ICOT Technical Memorandom: TM-0957

TM-0957

An Adaptive Model-Based Diagnostic System

by
Y. Koseki, Y. Nakakuki & M. Tanaka (NEC)

September, 1990

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191 ~5
Telex ICOT J32964

Institute for New Generation Computer Technology

An Adaptive Model-Based Diagnostic System

Yoshiyuki Koseki, Yoichiro Nakakuki, Midori Tanaka

C&C Systems Research Laboratories, NEC Corporation
4-1-1 Miyazaki Miyamae-ku, Kawasaki 216, JAPAN

Abstract

This paper describes an experimental diagnostic system which explores adaptability with learning capability from its experience. It takes a model based reasoning approach, utilizing experiential knowledge at the same time. Experiential knowledge consists of cached symptom-failure association rules and a probability model of the target system components. It is able to generate and select appropriate tests according to the failure probability distribution. With this capability, when the system has had a similar experience in the past, it can diagnose the failure faster and more efficiently by suggesting better tests. Therefore, this system gives a solution to both the knowledge acquisition bottleneck problem of rule-based systems and the efficiency problem of model-based systems. It is being implemented in ESP language on a prolog machine PSI-II. The effectiveness of the technique is shown by an experimental result.

1 Introduction

Since the creation of the MYCIN system [Shortliffe 1976], most of the expert systems, especially diagnostic systems, have incorporated the idea of representing their knowledge in a form of symptom-failure association rules. Those expert systems that take rule-based approach have two major inherent disadvantages. First, those systems lack robustness because they cannot deal with unexpected cases not covered by rules in its knowledge base. Second, their knowledge bases are expensive to create and maintain. Rule-based systems use symptom-failure association rules gathered by interview from experts experienced in a domain. Interviewing human experts requires substantial effort and it sometimes becomes bottleneck of the system development. Debugging and maintenance is difficult, since different kinds of knowledge, such as target device knowledge, failing component behavior and probability of failures, are condensed into a uniform rule-base. Therefore, even a small design change in a target device may require thorough review of the knowledge base.

There has been a series of research to tackle those problems. The most distinct ones are on *model-based methods*, i.e. *first-principle methods*. Model-based methods use design descriptions, such as the structure and behavior descriptions. Model-based methods include conventional Boolean-Logic-level diagnostic methods. However, those algorithms specialized in Boolean logic-level, suffer from a scaling problem in dealing with modern complex devices which have millions of logic-level subcomponents. Recently introduced model-based methods, such as [Genesereth 1984], [Davis 1984], [deKleer and Williams 1987], [Reiter 1987], and [Poole 1986], provide declarative device-independent design representation languages and device-independent diagnostic procedures. Since they are not restricted to Boolean logic-level, they are capable, utilizing hierarchical designs, of diagnosing complex devices.

Model-based methods are more robust than rule-based systems, because the diagnosis methods with correct design models can cover all of the possible failure modes for the components. Their knowledge bases are less expensive to create and flexible in regard to design changes since they are a straightforward representation of designs which are likely to be found in modern CAD environments.

However, model-based diagnostic systems are generally not as efficient as rule-based ones since they require more complex computation. Furthermore, they are not always able to pinpoint a failing component from the available symptom information and sometimes require many tests to reach a conclusive decision. This is because they lack heuristic knowledge which human experts usually utilize.

The authors have been working on a research to explore a general architecture to realize an adaptive diagnostic agent and introduced its basic architecture [Koseki 1989]. This paper describes an experimental system based on the architecture. It is a model based diagnostic system which realizes adaptability with learning capability from its experience. The experiential knowledge is represented in a form of cached symptom-failure association rules and a probability model of the target system components. With this experiential knowledge, it is able to diagnose a failing component faster with a fewer tests than pure model-based systems can.

In this paper, the authors are making a *single fault assumption*. That is, it is assumed that there is only one malfunctioning component in a failing device. This assumption makes it easy to rule out certain failure hypotheses using test results. The author also makes a *non-intermittency fault assumption*. That is, it is assumed that the behavior of a failing device does not change during the diagnosis process. This enables localizing a fault by testing the device after the symptom appears. In addition, the method is based on the nature of the fault locality. It is usually true that most of the faults which occur in a device are local to some particular subcomponents. That is to say, a failed subcomponent tends to fail again in a similar manner in the future.

