# Diagnosing Multiple Faults

**Johan de Kleer**
*Intelligent Systems Laboratory, XEROX Palo Alto Research Center, Palo Alto, CA 94304, U.S.A.*

**Brian C. Williams**
*Artificial Intelligence Laboratory, MIT, Cambridge, MA 02139, U.S.A.*

Recommended by Judea Pearl

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 many 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, exploiting the iterative nature of diagnosis. Fourth, a clear separation is drawn between diagnosis and behavior prediction, resulting in a domain (and inference procedure) independent diagnostic procedure. Fifth, GDE combines model-based prediction with sequential diagnosis to propose measurements to localize the faults. The normally required conditional probabilities are computed from the structure of the device and models of its components. This capability results from a novel way of incorporating probabilities and information theory into the context mechanism provided by assumption-based truth maintenance.*

## 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 common-sense 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. 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 (Fig. 1).

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.

Earlier research work (see Section 6) on model-based diagnosis concentrated on determining a single faulty component that explains all the symptoms. This paper extends that research by diagnosing systems with multiple failed components, and by proposing a sequence of measurements which efficiently localize the failing components.

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, referred to as the general diagnostic engine (GDE), for diagnosing failures due to any number of simultaneous faults. To achieve the needed efficiency, GDE exploits the features of assumption-based truth maintenance (ATMS) [8]. This is the topic of the first half of the paper.

Usually, additional measurements are necessary to isolate the set of components which are actually faulted. The best next measurement is the one which will, on average, lead to the discovery of the faulted set of components in a minimum number of measurements. Unlike other probabilistic techniques which require a vast number of conditional probabilities, GDE need only be provided with the a priori probabilities of individual component failure. Using an ATMS, this probabilistic information can be incorporated into GDE such that it is straightforward to compute the conditional probabilities of the candidates, as well as the probabilities of the possible outcomes of measurements, based on the faulty device's model. This combination of probabilistic inference and assumption-based truth maintenance enables GDE to apply a minimum entropy method [1] to determine what measurement to make next: the best measurement is the one which minimizes the expected entropy of candidate prob-

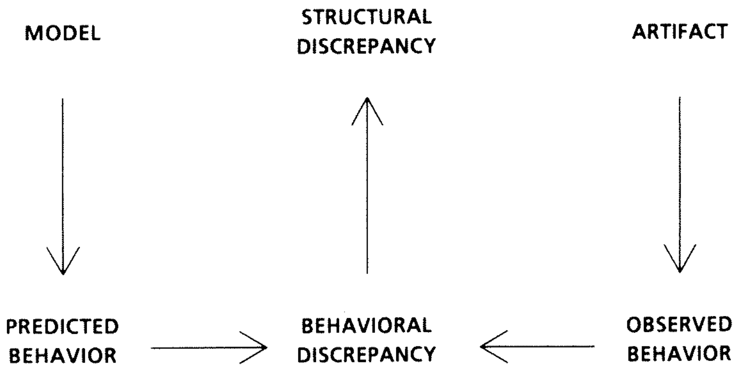

FIG. 1. Model-artifact difference.

abilities resulting from the measurement. This is the topic of the second half of the paper.

## 1.1. Troubleshooting circuits

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. The task for the diagnostician is to use the results of measurements to identify the cause of the variance when possible, and otherwise to determine which additional measurements must be taken.

For example, consider the circuit in Fig. 2, 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 $[\cdot]$): $[A_1]$, $[M_1]$, $[A_2, M_2]$, 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 $[M_1]$.

Next the value of $X$ is measured and the result is used to reduce the size of the candidate set. The candidate generation-measurement process continues until a single high-probability candidate remains.

[1] This circuit is also used by both [5] and [12] in explaining their systems.

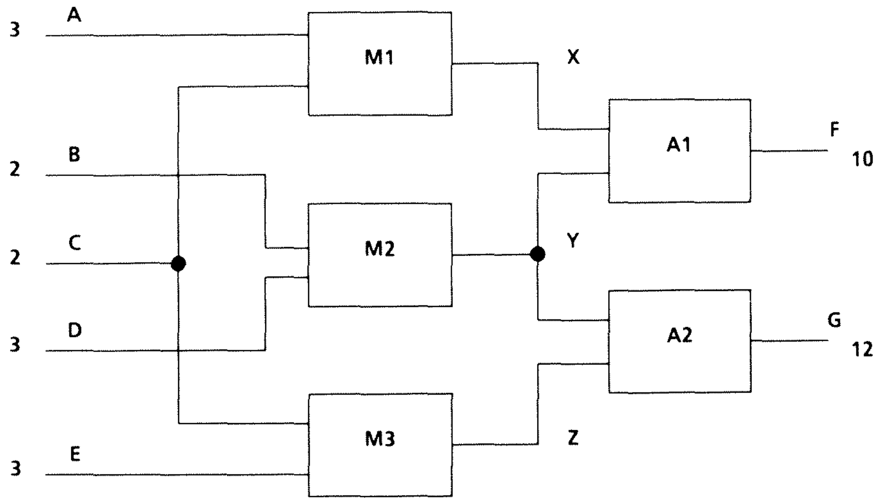

FIG. 2. A familiar circuit.

### 1.2. Some basic presuppositions

Although GDE considers multiple faults and probabilistic information, it shares many of the basic presuppositions of other model-based research. We presume that the act of taking a measurement (i.e., making an observation) has no affect on the faulty device. We presume that once a quantity is measured to be a certain value, that the quantity remains at the value. This is equivalent to assuming that no component's (correct or faulty) functioning depends on the passage of time. For example, this rules out flip-flops as well as intermittent components which spontaneously change their behavior. We presume that if a component is faulty, the distribution of input-output values becomes random (i.e., contains no information). We do not presume that if a component is faulty, that it must be exhibiting this faulty behavior—it may exhibit faulty behavior on some other set of inputs. These presuppositions suggest future directions for research, and we are extending GDE in these directions.

## 2. A Theory of Diagnosis

The remainder of this paper presents a general, domain-independent, diagnostic engine (GDE) which, when coupled with a predictive inference component provides a powerful diagnostic procedure for dealing with multiple faults. In addition the approach is demonstrated in the domain of digital electronics, using propagation as the predictive inference engine.

## 2.1. 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 Kirchhoff'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 [4,5,10,12,29] 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 an assumption 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.

