다중 상황공간을 이용한 다중 오류의 고장 진단

이 계 성[†] · 권 경 희[††]

요 약

고장진단 문제는 지식기반 시스템을 이용해 해결하려는 시도가 많이 있어왔다. 그러나 대부분의 방식은 휴리스틱 또는 모델기반 방식으로 단일 오류에 대한 문제에 많은 노력이 이루어져 왔다. 단일오류에 대한 고장진단 문제 해결방식을 다중 오류진단으로 확대할 때 발생하는 지수적인 계산비용은 피할 수 없는 문제점으로 지적되어왔다. 이 논문에서는 시스템 구성에 따라 블록으로 구분하면 전체 탐색 영역을 국소화할 수 있다는 점에 착안하여 다중오류 진단을 위한 효율적인 알고리즘을 제안한다. 이 알고리즘의 기본 원리는 오류진단을 위한 출력값 측정 지점에 따라 전체 회로를 블록으로 나누어 다중오류에 대한 발생원인의 지수적 증가를 줄임으로 효율화시킬 수 있다. 각각의 블록으로부터 발생하는 오류에 대해 결합하는 규칙을 개발하고 이를 통해 상호 논리적인 모순이 없는 최소 오류원인 집합을 구한다.

Diagnosing Multiple Faults using Multiple Context Spaces

Gyesung Lee[†] · Kyung Hee Kwon[††]

ABSTRACT

Diagnostic problem solving is a major application area of knowledge-based system research. However, most of the current approaches, both heuristic and model-based, are designed to identify single faults, and do not generalize easily to multiple fault diagnosis without exhibiting exponential behavior in the amount of computation required. In this paper, we employ a decomposition approach based on system configuration to generate an efficient algorithm for multiple fault diagnosis. The basic idea of the algorithm is to reduce the inherent combinatorial explosion that occurs in generating multiple faults by partitioning the circuit into groups that correspond to output measurement points. Rules are developed for combining candidates from individual groups, and forming consistent sets of minimal candidates.

1. Introduction

The design of diagnostic systems is a major application area of knowledge-based system research. A number of successful diagnostic systems have been developed for medical, industrial, and engineering applications. The complexity of present day electronic and digital circuits and their use in large numbers has accelerated the need for automated fault diagnosis systems that are efficient and effective in real world applications. Most current approaches, both heuristic and model-based, are designed to identify single faults, and do not generalize easily to multiple fault diagnosis without exhibiting exponential behavior. An excellent introduction to different AI techniques used for diagnosis: associational (e.g., MYCIN[2]), functional (e.g., [9]) and structural (e.g., [3]) is presented by [12].

† 정 회 원:단국대학교 전자계산학과 전임강사
†† 정 회 원:단국대학교 전자계산학과 조교수
논문접수:1996년 11월 5일, 심사완료:1996년 12월 18일

Recently a number of researchers ([3], [4], [10], and [14]) have proposed diagnosis systems based on model-based reasoning techniques as opposed to associational pattern matching techniques that link symptoms to potential causes [2]. They all view diagnosis as the process to determine the fault or faults responsible for an observed set of symptoms. Analysis by model-based reasoning techniques relies on generating behavior of a system from its structure and functionality derived from first principles.

Model-based approaches result in more general and robust techniques for diagnosis. Associational methods rely on the experience of human experts (source of associational knowledge), and, therefore, are mostly applicable in previously encountered situations (whereas model-based techniques are applicable to novel fault diagnosis). However, the methods proposed by [3] and [10] are effective only in single fault diagnosis. Davis represents the behavioral knowledge of the device as a constraint network, and dependency relations are established between output measurements, input data and components that influence the output measurement (pathways of causal interaction). A constraint suspension method is then applied to narrow down the set of candidates by relaxing constraints on individual component behavior (i.e., relaxing the constraint that describes correct behavior of a component), and propagating its effects to ensure that this relaxation does not contradict other measurements (to ensure global consistency). Note that this technique does not require the use of fault models, however, to apply it to multiple fault diagnosis would require applying constraint suspension to all possible combinations of components in a sequential manner. Similarly, Genesereth's diagnosis scheme uses design knowledge to identify a suspect set. Design knowledge corresponds to the structural and behavioral knowledge of devices. The scheme then applies a traditional test generation method to generate the test set used to confirm and disconfirm the candidate components in the suspect set.