This paper is organized as follows. The section to follow gives an overview of the system describing its structure and general flow. The next section explains reasoning methods including learning algorithms. Experimental results, showing promising data, are shown in the following section. The concluding last section discusses the future direction of this research.

2 System Overview

2.1 Structure

We can observe two kinds of intelligent behavior in maintenance expert's diagnostic procedure. First, they can quickly identify a faulty component with a little information according to their experience, if they have solved a similar problem in the past. Second, even if a novel symptom arises, the expert can reach a conclusion, by consulting with other information sources, such as design description manuals. They can reason which component might have gone wrong and caused the symptom to appear by knowing how the system is supposed to work. Interestingly, they can remember the experienced case, and when they again confront a similar case in the future, they can diagnose the failure using fewer tests without consulting the design description.

To realize those kinds of intelligent behavior, the system consists of several modules as shown in Figure 1. The knowledge base consists of *design knowledge* and *experiential knowledge*. The design knowledge represents a correct model of the target device. It consists of *structural description* which expresses component interconnections and *behavior description* which expresses component behavior. The experiential knowledge contains *symptom-failure association rules* and *component failure probability* for known component failures.

The *diagnosis module* utilizes those two kinds of knowledge and manipulates a *suspect-list(SL)*. A suspect-list represents a set of suspected component mis-behaviors at a state of a diagnosis session. The diagnosis module calculates an initial suspect-list from a given initial symptom using both of design knowledge and experiential knowledge. Every time the system gets a test result, it eliminates a subset of the suspect-list and reduces its size. The *test pattern generator/selector* gives an appropriate test to reduce the suspect-list according to the probability distribution. The *learning module* takes an experienced case to generate a symptom-failure association rule to represent a general description of the case. This rule acts as a cache for the design knowledge. It also records the number of the failures for each experienced case in terms of the failing components. This structure is novel in the sense that it utilizes both model-based and experience-based knowledge and it acquires knowledge incrementally from the past experience.

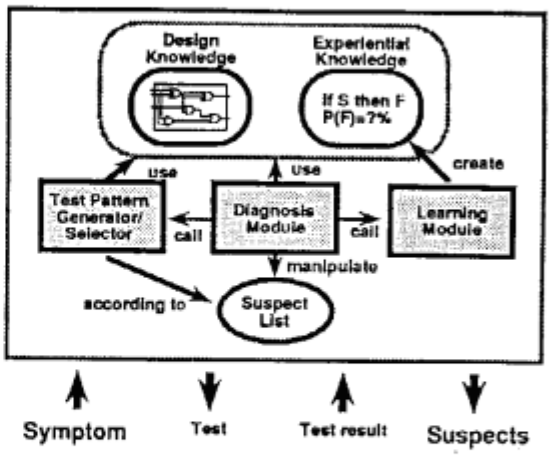

Figure 1: Structure of the System

2.2 Diagnosis Flow

The general flow of the diagnostic system is shown in Figure 2. The system keeps a set of suspected component mis-behavior as a suspect-list. And it takes *eliminate-not-suspected* strategy to reduce the number of the suspects in the suspect-list, repeating the test-and-eliminate cycle.

It starts with getting an initial symptom. A symptom is represented as a set of input signals of the target device and an incorrect output signal observed. It calculates an initial suspect list from the given initial symptom. It searches through symptom-failure association rules in the experiential knowledge base and if it finds a matching rule it gives a corresponding set of suspects. If it does not find a matching rule in the experiential knowledge, it performs a model-based reasoning to obtain a suspect-list using a correct design model and an expected correct output signal. To obtain an expected correct output signal for the given inputs, the system carries out simulation using the correct design model. The system also obtains a generalized symptom-failure association rule at the same time and stores it for the future use.

