Candidate Ordering and Elimination in Model-based Fault Diagnosis

Jiah-shing Chen and Sargur N. Srihari*
Department of Computer Science
State University of New York at Buffalo
Buffalo, New York 14260

Abstract

A major step in model-based fault diagnosis is the generation of candidate submodules which might be responsible for the observed symptom of malfunction. After the candidates are determined, each subrnodule can then be examined in turn. It is useful to be able to choose the most likely candidate to focus on first so that the faulty parts can be located sooner. We propose here a systematic method for initial candidate ordering that takes into account the structure of the device and the discrepancy in outputs between the observed and expected values. We also give effective methods for a system to adjust its focus according to new information acquired during diagnosis. Under the single fault assumption, the average length of diagnosis (number of submodules evaluated) is O(logm), where m is the number of submodules.

1 Introduction

Diagnostic reasoning based on structural and functional descriptions of a device, usually referred to as the "design model", has been shown to be viable by many AI researchers [Cantone *et al.*, 1983, Chandrasekaran and Milne, 1985, Davis *et al.*, 1982, de K'leer and Williams, 1987, Genesereth, 1984, Reiter, 1987]. A thorough survey on model-based troubleshooting can be found in [Davis and Harnscher, 1988]. In this approach reasoning uses "first principles", i.e., knowledge of how the device works rather than knowledge of how it fails. The knowledge needed for such a system is well-structured and usually available when a device is designed. A model-based system avoids many of the difficulties of an empirical failure-based diagnosis system, e.g., in knowledge acquisition, diagnosis capability (ability to diagnose previously unseen faults) and system generalization.

*This work was supported in part by the Air Force Systems Command, Rome Air Development Center, Criffiss Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Boiling AFB DC 20332 under Contract No. F30602-85-C-0008, which supports the Northeast Artificial Intelligence Consortium (NAIC).

Since a model-based fault diagnosis system reasons directly on the structure and function of a device, it usually follows a simple control structure. It starts from the top level of the structural hierarchy of the device and tries to find outputs that violate their expectations. After detecting the violated outputs, the system uses structural descriptions to find a subset of components, which might be responsible for the observed symptom of malfunction, at the next lower hierarchical level. This step, known as candidate generation, is a major step in fault diagnosis. This process is then continued with each of the candidate components in turn until the faulty parts are found. It is desirable to have a diagnostic system which is able to choose the most likely candidate to focus on first so that the faulty parts can be located sooner.

Previous work on diagnosis based on structure and behavior contains suggestions on initial candidate ordering. Davis [1984] developed a technique called *assumption relaxation* for diagnostic reasoning. The technique uses an ordered list of hypotheses (the kinds of things that can go wrong) and starts to generate candidates under the first hypothesis. If this fails to find the faulty part, it gives up the assumption and considers subsequent hypotheses in order. This approach, in some sense, orders candidates implicitly according to various assumptions, such as single point of failure, bridges, multiple points of failure, etc. However, when more than one candidate is generated using a particukir hypothesis, there is no ordering among the generated candidates.

In our previous work [Taie *ct al.*, 1987], candidates are explicitly sorted by their *fault possibilities* which are derived from their relationship to bad primary outputs. A subrnodule has a higher fault probability if it contributes to more bad primary outputs. Here we extend this statement to its dual a subrnodule has a lower fault probability if it contributes to more good primary outputs.

Most existing model-based troubleshooting systems focus primarily on diagnosing failures caused by a single faulted component. The GDE (general diagnostic engine) system [de Kleer and Williams, 1987] provides a mechanism for dealing with multiple faults. This is achieved by using an assumption-based truth maintenance system (ATMS) [de Kleer, 1986], which maintains both derived conclusions and their supporting assumptions. In GDE, the hypotheses are generated in two steps. First, whenever two contradictory predictions are made for the same point in a device, a conflict is constructed by taking the union of assumptions supporting the predictions. The complete set of hypotheses is then generated such that each hypothesis, represented as a set of false assumptions, has a non-empty intersection with every conflict. This process is computationally expensive and GDE incorporates the notion of minimality in both conflicts and hypotheses to reduce the complexity. However, even the set of minimal hypotheses grows exponentially in the worst case. GDE also uses additional measurements to discriminate hypotheses and proposes the best next measurement based on minimum entropy. The optimal measurement is the one that will minimize the expected entropy of hypothesis probabilities resulting from the measurement and, as a result, will minimize the expected number of measurements.

While measurements are used to differentiate among the competing hypotheses in GDE, they can be used to confirm or disconfirm suspected components too. The results of measurements also provide a diagnostic system with information about the relative fault probabilities of other candidates. To make use of these kinds of information, we develop candidate reordering and elimination methods. They modify the candidate list during the diagnosis according to the results of measurements in a way that actual faulty modules are moved forward in the list and modules which now become impossible to be responsible for any violations are removed from the list.

Section 2 of this paper is devoted to initial candidate ordering. Section 3 explains candidate reordering and elimination using examples. Section 4 gives an average case analysis of reordering and elimination.

2 Initial Candidate Ordering

We now describe heuristics for selecting the most likely candidate. They are derived by analyzing the circuit in a manner analogous to that employed by a human diagnostician. Typical examples are used to explain our points. In what follows, an output of a device is said to be bad or violated if for a certain assignment of values to the inputs, the observed output value is different from the expected output value. (An incorrect output is marked with a cross in the figures.) An output is good or corroborated if it is not violated. The expected value of an output can be computed by knowing the function of the device. For simplicity, in this section, we concentrate only on the modules A's and B's in the following figures and assume other modules are well-functioning.