## 2.2. 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 2.7 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

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

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

## 2.3. Conflicts

The diagnostic procedure is guided by the symptoms. Each symptom tells us about one or more assumptions that are possibly violated (e.g., component that may be faulty). Intuitively, a *conflict* is a set of assumptions which support a symptom, and thus leads to an inconsistency. In this electronics example, a conflict is a set of components which cannot all be functioning correctly. Consider the example symptom "$F$ is observed to be 10, not 12." The prediction that $F = 12$ depends on the correct operation of $A_1$, $M_1$, and $M_2$, i.e., if $A_1$, $M_1$, and $M_2$ were correctly functioning, then $F = 12$. Since $F$ is not 12, at least one of $A_1$, $M_1$, and $M_2$ is faulted. Thus the set $\langle A_1, M_1, M_2 \rangle$ is a conflict for the symptom (conflicts are indicated by $\langle \cdot \rangle$). Because the inference is monotonic with the set of assumptions, the set $\langle A_1, A_2, M_1, M_2 \rangle$, and any other superset of $\langle A_1, M_1, M_2 \rangle$ are conflicts as well; however, no subsets of $\langle A_1, M_1, M_2 \rangle$ are necessarily conflicts since all the components in the conflict were needed to predict 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 $A_2$, $M_2$, and $M_3$ 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]

[3] Representing the conflict space in terms of minimal conflicts is analogous to the idea of version spaces for representing plausible hypotheses in single concept learning [19].

### 2.4. Candidates

A *candidate* is a particular hypothesis for how the actual artifact differs from the model. For example "$A_2$ and $M_2$ are broken" is a candidate for the two symptoms observed for our example circuit. 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 $[\cdot]$). The assumptions explicitly mentioned are false, while the ones not mentioned are true. A candidate which explains the current set of symptoms is a set of assumptions such that if every assumption fails to hold, then every known symptom is explained. Thus each set representing a candidate must have a nonempty 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 candidate space is the set of candidates consistent with the observations. 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. 2 initially consists of $2^5 = 32$ candidates.

It is essential that candidates be represented concisely as well. Notice that, like conflicts, candidates have the property that any superset of a possible candidate for a set of symptoms must be a possible candidate as well. Thus the candidate space can be represented by the *minimal candidates*. Representing and manipulating the candidate space in terms of minimal candidates is crucial to our diagnostic approach. Although the candidate space grows exponentially with the number of potentially faulted components, it is usually the case that the symptoms can be explained by relatively few 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. 3). 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. 3.

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 minimal 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 2.6.

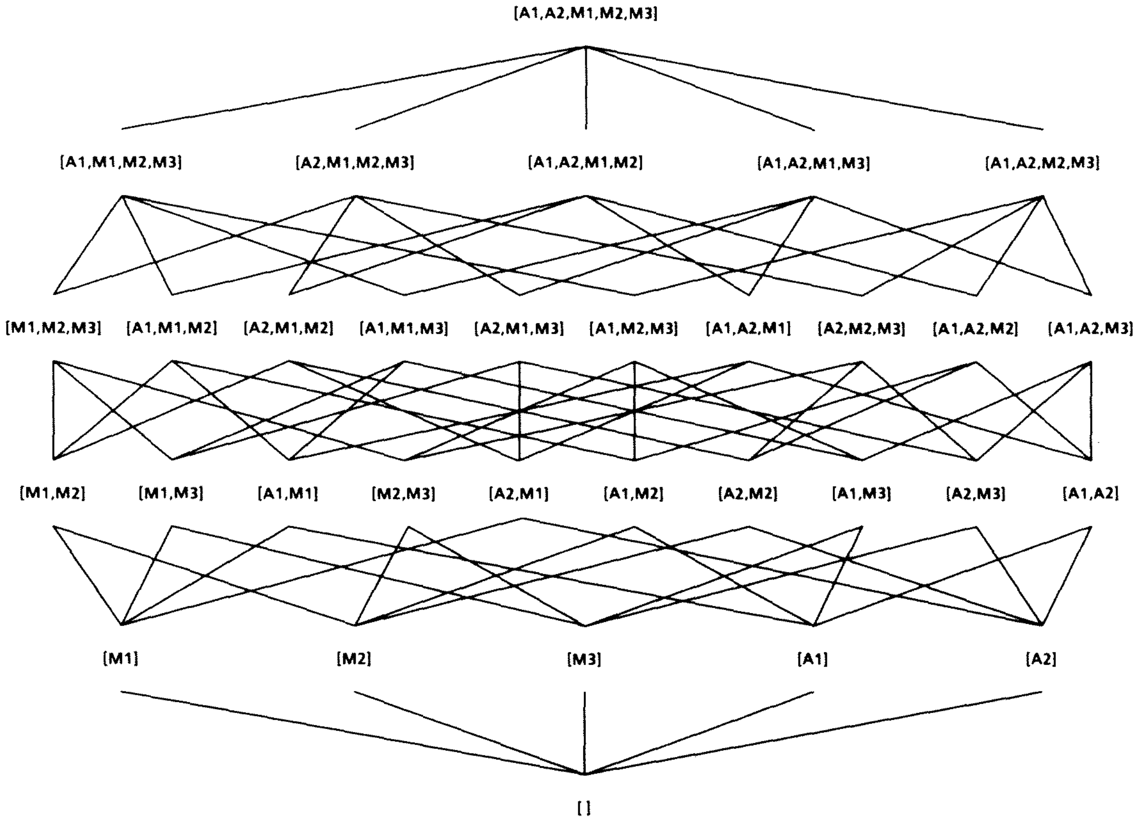

FIG. 3. Initial candidate space for the circuit example.

### 2.5. 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 containing 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 minimal conflict(s) and the old minimal candidate(s) to generate the new minimal candidate(s).

The set of minimal candidates is incrementally modified as follows. Whenever a new minimal 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 replacing the old minimal candidate with a set of new tentative minimal candidates each of which contains the old candidate plus one assumption from the new conflict. Any tentative new candidate which is subsumed or duplicated by another is eliminated; the remaining candidates are added to the set of new minimal candidates.

Consider our example. Initially there are no conflicts, thus the minimal candidate [ ] (i.e., everything is 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 containing one assumption of the conflict $[A_1]$, $[M_1]$, and $[M_2]$ are considered. None of these are duplicated or subsumed as there were no other old minimal candidates. The new minimal candidates are $[A_1]$, $[M_1]$, and $[M_2]$. This situation is depicted with the lattice in Fig. 4. All candidates above the line labeled by the conflict "C1: $\langle A_1, M_1, M_2 \rangle$" are valid candidates.

