REASONING ABOUT MULTIPLE FAULTS

Johan de Kleer

Intelligent Systems Laboratory
XEROX Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304

and

Brian C. Williams

M.I.T. Artificial Intelligence Laboratory
545 Technology Square
Cambridge, Massachusetts, 02139

ABSTRACT

Diagnostic tasks require determining the differences between a model of an artifact and the artifact itself. The differences between the manifested behavior of the artifact and the predicted behavior of the model guide the search for the differences between the artifact and its model. The diagnostic procedure presented in this paper is model-based, inferring the behavior of the composite device from knowledge of the structure and function of the individual components comprising the device. The system (GDE — General Diagnostic Engine) has been implemented and tested on examples in the domain of troubleshooting digital circuits.

This research makes several novel contributions: First, the system diagnoses failures due to multiple faults. Second, failure candidates are represented and manipulated in terms of minimal sets of violated assumptions, resulting in an efficient diagnostic procedure. Third, the diagnostic procedure is incremental, reflecting the iterative nature of diagnosis. Finally, a clear separation is drawn between diagnosis and behavior prediction, resulting in a domain (and inference procedure) independent diagnostic procedure.

1. Introduction

Engineers and scientists constantly strive to understand the differences between physical systems and their models. Engineers troubleshoot mechanical systems or electrical circuits to find broken parts. Scientists successively refine a model based on empirical data during the process of theory formation. Many everyday commonsense reasoning tasks involve finding the difference between models and reality.

Diagnostic reasoning requires a means of assigning credit or blame to parts of the model based on observed behavioral discrepancies observed. If the task is troubleshooting, then the model is presumed to be correct and all model-artifact differences indicate part malfunctions. If the task is theory formation, then the artifact is presumed to be correct and all model-artifact differences indicate required changes in the model. Usually the evidence does not admit a unique model-artifact difference. Thus, the diagnostic task requires two phases. The first, mentioned above, identifies the set of possible model-artifact differences. The second proposes evidence-gathering tests to refine the set of possible model-artifact differences until they accurately reflect the actual differences.

This view of diagnosis is very general, encompassing troubleshooting mechanical devices and analog and digital circuits, debugging programs, and modeling physical or biological systems. Our approach to diagnosis is also independent of the inference strategy employed to derive predictions from observations.

For troubleshooting circuits, the diagnostic task is to determine why a correctly designed piece of equipment is not functioning as it was intended; the explanation for the faulty behavior being that the particular piece of equipment under consideration is at variance in some way with its design (e.g., a set of components is not working correctly or a set of connections is broken). To troubleshoot a system, a sequence of measurements must be proposed, executed and then analyzed to localize this point of variance, or fault. For example, consider the circuit in Fig. 1, consisting of three multipliers, M_1, M_2, and M_3, and two adders, A_1 and A_2. The inputs are $A = 3$, $B = 2$, $C = 2$, $D = 3$, and $E = 3$, and the outputs are measured showing

that $F = 10$ and $G = 12$.[1] From these measurements it is possible to deduce that at least one of the following sets of components is faulty (each set is referred to as a candidate and is designated by [...]): $[A_1]$, $[M_1]$, $[A_2, M_2]$, or $[M_2, M_3]$. Furthermore, measuring X is likely to produce the most useful information in further isolating the faults. Intuitively, X is optimal because it is the only measurement that can differentiate between two highly probable singleton candidates: $[A_1]$ and $[M_1]$.

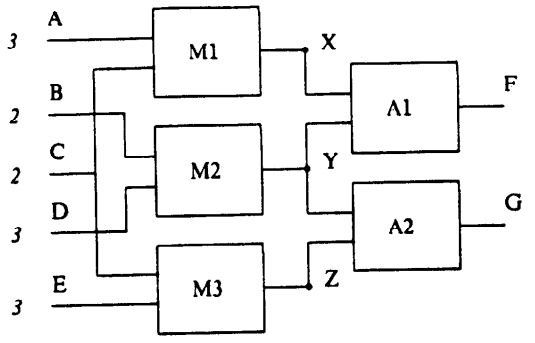

Fig. 1: A familiar circuit.

Earlier work in diagnosis has concentrated primarily on diagnosing failures produced by a single faulty component. When one entertains the possibility of multiple faults, the space of potential candidates grows exponentially with the number of faults under consideration. This work is aimed specifically at developing an efficient general method for diagnosing failures due to any number of simultaneous faults.