Intuitively, we have the following two heuristics: (a) a submodule is more likely to be faulty if it is connected to more bad primary outputs and (b) a submodule is less likely to be faulty if it is connected to more good primary outputs. While these rules seem plausible, they need to be further refined. For example, both A and B in Figure 1 are connected to the same number of bad primary outputs, i.e., one, and the same number of good primary outputs, i.e., zero. They are equally likely to be faulty according to the above two rules. A closer look reveals that A has two outputs connecting to the bad primary

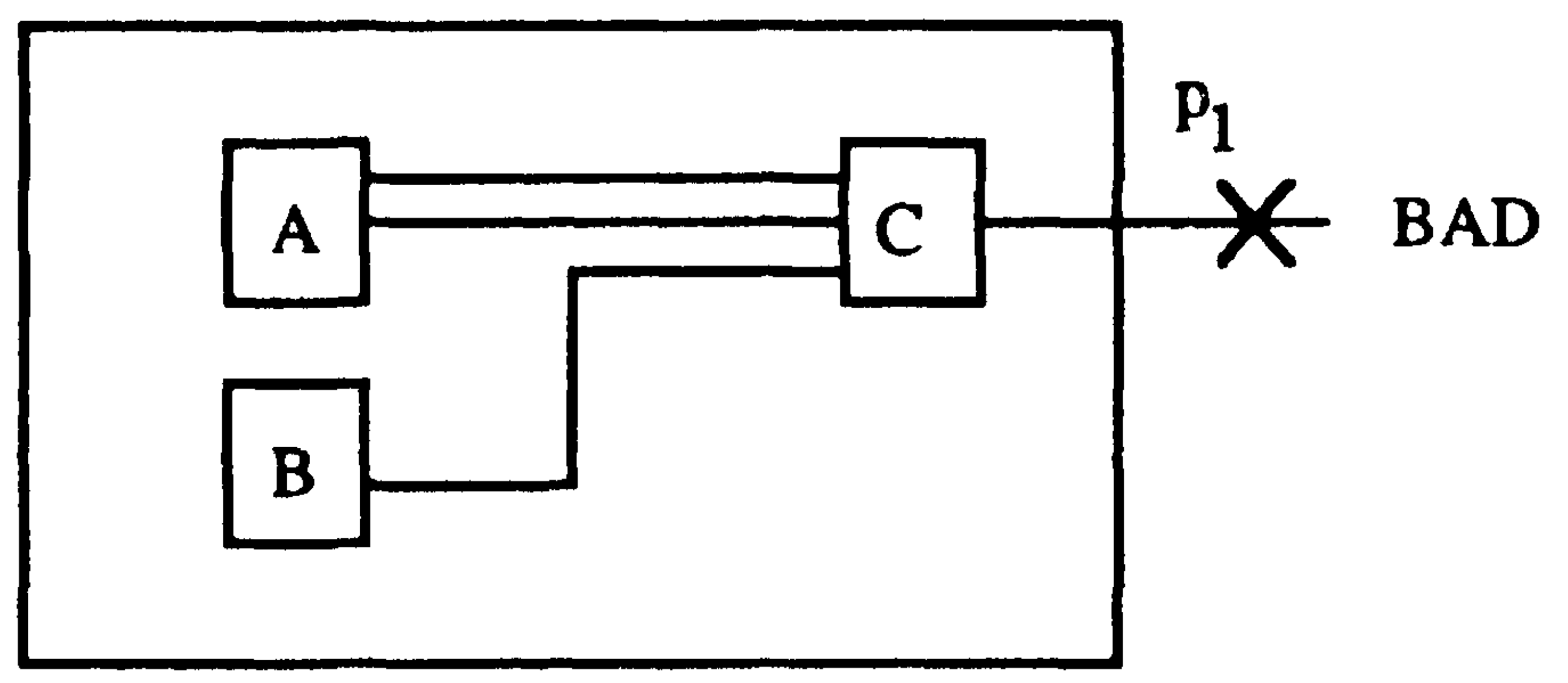

Figure 1: Number of submodule outputs connected to bad primary outputs: A has two, B and C have one each.