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 minimal candidates

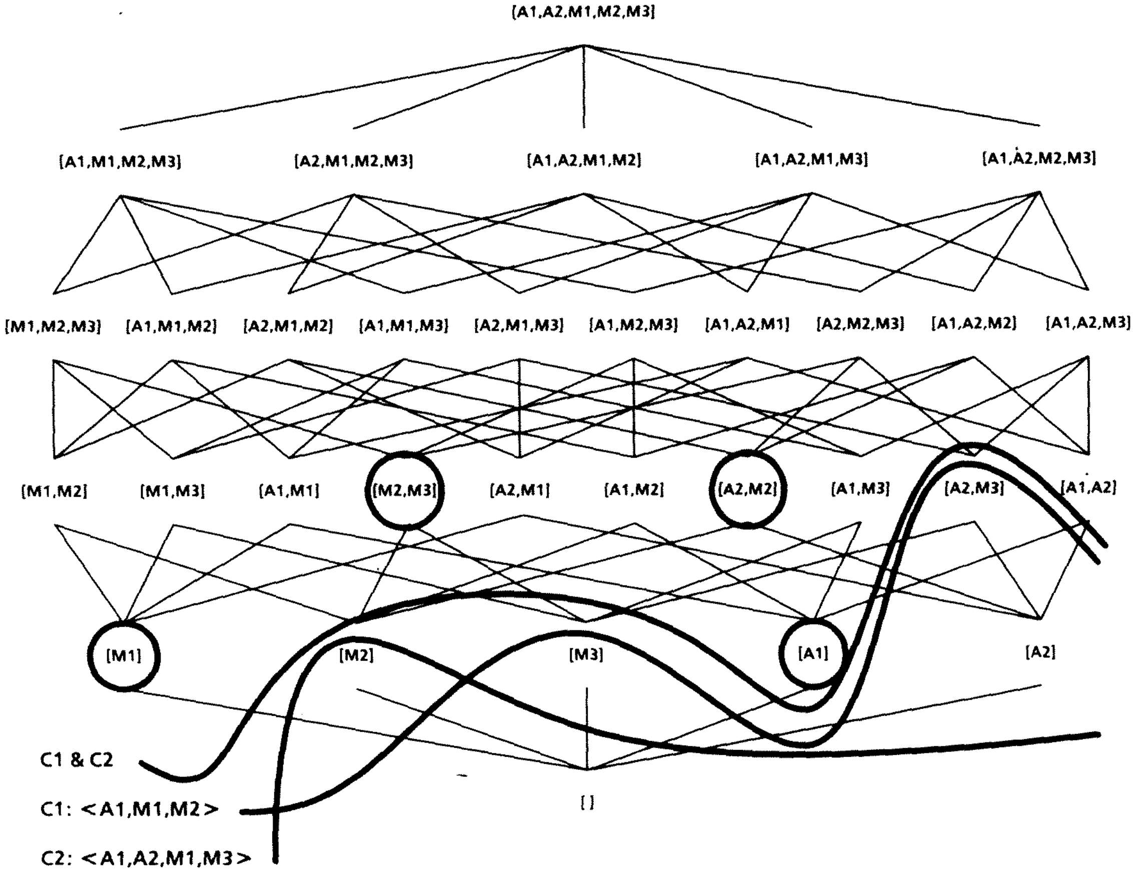

FIG. 4. Candidate space after measurements.

$[M_1]$, and $[A_1]$ remain. However, to complete the set of minimal candidates we must consider the immediate supersets of $[M_2]$ which cover the new conflict: $[A_1, M_2]$, $[A_2, M_2]$, $[M_1, M_2]$, 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 $[M_2, M_3]$. The line labeled by conflict "C2: $\langle A_1, A_2, M_1, M_3\rangle$" in Fig. 4 shows the candidates eliminated by the observation $G = 12$ alone, and the line labeled "C1 & C2" shows the candidates eliminated as a result of both measurements ($F = 10$ and $G = 12$). The minimal candidate which split the lattice into valid and eliminated candidates are circled.

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 eliminated minimal candidates are replaced by larger candidates. 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.

## 2.6. 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 $C(\text{OBS}, \text{ENV})$ 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 $G = 12$, $C(\{F = 10\}, \{A_1, M_1, M_2\})$ (leaving off the inputs) is false indicating the conflict $\langle A_1, M_1, M_2\rangle$. 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

[4] An environment should not be confused with a candidate or conflict. 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.

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 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 $\boldsymbol{C}$ 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 $\boldsymbol{P}$(OBS, ENV) be all behavioral predictions which follow from the observations OBS given the assumptions ENV. For example, $\boldsymbol{P}(\{A=3, B=2, C=2, D=3\}, \{A_1, M_1, M_2\})$ produces $\{A=3, B=2, C=2, D=3, X=6, Y=6, F=12\}$.

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

*Refinement* 2: *Monotonicity of measurements.* If input values are kept constant, measurements are cumulative and our knowledge of the circuit's structure grows monotonically. Given a new measurement $M$, $\boldsymbol{P}(\text{OBS} \cup \{M\}$, ENV) is always a superset of $\boldsymbol{P}$(OBS, ENV). Thus if we cache the values of every $\boldsymbol{P}$, 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 follows from the environment, then the addition of any assumption to that environment only expands this set. Therefore $\boldsymbol{P}$(OBS, ENV) contains $\boldsymbol{P}(\text{OBS}, E)$ for every subset $E$ of ENV. This makes the computation of $\boldsymbol{P}$(OBS, ENV) very simple if all its subsets have already been analyzed.

*Refinement* 4: *Redundant inferences.* $\boldsymbol{P}$ must be run on a large number of (overlapping) environments. Thus, the same rule will be executed over and over again on the same facts. 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 [11].

*Refinement* 5: *Exploiting the sparseness of the search space.* The 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 computationally with an exponential in the number of potential model-artifact differences. However, in practice, as the components are weakly connected, the inference rules are weakly connected. Therefore, it is more efficient to associate environments with rules than vice versa. 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.

### 2.7. Inference procedure architecture

To completely exploit the ideas discussed in the preceding section we need to modify and augment the implementation of $\boldsymbol{P}$. We presume that $\boldsymbol{P}$ 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, $\textsc{envs}(V)$, from which it follows (i.e., $\textsc{envs}(V) \equiv \{\text{env} \mid V \in \boldsymbol{P}(\textsc{obs}, \text{env})\}$). 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 environments of prediction change, then the supporting environments of its consequents are updated automatically by tracing the dependencies created when the rule was first run. This achieves the consequence of a deduction without rerunning the rule.

We control the inference process such that whenever more than one rule is runnable, the one producing a prediction in the smaller supporting environment is performed first. A simple agenda mechanism suffices for this. Whenever a symptom is recognized, the environment is marked a conflict and all rule execution stops on that environment. Using this control scheme predictions 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.

## 3. 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***. In this section, 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). Note however, if the two coinciding values are the same, then it is not necessarily the case that the components involved in the predictions are functioning correctly. Instead, it may be that the symptom simply does not manifest itself at that point. Also, it might be that one of these components is faulty, but does not manifest its fault, given the current set of inputs. (For example, an inverter with an output stuck at one will not manifest a symptom given an input of zero.) Thus if the coinciding values are in agreement then no information is gained.

### 3.1. Constraint propagation

Constraint propagation [33, 34] 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 recorded dependencies 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, this 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. 2. 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)):

$$\begin{aligned}
X &= 6 \quad (M_1 : A = 3, C = 2)\,, \\
Y &= 6 \quad (M_2 : B = 2, D = 3)\,, \\
Z &= 6 \quad (M_3 : C = 2, E = 3)\,, \\
F &= 12 \quad (A_1 : X = 6, Y = 6)\,, \\
G &= 12 \quad (A_2 : Y = 6, Z = 6)\,.
\end{aligned}$$

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 in 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.