The focus of this paper is the process of analyzing the results of measurements to identify potential causes of variance (see [3] for an extensive discussion on the use of probabilistic information to guide the measurement process). This paper describes a general framework for diagnosis which, when coupled with a predictive inference component provides a powerful diagnostic procedure for dealing with multiple faults. In addition it also demonstrates the approach in the domain of digital electronics, using propagation as the predictive inference engine.

2. Model-artifact Differences

The model of the artifact describes the physical structure of the device in terms of its constituents. Each type of constituent obeys certain behavioral rules. For example, a simple electrical circuit consists of wires, resistors and so forth, where wires obey Kirchoff's Current Law, resistors obey Ohm's Law, and so on. In diagnosis, it is given that the behavior of the artifact differs from its model. It is then the task of the diagnostician to determine what these differences are.

The model for the artifact is a description of its physical structure, plus models for each of its constituents. A constituent is a very general concept, including components, processes and even steps in a logical inference. In addition, each constituent has associated with it a set of one or more possible model-artifact differences which establishes the grain size of the diagnosis.

Diagnosis takes (1) the physical structure, (2) models for each constituent, (3) a set of possible model-artifact differences and (4) a set of measurements, and produces a set of candidates, each of which is a set of differences which explains the observations.

Our diagnostic approach is based on characterizing model-artifact differences as assumption violations. A constituent is guaranteed to behave according to its model only if none of its associated differences are manifested, i.e., all the constituent's assumptions hold. If any of these assumptions are false, then the artifact deviates from its model, thus, the model may no longer apply. An important ramification of this approach ([1,2,3,6,8,11]) is that we need only specify correct models for constituents — explicit fault models are not needed.

Reasoning about model-artifact differences in terms of assumption violations is very general. For example, in electronics the assumptions might be the correct functioning of each component and the absence of any short circuits; in a scientific domain a faulty hypothesis; in a commonsense domain an assumption such as persistence, defaults or Occam's Razor.

3. Detection of Symptoms

We presume (as is usually the case) that the model-artifact differences are not directly observable.[2] Instead, all assumption violations must be inferred indirectly from behavioral observations. In section 8 we present a general inference architecture for this purpose, but for the moment we presume an inference procedure which makes behavioral predictions from observations and assumptions without being concerned about the procedure's details.

Intuitively, a *symptom* is any difference between a prediction made by the inference procedure and an observation. Consider our example circuit. Given the inputs, $A = 3$, $B = 2$, $C = 2$, $D = 3$, and $E = 3$, by simple calculation (i.e., the inference procedure), $F = X \times Y = A \times C + B \times D = 12$. However, F is measured to be 10. Thus "F is observed to be 10, not 12" is a symptom. More generally, a symptom is *any* inconsistency detected by the inference procedure, and may occur between two predictions (inferred from distinct measurements) as well as a

[1] This circuit is also used by both [2] and [8] in explaining their systems.

[2] In practice the diagnostician can sometimes directly observe a malfunctioning component by looking for a crack or burn mark.

measurement and a prediction (inferred from some other measurements).

4. Conflicts

The diagnostic procedure is guided by the symptoms. Each symptom tells us about one or more assumptions that are possibly violated (e.g., components that may be faulty). Intuitively, a *conflict* is a set of assumptions supporting a symptom, and thus leads to an inconsistency. In electronics, a conflict might be a set of components which cannot all be functioning correctly. Consider our example symptom "F is observed to be 10, not 12." Our calculation that $F = 12$ depends on the correct operation of M_1, M_2 and A_1, i.e., if M_1, M_2, and A_1 were correctly functioning, then $F = 12$. Since F is not 12, at least one of M_1, M_2 and A_1 is faulted. Thus the set $\langle M_1, M_2, A_1 \rangle$ (conflicts are indicated by $\langle ... \rangle$) is a conflict for the symptom. Because the inference is monotonic with the set of assumptions, the set $\langle M_1, M_2, A_1, A_2 \rangle$, and any other superset of $\langle M_1, M_2, A_1 \rangle$ are conflicts as well; however, no subsets of $\langle M_1, M_2, A_1 \rangle$ are necessarily conflicts since all the components in the conflict were necessary to constrain the value at F.