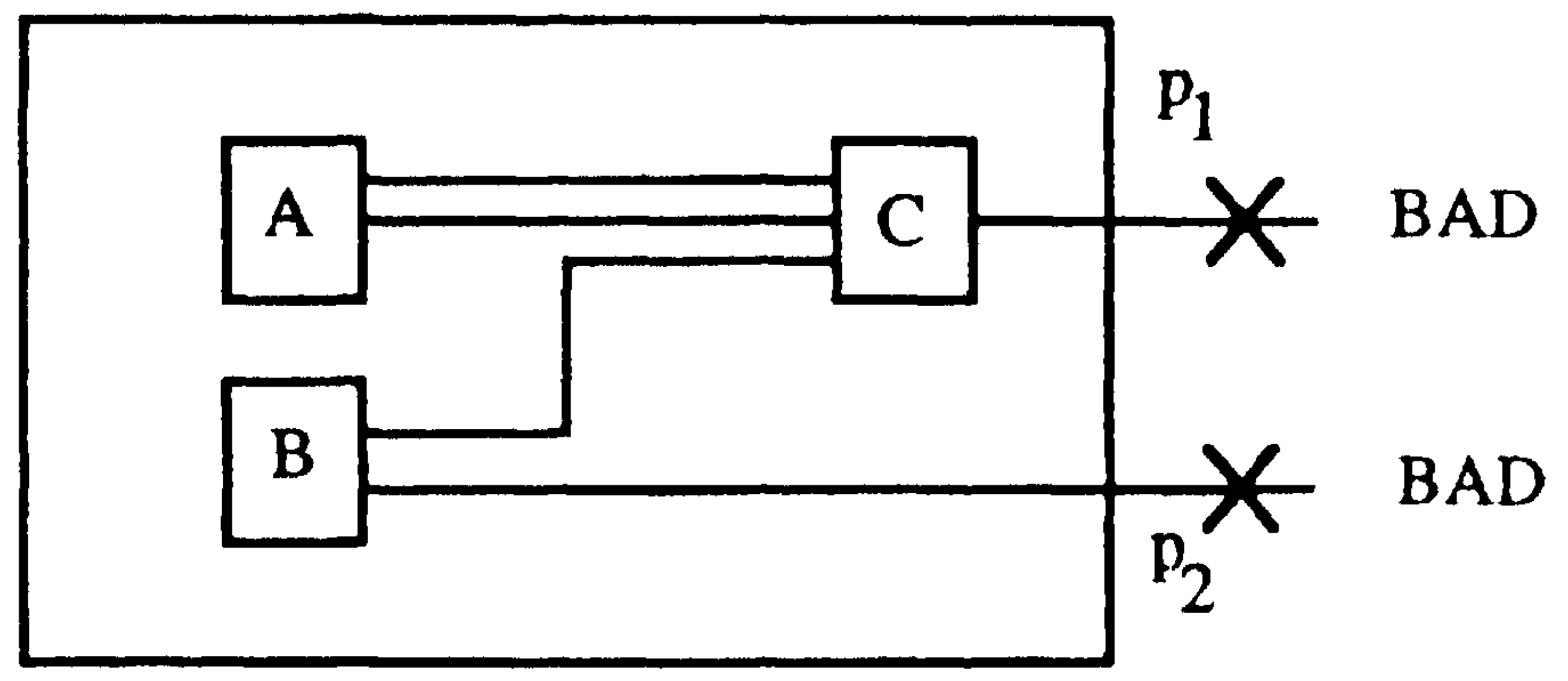

Figure 2: Number of covered bad primary outputs: A covers p_1 and B covers p_1 and p_2.

output while B has only one. This makes A more likely to be faulty than B since either of A's two outputs may be responsible for the observed output discrepancy and the probability of A being faulty is approximately the sum of the probabilities of each of its outputs being bad (assuming they are independent). This example shows that we have to consider submodule outputs individually and combine them in some way. Summing up the total number of bad (good) primary outputs connected to each output of a submodule works fine in this case.

The circuit in Figure 2 shows that the number of bad primary outputs covered by a submodule is more important than the number of potential bad outputs of a submodule. Here both A and B have two outputs, and each of them connects to one of the bad primary outputs. The total number of bad primary outputs connected to A's outputs, i.e., two, is the same as that of B's outputs. This time the distinction lies not on the total number but the different number of the bad primary outputs connected to their outputs. The fact that B's outputs cover two bad primary outputs while A's outputs cover only one gives us some evidence that B is more likely to be the faulty submodule. Thus in addition to summation, we also need to know the union of the sets of bad primary outputs connected to the outputs of a submodule.

Figure 3 shows the insufficiency of the previous rules. None of the rules will discriminate A and B since they both cover the same bad and good primary outputs.