### 3.2. 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 $[\![x, e_1, e_2, \ldots]\!]$ to represent the assertion $x$ with its associated supporting environments. Before any measurements or propagations take place, given only the inputs, the database consists of: $[\![A=3, \{\ \}]\!]$, $[\![B=2, \{\ \}]\!]$, $[\![C=2, \{\ \}]\!]$, $[\![D=3, \{\ \}]\!]$, and $[\![E=3, \{\ \}]\!]$. Observe that when propagating values through a component, the assumption for that 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 database. Analysis proceeds as follows (starting with the smaller environments 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: $[\![Z=6, \{A_2, M_2\}]\!]$, $[\![Y=6, \{A_2, M_3\}]\!]$, $[\![Z=8, \{A_1, A_2, M_1\}]\!]$, 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 database state is shown below.[5]

[5] The justifications are not shown but are the same as those in Section 3.1.

$$\begin{aligned}
&[\![A=3,\{\ \}]\!],\\
&[\![B=2,\{\ \}]\!],\\
&[\![C=2,\{\ \}]\!],\\
&[\![D=3,\{\ \}]\!],\\
&[\![E=3,\{\ \}]\!],\\
&[\![F=10,\{\ \}]\!],\\
&[\![G=12,\{\ \}]\!],\\
&[\![X=4,\{A_1,M_2\}\ \{A_1,A_2,M_3\}]\!],\\
&[\![X=6,\{M_1\}]\!],\\
&[\![Y=4,\{A_1,M_1\}]\!],\\
&[\![Y=6,\{M_2\}\ \{A_2,M_3\}]\!],\\
&[\![Z=8,\{A_1,A_2,M_1\}]\!],\\
&[\![Z=6,\{M_3\}\ \{A_2,M_2\}]\!].
\end{aligned} \tag{1}$$

This results in two minimal conflicts:

$$\langle A_1, M_1, M_2\rangle\,, \qquad \langle A_1, A_2, M_1, M_3\rangle\,.$$

The algorithm discussed in Section 2.5 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.

## 4. Sequential Diagnosis

In order to reduce the set of remaining candidates the diagnostician must perform measurements [14] which differentiate among the remaining candidates. This section presents a method for choosing a next measurement which best distinguishes the candidates, i.e., that measurement which will, on average, lead to the discovery of the actual candidate in a minimum number of subsequent measurements.

### 4.1. Possible measurements

The conflict recognition strategy (via ***P***(OBS, ENV)) identifies all predictions for each environment. The results of this analysis provides the basis for a differential diagnosis procedure, allowing GDE to identify possible measurements and their consequences.

Consider how measuring quantity $x_i$ could reduce the candidate space. GDE's database (e.g., (1)) explicitly represents $x_i$'s values and their supporting environments:

$$[\![x_i = v_{ik}, e_{ik1}, \ldots, e_{ikm}]\!]\,.$$

If $x_i$ is measured to be $v_{ik}$, then the supporting environments of any value distinct from the measurement are necessarily conflicts. If $v_{ik}$ is not equal to any of $x_i$'s predicted values, then every supporting environment for each predicted value of $x_i$ is a conflict. Given GDE's database, it is simple to identify useful measurements, their possible outcomes, and the conflicts resulting from each outcome. Furthermore, the resulting reduction of the candidate space is easily computed for each outcome.

Consider the example of the previous section. $X = 4$ in environments $\{A_1, M_2\}$ and $\{A_1, A_2, M_3\}$, while $X = 6$ in environment $\{M_1\}$. Measuring $X$ has three possible outcomes: (1) $X = 4$ in which case $\langle M_1 \rangle$ is a conflict and the new minimal candidate is $[M_1]$, (2) $X = 6$ in which case $\langle A_1, M_2 \rangle$ and $\langle A_1, A_2, M_3 \rangle$ are conflicts and the new minimal candidates are $[A_1]$, $[M_2, M_3]$ and $[A_2, M_2]$, or (3) $X \neq 4$ and $X \neq 6$ in which case $\langle M_1 \rangle$, $\langle A_1, M_2 \rangle$ and $\langle A_1, A_2, M_3 \rangle$ are conflicts and $[A_1, M_1]$, $[M_1, M_2, M_3]$, and $[A_2, M_1, M_2]$ are minimal candidates.

The minimal candidates are a computational convenience for representing the entire candidate set. For presentation purposes, in the following we dispense with the idea of minimal candidates and consider all candidates.

The diagnostic process described in the subsequent sections depends critically on manipulating three sets: (1) $R_{ik}$ is the set of (called remaining) candidates that would remain if $x_i$ were measured to be $v_{ik}$, (2) $S_{ik}$ is the set of (called selected) candidates in which $x_i$ *must* be $v_{ik}$ (equivalently, the candidates necessarily eliminated if $x_i$ is measured not to be $v_{ik}$), and (3) $U_i$ is the set of (called uncommitted) candidates which do not predict a value for $x_i$ (equivalently, the candidates which would not be eliminated independent of the value measured for $x_i$). This set $R_{ik}$ is covered by the sets $S_{ik}$ and $U_i$:

$$R_{ik} = S_{ik} \cup U_i , \qquad S_{ik} \cap U_i = \phi .$$

## 4.2. Lookahead versus myopic strategies

Section 4.1 describes how to evaluate the consequences of a hypothetical measurement. By cascading this procedure, we could evaluate the consequences of any sequence of measurements to determine the optimal next measurement (i.e., the one which is expected to eliminate the candidates in the shortest sequence of measurement). This can be implemented as a classic decision tree analysis, but the computational cost of this analysis is prohibitive. Instead we use a one-step lookahead strategy based on Shannon entropy [1, 20, 26]. Given a particular stage in the diagnostic process we analyze the consequences of each single measurement to determine which one to perform next. To accomplish this we need an evaluation function to determine for each possible outcome of a measurement how difficult it is (i.e., how many additional measurements are necessary) to identify the actual candidate. From decision and information theory we know that a very good cost function is the entropy ($H$) of the candidate probabilities:

$$H = -\sum p_i \log p_i ,$$

where $p_i$ is the probability that candidate $C_i$ is the actual candidate given the hypothesized measurement outcome.

Entropy has several important properties (see a reference on information theory [30] for a more rigorous account). If every candidate is equally likely, we have little information to provide discrimination—$H$ is at a maximum. As one candidate becomes much more likely than the rest $H$ approaches a minimum. $H$ estimates the expected cost of identifying the actual candidate as follows. The cost of locating a candidate of probability $p_i$ is proportional to $\log p_i^{-1}$ (cf. binary search through $p_i^{-1}$ objects). The expected cost of identifying the actual candidate is thus proportional to the sum of the product of the probability of each candidate being the actual candidate and the cost of identifying that candidate i.e., $\Sigma\, p_i \log p_i^{-1} = -\Sigma\, p_i \log p_i$. Unlikely candidates, although expensive to find, occur infrequently so they contribute little to the cost: $p_i \log p_i^{-1}$ approaches 0 as $p_i$ approaches 0. Conversely, likely candidates, although they occur frequently, are easy to find so contribute little to the cost: $p_i \log p_i^{-1}$ approaches 0 as $p_i$ approaches 1. Locating candidates in between these two extremes is more costly because they occur with significant frequency and the cost of finding them is significant.