A measurement might agree with one prediction and yet disagree with another, resulting in a symptom. For example, starting with the inputs $B = 2$, $C = 2$, $D = 3$, and $E = 3$, and assuming M_2, M_3 and A_2 are correctly functioning we calculate G to be 12. However, starting with the observation $F = 10$, the inputs $A = 3$, $C = 2$, and $E = 3$, and assuming that A_1, A_2, M_1, and M_3, (i.e., ignoring M_2) are correctly functioning we calculate $G = 10$. Thus, when G is measured to be 12, even though it agrees with the first prediction, it still produces a conflict based on the second: $\langle A_1, A_2, M_1, M_3 \rangle$.

For complex domains any single symptom can give rise to a large set of conflicts, including the set of all components in the circuit. To reduce the combinatorics of diagnosis it is essential that the set of conflicts be represented and manipulated concisely. If a set of components is a conflict, then every superset of that set must also be a conflict. Thus the set of conflicts can be represented concisely by only identifying the minimal conflicts, where a conflict is minimal if it has no proper subset which is also a conflict. This observation is central to the performance of our diagnostic procedure. The goal of *conflict recognition* is to identify the complete set of minimal conflicts.[3] Typically, but not always, each symptom corresponds to a single minimal conflict.

[3] Representing the conflict space in terms of minimal conflicts is analogous to the idea of version spaces for representing plausible hypothesis in single concept learning ([10]).

5. Candidates

A *candidate* is a particular hypothesis for how the actual artifact differs from the model. Ultimately, the goal of diagnosis is to identify, and refine, the set of candidates consistent with the observations thus far.

A candidate is represented by a set of assumptions (indicated by [...]). Every assumption mentioned in the set must fail to hold. As every candidate must explain every symptom (i.e., its conflicts), each set representing a candidate must have a non-empty intersection with every conflict.

For electronics, a candidate is a set of failed components, where any components not mentioned are guaranteed to be working. Before any measurements have been taken we know nothing about the circuit. The size of the initial candidate space grows exponentially with the number of components. Any component could be working or faulty, thus the candidate space for Fig. 1 initially consists of $2^5 = 32$ candidates.

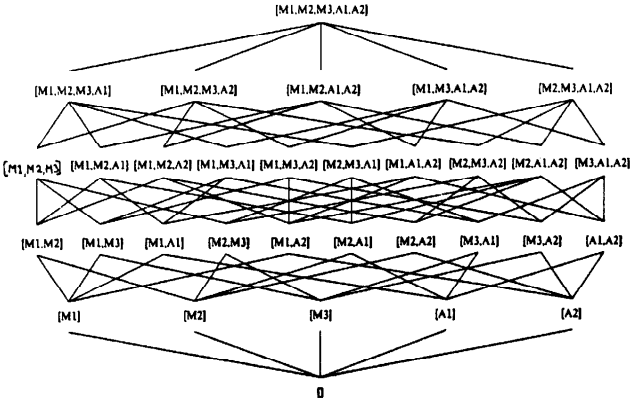

Fig. 2 Initial candidate space for circuit example.

It is essential that candidates be represented concisely as well. Notice that, like conflicts, candidates have the property that any superset of a candidate must be a candidate as well. Thus the space of all candidates consistent with the observations can be represented by the *minimal candidates*. The goal of *candidate generation* is to identify the complete set of minimal candidates. The space of candidates can be visualized in terms of a subset-superset lattice (Fig. 2). The minimal candidates then define a boundary such that everything from the boundary up is a valid candidate, while everything below is not.

Given no measurements every component might be working correctly, thus the single minimal candidate is the empty set, [], which is the root of the lattice at the bottom of Fig. 2.

To summarize, the set of candidates is constructed in two stages: conflict recognition and candidate generation. Conflict recognition uses the observations made along with a model of the device to construct a complete set of min-

imal conflicts. Next, candidate generation uses the set of minimal conflicts to construct a complete set of minimal candidates. Candidate generation is the topic of the next section, While conflict recognition is discussed in Section 7.

6. Candidate Generation

Diagnosis is an incremental process; as the diagnostician takes measurements he continually refines the candidate space and then uses this to guide further measurements. Within a single diagnostic session the total set of candidates must decrease monotonically. This corresponds to having the minimal candidates move monotonically up through the candidate superset lattice towards the candidate represented by the set of all components. Similarly, the total set of conflicts must increase monotonically. This corresponds to having the minimal conflicts move monotonically down through a conflict superset lattice towards the conflict represented by the empty set. Candidates are generated incrementally, using the new conflict(s) and the old candidate(s) to generate the new candidate(s).