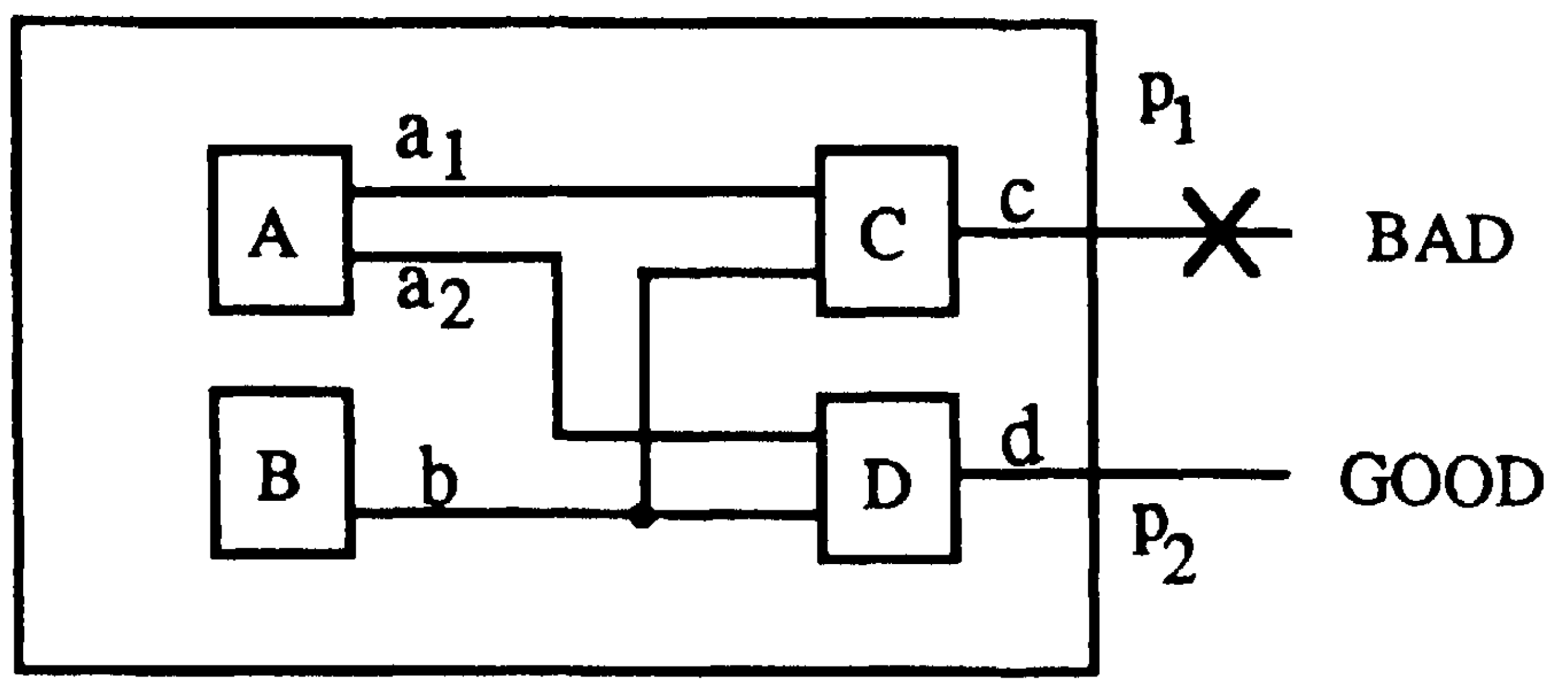

Figure 3: Outputs a_2, b and d are good interactive (GI), while a_1 and c are non-GI. In this example, the initial ordering derived is (C A B D).

Nevertheless, the fact that B's only output connects to both the bad and good primary outputs shows evidence of its being good and makes B less likely to be faulty. This makes it necessary to distinguish between the two types of outputs, output a_1 of A and output b of B. An output of a module is said to be good interactive (or GI) if it connects to any good primary output (e.g., a_2 and 6); otherwise, it is said to be non-GI (e.g., a_1). GT outputs are considered more likely to be good than non-GI outputs.

These examples show that the two heuristics using submodules as references are not adequate. We have to consider the individual outputs of submodules before considering the submodules themselves. In order to combine the results obtained from the individual outputs of a submodule, we need to use additions and set unions with special treatment on GI outputs.

We present in the following an algorithm which assigns a *rank* to a submodule *S* according to these heuristics. In addition, we also make use of heuristics about the kinds of faults that are more frequently encountered when the device has been functioning for a while. Here we assume that single faults occur more often than other faults.

1. Let *B* and *G* be the sets of bad and good primary outputs, respectively. Assume o\, 02,..., *Ok* are the outputs of submodule 5.
2. For each oi, compute *bi* and *gi,* which are the sets of bad and good primary outputs o_i connects to, respectively. Note that if *gi* — 0 then *oi* is a non-GI output.
3. Compute the *rank* of 5, r = (r_1, r_2, r_3, r_4, r_5).
   - $r_1 = |\cup_{i=1}^{k} b_i|$, which is the total number of bad primary outputs (without repetition) connected to the outputs of 5.
   - $r_2 = |\cup_{g_i=\emptyset} b_i|$, which is the total number of bad primary outputs (without repetition) connected to the non-GI outputs of 5.
   - $r_3 = \sum_{i=1}^{k} |b_i|$, which is the total number of bad primary outputs (with repetition) connected to the outputs of *S*.
   - $r_4 = \sum_{g_i=\emptyset} |b_i|$, which is the same as r_2 but with repetition.
   - $r_5 = -\sum_{i=1}^{k} |g_i|$, which is the negative of the total number of good primary outputs (with repetition) connected to the outputs of *S*.
4. Determine whether 5 is a candidate under the fault assumption we are using according to its rank. For example, 5 is a candidate under the *single fault assumption* (SFA) if r_1 = |B|, or 5 is a candidate under the *unidirectional ports assumption* (UPA)[1] if $r_1 > 0$.

After each of the candidate submodules has been identified and assigned a rank, we can now order them according to their ranks. We use r_1 as the first key, r_2 as the second key, etc. and order candidates in decreasing ranks. The reasons we select these keys are explained as follows. The first key, r_1 orders candidates by the number of different bad primary outputs covered by their outputs. The more bad primary outputs a module covers, the more likely its being faulty can explain the observation. The second key, r_2, counts the number of bad primary outputs covered by the non-GI outputs of a module since non-GI outputs have higher probabilities of being bad. The third and fifth keys, r_3 & r_5, are straightforward realization of the heuristic implied by Figure 1. The fourth key, r_4, serves as a tie breaker for the third key.

To see that the rank assignments actually reflect the heuristics described above, we review the circuit in Figure 3 again according to the algorithm. All modules, except D, have one output connected to the bad primary output, p_1, so $r_1(A)$ = $r_1(B)$ = $r_1(C)$ = $r_3(A)$= $r_3(B)$ = $r_3(C)$ = 1 and $n(D)$ = $r_3(D)$ = 0. Both modules A and C have only one non-GI output connected to the bad primary output, so $r_2(A)$ = $r_4(A)$ = 1 and $r_2(C)$ = $r_4C)$ = 1. On the other hand, B and D have no non-GI outputs so $r_2(B)$ = $r_4(B)$ = 0 and $r_2(D)$ = $r_4(D)$ = 0. Similarly, $r_5(A)$ — - 1 , $r_5(B)$ = - 1 , $r_5(C)$ = 0 and $rr_0(D)$ = - 1 . Thus we have $r(C) > r(A) > r(B) > r(D)$, which is consistent with the result derived from our heuristics.

3 Candidate Reordering and Elimination

While the initial candidate ordering looks promising, there is no guarantee that actual faulty components are ordered in the front of the candidate list. In fact, for any device and good/bad output pattern, it is not difficult to come up with a counterexample on which our method does poorly in the sense that the actual faulty component is put at the last few places in initial ordering. For example, in Figure 3, the culprit might actually be B which is the last candidate (under SFA) according to our initial ordering. To address this problem, we reorder or

[1] Unidirectional ports assumption assumes that the inputs always receive data and outputs always send data, not the other way around. In this paper, we also implicitly assume UPA as we assume SFA.