## 4.3. Minimum entropy

Under the assumption that every measurement is of equal cost, the objective of diagnosis is to identify the actual candidate in a minimum number of measurements. This section shows how the entropy cost function presented in the previous section is utilized to choose the best next measurement. As the diagnosis process is sequential, these formulas describe the changes in quantities as a consequence of making a single measurement.

The best measurement is the one which minimizes the expected entropy of candidate probabilities resulting from the measurement. Assuming that the process of taking a measurement doesn't influence the value measured, the expected entropy $H_e(x_i)$ after measuring quantity $x_i$ is given by:

$$H_e(x_i) = \sum_{k=1}^{m} p(x_i = v_{ik}) H(x_i = v_{ik}) .$$

Where $v_{i1}, \ldots, v_{im}$ are all possible values[6] for $x_i$, and $H(x_i = v_{ik})$ is the

[6] These results are easily generalized to account for an infinite number of possible values since, although a quantity may take on an infinite number of possible values, only a finite number of these will be predicted as the consequences of other quantities measured. Further the entropy resulting from the measurement of a value not predicted is independent of that value. Thus the system never has to deal with more than a finite set of expected entropies.

entropy resulting if $x_i$ is measured to be $v_{ik}$. $H(x_i = v_{ik})$ can be computed from the information available.

At each step, we compute $H(x_i = v_{ik})$ by determining the new candidate probabilities, $p'_l$ from the current probabilities $p_l$ and the hypothesized result $x_i = v_{ik}$. The initial probabilities are computed from empirical data (see Section 4.4). When $x_i$ is measured to be $v_{ik}$, the probabilities of the candidates shift. Some candidates will be eliminated, reducing their posterior probability to zero. The remaining candidates $R_{ik}$ shift their probabilities according to (see Section 4.5):

$$p'_l = \begin{cases} \dfrac{p_l}{p(x_i = v_{ik})}, & l \in S_{ik}, \\[2ex] \dfrac{p_l/m}{p(x_i = v_{ik})}, & l \in U_i. \end{cases}$$

If every candidate predicts a value for $x_i$, then $p(x_i = v_{ik})$ is the combined probabilities of all the candidates predicting $x_i = v_{ik}$. To the extent that $U_i$ is not empty, the probability $p(x_i = v_{ik})$ can only be approximated with error $\varepsilon_{ik}$:

$$p(x_i = v_{ik}) = p(S_{ik}) + \varepsilon_{ik}, \quad 0 < \varepsilon_{ik} < p(U_i),$$

$$\sum_{k=1}^{m} \varepsilon_{ik} = p(U_i),$$

where,

$$p(S_{ik}) = \sum_{C_j \in S_{ik}} p_j, \qquad p(U_i) = \sum_{C_j \in U_i} p_j.$$

At any stage of the diagnostic process only some (say the first $n$ of the $m$ possible) of the $v_{ik}$ are actually predicted (i.e., those with nonempty $S_{ik}$) for $x_i$. If a candidate does not predict a value for a particular $x_i$, we assume each possible $v_{ik}$ is equally likely[7]:

$$\varepsilon_{ik} = p(U_i)/m.$$

So,

$$p(x_i = v_{ik}) = p(S_{ik}) + p(U_i)/m.$$

Notice that for unpredicted values $S_{ik}$ is empty, so $p(x_i = v_{ik}) = p(U_i)/m$.

[7] We could assume that if a component were faulted (i.e., a member of the actual candidate), then its current observed inputs and outputs would be inconsistent with its model. Under such an assumption, the distribution would be skewed away from those $v_{ik}$ predicted from the set of assumptions of the candidate (i.e., viewing a candidate as an environment). We do not make this assumption because a component may appear to be functioning correctly, but actually be faulted producing incorrect outputs for a different set of inputs.

The expected entropy can be computed from predicted quantities:

$$H_{\mathrm{e}}(x_i) = \sum_{k=1}^{n} (p(x_i = v_{ik}) + \varepsilon_{ik})H(x_i = v_{ik}) + \sum_{k=n+1}^{m} \varepsilon_{ik} H_{U_i} ,$$

where $H_{U_i}$ is the expected entropy if $x_i$ is measured to have an unpredicted value (i.e., all but the candidates $U_i$ are eliminated):

$$H_{U_i} = - \sum_{C_l \in U_i} p'_l \log p'_l .$$

$H_{U_i}$ and $\varepsilon_{ik}$ are independent of the unpredicted value measured, thus rewriting we obtain:

$$H_{\mathrm{e}}(x_i) = \sum_{k=1}^{n} (p(x_i = v_{ik}) + \varepsilon_{ik})H(x_i = v_{ik}) + \frac{(m-n)}{m} p(U_i) H_{U_i} .$$

Substituting and simplifying gives:

$$H_{\mathrm{e}}(x_i) = H + \Delta H_{\mathrm{e}}(x_i) .$$

Where $H$ is the current entropy, and $\Delta H_{\mathrm{e}}(x_i)$ is:

$$\sum_{k=1}^{n} p(x_i = v_{ik}) \log p(x_i = v_{ik}) + p(U_i) \log p(U_i) - \frac{np(U_i)}{m} \log \frac{p(U_i)}{m} .$$

The expected entropy can be calculated from the current candidate probability distribution—there is no necessity to explicitly construct the possible posterior probability distributions and compute their entropies. Thus the best, on average, measurement is the one that minimizes $\Delta H_{\mathrm{e}}(x_i)$.

The choice of base for the logarithm does not affect the relative order of costs. Purely for convenience GDE computes base e (this corresponds to measurements, on average, having e known outcomes). To obtain a positive cost, GDE adds one to this equation. This cost indicates the quality of a hypothesized measurement. The cost is the expected increase in total (i.e., in the entire diagnostic session) number of measurements that need to be made to identify the candidate after making the measurement. A cost of 1 indicates that no information at all is obtained. A cost of 0 is ideal as it indicates perfect information gain.

### 4.4. Independence of faults

The initial probabilities of candidates are computed from the initial probabilities of component failure (obtained from their manufacturer or by observation). We make the assumption that components fail independently. (This approach could be extended to dependent faults except that voluminous data is required.) The initial probability that a particular candidate $C_i$ is the actual candidate $C_\mathrm{a}$ is given by:

$$p_i = \prod_{c \in C_i} p(c \in C_\mathrm{a}) \prod_{c \notin C_i} (1 - p(c \in C_\mathrm{a})) .$$

### 4.5. The conditional probability of a candidate

Given measurement outcome $x_i = v_{ik}$, the probability of a candidate is computed via Bayes' rule (see Section 6.6.):

$$p(C_l | x_i = v_{ik}) = \frac{p(x_i = v_{ik} | C_l) p(C_l)}{p(x_i = v_{ik})} .$$

