchy, and then slowly moving down to lower and lower levels.

At each level in this abstraction hierarchy tree, the troubleshooter must decide where to test first, and then based on the outcome of that test, where to test next, etc. In the computer programs that guide Automatic Test Equipment (ATE) these decisions are stored in a binary *{pass/fail)* decision tree. Each node in the decision tree represents the *best* test to make next, given the test outcomes represented by the arcs which lead to it from the root node. The root node represents the initial symptoms. This is similar to the decision trees produced by artificial intelligence *game tree* search algorithms, such as the A* algorithm and the alpha-beta minimax algorithm (see [Nilsson, 1980] for an excellent discussion of these algorithms.) In these decision trees each node represents the best move for a player to make next The arcs which lead to each node represent his opponent's responses to his earlier moves.

The A* algorithm can find an solution tree which is optimal (in cost) for and/or trees, but requires a depth-first search to termination (eg., by looking ahead until a faulty component is found.) When there are many possible tests in a circuit (or many possible moves in a game) the A* algorithm can be impractical* and a shallow, suboptimal search strategy may be required. The *Alpha-beta minimax* method can allow this to be done efficiently, yet the alpha-beta method does not take into account the cost of each test. While the cost of moves in a game may be inconsequential, the cost of a test in troubleshooting can be crucial.

* Even in this case, where there are only two branches at each "and" node (i.e., *pass/fail),* it has in fact been shown that the general problem of finding an optimal solution tree is NP-hard (see [Hyafil and Rivest, 1976] and also [Loveland, 1979].)

Still another problem with using *any* minimax method for this application is the built-in assumption that the "and" branches in the and/or search space are decided by an *opponent* In physical systems, such as electronic devices, the "and" branches are governed by nature, or chance, and can be estimated using probability. This observation by Slagle and Lee motivated their creation of the *gamma miniaverage method* [Slagle and Lee, 1971]. In the miniaverage method, the backed-up value of an "and" branch is not a maximum, but a weighted average - weighted by the conditional probabilities associated with each branch. It allows for efficient pruning of the search space with cut-offs, similar to *alpha-beta* cut-ofls, called *gamma* cut-offs. It also allows for termination of the search when the cost of continuing the search exceeds the cost of simply replacing the suspect components.

While the cost of each test can often be estimated *a priori,* two other parameters that guide the gamma miniaverage method - the conditional probability of a test outcome, discussed above, and a static evaluation function, which estimates the proximity to a solution - are based on the results of earlier tests. Even with only 2 possible outcomes (i.e., *pass/fail)* for a test, for *n* possible tests there may be as many as 3^n possible combinations to consider (since each test could either *pass, fail,* or not be made.) Rather than require that values for these two parameters be established a *priori* for all possible combinations of test results, IN-ATE estimates these values through rule-based *simulation.*

Figure 1 shows the overall structure of the rule-based reasoning component, which is the heart of the IN-ATE system. Test results, real or hypothetical, are introduced one at a time to the Inference Engine Given a test result, the Inference Engine uses a single, triggered rule to update a dynamic, *a posteriori* model of an expert technician's *beliefs* about what in the UUT could be bad and what should be good.

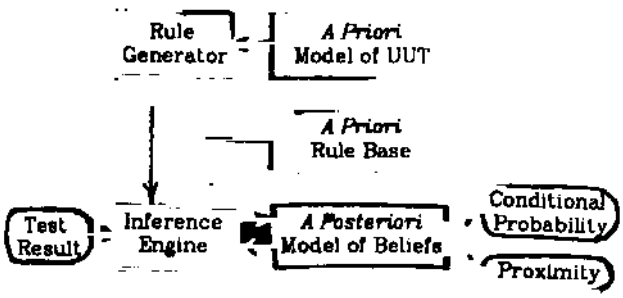

Figure 1: Rule-Based Reasoning in IN-ATE

In the traditional *expert systems* approach, the rules used to update these beliefs would come from a stored, a *priori* collection of expert-supplied *symptom --> possible-cause* rules. Randall Davis [1982a. 1982b] has pointed out that the task of extracting a separate rule from the expert for every conceivable symptom that might arise in a sophisticated piece of electronics equipment is unrealistic. Because of this, Davis stresses the importance of an alternative means with which to reason - a *model* of the system being diagnosed. Davis argues that expert systems should have *both* compiled rules (which he calls "compiled experience") and the ability to reason from a model. IN-ATE incorporates these two knowledge sources through its small expert-supplied, device-specific *a priori* rulebase, and a model-guided, device-independent *Rule Generator.*