The set of candidates is incrementally modified as follows. Whenever a new conflict is discovered, any previous minimal candidate which does not explain the new conflict is replaced by one or more superset candidates which are minimal based on this new information. This is accomplished by moving up along the lattice from the old minimal candidate, recording the first candidate reached which explains the new conflict; i.e., when the candidate's intersection with the new conflict is non-empty. When moving up past a candidate with more than one parent a consistent candidate must be found along each branch. Eliminated from those candidates recorded are any which are subsumed or duplicated; the remaining candidates are added to the set of new candidates.

Consider our example. Initially there are no conflicts, thus the minimal candidate [] (i.e., everything working) explains all observations. We have already seen that the single symptom "$F = 10$ not 12" produces one conflict $\langle A_1, M_1, M_2 \rangle$. This rules out the single minimal candidate []. Thus, its immediate supersets $[M_1]$, $[M_2]$, $[M_3]$, $[A_1]$, and $[A_2]$ are examined. Each of the candidates $[M_1]$, $[M_2]$, and $[A_1]$ explain the new conflict and thus are recorded; however, $[A_2]$ and $[M_3]$ do not. All of their immediate superset candidates except for $[A_2, M_3]$ are supersets of the three minimal candidates discovered above. $[A_2, M_3]$ does not explain the new conflict, however, its immediate superset candidates are supersets of the three minimal candidates and thus are implicitly represented. Therefore, the new minimal candidate set consists of $[M_1]$, $[M_2]$, and $[A_1]$.

The second conflict (inferred from observation $G = 12$), $\langle A_1, A_2, M_1, M_3 \rangle$, only eliminates minimal candidate $[M_2]$; the unaffected candidates $[M_1]$, and $[A_1]$ remain minimal. However, to complete the set of minimal candidates we must consider the supersets of $[M_2]$: $[A_1, M_2]$, $[A_2, M_2]$, $[M_1, M_2]$, and $[M_2, M_3]$. Each of these candidates explains the new conflict, however, $[A_1, M_2]$ and $[M_1, M_2]$ are supersets of the minimal candidates $[A_1]$ and $[M_1]$, respectively. Thus the new minimal candidates are $[A_2, M_2]$, and $[M_2, M_3]$, resulting in the minimal candidate set: $[A_1]$, $[M_1]$, $[A_2, M_2]$, and $[M_2, M_3]$.

Candidate generation has several interesting properties: First, the set of minimal candidates may increase or decrease in size as a result of a measurement; however, a candidate, once eliminated can never reappear. As measurements accumulate the sizes of the minimal candidates never decrease. Second, if an assumption appears in every minimal candidate (and thus every candidate), then that assumption is necessarily false. Third, the presupposition that there is only a single fault (exploited in all previous model-based troubleshooting strategies), is equivalent to assuming all candidates are singletons. In this case, the set of candidates can be obtained by intersecting all the conflicts.

7. Conflict Recognition Strategy

The remaining task involves incrementally constructing the conflicts used by candidate generation. In this section we first present a simple model of conflict recognition. This approach is then refined into an efficient strategy.

A conflict can be identified by selecting a set of assumptions, referred to as an *environment*, and testing if they are inconsistent with the observations.[4] If they are, then the inconsistent environment is a conflict. This requires an inference strategy C(OBS,ENV) which given the set of observations OBS made thus far, and the environment ENV, determines whether the combination is consistent. In our example, after measuring $F = 10$, and before measuring $G = 12$, C($\{F = 10\}$, $\{M_1, M_2, A_1\}$) (leaving off the inputs) is false indicating the conflict $\langle M_1, M_2, A_1 \rangle$. This approach is refined as follows:

Refinement 1: Exploiting minimality. To identify the set of minimal inconsistent environments (and thus the minimal conflicts), we begin our search at the empty environment, moving up along its parents. This is similar to the search pattern used during candidate generation. At each environment we apply C(OBS,ENV) to determine

[4] An environment should not be confused with a candidate. An environment is a set of assumptions all of which are assumed to be true (e.g., M_1 and M_2 are assumed to be working correctly), a candidate is a set of assumptions all of which are assumed to be false (e.g., components M_1 and M_2 are not functioning correctly). A conflict is a set of assumptions, at least one of which is false. Intuitively an environment is the set of assumptions that define a "context" in a deductive inference engine, in this case the engine is used for prediction and the assumptions are about the lack of particular model-artifact differences.

whether or not ENV is a conflict. Before a new environment is explored, all other environments which are a subset of the new environment must be explored first. If the environment is inconsistent then it is a minimal conflict and its supersets are not explored. If an environment has already been explored or is a superset of a conflict then **C** is not run on the environment and its supersets are not explored.