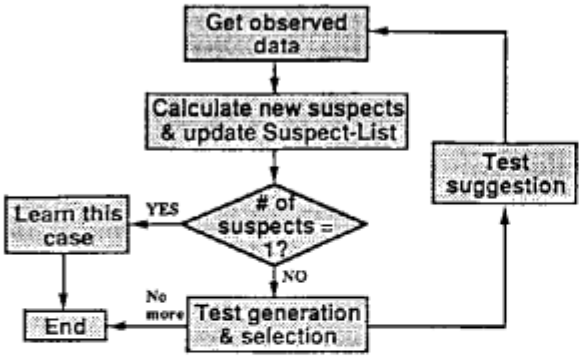

Figure 2: Diagnosis Flow

After obtaining the initial suspect-list, the system repeat a test-and-eliminate cycle, while the number of suspects is greater than one and an effective test exists. A set of tests is generated by the test pattern generator for the failing behaviors remaining in the suspect list. Among the generated tests, the most cost effective one is selected as the next test to be performed. The selected test is suggested and fed into the target device. By feeding the test into the target device, another set of observation is obtained as a test result and is used to eliminate the non-failure components. This test-and-eliminate cycle is repeated until the number of suspects is significantly reduced or until there is no test available. Finally the system suggests a repair of the remaining suspects.

From the successful cases where the suggested component was in fact the failing device, the system adds up the failure counter for the component and renews the probability distribution.

3 Reasoning Methods

This section briefly describes frameworks for reasoning methods of the system. More detailed discussion on the design model representation and model-based reasoning can be found in [Koseki 1989]. In the examples throughout the paper, all descriptions are expressed in the form of first order predicate calculus, with the conventions used in [Genesereth and Nilsson 1987].[1]

3.1 Design Representation

The design description for the target device consists of descriptions of its structure and its behavior. The structure description consists of a list of subcomponents and interconnections among them. The connections are represented as the facts like shown in the following example. The dash character - is used as an infix operator to represent a subport for a component. The following fact states that output signal Out of component Comp1 is connected to input signal In of component Comp2.

Conn(Comp1-Out, Comp2-In).

The behavior descriptions are specified by a set of horn clauses. For example, the behavior for an AND gate, with zero time delay, is given by the rules given below. The first rule states that output Out of gate AND is 1 at time t, if input In1 of AND is 1, input In2 of AND is 1, and AND is working correctly for inputs In1 and In2. The remaining rules state the behaviors for other combinations of input signals.

Val(AND-Out, 1, t) ⇐
  Val(AND-In1, 1, t) ∧ Val(And-In2, 1, t)
  Working(AND, [1, 1]).
Val(AND-Out, 0, t) ⇐
  Val(AND-In1, 0, t) ∧ Val(And-In2, x, t)
  Working(AND, [0, x]).
Val(AND-Out, 0, t) ⇐
  Val(AND-In1, x, t) ∧ Val(And-In2, 0, t)
  Working(AND, [x, 0]).

3.2 Reasoning Framework

The diagnosis inference and the test pattern generation are based on an abduction inference method which is used in DART [Genesereth 1984] and SATURN [Singh 1987]. This method is functionally equivalent to the *constraint suspension* technique of [Davis 1984] and the Theorist framework of [Poole 1986]. This inference procedure uses a domain theory, a goal, and a specification of facts called *assumables*, which can be assumed to prove the goal. It deduces a set of assumable facts which, together with the domain theory, entails the goal.

To learn symptom-failure association rules and to generate generalized rules for caching, the inference method is extended using *Explanation Based Generalization* (EBG) techniques [Mitchell et al. 1986, Dejong and Mooney 1986]. The extended inference procedure is called REBG. REBG generates a generalized compact rule expressed with a fixed vocabulary called *operational criteria*, while it performs the abduction inference. This method performs two functions in addition to the abduction inference. First, it examines the given inference explanation and re-expresses it in a compact rule form in terms of operational criteria. The operational criteria depicts predicates which can be used to form the rule. The generated rules can be used as a cache to the inference procedure, contributing to the performance gain. Second, it generalizes the obtained rule by replacing instantiated variables by as many variables as possible. This function makes the learned rule applicable to other similar cases.

[1] The lower case letters stand for universally quantified variables, the upper case letters for constants, ∧ for logical conjunction, ∨ for logical disjunction, ¬ for negation, and ⇐ and ⇒ for logical implication.

3.3 Model-Based Suspect Calculation

The input to the model-based suspect calculation procedure is a design description D and a symptom description $< I, O >$, where I is a conjunction of input (controllable) port values and O is an expected output(observable) port value. The procedure produces a set of possible failures (suspects) in terms of working-conditions W for the components. The domain theory corresponds to design description D and input port values I. Assumable facts specify working conditions W for the components, and the goal is output port value O. A set of assumable working conditions W is then computed so that $I \wedge D \wedge W \Rightarrow O$ is true. This is equivalent to $I \wedge \neg O \wedge D \Rightarrow \neg W$. Since design D is assumed to be correct, this formula corresponds to the diagnosis rule $I \wedge \neg O \Rightarrow \neg W$. Note that this procedure does not require any subcomponent failure models. This design was considered important because of the extreme difficulty of assuming failure models, such as stuck-at-faults, in general devices.