The Rule Generator can dynamically produce rules, with the same syntax as the compiled rules, by consulting an *a priori* model of the UUT. The model of the UUT that has been implemented, thus far, is an *augmented block diagram** - augmented in the sense that it contains such non-traditional information as component failure rates Although the rules that IN-ATE generates from its model may be less accurate than expert-supplied rules, the generated rules can fill in whatever gaps exist in the expert-supplied rulebase, and then only when needed. The motivation for this design was a statement by our domain expert that in troubleshooting any sophisticated piece of electronic equipment, including equipment with which he is very familiar, a set of block/schematic diagrams is indispensible He also says that given a *good* set of such diagrams he can troubleshoot essentially *any* piece of analog electronics The Rule Generator tries to capture this generality by producing *approximations* of the rules which normally make up "compiled experience "

The rules in IN-ATE, both compiled and generated, are probabilistic (or, more properly, evidential) rules of belief Each rule is triggered by a single test result. The consequences of each rule are probabilistic assignments of guilt or innocence, given this one particular finding, to various components and lines in the UUT. These probabilistic assignments are used by the system's inference engine to update the system's *a posteriori* model of beliefs about what in the UUT is good and what is bad. Since the system does not make use of a "single fault assumption" that at most one of the UUT's components is faulty, for each component and line the system gathers *both* evidence that is it good *and* evidence that it is bad. And, because of the uncertainty inherent with approximated probabilities, the probabilities associated with these two mutually exclusive possibilities may not necessarily sum to 1. The system combines evidence from separate, independent sources (i.e., test results) using Dempster's Rule [Shafer, 1976]. Dempster's Rule does allow for a sum less than 1, yet reduces to the traditional Bayesian approach when the sum is exactly 1.

Dempster's Rule works fine for combining evidence from sources that are *independent.* But because of circuit connectivity, test results are not always independent. In IN-ATE, non-independent test results are defined as those that are taken from a common path in the UUT circuitry. The interpretation assigned by the inference engine given two non-independent results depends upon whether the tests passed or failed and upon which result is upstream or downstream from the other. For example, if two non-independent results are both bad (good), then only the more useful - the upstream (downstream) result - is retained and interpreted by the inference engine. As another example, when one *passed* test result later clears away the blame from some component - assigned as a result of an earlier *failed* test result - that blame is *removed* from the component and the original *failed* test result is re-interpreted in light of this new information. This form of non-monotonicity has been implemented using the basic truth maintenance facility in the electronics simulation system EL [Stallman and Sussman, 1979].

The current IN-ATE system is made up of two basic components. Section 2 of this paper discusses the probabilistic rule-based reasoning component that guides the heuristic search component discussed in Section 3. Section 4 is a discussion of future plans.

* Fault isolation and testability analysis based upon block diagrams has come to be known as *logic modeling* or *logic model analysis.* One notable example of this is the STAMP system [Simpson and Balaban, 1982] which makes use of Information theory to generate, in a proprietary manner, binary decision trees or the type generated by the IN-ATE system.

2. Probabilistic Rule-Baaed Reasoning

The rules in 1N-ATE incorporate probabilistic measures of belief. Probabilistic measures of belief, or certainty factors, have played an important role in such successful rule-based expert consultant systems as MYCIN [Shortliffe, 1976] and PROSPECTOR [Duda et.al.. 1979]. The expert consultant system for electronics fault isolation ARBY [McDermott, D. and Brooks, R., 1982] totals "amounts" of evidence for and against hypotheses, but since these "amounts" are not related to probabilities, it does so in a very ad hoc manner [Brooks, R., 1982].