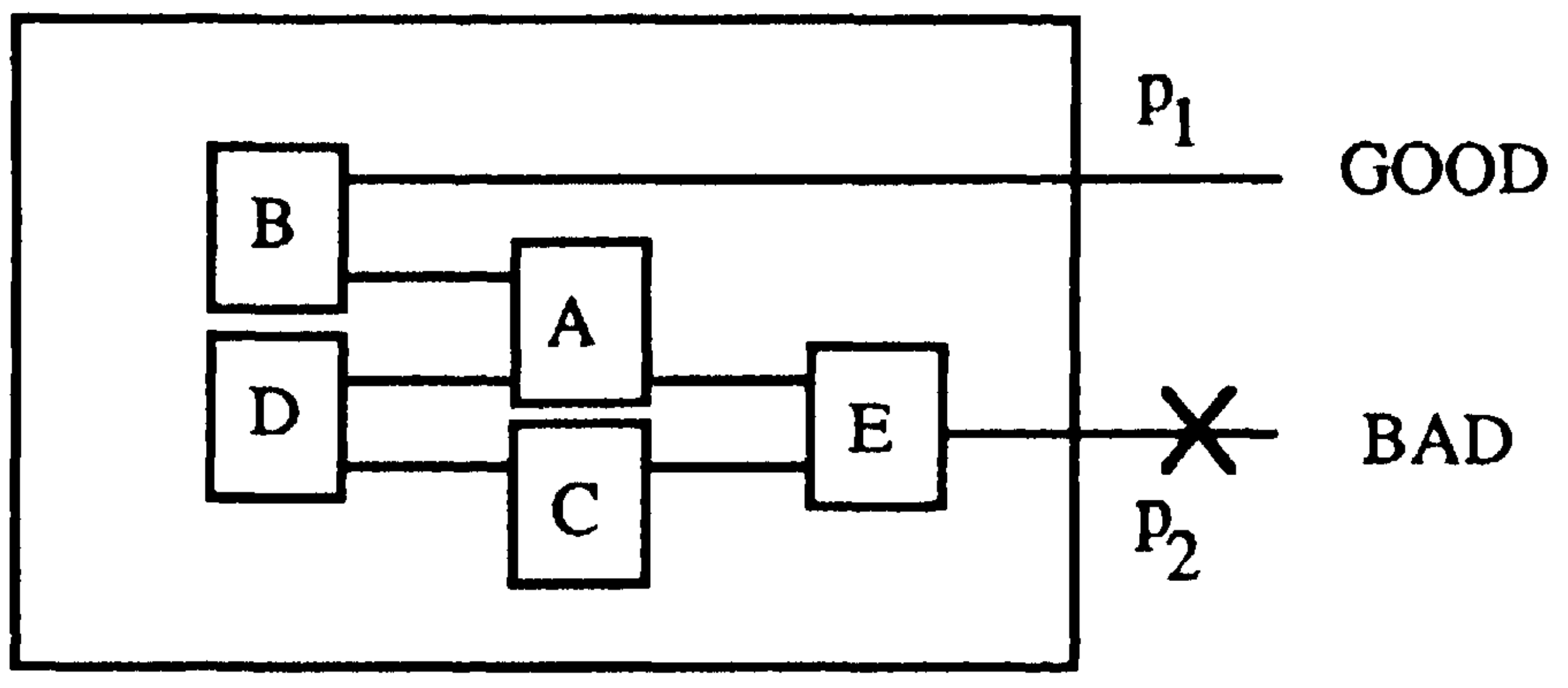

Figure 4: An example for candidate elimination: (A B C D E) is the assumed initial ordering with E the culprit.