Where $p_l = p(C)$ where $p_l' = p(C_l | x_i = v_{ik})$. There are three cases for evaluating $p(x_i = v_{ik} | C_l)$. If $C_l$ predicts $x_i = w_{ik}$ where $w_{ik} \neq v_{ik}$, then the conditional probability is 0:

$$p(x_i = v_{ik} | C_l) = 0 , \quad \text{if } C_l \notin R_{ik} .$$

If $w_{ik} = v_{ik}$, then the conditional probability is 1:

$$p(x_i = v_{ik} | C_l) = 1 , \quad \text{if } C_l \in S_{ik} .$$

In the third case, $C_l$ predicts no value for $x_i$. We assume that every possible value for $x_i$ (there are $m$ of them) is equally likely:

$$p(x_i = v_{ik} | C_l) = 1/m , \quad \text{if } C_l \in U_i .$$

Substituting these probabilities into Bayes' rule we obtain:

$$P(C_l | x_i = v_{ik}) = \begin{cases} 0, & \text{if } C_l \notin R_{ik} , \\ \dfrac{p(C_l)}{p(x_i = v_{ik})} , & \text{if } C_l \in S_{ik} , \\ \dfrac{p(C_l)7m}{p(x_i = v_{ik})} , & \text{if } C_l \in U_i , \end{cases}$$

where

$$p(x_i = v_{ik}) = p(S_{ik}) + p(U_i)/m .$$

The estimate $p(x_i = v_{ik}|C_l) = 1/m$ is suspect and introduces various computational complexities. Fortunately, $U_i$ contains primarily low probability candidates and thus any error tends to be minor. On average, the candidates in $U_i$ are nonminimal (because minimal candidates tend to assign values to most of the device's variables) and as nonminimal candidates have lower probability than minimal candidates, the candidates in $U_i$ have relatively lower probability. In practice, using $P(x_i = v_{ik}|C_l) = 1$ for $C_l \in R_{ik}$ greatly simplifies the computations and rarely affects the number of measurements required.

### 4.6. Examples

This section presents a series of examples to illustrate some of the intuitions behind our technique.

4.6.1. *Cascaded inverters*: $a = 1$

Consider circuit of Fig. 5. Assume the initial probability of component failure is 0.01 and input $a = 1$ is given. As initially no symptom has been detected on average, little is gained by making any measurement—all costs are nearly 1. Compare the advantages and disadvantages of measuring $b$ or $e$. If the value measured differs from the value predicted, then measuring closer to the input (or any previously measured value) produces as a smaller conflict, and thus more information. For example, if we measure $b$ to be 1, then we learn that $A$ must be faulted, where measuring $e = 0$ tell us only that some component is faulty. On the other hand, measuring further away from the input is more likely to produce a discrepant value. That is, as there is a large number of intervening components, it is more likely that one will be faulted and thus produce an incorrect value. These two effects tend to cancel each other, however, the probability of finding a particular value outweighs the expected cost of isolating the candidate from a set. In terms of entropy, every possible outcome contributes $p_i \log p_i^{-1}$ but $p_i$ dominates $\log p_i^{-1}$. In general for any cascaded sequence of components for which only the input is known, the best next measurement is the output of the sequence. This is illustrated in Table 1.

4.6.2. *Cascaded inverters*: $a = 1, e = 0$

Suppose $e$ is measured to be 0, violating the prediction and thus telling us that at least one of the components is faulted. The best next measurement is the one that is equidistant from the previous two measurements, $c$ (Table 2).

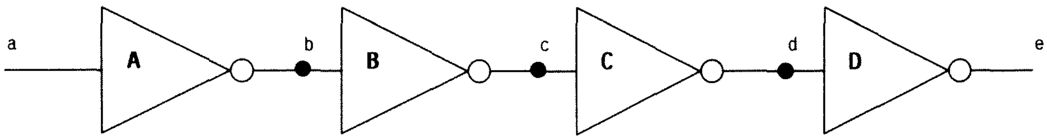

Fig. 5. Cascaded inverters.

TABLE 1. Expected costs after $a = 1$ with $p = 0.01$

| | |
|---|---|
| $a$ | 1 |
| $b$ | 0.98 |
| $c$ | 0.96 |
| $d$ | 0.94 |
| $e$ | 0.93 |

TABLE 2. Expected costs after $a = 1$, $e = 0$ with $p = 0.01$

| | |
|---|---|
| $a$ | 1 |
| $b$ | 0.45 |
| $c$ | 0.31 |
| $d$ | 0.44 |
| $e$ | 1 |

TABLE 3. Expected costs after $a = 1$, $e = 1$ with $p = 0.01$

| | |
|---|---|
| $a$ | 1 |
| $b$ | 0.999 |
| $c$ | 0.999 |
| $d$ | 0.999 |
| $e$ | 1 |

### 4.6.3. *Cascaded inverters*: $a = 1$, $e = 1$

Suppose that instead of measuring $e$ to be 0 it was measured 1 as predicted. The circuit must have a relatively unlikely double fault (if it has any fault at all) where one fault shadows the effect of the other. Nevertheless the best measurement to identify a possible double fault is still $c$ (Table 3).

### 4.6.4. *Cascaded inverters*: $a = 1$, $e = 0$, $p(a) = 0.025$

If we are given that the first inverter $A$ is more likely to be faulted, then the best measurement point is no longer equidistant from the previous observations. The advantage of measuring $b$ first instead of $c$ is that we might identify that inverter $A$ is faulted immediately. As $A$ is far more likely to be faulted, it is better to measure $b$ first (Table 4).

### 4.6.5. *Diagnosing a fault*

Consider the circuit of Fig. 2. Suppose that $M_2$ is faulted exhibiting behavior $Y = B \times C - 2$ and $A_2$ is faulted with behavior $G = Y \times Z + 2$. The inputs are the same as earlier. The following illustrates GDE's strategy to localize this fault. Note that this is an unusual fault in that the effect of the two faults cancel, producing the value predicted for $G$. Thus, several measurements are necessary to identify the fault. We assume that components fail with initial probability 0.01 and that $m = 16$.

Initially the most probable candidate is $[\;]$, $p([\;] = C_{\mathrm{a}}) = 0.951$. The expected cost associated with each hypothetical measurement is shown in Table 5. All the costs are nearly one, indicating that little is to be gained by any of the measurements, this is because on average components are not faulted, and there is yet no evidence that anything is malfunctioning. Not surprisingly, $F$ and $G$ as well as $X$, $Y$, $Z$ are treated symmetrically. $F$ and $G$ have slightly lower cost because their predicted values depend on three components functioning correctly, while $X$, $Y$, and $Z$ depend on two (i.e., $F$ and $G$ are more likely to have discrepant values).

Suppose $F$ is measured with result $F = 10$ (Table 6). The most probable candidates are now $[A_1]$, $[M_1]$, and $[M_2]$, all having probability 0.323. $G$, $X$ and $Y$ are all good measurements because each one differentiates among the

TABLE 4. Expected costs after $a = 1$, $e = 0$ with $p(a) = 0.025$

| | |
|---|---|
| $a$ | 1 |
| $b$ | 0.33 |
| $c$ | 0.36 |
| $d$ | 0.53 |
| $e$ | 1 |

TABLE 5. Initial expected costs

| | |
|---|---|
| $F$ | 0.88 |
| $G$ | 0.88 |
| $X$ | 0.95 |
| $Y$ | 0.95 |
| $Z$ | 0.95 |

TABLE 6. Costs after $F = 10$

| | |
|---|---|
| $F$ | 1 |
| $G$ | 0.28 |
| $X$ | 0.34 |
| $Y$ | 0.34 |
| $Z$ | 0.95 |

high-probability single-fault candidates. Each high-probability candidate predicts $Z = 6$ so measuring $Z$ provides little new information. $G$ is a slightly better point to measure because the candidates are more balanced between $G$'s two predicted outcomes (the best measurement is one whose predicted outcomes all have equal probability and which cover all the candidates).