Probabilistic measures of belief can play an important role in troubleshooting electronic devices. For example, consider the simple circuit depicted by Figure 2,

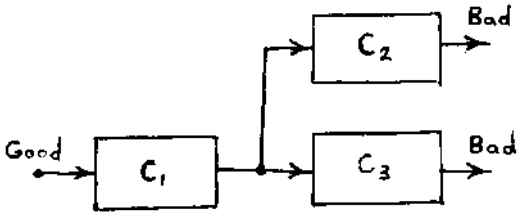

Figure 2

where it is known, through testing, that the input is good and that both outputs are bad. One explanation for these test results could be that both components C_2 and C_3 are faulty. While this is possible, and should not be ruled out, the possibility that C_1 is broken is much more *probable*.

Also, consider the slightly different circuit depicted by Figure 3, where it is known that the input of C_x is good, the output of C_2 is bad. and the output of C_3 is good. Since the output of C_3 is good, its input must be good, which also

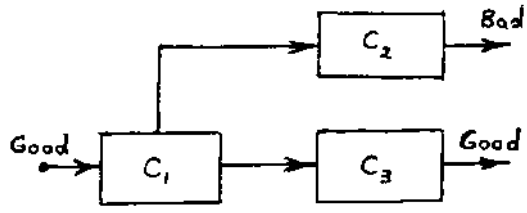

Figure 3

means that one of the outputs of C_x is good. This give *some* evidence that C_1 is good. (If it were known that *both* of the outputs of C_1 are good then we could be *completely* certain that C_1 is good.) So, in this case, C_1 is slightly *less* suspect than C_2, since there is some evidence that C_1 is good while there is none at all that C_2 is good.

2.1. The Inference Rules

In expert consultant systems, relative measures of belief, such as those discussed above, have been quantized using *antecedent -> plausible-consequences* rules. In IN-ATE there are two types of rules - rules for *passed* tests and rules for *Jailed* tests. Also, each rule has two parts - one suggests consequences about components, while the other suggests consequences about other test-points in the circuit. The basic syntax of the rules is described using the following rule schemata:

Rule-1a // there is evidence with probability z that an output of a component is bad*, *then* if there are n components which could have caused this bad output, then for each such component there is evidence (with probability $\frac{x}{n}$) that this component is broken.

Rule-1b For each input I of each of the n components indicated by Rule-1a, *if* m of these n components feed line I, *then* there is evidence (with probability $\frac{m \cdot x}{n}$) that I is bad.

Rule-2a *If* there is evidence with probability x_i that output O_i of a component with n outputs is good, *then* for each line that feeds this component, through some path, there is evidence (with probability $\sum_{i=1,n} \frac{x_i}{n}$) that this line is good.

Rule-2b *If* there is evidence with probability x_i that output O_i of a component with n outputs is good, *then* there is evidence (with probability $\sum_{i=1,n} \frac{x_i}{n}$) that this component is good.

These schemata have actually been used by our domain expert as a guide for writing device-specific rules for the Tektronix 465 Oscilloscope, with the probabilities in parentheses merely suggestions.

Rather than requiring that a domain expert provide the system with a complete and exhaustive set of such rules for every test-point in a specific UUT, the system's *Rule Generator* can approximate missing rules using information stored in the system's *a priori* model of the UUT. If information is missing from the model, the Rule Generator will use the probabilities within the parentheses in the schemata above as defaults. The types of information presently stored in the *a priori* model of the UUT and used by the Rule Generator are: the circuit structure (e.g., topology), sub-structure, and relative component failure rates. For example, the Rule Generator can approximate a Rule-1a rule by making use of information about:

- *Topology:* The grossest estimate of n is the total number of components in the entire circuit. This can be cut down significantly, using simple signal tracing, by considering only those components that lie upstream from the bad output.
- *Component Failure Rates:* The probability $\frac{x}{n}$ is a default probability. The failure rate of each component is used as a weighting factor. For example, if one of the n components has exactly twice the failure rate of each of the other $n-1$ components then the probability of the evidence assigned to it would be $\frac{2x}{n+1}$ and $\frac{x}{n+1}$ for each of the others.

The Rule Generator can custom tailor a Rule-2b rule using information about:

- *Component Sub-Structure:* Some of the outputs of a component could be more informative than others. For example, output i may be affected by 90% of the component's internal sub-components. By making use of such information, the probability assigned is a weighted average rather than a simple average.