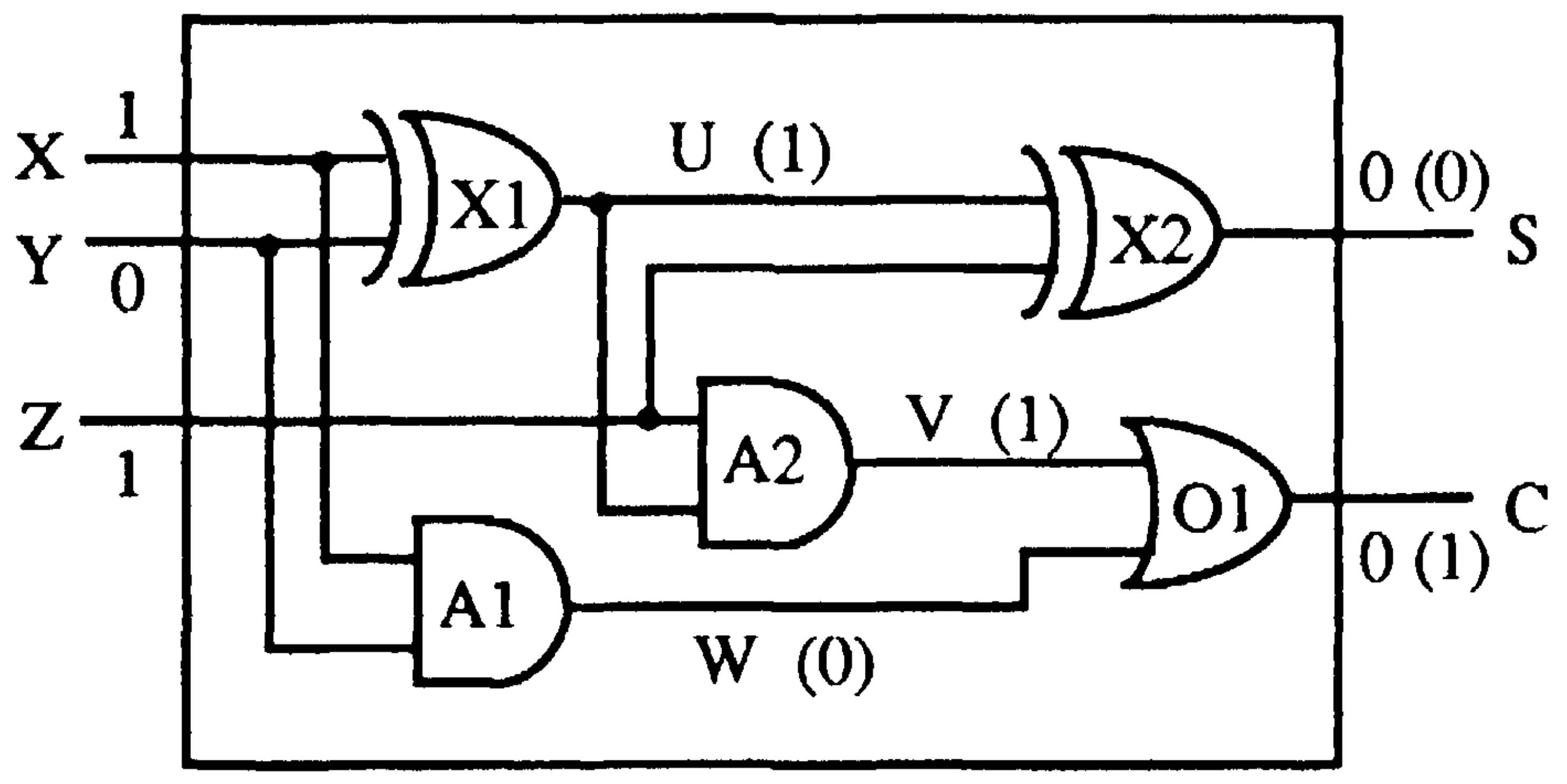

Figure 5: A faulty full adder with measured input and output values (parenthetical values are predictions).

eliminate candidates whenever some intermediate values are measured.

We now explain the reordering principle by considering the example of Figure 3. Assume there is only one fault and B is the culprit (or the actual faulty module). As shown in previous section, the initial candidate list is (C A B). After we check the current candidate C by measuring its inputs and outputs, we will find that the first input of C is consistent with the expected value derived from primary inputs but the second input is not since B is the one at fault. Now we can use this information to shove B to the front of candidate list and A to the tail because B (A) becomes more (less) likely to be faulty. Therefore, the candidate list becomes (B A) and B will be checked next. This shows that B is not the last one to be checked due to candidate reordering although it was at the last place initially. More formally, candidate reordering works as follows. After the inputs of current candidate are measured, candidates connected to its incorrect inputs are shoved to the front of candidate list and candidates connected to correct inputs but not to incorrect inputs are shoved to the tail.

To explain candidate elimination, we use the example in Figure 4. We assume E is the only culprit and the initial candidate list is (A B O D E). Since E is the only culprit, the measured value of A's output is consistent with its predicted value and A is said to be *non-error-propagatzng.* Module A cannot be responsible for the observed error in p2 nor can B. This is because, B's output must go through A to affect p_2 but A's output has been known to be consistent with what it is supposed to be. Therefore B can be removed from the candidate list. In contrast to B, D cannot be removed at this time due to the existence of a path passing C to p_2. This leaves (O D E) as the new candidate list. By the same argument, D will be removed after C's output is found to be consistent. As a result, E becomes the only candidate left and the diagnosis at this level can be terminated. In short, whenever a candidate is found to be non-error-propagating, candidates that no longer have a path to violations are removed from the candidate list.

To have more effective reordering and elimination results and a more efficient termination condition for diagnosis, the predictions have to be updated after new measurements are made. This makes it necessary to generalize the notion of violation and corroboration to intermediate points as well as primary outputs. Reordering and elimination are most effective since they are based on the most current predictions. Also diagnosis can be terminated when there are no more violations, except those at outputs of known faulty components. This way the diagnostic procedure is able to find a minimal set of faulty components to account for the observed misbehavior of a device, which is similar to the "diagnosis" defined according to the *principle of parsimony in* [Reiter, 1987]. It is worth pointing out that since the predictions are updated as measurements are made, violations and corroborations are updated correspondingly. Due to the effect of canceling faults, a corroboration may later become a violation (and *vice versa)* and hence components which were not candidates before may need to be added to the candidate list.

We shall demonstrate in the following full adder example [Genesereth, 1984] h ow the aforementioned ideas can be used in a diagnostic procedure. Suppose the symptom of the full adder is that with inputs XYZ = 101 it produces outputs SC = 00 and assume both XOR gates XI and X2 are faulty. The expected values at intermediate points and primary outputs are shown in parentheses in Figure 5. Since output C is a violation and output S is a corroboration, the initial candidate list is (AI A2 OI XI). The output of AI, W, is measured to be 0 and AI is non-error-propagating. The AND gate AI and all upstream components which can no longer be responsible for the violation at C are eliminated (here only AI is excluded) and the candidate list becomes (A2 OI XI). Next, A2's output, V, is measured to be 0 which is different from its predicted value, 1. The second input of A2, U, is then measured to determine if A2 is at fault and the measurement is 0 due to the assumption that XI is faulty. Therefore A2 is found not faulty and the candidate list becomes (XI OI) because U is an inconsistent input of A2 and XI is shoved to the front. The predictions are updated according to the measurements at U and V as shown in Figure 6. Since C is not a violation now, O1 is not a candidate any more and the candidate