We presume the inference strategy operates entirely by inferring hypothetical predictions (e.g., values for variables in environments given the observations made). Let **P**(OBS,ENV) be all behavioral predictions which follow from the observations OBS given the assumptions ENV. For example, **P**($\{A = 3, B = 2, C = 2, D = 3\}$, $\{A_1, M_1, M_2\}$) produces $\{A = 3, B = 2, C = 2, D = 3, X = 6, Y = 6, F = 12\}$.

C can now be implemented in terms of **P**. If **P** computes two distinct values for a quantity (or more simply both x and $\sim x$), then ENV is a conflict.

Refinement 2: Monotonicity of measurements. If inputs are kept constant, measurements are cumulative and our knowledge of the circuit's structure grows monotonically. Given a new measurement M, **P**(OBS$\cup\{M\}$, ENV) is always a superset of **P**(OBS,ENV). Thus if we cache the values of every **P**, when a new measurement is made we need only infer the incremental addition to the set of predictions.

Refinement 3: Monotonicity for assumptions. Analogous to refinement 2, the set of predictions grows monotonically with the environment. If a set of predictions follow from an environment, then the addition of any assumption to that environment only expands this set. Therefore **P**(OBS,ENV) contains **P**(OBS,E) for every subset E of ENV. This makes the computation of **P**(OBS,ENV) very simple if all its subsets have already been analyzed.

Refinement 4: Redundant Inferences. **P** must be run on every possible environment. Thus, we need a large set of data-bases, and the same rule will be executed over and over again on the same antecedents. All of this overlap can be avoided by utilizing ideas of Truth Maintenance such that every inference is recorded as a dependency and no inference is ever performed twice [7].

Refinement 5: Exploiting locality. This is primarily an observation of why the previous refinements are successful. The first four refinements allow the strategy to ignore (i.e., to the extent of not even generating its name) any environment which doesn't contain some interesting inferences absent in every one of its subsets. If every environment contained a new unique inference, then we would still be faced with a computation exponential in the number of potential model-artifact differences. However, in practice, as the components are weakly connected, the inferences rules are weakly connected. Our strategy depends on this empirical property. For example, in electronics the only assumption sets of interest will be sets of components which are connected and whose signals interact — typically circuits are explicitly designed so that component interactions are limited.

8. Inference Procedure Architecture

To completely exploit the ideas discussed in the preceding section we need to modify and augment the implementation of **P**. We presume that **P** meets (or can be modified to) the two basic criteria for utilizing truth maintenance: (1) A dependency (i.e., justification) can be constructed for each inference, and (2) belief or disbelief in a datum is completely determined by these dependencies. In addition, we presume that, during processing, whenever more than one inference is simultaneously permissible, that the actual order in which these inferences are performed is irrelevant and that this order can be externally controlled (i.e., by our architecture). Finally, we presume that the inference procedure is monotonic. Most AI inference procedures meet these four general criteria. For example, many expert rule-based systems, constraint propagation, demon invocation, taxonomic reasoning, qualitative simulations, natural deduction systems, and many forms of resolution theorem-proving fit this general framework.

We associate with every prediction, V, the set of environments, ENVS(V), from which it follows (i.e., ENVS(V) $\equiv \{env | V \in \mathbf{P}(OBS, env)\}$). We call this set the supporting environments of the prediction. Exploiting the monotonicity property, it is only necessary to represent the minimal (under subset) supporting environments.

Consider our example after the measurements $F = 10$ and $G = 12$. In this case we can calculate $X = 6$ in two different ways. First, $Y = B \times D = 6$ assuming M_2 is functioning correctly. Thus, one of its supporting environments is $\{M_2\}$. Second, $Y = G - Z = G - (C \times E) = 6$ assuming A_2 and M_3 are working. Therefore the supporting environments of $Y = 6$ are $\{\{M_2\}\{A_2, M_3\}\}$. Any set of assumptions used to derive $Y = 6$ is a superset of one of these two.

By exploiting dependencies no inference is ever done twice. If the supporting environment of a fact changes, then the supporting environments of its consequents are updated automatically by tracing the dependencies created when the rule was first run. This achieves the effect of rerunning the rule without incurring any computational overhead.