2.2. The Inference Engine

All the rules are triggered by a test result. When a new test result is entered into the system, the inference engine first looks in the small rulebase, which comprises compiled experience, for a rule that is appropriate. If there is none, it will call upon the Rule Generator to generate one. With either the retrieved or generated rule, the

• Here, a "bad" input or output line just means that the signal on the line is not the same as when the entire UUT is good.

inference engine can then update the *a posteriori* model of beliefs about what in the UUT is good and what is bad

In updating the model of beliefs, there are two cases that the inference engine must consider. The new test result could either contribute new independent evidence that should be *combined* with earlier evidence, or it could provide new information upon which earlier plausible inferences should be *revised*.

Combining New Independent Evidence

At any point in time, we may have evidence, say for a particular component C, that indicates with probability x_1 that C is good and y_1 that C is bad. Since we allow that $z_1 = 1-(x_1+y_1)$ may be greater than 0, these are not classical Bayesian probabilities.

When a new piece of independent evidence is introduced concerning C, the new evidence can be combined with the old evidence using Dempster's Rule [Schafer, 1976].* For example, suppose that new evidence suggests, with probability x_2, merely that C is *good*. The uncertainty z_2 associated with this evidence is simply $1-x_2$ (since, in this case, $y_2=0$.) According to Dempster's Rule, the updated probabilities x_3 (good), y_3 (bad), and z_3 (uncertain) would then be computed as follows. First, using a cross product, let:

$$x'_3 = x_2 \cdot x_1 + x_2 \cdot z_1 + z_2 \cdot x_1$$

$$y'_3 = z_2 \cdot y_1$$

$$z'_3 = z_2 \cdot z_1$$

$$N = (x'_3+y'_3+z'_3)^{-1}$$

and then, through re-normalization:

$$x_3 = N \cdot x'_3$$

$$y_3 = N \cdot y'_3$$

$$z_3 = N \cdot z'_3$$

(There are similar equations for combining new evidence that C is bad.)

For example, suppose that at some moment the total probability that component C is good is $x_1=1/3$ and $y_1=1/3$ that C is bad (with uncertainty $z_1=1/3$.) If new evidence is introduced with certainty $x_2=1/3$ that C is good (and with uncertainty $z_2=2/3$), then the updated probabilities would be $x_3=1/2$ that C is good and $y_3=1/4$ that C is bad (with uncertainty $z_3=1/4$.)

The associativity and commutativity of Dempster's Rule are very important in this application since the results of independent tests should be independent of the order in which the tests were made. The Rule's sound treatment of uncertainty is also important here, since in troubleshooting *exact* probabilities are often difficult or impossible to calculate.

Revising Earlier Plausible Inferences ("Shifting Blame")

Inferences made earlier in a troubleshooting session might need to be re-evaluated in light of new information. For example, Rule-1a rules distribute the blame for a fault among n components. If m (for $m \leq n$) of these are later cleared by a *passed* test result, then the blame assigned to these m components should be removed from them and re-distributed among the other n-m components. Similarly, when the plausible blame attributed to some line L (by an instance of Rule-1b) is later *confirmed* by a *failed* test result, then the blame that had been assigned to the m components that line L feeds (by the corresponding instance of Rule-1a) should be removed* and re-distributed to the other n- m components using a *new* instance of Rule-1a - one triggered by the *new* test result.

This form of non-monotonicity has been implemented using the basic truth maintenance system incorporated in the electronics simulation system EL [Stallman and Sussman, 1979], Borrowing the terminology of Stall man and Sussman. at each moment an assertion can be believed with some certainty, and labeled *in,* if there is some well-founded support behind it (i.e., a test result), otherwise it LS labeled *out* An assertion that is less than completely certain can move from *in* to *out* by the introduction of contradictory evidence that is completely certain. To implement this capability we store with each discrete piece of evidence that is less than completely certain the test result and rule that originally introduced it to *in.* This LS the only information necessary for removing and re-evaluating the plausible consequences of a rule in light of new information