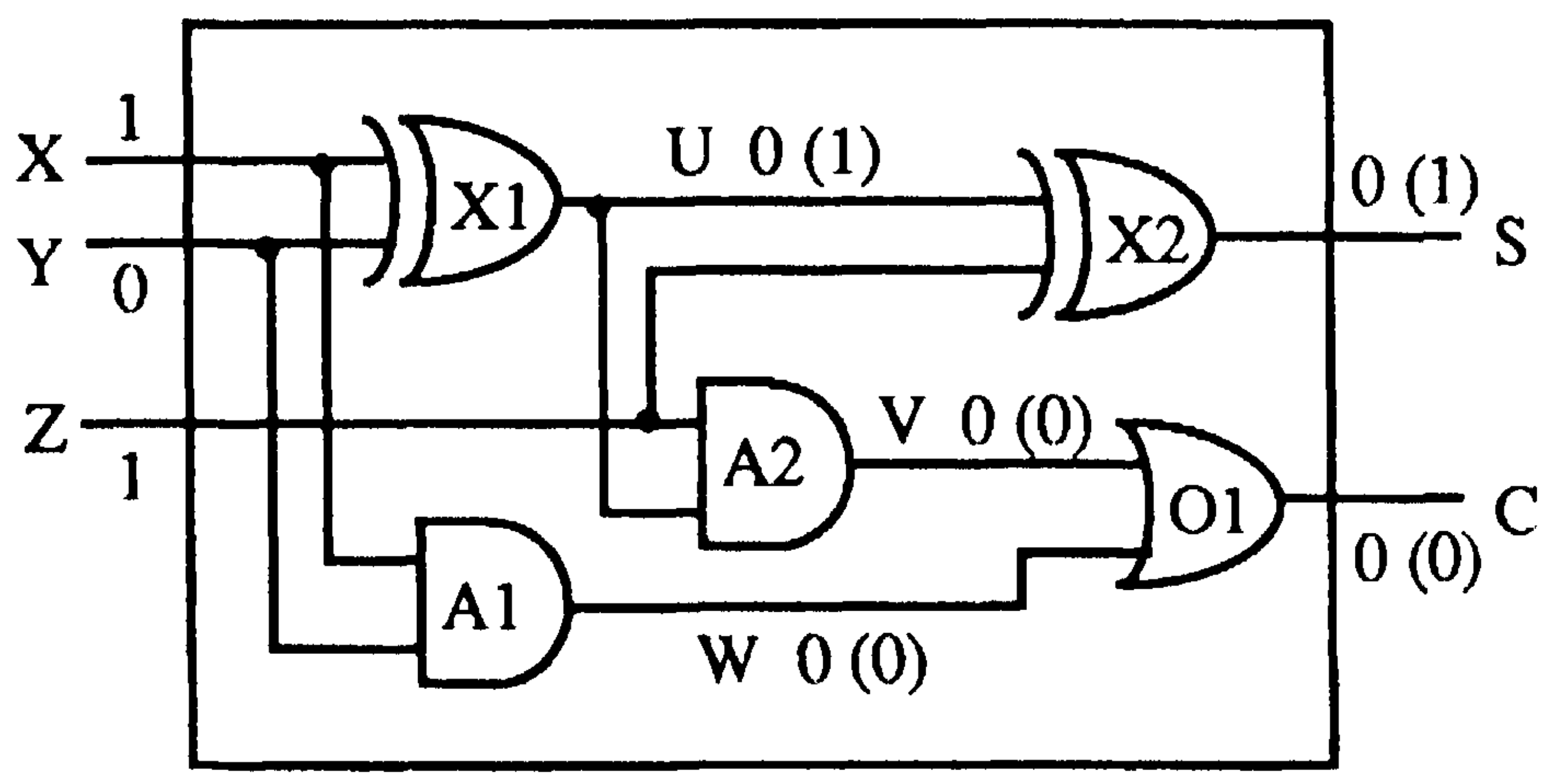

Figure 6: A full adder with requested intermediate measurements.

list becomes (XI). However, S becomes a violation so X2 is now added to the candidate list making it (X2 XI). Since all inputs and outputs of XI and X2 are known at this point, X1 and X2 are found faulty and there remains no more candidate. The diagnosis terminates with the conclusion that both X1 and X2 are faulty.

In this simple example, it turns out that all intermediate points are measured before a conclusion can be reached. It should be noted that in general it is not necessary to check all components when reordering and elimination techniques are used. We shall show in the next section that these techniques are indeed helpful at least for the single fault cases.

4 Analysis

To show that candidate reordering and elimination really help shorten the length of a diagnosis, we compute the average number of components that have to be checked before the culprit is found, under single fault assumption (SFA). Note that under SFA, shoving candidates connected to incorrect inputs of current candidate to the front of candidate list is equivalent to removing other candidates as far as the length of diagnosis is concerned. The culprit is guaranteed to be among those being shoved to front since they contribute to some violations while others don't. Throughout the analysis, the probability distribution is assumed to be uniform. For example, each candidate has equal failure rate and a candidate has a probability of 0.5 of being non-error-propagating if it is not faulty.