3.4 Knowledge Caching by REBG

Model-based reasoning involves finding a proof for the correct output value by examining the structure and behavior of the device. One way to avoid this computation is to compile all of the cases into symptom-failure association rules in advance. This approach may only be feasible if probable subcomponent misbehaviors are known in advance.

As an alternative to compiling all the cases, this system partially compiles and caches its model-based knowledge into compact generalized rules for the cases it has experienced. The REBG procedure generates such rules while it computes facts to be assumed. It is essential to store generalized rules rather than instance rules in order to cover the entire class of symptoms related to cases experienced.

3.5 Test Generation

The test-pattern generation algorithm is based on the work of SATURN system [Singh 1987]. It is an algorithmic test generation system, similar to the D-algorithm [Roth et al. 1967]. However, it works on designs specified at arbitrary abstraction levels rather than at the Boolean logic level. The procedure takes as inputs the design description D and the subcomponent input-output behavior to be tested $< Is, Os >$. It propagates Is to a set of controllable input values I, and Os to a set of observable output values O, by using the same method as the one used for suspect calculation.

3.6 Test Selection

The former implementation of the system described in the reference [Koseki 1989] selects a test generated for the most suspected component with the highest probability. However, this method does not always select the most effective one, and it does not consider cost for the test execution. The current implementation has a test selection mechanism which selects a test with the highest effectiveness per cost. This section describes this mechanism.

To compute test effectiveness, the system uses probability distribution for each component. The mechanism employed in the system is basically same as the one found in the reference [deKleer and Williams 1989]. It is so called *minimum entropy* technique where entropy is calculated from the fault probability for each suspected component. The entropy gain (information gain) is used to evaluate a test to be carried out next. That is, it calculates the difference between the entropy before the test execution and the expected entropy after the test for each possible test and it selects the one with the highest gain.

First, we define an entropy $E(SL)$ of a suspect-list SL during a diagnosis session, in terms of the estimated probabilities of the component failures. Let SL denote the set of suspected components,

$$SL = \{S_1, S_2, \ldots S_n\},$$

and let $p_1, p_2, \ldots p_n (\sum P_i = 1, \ p_i > 0)$ be failure probabilities of suspects $S_1, S_2, \ldots S_n$.

Then an entropy E(SL) is defined as

$$E(SL) = -\sum_{i=1}^{n} p_i \log p_i.$$

Let a set of suspect lists after executing a test T be denoted by $SL_1, SL_2, \ldots SL_k$, when T has multiple possible results, $t_1, t_2, \ldots t_k$. And let occurrence probability for a test result t_j be expressed by q_j. Then the effect of the test T is defined as

$$gain(T) = \sum_{j=1}^{k} q_j (E(SL) - E(SL_j)).$$

This function is based on the same idea as the one for the gain function of Quinlan's decision tree learning algorithm ID3 [Quinlan 1986]. The system calculates $gain(T)$ for all of the available tests and decides the one with the highest value as the most effective one.

Let us see an example following this method. Suppose there is a test T with two possible results t_1 and t_2, and the relation between suspect-list SL and test results (t_1, t_2) of the test T is given as in the table below.

Suspect	S_1	S_2	S_3	S_4
Probability	0.1	0.2	0.3	0.4
t_1	○	○	○	×
t_2	×	○	○	○

For example, if the test result is t_1, S_4 is not suspected and either $S_1, S_2, or S_3$ is failing. Therefore, if S_1 is the failing component, test result must be t_1. Similarly, if S_4

is failing, the result should be t_2. In other cases, such as when either S_2 or S_3 is failing, we assume the same probability for t_1 and t_2. Therefore, the estimated probability q1 and q2 for test results t1 and t2 are presumed to be as follows.

$$q_1 = 0.1 + \frac{0.2 + 0.3}{2} = 0.35$$

$$q_2 = \frac{0.2 + 0.3}{2} + 0.4 = 0.65$$

Here, the system assumes that the probability distribution after the test remains same, therefore, it predicts the suspect list $SL1$ and $SL2$ for each test result, as follows.