3. Deciding the Best Test

The *a posteriori* model of beliefs has two basic purposes Firstly, the system can present to the user a list of suspect components, ordered by their current probability of being faulty This can either be done at any time, upon request from the user, or automatically, when the maximum such probabability exceeds some user established threshold Secondly, the system can use the model of beliefs, in a *hypothetical mode,* to help decide the best test for the user to make next If there are no initial symptoms the best test model of beliefs can be used to construct, a *priori,* a binary decision of best tests to be performed at each step of such a troubleshooting session If there *are* initial symptoms, a custom tailored binary decision tree which takes these initial "test results" into account can be computed at the beginning of the troubleshooting session (or dynamically recomputed if the user later contributes, through his own initiative, other test results)

The best test is basically that test which, for its cost, will bring the system closest to isolating the fault to a single component. Since there are two possible outcomes of each test (i.e..pass *or fail),* the system must consider both of them separately. It does this by simulating the effect that each such outcome would have on its model of beliefs. A heuristic static evaluation function is applied, in each case, to the model of beliefs to estimate how far the system would then be to having isolated the problem to one faulty component.

We have applied the gamma miniaverage method in a straight-forward manner (and refer the reader to [Slagle and Lee, 1971] for a detailed discussion of the method itself.) The nodes in the search tree represent, on alternating levels, the possible tests that can be performed and the potential outcomes o_4 of the possible tests. A static evaluation function G applied to an outcome node on the frontier of the search represents the loss, or cost, of terminating at that point. The value of the static evaluation function G which we have empirically chosen is the size of the component ambiguity set (the still suspect components.) G could also represent the total cost of replacing all of the components in the ambiguity set, if this data were known.

* One notable use of Dempster's Rule has been by Garvey for integrating sensor information [Garvey, 1981]. It has also influenced Friedman's Extended Plausible Reasoning [Friedman, 1981].

* Although that blame is removed from these components, the components are not cleared since this test result provides no evidence that they are good, but merely that one of their inputs is bad.

To calculate the backed-up value of the "and" node which represents the cost R of terminating the search after performing test T_k, the static values $G([o_i, T_k, E])$ for each possible outcome o_i of T_k (where $[o_i, T_k, E]$ represents the model of beliefs after outcome o_i of test T_k is incorporated into the model of beliefs E) are combined with the cost C_{T_k} of performing test T_k and the conditional probabilities $P_{T_k}(o_i \mid E)$ that test T_k will have outcome o_i (given the evidence E that has been accumulated thus far) using the formula

$$R(E, T_k) = C_{T_k} + \sum_i P_{T_k}(o_i \mid E) \cdot G([o_i, T_k, E])$$

An approximation of the *a posteriori* probability $P_{T_k}(o_i \mid E)$ is maintained in the system's *a posteriori* model of beliefs (see, for example, rule schemata Rule-1b and Rule-2a.) The value of $G([o_i, T_k, E])$ is calculated by statically evaluating the *a posteriori* model of beliefs. The risk R basically represents the total of the actual cost and the potential cost if the troubleshooting session were to be terminated at that point.

Since we want to choose the test T_k of minimum risk $R(E, T_k)$, the backed-up value of the "or" node which represents this decision is

$$R(E) = \min_k R(E, T_k).$$

To allow for multiple levels of look-ahead, this backing-up process can be repeated as many levels as resources permit Not only can these backed-up miniaverage values be used for estimating the best test to make next, but using gamma cut-offs, this search can be done efficiently.

4. Future Work

The algorithms have been implemented in Franz LISP on a VAX 11/780 computer and produces reasonably good decision trees for block diagrams that include series, parallel, and feedback circuits. For example, in the case of a simple series circuit where all test costs are equal, the system reduces to the standard half-split method (i.e., binary search.) In the more realistic case of a 2-dimensional circuit, where there is a real ambiguity about where the "middle" is, the system performs much better than the half-split method. In the next few months this basic system will first be applied to troubleshooting a standard oscilloscope, the Tektronix Model 465, and its performance will be evaluated by our domain expert. To test the generality of the system, it will then be applied to troubleshooting a sophisticated piece of military electronics, such as the WLQ-4 Intercept Receiver.