Next, suppose $G$ is measured with result $G = 12$ (Table 7). The most probable candidates are now $[A_1]$ and $[M_1]$ both with probability 0.478. At this point $X$ is the best measurement because it splits the two high-probability candidates.

Next, suppose $X$ is measured with result $X = 6$ (Table 8). This results in a single high-probability candidate $[A_1]$ with probability 0.942. The next seven most likely candidates are: $[A_1, M_3]$, $[A_1, M_2]$, $[A_1, M_1]$, $[A_1, A_2]$, $[A_2, M_2]$, and $[M_2, M_3]$, all with probability 0.00951.

Next, suppose $Y$ is measured with result $Y = 4$. $M_2$ is now necessarily faulted and at least one other fault exists. The costs are now as shown in Table 9.

Finally, suppose $Z$ is measured with result $Z = 6$. There are six remaining candidates: $[A_2, M_2]$ with probability 0.970, $[A_2, M_1, M_2]$, $[A_1, A_2, M_2]$, $[A_2, M_2, M_3]$, with probabilities 0.0098, $[A_1, A_2, M_1, M_2]$, $[A_1, A_2, M_2, M_3]$, $[A_2, M_1, M_2, M_3]$, with probabilities 0.0001, and $[A_1, A_2, M_1, M_2, M_3]$ with probability 0.000001. Components $M_2$ and $A_2$ are necessarily faulted, and $A_1$, $M_1$, and $M_3$ are possibly faulted, each with probability 0.01. No measurement points remain in the circuit, so no further information can be obtained.

TABLE 7. Costs after $F = 10$, $G = 12$

| | |
|---|---|
| $F$ | 1 |
| $G$ | 1 |
| $X$ | 0.28 |
| $Y$ | 0.94 |
| $Z$ | 0.97 |

TABLE 8. Costs after $F = 10$, $G = 12$, and $X = 6$

| | |
|---|---|
| $F$ | 1 |
| $G$ | 1 |
| $X$ | 1 |
| $Y$ | 0.90 |
| $Z$ | 0.94 |

TABLE 9. Costs after $F = 10$, $G = 12$, $X = 6$, and $Y = 4$

| | |
|---|---|
| $F$ | 1 |
| $G$ | 1 |
| $X$ | 1 |
| $Y$ | 1 |
| $Z$ | 0.141 |

### 4.7. Logical incompleteness

In practice, for diverse reasons, the underlying inference process is usually incomplete. One of the consequences of this incompleteness is that it becomes more difficult to evaluate the results of hypothetical measurements—the $U_i$ are larger than ideal. As any incompleteness degrades GDE's performance, it is instructive to examine the sources and types of this incompleteness.

An often avoidable form of incompleteness occurs when the conflict recognition strategy misses some conflicts. This was discussed earlier. Here we assume the conflict recognition strategy is complete. $U_i$ will be larger than ideal only if the prediction process is incomplete. This incompleteness manifests itself in two ways. First, an incomplete inference process may result in missing predicted values and missing supporting environments. As a consequence, the hypothesized conflicts resulting from measuring a quantity will be incomplete. For example, consider an inverter which is incompletely modeled by a rule which predicts its output from its input, but not its input from its output. Even though the inverter's output is measured to be one, the prediction that its input is zero is not made. Thus, GDE does not consider measuring its input. A second source of incompleteness is inherent in the $x_i = v_{ik}$ representation—there may be many additional properties that could be inferred about $x_i$, but as there is no way to represent them they cannot be used by our strategy. For example, it cannot represent $x \neq 1$. Thus, although $x \neq 1$ might be derivable from an environment, GDE cannot foresee the resulting conflict when considering, say, $x = 2$.

Assuming the basic conflict recognition strategy is complete, both these sources of incompleteness can be avoided at prohibitive computational costs. The first source of incompleteness can be avoided with a complete inference process. The second source of incompleteness can be avoided with a more general representation.[8]

## 5. Pragmatics

### 5.1. Most probable candidates

Computing all candidates is computationally prohibitive. In practice it is only necessary to compute the more probable candidates. There is no way to tell whether a single candidate *actually* has a high probability without knowing the overall normalization factor. This suggests using a best-first search of the lattice to find candidates in increasing order of probability. This search is arbitrarily

[8] Assuming the number of possible values for quantities are finite, GDE could hypothesize each possible measurement outcome, run its complete inference procedure and precisely compute the $R_{ik}$ from which the $S_{ik}$ could then be computed.

stopped for candidates below some threshold fraction (e.g., $\frac{1}{10}$ of the highest probability candidate). Although the most probable candidate is a minimal candidate, the remaining minimal candidates need not be very probable.

### 5.2. When to stop making measurements

If there are many points in a device that could be measured choosing the best point to measure can be computationally expensive. A heuristic is to make the first reasonable measurement whose cost is computed to be less than $1 - \log 0.5 = 0.7$ as this measurement on average, splits the candidate space in half.

The point at which measurement should stop depends greatly on the seriousness of a misdiagnosis as well as on the a priori probabilities of component failure. When a candidate is found whose probability approaches some threshold (e.g., 0.9), diagnosis can stop. If the cost of misdiagnosis is high, then the threshold should be increased.

## 6. Comparison to Other Work

### 6.1. Circuit diagnosis based on structure and behavior

Work in the AI community on model-based hardware diagnosis has grown out of a desire to move away from the domain and device specific fault models used in traditional circuit diagnosis.[9] Instead, during candidate generation the model-based approach reasons from a small set of component behavior models plus the structure of the device. The requirement for device specific fault models is eliminated by basing the diagnostic approach solely on the knowledge that, if a component's behavior is inconsistent with its model, then it must be faulty. This results in a domain independent diagnostic technique. A number of systems have followed this approach for diagnosing both analog (e.g., LOCAL [10] and SOPHIE [4]) and digital circuits (e.g., HT [5] and DART [12]).

Our work naturally extends this approach along a number of dimensions. First, unlike earlier approaches, our work is aimed specifically at coping with the problem of diagnosing multiple faults. Earlier work focused primarily on the case where all symptoms could be explained by a single component being faulty. As Davis points out, the obvious extension to his work to handle multiple faults results in an algorithm which is exponential in the number of potential faults. As our approach represents the candidate space implicitly in terms of the minimal candidates, we need only be concerned with the growth of this smaller set. Typically the size of each minimal candidate is relatively small (i.e., the symptom can usually be explained by one, two or three

[9] See Davis' [5], Sections 5.1., 12.1. and 13 for a discussion of traditional circuit diagnosis and the advantages of the model-based approach. Also see [16].

components being simultaneously faulty), thus in practice our approach tends to grow with the square or cube of the number of potential faults.