We control the inference process such that whenever two inferences are possible, the one producing a datum in the smaller environment is performed first. A simple agenda mechanism suffices for this. Whenever a symptom is recognized, the environment is marked a conflict and all inferencing stops on that environment. Using this control scheme facts are always deduced in their minimal environment, achieving the desired property that only minimal

conflicts (i.e., inconsistent environments) are generated.

In this architecture P can be incomplete (in practice it usually is). The only consequence of incompleteness is that fewer conflicts will be detected and thus fewer candidates will be eliminated than the ideal — no candidate will be mistakenly eliminated.

9. Circuit Diagnosis

Thus far we have described a very general diagnostic strategy for handling multiple faults, whose application to a specific domain depends only on the selection of the function **P**. During the remainder of this paper, we demonstrate the power of this approach, by applying it to the problem of circuit diagnosis.

For our example we make a number of simplifying presuppositions. First, we assume that the model of a circuit is described in terms of a circuit topology plus a behavioral description of each of its components. Second, that the only type of model-artifact difference considered is whether or not a particular component is working correctly. Finally, all observations are made in terms of measurements at a component's terminals. Measurements are expensive, thus not every value at every terminal is known. Instead, some values must be inferred from other values and the component models. Intuitively, symptoms are recognized by propagating out locally through components from the measurement points, using the component models to deduce new values. The application of each model is based on the assumption that its corresponding component is working correctly. If two values are deduced for the same quantity in different ways, then a coincidence has occurred. If the two values differ then the coincidence is a symptom. The conflict then consists of every component propagated through from the measurement points to the point of coincidence (i.e., the symptom implies that at least one of the components used to deduce the two values is inconsistent).

10. Constraint Propagation

Constraint propagation [12,13] operates on cells, values, and constraints. Cells represent state variables such as voltages, logic levels, or fluid flows. A constraint stipulates a condition that the cells must satisfy. For example, Ohm's law, $v = iR$, is represented as a constraint among the three cells v, i, and R. Given a set of initial values, constraint propagation assigns each cell a value that satisfies the constraints. The basic inference step is to find a constraint that allows it to determine a value for a previously unknown cell. For example, if it has discovered values $v = 2$ and $i = 1$, then it uses the constraint $v = iR$ to calculate the value $R = 2$. In addition, the propagator records R's dependency on v, i and the constraint $v = iR$. The newly recorded value may cause other constraints to trigger and more values to be deduced. Thus, constraints may be viewed as a set of conduits along which values can be propagated out locally from the inputs to other cells in the system. The dependencies recorded trace out a particular path through the constraints that the inputs have taken. A symptom is manifested when two different values are deduced for the same cell (i.e., a logical inconsistency is identified). In this event dependencies are used to construct the conflict.

Sometimes the constraint propagation process terminates leaving some constraints unused and some cells unassigned. This usually arises as a consequence of insufficient information about device inputs. However, it can also arise as the consequence of logical incompleteness in the propagator.

In the circuit domain, the behavior of each component is modeled as a set of constraints. For example, in analyzing analog circuits the cells represent circuit voltages and currents, the values are numbers, and the constraints are mathematical equations. In digital circuits, the cells represent logic levels, the values are 0 and 1, and the constraints are boolean equations.

Consider the constraint model for the circuit of Fig. 1. There are ten cells: A, B, C, D, E, X, Y, Z, F, and G, five of which are provided the observed values: $A = 3$, $B = 2$, $C = 2$, $D = 3$ and $E = 3$. There are three multipliers and two adders each of which is modeled by a single constraint: $M_1 : X = A \times C$, $M_2 : Y = B \times D$, $M_3 : Z = C \times E$, $A_1 : F = X + Y$, and $A_2 : G = Y + Z$. The following is a list of deductions and dependencies that the constraint propagator generates (a dependency is indicated by (component : antecedents):

$$X = 6 \ (M_1 : A = 3, C = 2)$$

$$Y = 6 \ (M_2 : B = 2, D = 3)$$

$$Z = 6 \ (M_3 : C = 2, E = 3)$$

$$F = 12 \ (A_1 : X = 6, Y = 6)$$

$$G = 12 \ (A_2 : Y = 6, Z = 6)$$

A symptom is indicated when two values are determined for the same cell (e.g., measuring F to be 10 not 12). Each symptom leads to new conflict(s) (e.g., in this example the symptom indicates a conflict $\langle A_1, M_1, M_2 \rangle$).