De Kleer and Williams [7] extended Davis' work [3] and developed GDE (Generalized Diagnosis Engine), a general-purpose model-based approach for multiple fault diagnosis. They start off with a device model, which includes

- The device schematics-components that make up the circuit and their interconnections, and
- A behavior generating model-given a set of input values, the model can compute the correct (or expected) values at the output and other points at which measurements can be made.

Their diagnostic scheme, based on a two step approach: conflict recognition and candidate generation, is initiated when discrepancies are detected at measurement points. An assumption-based (ATMS) reasoning technique keeps track of multiple sets of consistent and inconsistent environments (possible faulty component sets). Rather than generate all possible candidates, their algorithm is made more efficient by generating only minimal sets of faulty candidates. They couple this mechanism with a probabilistic information theoretic scheme that defines an efficient probing strategy to help refine the candidate sets generated by the first step. [14] also presents a similar technique, but develops a more formal model based on a finite set of faults, symptoms, and causal connections between faults and symptoms. Recently there have been attempts by [15] and [6] to integrate fault models into model based diagnosis to further refine the candidate sets derived from the above analysis.

The primary motivation of this paper is that in real world circuit diagnosis, which involves MSI, LSI, and VLSI circuits, the complexity of the system being diagnosed is many orders of magnitude greater than the examples presented in above papers. It is not clear as to how well these algorithms would scale up when the complexity of the circuits to be diagnosed increases greatly in terms of the number of components and the complexity of the interconnections used. As a first step towards achieving better bounds on the computational complexity, we propose a new algorithm that is conceptually simpler than the [7] algorithm, and probably computationally less complex.

Our approach starts by decomposing the circuit [8] being investigated into groups of smaller number of components based on output measurement points. The idea is to prevent inherent combinatorial explosion that occurs when generating multiple faults. The implementation requires that a circuit be defined as a network of connected components (the components are the nodes of the network), the correct behavior of all individual components are known, and, therefore, correct behavior can be computed at all measurement points using the model. Like the [7] approach, an ATMS [5] is used for maintaining the partial set of minimal candidates and dependency lists, so that global consistency can be ensured as partitions are combined. Formation of the set of minimal candidates is initiated from a single partition with a faulty output measurement, and then sequentially combined with neighboring partitions to obtain globally consistent candidate sets.

A point to note is that though the paper was motivated by digital circuits, the multi-fault diagnostic technique presented can be applied to any system or device provided: (i) they can be represented as a system of interconnected components, and (ii) given that behaviors of individual components are known, the network model can be used to completely derive the expected behavior of the system for different sets of input values.

Section 2 outlines the basic ideas employed by the multiple fault diagnosis algorithm. Section 3 presents the algorithm for candidate generation, and Section 4 talks about some implementation issues. Section 5 discusses a complete example that demonstrates the working of the algorithm. Section 6 presents conclusions, and ideas for future extensions of this approach to diagnosis.

2. Diagnosing multiple faults

de Kleer and Williams define diagnostic reasoning as "a means of assigning credit or blame to parts of a model based on observed behavioral discrepancies" [7]. Diagnosis involves two primary aspects: (i) identifying model-artifact differences, and (ii) using these differences to identify possible sets of faulty components that are consistent with the measurements made. In reality, diagnosis includes a very important third component: (iii) determining a sequence of evidence-gathering tests that help refine the results of step (ii). This third step aids in narrowing down possible faulty candidates till ultimately the actual faulty components can be identified.

This paper assumes that a schematic of the system being diagnosed is available in the form of actual components that make up the system and how they are interconnected. It is also assumed that a complete device model is available, therefore, model-artifact differences can be easily identified by comparing observed measurements to expected measurements computed from the model. These differences are recorded as symptoms. Furthermore, it is assumed that the device or system being diagnosed has multiple output ports, and all discrepancies at the output ports have been recorded, before step (ii) of the diagnostic process is initiated.

A traditional (or manual) diagnostic scheme would likely combine steps (ii) and (iii) of the diagnostic process, however, we assume an automated diagnosis scenario where additional test points are not easily accessible. For example, consider an automated diagnostic system for PCB's (Printed Circuit Boards). Measurements at the I/O pins can be easily performed, but probing measurements at arbitrary points of the board may be much harder to accomplish by an automated system. Performing additional tests or making further measurements is much more expensive than performing the set of computations in step (ii) in its entirety. Therefore, the algorithm we develop concentrates on determining all possible minimal faulty candidates. Like the de Kleer and Williams approach [7] further testing can be initiated based on the candidate sets extracted in step (ii), however, we do not address that phase of the sequential diagnosis process in this paper.