SL_1

Component	S_1	S_2	S_3
Probability	0.17	0.33	0.5

SL_2

Component	S_2	S_3	S_4
Probability	0.22	0.33	0.45

Therefore, the entropies for each SL are calculated as
$E(SL) = -0.1 \log 0.1 - 0.2 \log 0.2 - 0.3 \log 0.3 - 0.4 \log 0.4 = 1.85$
$E(SL_1) = -0.17 \log 0.17 - 0.33 \log 0.33 - 0.5 \log 0.5 = 1.45$
$E(SL_2) = -0.22 \log 0.22 - 0.33 \log 0.33 - 0.45 \log 0.45 = 1.52.$
Finally, the gain of the test T is obtained as
$gain(T) = 0.35(1.85 - 1.45) + 0.65(1.85 - 1.52) = 0.35$ (bits).
This gain indicates the effectiveness of the test.

In addition to this value, we should consider the test execution cost to select a cost effective test. For example, suppose there are two test sequences TS1 and TS2, when SL is A, B, C, D.

TS1 The first test can pinpoint a faulty component. The test takes 10 minutes.

TS2 The first test can narrow down the SL to A, B or C, D, and the second test can pinpoint a faulty one. Each test takes 4 minutes each.

The first test in the sequence TS1 is more effective than the tests in TS2. However, TS2 is more desirable because it takes less time than TS1 does in overall. As we can see in this simple example, the system should consider the test cost in order to select a test to be performed next. This is because the goal is to minimize time for the overall diagnosis procedure. Therefore, the system selects a test, according to the following evaluation function.

$$\frac{gain(T)}{cost(T)}$$

In the current implementation, the test costs must be given in advance. In general, other factors, such as amount of service interruption and test equipment availability might have to be considered at the same time.

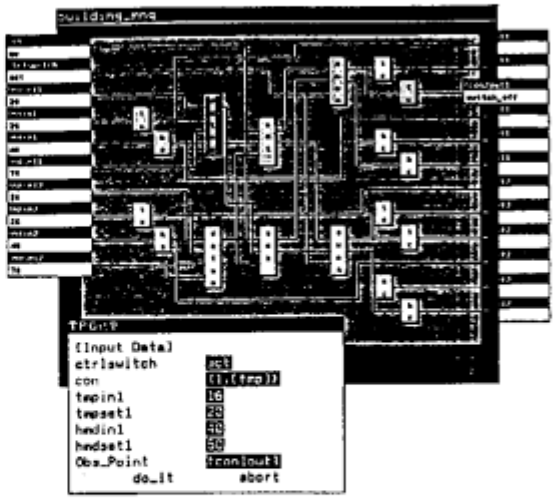

Figure 3: Display of the System

3.7 Probability Estimation

The performance of the test selection mechanism relies on the correctness of the presumed probability distribution of components. However, it is not easy to predict appropriate probability for each component from observed data, especially when the number of observed data is small. Although the current implementation employs Bayes presumption method, a novel mechanism to induce the probability distribution is being carried out by the authors and is reported in the reference [Nakakuki et al. 1990]. This mechanism is being incorporated into the system.

4 Experimental Result

This section summarizes an experimental result on the effect of experience learning. Especially, we concentrate on the effect of the test selection mechanism. Another effect of learning, the performance improvement achieved by knowledge caching for a model-based suspect calculation can be found in the reference [Koseki 1989].

The experimental system was implemented on a Prolog machine PSI-II. Figure 3 shows an example display of the system. The connectivity knowledge can be edited on the schematic editor on the display. The darkened boxes represent suspected components. The darker the box is, the higher probability the box has.

A simple model for an air conditioning controller shown on the example display was used to measure an effect of the experience learning mechanism. The model consists of 16 components and it has 10 input signals and 12 output signals. As a measure, the number of tests required to identify a fault is counted for hypothetical faults for each component. The number of test are compared between two cases, the one without any experience (same probabilities for all of the components) and the one with

a set of predefined probabilities.