This approach has some important properties. First, it is not necessary for the starting points of these paths to be inputs or outputs of the circuit. A path may begin at any point in the circuit where a measurement has been taken. Second, it is not necessary to make any assumptions about the direction that signals flow through components. In most digital circuits a signal can only flow from inputs to outputs. For example, a subtractor cannot be constructed by simply reversing an input and the output

of an adder since it violates the directionality of signal flow. However, the directionality of a component's signal flow is irrelevant to our diagnostic technique: a component places a constraint between the values of its terminals which can be used any way desired. To detect discrepancies, information can flow along a path through a component in any direction. For example, although the subtractor does not function in reverse, when we observe its outputs we can infer what its inputs must have been.

11. Generalized Constraint Propagation

Each step of constraint propagation takes a set of antecedent values and computes a consequent. We have built a constraint propagator within our inference architecture which explores minimal environments first. This guides each step during propagation in an efficient manner to incrementally construct minimal conflicts and candidates for multiple faults.

Consider our example. We ensure that propagations in subset environments are performed first, thereby guaranteeing that the resulting supporting environments and conflicts are minimal. We use $[\![x, e_1, e_2, ...]\!]$ to represent the assertion x with its associated supporting environments. Before any measurements or propagations take place, given only the inputs, the data base consists of: $[\![A = 3, \{\}]\!]$, $[\![B = 2, \{\}]\!]$, $[\![C = 2, \{\}]\!]$, $[\![D = 3, \{\}]\!]$, and $[\![E = 3, \{\}]\!]$. Observe that when propagating values through a component, the assumption for the component is added to the dependency, and thus to the supporting environment(s) of the propagated value. Propagating A and C through M_1 we obtain: $[\![X = 6, \{M_1\}]\!]$. The remaining propagations produce: $[\![Y = 6, \{M_2\}]\!]$, $[\![Z = 6, \{M_3\}]\!]$, $[\![F = 12, \{A_1, M_1, M_2\}]\!]$, and $[\![G = 12, \{A_2, M_2, M_3\}]\!]$.

Suppose we measure F to be 10. This adds $[\![F = 10, \{\}]\!]$ to the data base. Analysis proceeds as follows (starting with the smaller assumption sets first): $[\![X = 4, \{A_1, M_2\}]\!]$, and $[\![Y = 4, \{A_1, M_1\}]\!]$. Now the symptom between $[\![F = 10, \{\}]\!]$ and $[\![F = 12, \{A_1, M_1, M_2\}]\!]$ is recognized indicating a new minimal conflict: $\langle A_1, M_1, M_2 \rangle$. Thus the inference architecture prevents further propagation in the environment $\{A_1, M_1, M_2\}$ and its supersets. The propagation goes one more step: $[\![G = 10, \{A_1, A_2, M_1, M_3\}]\!]$. There are no more inferences to be made.

Next, suppose we measure G to be 12. Propagation gives: $[\![Z = 6, \{A_2, M_3\}]\!]$, $[\![Y = 6, \{A_2, M_2\}]\!]$, $[\![Z = 8, \{A_1, A_2, M_1\}]\!]$, and $[\![X = 4, \{A_1, A_2, M_3\}]\!]$. The symptom "$G = 12$ not 10" produces the conflict $\langle A_1, A_2, M_1, M_3 \rangle$. The final data-base state is:[5]

$A = 3, \{\}$
$B = 2, \{\}$
$C = 2, \{\}$
$D = 3, \{\}$
$E = 3, \{\}$
$F = 10, \{\}$
$G = 12, \{\}$
$X = 4, \{A_1, M_2\}\{A_1, A_2, M_3\}$
$\quad 6, \{M_1\}$
$Y = 4, \{A_1, M_1\}$
$\quad 6, \{M_2\}\{A_2, M_3\}$
$Z = 8, \{A_1, A_2, M_1\}$
$\quad 6, \{M_3\}\{A_2, M_2\}$

This results in two minimal conflicts:

$$\langle A_1, M_1, M_2 \rangle$$

$$\langle A_1, A_2, M_1, M_3 \rangle$$

Note that at no point during propagation is effort wasted in constructing non-minimal conflicts.