The algorithm for multiple fault diagnosis is based on candidate generation in local subspaces, and then combining the results from local subspaces in a manner that achieves global consistency over the entire space. Three factors form the primary issues in algorithm development: (i) decomposition of the system, (ii) candidate generation in individual partitions, and (iii) the combination scheme, where results from individual partitions are put together to achieve global consistency. Each one of these are discussed in some detail below.

2.1 Decomposing the circuit

Fig. 1 and 3 are examples of two circuits on which we apply the diagnostic algorithm. These circuits have one or more input and output points, and the circuit configuration can be represented in network form with components mapped into nodes, and interconnections into links. Given that these circuits can, in general, be made up of large numbers of components with complex interconnections, a natural approach to solve the problem is by a divide and conquer or partitioning approach. The idea, in general, is to partition the problem into smaller parts, find solutions for the parts, and then combine the solutions for the parts into a solution for the whole. The advantage of this method is that even if the solution process is computationally complex, the adverse effects of complexity are not as drastic when one deals with small problem sizes. Intuitively, this notion can be exing the search process. The cost of path analysis is less in smaller spaces, therefore, searching and eliminating unlikely paths in earlier stage of the search helps in reducing search cost.

As discussed earlier, it is assumed that the first set of observations are made at the terminal (output) points of the circuit, therefore, partitions are formed by grouping together all components that affect the same terminal point. This structural division is created by following backward from the output and picking up all components in the path till input points are reached. The components of a partition are said to create a context space. Note that a component can belong to more than one context space. It turns out that the common components (those shared by more than one partition) play a key role in determining global consistency of the faulty component sets.

2.2 Consistent candidate set

As discussed earlier, the diagnostic procedure is initiated by the detection of symptoms. Symptoms are defined as discrepancies between actually observed behavior (measurements) made on the system, and behavior (measurements) predicted by the system model. When the system is working (behaving) correctly, the output measurements can be justified by the fact that individual components that influence the output are also behaving correctly, i.e., given M_1, $M_2, \cdots$, M_k are directly related to output, V_i, if M_1, $M_2, \cdots$, M_k are o.k., then V_i is good. We assume that there are no design and fabrication errors. In AI terminology, the set $\langle M_1, M_2, \cdots, M_k \rangle$ can be defined as an assumption that explains V_i. However, if the output V_i is observed to be faulty, then the above assumption is no longer valid, and is said to be inconsistent with the observed measurement. Consistency is restored when we identify a set (or sets) of faulty components that can explain the symptoms and other measurements in the circuit.

Note that inconsistencies are detected locally, but must be resolved globally, i.e., determining V_i is faulty implies that $\langle M_1, M_2, \cdots, M_k \rangle$ is inconsistent, however, choosing $M_j \in M_1, M_2, \cdots, M_k \rangle$ as faulty (indicated by M_j) may be an assumption that explains V_i (i.e., achieves local consistency) but it may not be globally consistent because M_j is not consistent with measurement V_m being correct. For example, in Fig.1, assuming output at M_2 is faulty is a plausible reason for f being incorrect, but by itself is inconsistent with the fact that measurement g is observed to be correct. In other words, individual symptoms give rise to inconsistences that are resolved locally but need to be propagated globally to ensure that the set of assumptions about the overall system (which components are faulty, and which are o.k.) explain all the output measurements. Our algorithm uses the partitioning scheme discussed in section 2.1 to reduce the combinatorial problem that arises in achieving global consistency, especially when multiple candidates can be considered to be faulty.

The first step in the multiple fault identification process involves selecting possible faulty components in individual partitions. For the faulty partition, each component could be a candidate. In the above example, given V_i is incorrect, $[M_1]$, $[M_2]$,---,$[M_k]$ are possible fault candidates based on that one measurement. Of course, any superset of a candidate set may also be a candidate, however, for this study we are only interested in minimal candidate sets. The minimality property implies that particular candidate set includes a minimum number of faulty candidates that explains all measurements consistently; any smaller number cannot correctly explain all measurements. Note that this does not cover the entire spectrum of errors that could occur. For example, even if the output measurement is correct for a certain set of input values, a pair of components that cancel out each other's errors, could still be faulty. Sometimes errors are blocked out by particular characteristics of the components. For instance, an and-gate will not transmit an error in an input line if any of its other inputs is low. These issues, which require deeper behavioral analysis, are not dealt with in this paper.