The number of tests are reduced from 7 to 4 after learning for a test case, where the faulty component has a higher probability than the others. The predefined failure distribution for the 16 components are, zero times for 10 components, five times for 4 components, fifteen times for 2 components, and thirty times for 2 components, one of which is the faulty component. On the other hand, when the faulty component has low probability, the number of tests required was larger than the unexperienced case. Since this happens with low probability, it can be estimated that the number of the tests is reduced in average.

With only this particular result, it is still dangerous to conclude that the learning mechanism described here is always effective, since the result heavily depends on the characteristics of the particular target device. However, we can conjecture the effectiveness of the method.

5 Conclusion and Future Problems

This paper has described an experimental model-based diagnostic system that learns from experience. Its architecture offers a solution to the two main problems encountered in developing conventional rule-based systems, brittleness and knowledge acquisition bottlenecks. It also solves the efficiency problem in model-based systems.

Several problems and research tasks still remain. At present, the system works on one level of abstraction. Future work is to include the use of hierarchical design models for diagnosis and for test pattern generation. The system requires complete design knowledge. However, to realize more robust diagnosis, it is desirable to be able to diagnose with incomplete design knowledge. The described learning mechanism only learns from the successful diagnosis experience. A research on learning mechanism from failed cases is planned. The behavioral description employed in the system can only represent a component behavior without internal states. Although internal states can be represented by providing pseudo input and output signal ports outside of components, more systematic mechanism to deal with the internal states is desired.

Acknowledgments

This research has been carried out as a part of the Fifth Generation Computer Project. The authors would like to thank Katsumi Nitta and Kenji Ikoma of the Institute for New Generation Computer Technology for their support. They also express their appreciation to Yoshihiro Nagai, Takeshi Yoshimura, and Tomoyuki Fujita of NEC Corporation for giving them the opportunity to pursue this research.

References

[Davis 1984] R. Davis, "Diagnostic Reasoning Based on Structure and Behavior," *Artificial Intelligence* 24, 1984, pp. 347–410.

[Dejong and Mooney 1986] G. Dejong and R. Mooney, "Explanation-Based Learning: An Alternative View," *Machine Learning* 1:2, 1986, pp. 145–176.

[deKleer and Williams 1987] J. de Kleer and B. C. Williams, "Diagnosing Multiple faults," *Artificial Intelligence* 32, 1987, pp. 97–130.

[deKleer and Williams 1989] J. de Kleer and B. C. Williams, "Diagnosis With Behavioral Modes," *Proc. IJCAI-89*, Vol. 2, 1989, pp. 1324–1330.

[Genesereth 1984] M. R. Genesereth, "The Use of Design Descriptions in Automated Diagnosis," *Artificial Intelligence* 24, 1984, pp. 411–436.

[Genesereth and Nilsson 1987] M. R. Genesereth and N. J. Nilsson, *Logical Foundation of Artificial Intelligence*, Los Altos, CA: Morgan Kaufmann, 1987.

[Koseki 1989] Y. Koseki, "Experience Learning in Model-Based Diagnostic Systems," *Proc. IJCAI-89*, Vol. 2, 1989, pp. 1356–1361.

[Mitchell et al. 1986] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabeli, "Explanation-Based Generalization: A Unifying View," *Machine Learning* 1:1, 1986, pp. 47–80.

[Nakakuki et al. 1990] Y. Nakakuki, Y. Koseki, and M. Tanaka, "Inductive Learning in Probabilistic Domain," *Proc. AAAI-90*, 1990, pp. 809–814.

[Poole 1986] D. L. Poole, "Default Reasoning and Diagnosis as Theory Formation," Technical Report CS-86-08, Department of Computer Science, University of Waterloo, March, 1986.

[Quinlan 1986] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning* 1:1, 1986, pp. 81–106.

[Reiter 1987] R. Reiter, "A Throry of Diagnosis From First Priciples," *Artificial Intelligence* 32, 1987, pp. 57–95.

[Roth et al. 1967] J. P. Roth, W. G. Buricius, and P. R. Schneider, "Programmed algorithms to compute tests to detect and distinguish between failures in logic circuits," *IEEE Trans. Electron. Comput.*, vol. EC-16, no. 10, 1967, pp. 567–580.

[Shortliffe 1976] E. J. Shortliffe, *Computer Based Medical Consultations: MYCIN*, Elsevier, New York, 1976.

[Singh 1987] N. P. Singh, *An Artificial Intelligence Approach to Test Generation*, Norwell, MA: Kluwer Academic, 1987.