The algorithm discussed in section 6 uses the two minimal conflicts to incrementally construct the set of minimal candidates. Given new measurements the propagation/candidate generation cycle continues until the candidate space has been sufficiently constrained

12. Connected Research

Our approach has been completely implemented and tested on numerous examples. Our implementation consists of four basic modules. The first maintains the minimal supporting environments for each prediction and constructs minimal conflicts. It is based on Assumption-Based Truth Maintenance [4]. The second controls the inference such that minimal conflicts are discovered first and records the dependencies of inferences. It is based on the consumer and agenda architectures of [5]. The third is a general constraint language based on the first two modules. The last module, the candidate generator, incrementally constructs the minimal candidates from the minimal conflicts.

As all the work within the model-based paradigm, our approach presumes measurements and potential model-artifact differences are given. In [3] we exploit the framework of this paper in two ways to generate measurements which are information-theoretically optimal. First, the data structures constructed by our strategy (e.g., the data base state of Section 11) make it easy to consider and evaluate hypothetical measurements. Second, as we construct all minimal environments, conflicts, and candidates, it is relatively straight forward to compare potential measurements (using probabilistic information of component failure rates).

The work presented here represents another step towards the goal of automated diagnosis, nevertheless there remains much to be done. Plans for the future include: 1) incorporating the predictive engine discussed in [14] in order to diagnosis systems with time-varying signals and

[5] The justifications are not shown but are the same as those in section 10.

state, and 2) controlling the set of model-artifact differences being considered.

13. Related Work

This research fits within the model-based debugging paradigm: [1,2,3,6,8, 9,11]. However, unlike [1,2,6,8, 9], we propose a general method of diagnostic reasoning which is efficient, incremental, handles multiple faults, and is easily extended to include measurement strategies. Reiter [11] has been exploring these ideas independently and provides a formal account of many of our "intuitive" techniques of conflict recognition and candidate generation.

ACKNOWLEDGMENTS

Daniel G. Bobrow, Randy Davis, Kenneth Forbus, Matthew Ginsberg, Frank Halasz, Walter Hamscher, Tad Hogg, Ramesh Patil, provided useful insights. We especially thank Ray Reiter for his clear perspective and many productive interactions.

BIBLIOGRAPHY

1. Brown, J.S., Burton, R. R. and de Kleer, J., Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III, in: D. Sleeman and J.S. Brown (Eds.), *Intelligent Tutoring Systems*, (Academic Press, New York, 1982) 227–282.
2. Davis, R., Shrobe, H., Hamscher, W., Wieckert, K., Shirley, M. and Polit, S., Diagnosis based on description of structure and function, in: *Proceedings of the National Conference on Artificial Intelligence*, Pittsburgh, PA (August, 1982) 137–142.
3. de Kleer, J. and Williams, B.C., Diagnosing multiple faults, *Artificial Intelligence* (1986) *forthcoming.*
4. de Kleer, J., An assumption-based truth maintenance system, *Artificial Intelligence* **28** (1986) 127–162.
5. de Kleer, J., Problem solving with the ATMS, *Artificial Intelligence* **28** (1986) 197–224.
6. de Kleer, J., Local methods of localizing faults in electronic circuits, Artificial Intelligence Laboratory, AIM-394, Cambridge: M.I.T., 1976.
7. Doyle, J., A truth maintenance system, *Artificial Intelligence* **24** (1979).
8. Genesereth, M.R., The use of design descriptions in automated diagnosis, *Artificial Intelligence* **24** (1984), 411–436.
9. Hamscher, W., and Davis, R., Diagnosing circuits with state: an inherently underconstrained problem, in: *Proceedings of the National Conference on Artificial Intelligence,* Austin, TX (August, 1984) 142–147.
10. Mitchell, T., Version spaces: An approach to concept learning, Computer Science Department, STAN-CS-78-711, Palo Alto: Standford University, 1978.
11. Reiter, R., A theory of diagnosis from first principles, *Artificial Intelligence*, forthcomming. Also: Department of Computer Science Technical Report 187/86, (University of Toronto, Toronto, 1985).
12. Steele, G.L., The definition and implementation of a computer programming language based on constraints, AI Technical Report 595, MIT, Cambridge, MA, 1979.
13. Sussman, G.J. and Steele, G.L., CONSTRAINTS: A language for expressing almost-hierarchical descriptions, *Artificial Intelligence* **14** (1980) 1-39.
14. Williams, B.C., "Doing Time: Putting Qualitative Reasoning on Firmer Ground," *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, Penn., (August, 1984).