We will also be adding a learning component, in the style of the Self-Improving Diagnostics system SID [Hughes Aircraft Corp.. I960], to retain and periodically refine generated rules. The probabilities within a rule, Initially assigned by the Rule Generator, would be changed to reflect the actual distribution encountered by the system over time.

A more difficult improvement that we will be exploring is a more *informed* Rule Generator. We will attempt to incorporate an ability to reason with knowledge about component *function* (in addition to circuit topology and component failure rates.) Also the Rule Generator should know *when* to use assumptions that narrow the set of possible faults (e.g., that faults are *always* upstream from a bad line) but are *not always* correct (see [Davis, 1982b]).

ACKNOWLEDGEMENTS

We would like to thank our domain expert, master technician Mr. Mark Hulbert, for his time and patience, Prof. Kenneth De Jong. Prof. Michael Gaynor and Dr. James Slagle for their advice; Dr. Randall Schumaker (NAVA1R) for directing our attention toward the problem of automatic test program generation (ATPG), and Dr. Jude Franklin, especially, for introducing us to the general problem of electronics troubleshooting

REFERENCES

[I] Brooks, Ruven, personal communication (June, 1982).

[2] Davis, Randall, "Expert Systems: Where are we? And where do we go from here?", pgs. 3-22, *The AI Magazine,* in, Number 2 (1982a).

[3] Davis, Randall, et. al ."Diagnosis Based on Description of Structure and Function", in *Proceedings of the 1962 National Conference on Artificial Intelligence (AAA1-82),* (1982b)

[4] Duda. R.O, Hart, P.E., Konolige, K. and Reboh, R., "A Computer-Based Consultant for Mineral Exploration," Artificial Intelligence Center, SRI International. Menlo Park, CA(Sept.. 1979).

[5] Friedman, Leonard, "Extended Plausible Inference," *Proceedings, International Joint Conference on Artifical Intelligence (IJCAI-81),* (August, 1981)

[6] Garvey, Thomas D., et. ad., "An Inference Technique for Integrating Knowledge from Disparate Sources." in *Proc. IJCAI-81.,* (August, 1981).

[7] Genesereth, Michael R., "Diagnosis Using Hierarchical Design Models." in *Proc. AAAI-82,* (August, 19e2)

[8] Hughes Aircraft Corp., "Knowledge Based Systems (KBS) Study for Advanced MATE Systems," HAC Report No. FR80-75-869 (May, I960).

[9] Hyafil, Laurent and Rivest, Ronald L., "Constructing Optimal Binary Decision Trees is is NP-Complete," pgs. 15-17, *Information Processing Letters,* 5, 1 (May, 1976).

[10] Loveland. Donald W., "Selecting Optimal Test Procedures from Incomplete Test Sets," in *Proc. First Intern*. Symposium on Policy Analysis and Information Science,* Duke University, Durham, N.C., pgs. 228-235 (June, 1979)..

[II] McDermott, Drew and Brooks, Ruven, "ARBY: Diagnosis with Shallow Causal Models", in *Proc. AAAI-82,* (August, 1982).

[12] Nilsson, Nils J., *Principles of Artificial Intelligence,* Tioga Pub. Co., Palo Alto, CA (1980).

[13] Shafer. G., *A Mathematical Theory of Evidence,?rmce\.on* University Press, Princeton, NJ (1976).

[13] Shortiiffe, Edward H., co*mputer-Based Medical Consultations: MYCIN,* American Elsevier Pub. Co., New York, NY (1976).

[14] Simpson, William R. and Balaban, Harold S., "The ARINC Research System Testability and Maintenance Program (STAMP)", in *Proc. 1982IEEE AUTOTESTCON Conference,* Dayton, Ohio (October, 1982).

[15] Slagle, James R. and Lee, Richard C.T., "Application of Game Tree Searching Techniques to Sequential Pattern Recognition," p. 103-110, *Communications of the ACM,* 14, 2 (February, 1971).

[16] Stallman, Richard M. and Sussman. Gerald J., "Problem Solving About Electrical Circuits," in *Artificial Intelligence: An MIT perspective,* Winston, Patrick Henry and Brown. Richard Henry (eds), The MIT Press, pgs.33-91 (1979).