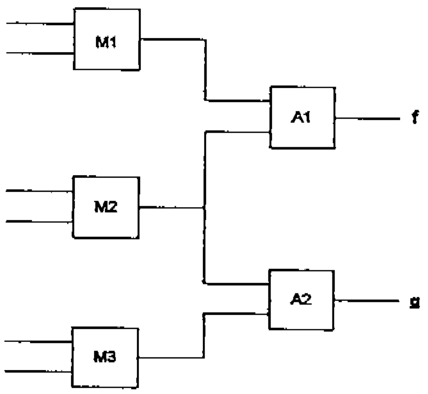
As discussed, when a symptom is observed, the possible candidates are generated by local resolution of assumptions within the partition, then the candidate sets are successively extended to increasingly distant neighboring partitions since the overall symptoms need to be explained in terms of the combined set of faults from different partitions. Two partitions that share at least one component are called structurally adjacent (or connected). Partitions may be structurally connected through a chain of partitions (e.g., P_i is adjacent to P_j, and P_j to P_k, then P_i is structurally connected to P_k through P_j. Two partitions that are directly adjacent are said to be one apart. The shortest distance between two partitions is $i+1$ if there are at least i partitions in any chain from one to the other.

The candidate generation scheme starts with a faulty partition (i.e., a partition with a faulty measurement), identifies all possible single faults, and then combines the results from this partition with adjacent partitions sequentially in the order of increasing distance till the entire circuit is covered. The combination scheme discussed in the next section maintains global consistency during the combination process. Fault diagnosis can be done independently in disjoint sections of the system.

2.3 Combination scheme

The combination scheme is based on the intuitive assumption that single faults are more likely than double, which are more likely than triple faults, and so on. This assumption can be exploited so that only minimal sets of multiple candidates are generated. The implications is that any superset of a generated candidate is a possible fault candidate but with a much less likelihood of occurrence. Based on this idea for generating only minimal sets, the diagnostic algorithm begins by forming single fault candidate sets, i. e., for every partition whose output measurement is faulty, every component in that partition is considered a singleton fault candidate. At the first step, partitions whose output measurements are not faulty are ignored. The rest of the algorithm involves iterative combination with adjoining partitions till the entire circuit is covered.

Note that two cases can occur-(i) candidates from a faulty partition may need to be combined with an adjacent partition with no faults; in this situation we have to consider double faults for components that are shared between the partitions. A faulty component in the intersection must be compensated for by a faulty component in the good partition, so that measurements at the terminal point are observed to be good, and (ii) the two adjacent partitions are faulty; in this case, form all pairs of components that are not shared between the two partitions (one from each). The components shared between the two partitions form singleton candidate sets. The combined partitions now can be considered a new faulty partition, and the combination step is repeated for another neighboring partition.



(Fig. 1) Example circuit 1

The simple circuit shown in Fig. 1 is used as an example in a number of previous papers ([7], [5], etc.). There are two adders and three multipliers. Each module has two inputs and one output. Applying the decomposition process to this circuit produces two partitions F and G corresponding to output measurement points f and g, respectively. Partition F=[M_1, M_2, A_1]. The components are obtained by tracing back from output terminal f to input terminals.

When f (F partition) is observed to be faulty in the example circuit, the assumption ⟨A_1, M_1, M_2⟩ is violated because f would be necessarily correct if A_1, M_1, and M_2 are all working correctly. The corresponding candidate sets are three singletons: {[A_1], [M_1], [M_2]}. Given that g is observed to be good, the combination scheme is invoked in context (i) (i.e., combine fault sets from a faulty and a good partition). The scheme looks for common elements in the partition F and G. In this case it is M_2. Note that M_2 by itself being faulty contradicts the observed output measurement at g. To achieve consistency, at least one other component in G must be faulty, so that the two faults compensate each other. Based on this analysis the new set of minimal candidates is {[M_1], [A_1], [M_2, A_2], [M_2, M_3]}. Further testing or checking actual functionalities may reduce the candidate set but that is not an issue discussed in this paper.

Note that the candidate size is one or two. When partitions are merged, the maximum candidate size is the maximum of the distances between two partitions. Note also that candidate sets whose size is greater than the total number of fault symptoms observed, include components whose errors compensate each other.

3. Algorithm for Candidate Generation

The overall algorithm for candidate generation is summarized as follows:

```
Partition the circuit
Check measurements
if no discrepancy detected, stop
Pick up one faulty partition
Repeat
  Choose a neighbor partition
  Nogood checking
  if it is good partition
    call fault vs good combination procedure
  else
    call fault vs faulty combination procedure
  Backward search to find the missing
```

```
        candidates
    Combined partition becomes a new faulty
        partition
until all partitions are combined
```

The individual steps of the algorithm are discussed below. The combining algorithm admits two possibilities :(i) combine a faulty partition with a good one, and (ii) combine two faulty partitions.