Let /(n) denote the average length of a diagnosis (number of candidates checked) when there are *n* candidates. *l(n)* is given by the following recurrence relation:

$$
\begin{aligned}
l(n) \;=\; & \frac{1}{n}\cdot 1 \\
& + \frac{n-1}{n}\cdot\frac{1}{2}\cdot[1+\sum_{i=1}^{n-1}\frac{1}{n-1}\cdot l(i)] \\
& + \frac{n-1}{n}\cdot\frac{1}{2}\cdot[1+\sum_{i=1}^{n-1}\frac{1}{n-1}\cdot l(i)]
\end{aligned}
$$

The first term represents the case that the first candidate is faulty (with probability 1/n) and only one component is checked. If the first candidate is intact (with probability (n — l)/n) and non-error-propagating (with conditional probability 0.5), 0 to n—2 candidates may be eliminated (i.e., 1 to *n* — 1 candidates are left) and each case has an equal probability of l/(n — 1). The average number of checked components for this case is computed by the second term. Similarly, if the first candidate is intact (probability = (n — l)/n) and error-propagating (probability = 0.5), 1 to *n*— 1 candidates may be shoved to the front of candidate list. This is described by the last term. The above expression simplifies to

$$l(n) = 1 + \frac{1}{n}\sum_{i=1}^{n-1} l(i) \qquad (1)$$

Substituting $n-1$ for n in (1), we get

$$l(n-1) = 1 + \frac{1}{n-1}\sum_{i=1}^{n-2} l(i) \qquad (2)$$

It follows that

$$\sum_{i=1}^{n-2} l(i) = (n-1)\cdot[l(n-1)-1] \qquad (3)$$

From (1) and (3), we have

$$l(n) = \frac{1}{n} + l(n-1) = \sum_{i=1}^{n}\frac{1}{i} = \Theta(\log n) \qquad (4)$$

If there are m candidates initially (m is bounded by the number of components), the average length of diagnosis using candidate reordering and elimination is Θ(log m). This is much better than random or sequential examination whose expected length is -1/2m.

5 Discussion

Our system accomplishes the minimal diagnosis for multiple faults by taking additional measurements at intermediate points. As in GDE, we assume complete visibility of the diagnosed device. Unlike GDE, we avoid the problem of enumerating the power set of components by putting all potential faulty components in one candidate list and manipulating the list according to our reordering and elimination principles which take advantage of both violations and corroborations. We have also shown the effectiveness of these techniques by analyzing the average length of diagnosis for single fault cases. However, the measurement selection method of GDE does have the advantage of becoming a binary search procedure on a cascaded chain of components where our methods fail to be discriminating.

The techniques of candidate ordering, reordering and elimination have been implemented in our Versatile Maintenance Expert System (VMES) [Shapiro *et* a/., 1986, Taie *et* a/., 1987]. VMES is a model-based fault diagnosis system which assists inexperienced maintenance technicians in troubleshooting electronic circuits. As expected, the system shows reasonable improvement in diagnostic performance after these techniques are employed.

Although our average case analysis does not depend on the initial ordering of the candidates, presumably a good initial ordering would prevent a diagnosis from going into worst cases. Incidentally, our initial candidate ordering scheme can be easily augmented to include other information useful for a system to produce a better initial ordering. All we need to do is introduce new keys representing the new information in appropriate positions. For example, if we have information about the failure rate and test cost of each component type, we can simply add them as, say, the second and fourth keys.

We view diagnosis as a progressive process. Until the culprits of a malfunctioning device are found, a diagnostic system must be able to produce a new answer and confirm the answer. The system is allowed to make mistakes but should not only recover from them but also discover useful information from them. The candidate reordering and elimination techniques are motivated by these ideas. They are somewhat limited due to the need for measuring values at intermediate points of the diagnosed device. Our future work will include the development of similar "self-adjusting" search methods that are based on other kinds of information which can be acquired during diagnosis. The behavior of the diagnosed device after some boards are replaced or swapped, and its responses to different primary (external) inputs, are currently under consideration.

Acknowledgments

The authors would like to thank Mr. Dale Richards of Rome Air Development Center and Dr. Shambhu J. Upadhyaya for their helpful comments and discussions about this paper.

References

[Cantone *et al.,* 1983] Richard R. Cantone, Frank J. Piptone, and W. Brent Lander. Model-based probabilistic reasoning for electronics troubleshooting. In *Proceedings of 1JCAI,* pages 207-211, 1983.

[Chandrasekaran and Milne, 1985] Balakrishnan Chandrasekaran and Rob Milne. Reasoning about structure, behavior and function. *ACM SIGART Newsletter,* 93:4-7, July 1985.

[Davis and Hamscher, 1988] Randall Davis and Walter Hamscher. Model-based reasoning: Troubleshooting. In Howard Shrobe, editor, *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence,* chapter 8, pages 297-346. Morgan Kaufmann, San Mateo, CA, 1988.

[Davis *et al,* 1982] Randall D avis, Howard Shrobe, Walter Ilamscher, Karen Wieckert, Mark Shirley, and Steve Polit. Diagnosis based on description of structure and function. In *Proceedings of AAAI,* pages 137-142. Morgan Kaufmann, 1982.

[Davis, 1984] Randall Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence,* 24(3):347-410, 1984.

[de Kleer and Williams, 1987] Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence,* 32(1):97-130, 1987.

[de Kleer, 1986] Johan de Kleer. An assumption-based TMS. *Artificial Intelligence,* 28:127-162, 1986.

[Genesereth, 1984] Michael R. Genesereth. The use of design description in automated diagnosis. *Artificial Intelligence,* 24(3):411-436, 1984.

[Reiter, 1987] Raymond Reiter. A theory of diasgnosis from first principles. *Artificial Intelligence,* 32(1):57-96, 1987.

[Shapiro *et al.,* 1986] Stuart C. Shapiro, Sargur N. Srihari, Mingruey R. Taie, and James Geller. VMES: A network-based versatile maintenance expert system. In *Proceedings of the 1st International Conference on Applications of AI to Engineering Problems,* pages 925-936. Springer-Verlag, April 1986. Volume II.

[Taie *et al,* 1987] Mingruey R. Taie, James Geller, Sargur N. Srihari, and Stuart C. Shapiro. Knowledge based modeling of circuit boards. In *Proceedings of 1987 Annual Reliability and Maintainability Symposium,* pages 422 427, January 1987.