3.1 Partitioning and measuring

3.1.1 Partition the circuit.

The circuit is divided into partitions by the method described in section 2.1. If partitions are not all structurally linked, i.e., there exist group of partitions that share no components with others, form disjoint groups and apply the algorithm separately to each group of partitions.

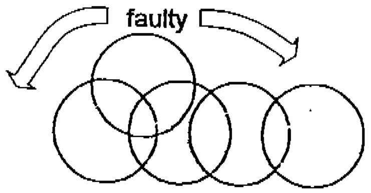
3.1.2 Check measurements.

Measurements are taken at preselected (usually terminal) points. Then they are compared with expected values computed from a model of the correctly functioning system. If there are no discrepancies, it is assumed that the system is functioning correctly. Note that this need not be true in general, e.g., faults may cancel each other for input set.

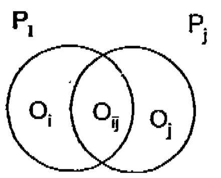
Realistic diagnostic practice often involves performing multiple tests in sequence to get a better hold on the nature of the faults observed. As discussed earlier, a particular input set may not detect an existing fault, but some other might. We again skirt this issue in this paper, and assume the right set of inputs have been applied to detect all faults that exist. Of course, that may require more than one input set, but it really does not affect the part of the process we concentrate on.

3.2 Combining the partitions

As discussed earlier, a faulty partition is chosen as the starting point for candidate generation. Each component in this partition represents a singleton candidate that may minimally explain the faulty behavior at the output port. This partition is then combined with one of its neighbors. Two cases can occur: the second partition output is faulty, or the second partition output is good. Different scheme developed for each case are presented later. The combined partition can now be evoked upon as a single faulty partition, which is then combined with another neighbor. This sequential process (see Fig. 2) continues till all overlapping partitions are included. Note that for every new combination, the size of the maximum candidate set included may increase by 1. Therefore, if there are k partitions, the maximal size of a candidate set will be k. Also, if certain partitions are completely disjoint then the candidate generation algorithm can be applied to each separately.



(Fig. 2) Combining Order



(Fig. 3) Combining two partitions

3.2.1 Combining method

Fig. 3 shows two partition P_i and P_j being combined. Let us assume P_i is a result of previous partition combinations, and P_j is a single partition to be combined with P_i. Let O_i represent the set of modules in P_i that are not in P_j, O_j represent the set of modules in P_j that are not in P_i, and O_{ij} represent the set of modules that are common to P_i and P_j. If the output measurement at P_j is good, then the combination process invoked is a faulty+good scheme. (Once the P_i and P_j combination is complete, P_{ij} will be renamed P_i to keep the number of subscripts under control.).

The first step in the combination process is to generate nogood candidate sets (Note that this is ATMS terminology [7]). A nogood really implies a set of modules that in no way can justify observed measurements, and once a set is established to be nogood, it and any superset of nodes that includes these nodes remain throughout the computation. For example, consider the circuit in Fig. 5. If partition F is found to be faulty, and G is good, then any pair of modules in the intersection of F and G, e.g., (M_2, M_{10}) is a nogood candidate. This is because they have contradictory roles in the two partitions. In partition F they are together supposed to explain a faulty measurement, whereas in partition G they are supposed to cancel each other's faults and produce a correct measurement. Therefore, identifying and recording nogoods enables to cut down the search for possible candidates. For example, the pair M_2, M_{10} or any superset of M_2, M_{10} can be considered to be a possible minimal candidate set.

Following this scheme, the same is done for modules in O_{ij} that are shared between $P_m \in P_i$ and P_j. Note nogoods are established only for partition pairs where one of them has a faulty measurement and the other does not (e.g., P_m faulty and P_j good, or vice versa). Then, if any superset of the newly established nogoods already exist in the current minimal candidate set, they are removed. The establishment and maintenance of nogood labels is implemented using the ATMS scheme discussed earlier. We next elaborate on the two possible scenarios for combinations:

a) faulty+faulty

This case holds for combination of two neighbor partitions, P_i and P_j whose measurements are both faulty, or $\mathbf{P_i}$ is a previously merged partitions in which at least one component partition was faulty, and P_j is a single partition that neighbors $\mathbf{P_i}$ but has not yet been merged into $\mathbf{P_i}$. P_j also has a faulty measurement. The combination scheme can be summarized as forming a new partial set CS'_i from CS_i (the existing candidate set) and C_i (a candidate) according to:

$\{[C_i \cup m_j] \mid C_i \in CS_i$, and $\forall m_i \in C_i$ also $\in O_i$ and $m_j \in O_j\} \cup \{[C_i] \mid C_i \in CS_i$ if all $m_i \in C_i$ also $\in O_{ij}\}$

The rationale for the above is every individual candidate set from the nonoverlapping sets O_i and O_j are merged to form a larger minimal candidate set, whereas a candidate set from the overlap O_{ij} remains as is, because it may also account for the new faulty measurement without creating an inconsistency.

b) good+faulty

In this case, partition $\mathbf{P_i}$ (single or merged) is faulty, and its neighbor P_j (not yet merged) has a good output measurement. Step 1 here is the same as before, remove all candidates that are nogoods or supersets of nogoods. The partial candidate set CS'_i is computed from CSi as follows:

$\{[C_i] \mid C_i \in CS_i$, and $\forall m_i \in C_i$, $m_i \in O_j\} \cup \{[C_i \cup m_j] \mid C_i \in CS_i$, and $\exists m_i \in C_i$, $m_i \in O_{ij}$, $m_j \in O_j\}$

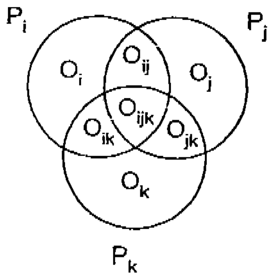
The rationale for the first component above is that any minimal candidate that has no overlap with non-faulty partition P_j remains consistent, and continues to be a minimal candidate. However, if it overlaps with P_j it needs to pick up an additional module from O_j, so that a pair of faulty modules within P_j cancel each other, and thereby produce a correct result.

Note that if C_i contained a pair (or more) modules that was in O_{ij}, it would be previously removed as a nogood. Therefore, each C_i that forms the second union, usually has only one module, m_i O_{ij}.

3.3 Backward search

When two partitions are merged and one partition is made up of previously merged partition, it may happen that the above procedure may miss some candidates in a particular configuration. Fig. 4 depicts such a situation. Consider a previously merged partition P_i being merged with a neighbor partition P_j. Candidates are usually missed when P_j overlaps with P_m P_i, and P_m is faulty but P_j is good. As a more specific example, consider three partitions P_i, P_j, and P_k. Suppose that P_i is faulty and P_j and P_k are good and now P_k is going to be merged into the P_i, P_j combination. To illustrate, let us subdivide the overlapped section into 3 parts: O_{ik}, O_{jk}, and O_{ijk}. The faulty +good procedure takes individual candidates from O_{ik} combines them with candidates in O_k to form double faults, and candidates with elements in O_{ijk} and O_{jk} to from triple faults. But O_{ik} and O_{jk} are never taken into account even though there are possible candidates that could be generated from them by combining with components in the non-overlapped part of good partition $P_j(O_j)$. The missing candidates in this configuration are represented as follows:

$\{[m_a, m_b, m_c] \mid m_a \in O_{ik}, m_b\ O_{jk}$, and $m_c\ O_j\}$.



(Fig. 4) Backw

If a faulty partition is merged into the previously merged one, then the components that need to be considered are from the intersection between the faulty partition and good partitions from the previously merged partition that are direct neighbors. Suppose in this case that P_k is faulty. Then only part to be taken care of is O_{jk}. Other intersected subsections (O_{ik} and O_{ijk}) are normally processed as P_j is merged. But O_{jk} will not combine with O_j and result in a candidate with any component from O_i. Additional candidates are generated as follows:

$\{[m_a, m_b, m_c] \mid m_a \in O_{jk}, m_b\ O_i$, and $m_c\ O_j\}$

3.4 Stop combination procedure

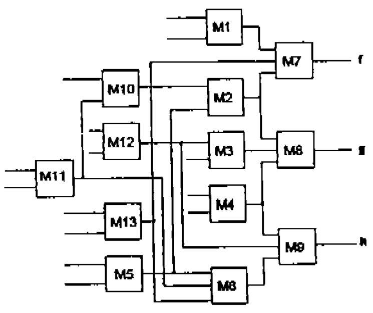
In case of very large, complex systems or circuits where the number of components and partitions (i.e., output ports) are large, if only a few symptoms are observed (e.g., only 1 or 2 faulty partitions among a total of 15), a stopping criteria may be applied so as not to include any candidate set of size above a certain threshold, t. This criteria is based on the premise that multiple faults of large numbers are much less likely to occur (failure probability will be very low). In addition, using reasoning similar to Davis' adjacency principle [3], sequential combination starting from a faulty partition may progress till the t nearest neighbors are considered. This may be repeated for all faulty partitions, and the total candidate set is the union of the individual ones.

4. Implementation issue

The multiple fault diagnosis system implementation is based on two schemes: a network-type computing scheme for model-based reasoning, and an ATMS for maintaining consistent minimal fault sets. A given circuit can be easily mapped into a network-type structure in which nodes represent the modules and links the interconnections. Each node is described in terms of functional composition of three elements: input, output and function. The function is coded in a demon-like procedure which is initiated to run and compute the result based in input values, and pass it onto the next module. The only failures considered are the ones in modules. In other words, there are no bridge faults, intermittent errors, design errors, or assembly errors. Each module can form a singleton fault set or form an element of a multiple fault set. Each fault set is called an assumption. Assumptions are managed using the ATMS scheme which ensures that consistency is specified in individual context spaces and retained in the combined space. Since the justifications for the symptoms are possible fault sets, which are equivalent to candidate sets. When new partitions are considered, new environments are introduced. Each environment is checked to see if it is subsumed by already existing environment. If their relationship is superset, only minimal set is retained in the ATMS database. Nogood is also recorded in the database and any superset of nogood node is also nogood. Therefore, the ATMS performs important bookkeeping functions.

5. Example

A detailed example which involves determining minimal candidate sets for the circuit shown in Fig. 5 is discussed to illustrate the details of the algorithm presented in Section 3 and 4. The circuit has three outputs **f**, **g**, and **h** with corresponding partitions F, G, and H. Candidates are represented by capital letters enclosed by square brackets. Suppose **f** is



(Fig. 5) Circuit 2

faulty and g and **h** are good. Since **f** is faulty, **g** or **h** can be combined with **f**. It can be shown that the combination sequence is immaterial, however, for this example we first combine F and G, and then combine H with the FG combination. From Fig. 5, we derive partitions F, G, and H:

$F=\{M_1, M_2, M_5, M_7, M_{10}, M_{11}, M_{13}\}$
$G=\{M_2, M_3, M_4, M_5, M_8, M_{10}, M_{11}, M_{12}\}$
$H=\{M_4, M_5, M_6, M_9, M_{11}, M_{12}, M_{13}\}$

Step1 : Consider F. The minimal candidate sets are single components:
{[M_1], [M_2], [M_5], [M_7], [M_{10}], [M_{11}], [M_{13}]}

Step 2: Combine results of F with partition G. Note this is a fault + good combination. The resultant candidate set is:

{[M_1]. [M_7], [M_{13}],
[$M_{10}M_3$], [$M_{10}M_8$], [$M_{10}M_4$], [$M_{10}M_{12}$],
[M_2M_3], [M_2M_8], [M_2M_4], [M_2M_{12}],
[M_5M_3], [M_5M_8], [M_5M_4], [M_5M_{12}],
[$M_{11}M_3$], [$M_{11}M_8$], [$M_{11}M_4$], [$M_{11}M_{12}$]}

Single faults represent components or modules in F that have no effect on G, and double faults arise because a candidate in F also affects the G output, therefore, a second component from G is needed to compensate for this candidate. An example of a nogood node generated is (M_2, M_{10}). After combination, FG is treated as a single faulty partition.

Step3 : Now partition H is combined with the FG combination. Modules in the intersection of FG with H are:

$\{M_4, M_5, M_{11}, M_{12}, M_{13}\}$

Any candidate from FG, which does not belong to H partition, remains in the new candidate set FGH:

$\{[M_1], [M_7], [M_{10}M_3], [M_{10}M_8], [M_2M_3], [M_2M_8]\}$

Pairs from intersection F and H, $\{M_5, M_{11}, M_{13}\}$ are recorded as nogoods because they are shared between a faulty partition F and a good partition G. Candidates of FG which contain overlapped modules are combined with modules of the H partition which are not shared, producing new candidates. For instance, $[M_{10}M_{12}]$ is joined with M_6 or M_9 to produce new candidates; $[M_{10}M_{12}M_6]$ and $[M_{10}M_{12}M_9]$. M_{10} belongs only to F so faulty f is explained with this singleton candidate. $[M_{10}M_{12}]$ and $[M_{12}M_6]$ explain the fault plus the fact that G and H are o.k. Candidate like $[M_5M_4]$ in the intersection can also explain all the symptoms of combined partition.

Step 4: The backtrace algorithm is invoked to find missing candidates. To calculate these extra candidates by backtrace, the system looks for possible double candidates from the intersection of F and H like $[M_{13}M_4]$ and $[M_{13}M_{12}]$, and then combines them with other modules from other partitions of FG, in this case G. For instance, M13M4 can explain both f and h but not g so module from G is combined and results in a new candidate, $[M_{13}M_4M_3]$. Another possibility is $[M_{13}M_4M_8]$.

6. Discussion

The candidate generation algorithm developed here bases its analysis on circuit schematics. By employing the partitioning structure, it provides a method for avoiding combinatorial explosion (considering all possible combinations) in multiple fault generation. Using the ATMS, as a method for bookkeeping makes the computational scheme more efficient. Behavioral description may be invoked to validate possible candidates. Furthermore, fault models can be substituted for faulty candidates to confirm whether they can generate the observed symptom (actually measurement). This seems to be a good method for further reducing possible candidates [6].

Our method could also be extended to cover different kinds of fault combinations. For example, the double fault cancellation may not explain the symptom, rather the double faults together may cause the faulty symptom. This would require an additional ATMS space to keep the minimality of the candidates. Our method currently does not take care of other characteristics of the components such as measurement cost, failure rate, etc., which can be used to make decisions on the next measurement point based on which fault sets can be further reduced. These parameters will be introduced into the diagnosis scheme to build a more practical system.

REFERENCES

[1] J. S. Brown, D. Burton, and J. deKleer, "Pedagogical natural language and knowledge engineering techniques in SOPHIE I, II and III," in Intelligent Tutoring System, D. Sleeman and J.S. Brown(Eds.), pp. 227-282, 1981.

[2] B.G. Buchanan and E.H. Shortliffe, 'Rule-based Expert Systems', The MYCIN experiments of the Stanford Heuristic Programming Project, Addison-Wesley, Reading MA, 1984.

[3] R. Davis, "Diagnostic reasoning based on structure and behavior," Artificial Intelligence, 24, pp. 347-410, 1984.

[4] J. deKleer, A.K. Mackworth, and R. Reiter, "Characterizing diagnoses and systems," Artificial Intelligence, 56, pp. 197-222, 1992.

[5] J. deKleer, "An assumption-based TMS," Artificial Intelligence 28, pp. 127-162, 1986.

[6] J. de Kleer and B.C. Williams, "Diagnosis with behavioral modes," in Proceedings IJCAI, pp. 1324-1330, 1989.

[7] J. deKleer and B. C. Williams, "Diagnosing multiple faults," Artificial Intelligence 32, pp. 97-130, 1987.

[8] Y.E. Fattah and R. Dechter, "Diagnosing

tree-decomposable circuits," in Proceedings IJCAI, pp. 1742-1748, 1995.

[9] P.K. Fink and J.C. Lush, "Expert system and diagnostic expertise in mechanical and electrical domains," IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-17, pp. 340-349, 1987.

[10] M.R. Genesereth, "The use of design descriptions in automated diagnosis," Artificial Intelligence 24, pp. 411-436, 1984.

[11] J.P. Martins and S.C. Shapiro, "A model for belief revision," Artificial Intelligence 35, pp. 25-79, 1988.

[12] R. Milne, "Strategies for Diagnosis," IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-17, pp. 333-339, 1987.

[13] F. Pipitone, "The FIS electronics troubleshooting system," Computer, July, pp. 68-75, 1986.

[14] R. Reiter, "A theory of diagnosis from first principles," Aritificial Intelligence 32, pp. 57-95., 1987.

[15] P. Struss and O. Dressler, "Physical Negation-Integrating fault models into the general diagnostic engine," in Proceedings IJCAI, pp. 1318-1323, 1989.

이 계 성

1980년 서강대학교 전자공학과(학사)
1982년 한국과학기술원 전산학과(석사)
1994년 Vanderbilt대학 전산학과(Ph.D)
1982년~1985년 경제기획원 조사통계국 전산처리관
1994년~1996년 대구대학교 전산정보학과 전임강사
1996년~현재 단국대학교 전자계산학과 전임강사
관심분야:전문가 시스템, 기계학습, ITS, Data mining

권 경 희

1976년 고려대학교 물리학과(이학사)
1986년 Old Dominion Univ, Department of Computer Science(M.S)
1991년 Louisiana State Univ. Dept. of Computer Science(Ph.D)
1979년~1984년 한국산업연구원 연구원
1993년~현재 단국대학교 전자계산학과 조교수
관심분야:병렬처리, 알고리즘, 연결망