Second, our approach to diagnosis is inference procedure independent, as well as domain independent. LOCAL, SOPHIE, and Davis' system all represent circuit knowledge as constraints and then use some form of constraint propagation to infer circuit quantities and their dependencies. The motivation behind this approach is that constraints naturally reflect the local interaction of behavior in the real world. These are similar (although not identical) to the constraint propagation technique used to demonstrate GDE in this paper.

On the other hand, DART expresses circuit knowledge as logical propositions and uses an inference system based on resolution residue to infer circuit properties. This same general inference procedure allows DART to deduce which components are suspect and how to change circuit inputs to reduce the suspect set to a singleton. Such a resolution-based inference strategy can be incorporated into the candidate generation and conflict recognition portion of GDE by recording, for each resolution step, the dependence of the resolvent on the formulas used to perform the resolution. Then, as in the case for constraint propagation, GDE can guide the inference process in identifying minimal candidates first.

Resolution is highlighted as having an advantage over constraint propagation in that it is logically complete with respect to a first-order theory, where most constraint propagators are not. However, this can be misleading. A significant computational cost is incurred using a logically complete inference strategy, such as resolution. Yet a logically complete inference strategy does not guarantee that the set of predictions will also be complete. For example in an analog domain, producing exact predictions often involves solving systems of higher-order nonlinear differential equations. As this type of equation is not generally solvable with known techniques, completeness in the predictor is currently beyond our reach. In practice propagation of constraints (without symbolic algebra) provides a good compromise between completeness and computational expense.

Third, our approach is incremental. This is crucial as diagnosis is an iterative process of making observations, refining hypotheses, and then using this new knowledge to guide further observations.

Finally, our approach is unique in that it combines model-based reasoning with probabilistic information in a sequential probing strategy. SOPHIE also proposes measurements to localize the failing component, but is based on an ad hoc half-split method. Its design is based on the presupposition that the circuit contains a single fault. Thus, candidate generation is trivial (set intersection), and identifying good measurements is easy.

### 6.1.1. *Possible extensions*

Representing and manipulating candidates in terms of minimal candidates gives us a significant computational advantage over previous approaches; however, even this representation can grow exponentially in the worst case. To effectively cope with very large devices, additional techniques must be incorporated.

One approach involves grouping components into larger modules and modifying the candidate generation strategy to deal with hierarchical decompositions. Although we have not implemented this, the strategies proposed by Davis and Genesereth, as they are model-based, apply. Their basic idea is to troubleshoot at the most abstract level in the hierarchy first and only analyze the contents of a module when there is reasonable evidence to suspect it. If there is no conflict involving the module, then there is no reason to suspect it.

Complexity is directly related to the number of hypothetical faults entertained. Thus another approach, proposed by Davis, involves enumerating and layering categories of failures based on their likelihood. Troubleshooting begins by considering faults in the most likely category first, moving to less likely categories if these fail to explain the symptoms. GDE uses a variation on this approach whereby candidates can be enumerated based on their likelihood. However, no theory has been developed about failure categories.

LOCAL and SOPHIE also address a number of issues which GDE does not. LOCAL and SOPHIE use corroborations as well as conflicts to eliminate candidates. To take advantage of corroborations, SOPHIE includes an *exhaustive* set of fault modes for each component. As a consequence, SOPHIE can identify a component as being unfaulted by determining that it is not operating in any of its fault modes (i.e., none of the components fault modes can explain the failure, and we know all the component's modes of failure, so the component must be working). Thus, SOPHIE reduces the candidate space from both sides: conflicts eliminate candidates that do not explain a conflict, and corroborations eliminate candidates which include known working components.

In addition, SOPHIE and LOCAL dealt with the problem of imprecise models and values as well. This makes constraint propagation very difficult because it is hard to tell whether two differing values corroborate or conflict within the precision of the analysis.

GDE proposes optimal measurements given a fixed set of inputs. Both Shirley's system [31] and DART generate test vectors to localize circuit faults. This approach is useful in cases where it is both difficult to make internal measurements, and easy to change circuit inputs. Both approaches produce tests which are likely to give useful information; however, no serious attempt is made at selecting the optimal tests. A fairly simple extension can be made to our cost function to account for tests, by taking into consideration, for example the cost of modifying the inputs.

In Shirley's approach, each test isolates a single potentially faulted component by using known good components to guide the input signals to the faulty component and the output of that component to a point which can be measured. However, it is not always possible to route the test signals such that all other suspects are avoided. Using the information constructed by GDE, this approach can be modified, so that tests are constructed first for components which are most likely to be faulted, while propagating the signal along a path with the lowest probability of having a faulty component.

### 6.2. Counterfactuals

Ginsberg [13] points out how David Lewis' possible world semantics could be applied to diagnosis. Each component is modeled by a counterfactual, a statement such as "if $p$, then $q$," where $p$ is expected to be false. Thus $A_1$ is modeled by the counterfactual "if $A_1$ fails, then $F = X + Y$ need not hold." Counterfactuals are evaluated by considering a "possible world" that is as similar as possible to the real world where $p$ is true. The counterfactual is true if $q$ holds in such a possible world. In this framework the most similar worlds (i.e., those closest to the one in which the circuit functions correctly) correspond directly to our minimal candidates. A most similar world is one in which as few of the $p$'s (e.g., "$A_1$ fails") as possible become true. Thus, in this instance Ginsberg uses counterfactuals for hypotheticals.

Ginsberg's approaches handles multiple faults, but he does not discuss measurement strategies or probabilities. Both could be integrated into his approach.

### 6.3. Reiter

Reiter [29] has been independently exploring many of the ideas incorporated in GDE. His theory of diagnosis provides a formal account of our "intuitive" techniques for conflict recognition and candidate generation. However, his theory does not include a theory of measurements nor how to exploit probabilistic information.

Reiter's theory uses McCarthy's [18] AB predicate. Reiter writes $\neg \mathrm{AB}(A_1)$ (i.e., adder $A_1$ is not ABnormal), while we write $A_1$ (i.e., the assumption that adder $A_1$ is working correctly). Under this mapping, Reiter's definition of diagnosis is equivalent to our definition of minimal candidate, and his definition of conflict set is equivalent to our definition of conflict.

Reiter proposes (unimplemented) a diagnostic algorithm based on his theory. This architecture is quite different from GDE's. The theorem prover of his architecture corresponds roughly to our inference engine. While GDE's procedure first computes all minimal conflicts resulting from a new measurement before updating the candidate space, Reiter's algorithm intermingles conflict recognition with candidate generation to guide the theorem proving to "prevent the computation of inessential variants of refutations, without imposing any constraints on the nature of the underlying theorem proving system." Given GDE's inference architecture (and the necessity to choose the best next measurement), GDE avoids almost all these inessential variants and also avoids much of the computation Reiter's architecture demands.

### 6.4. SNePS

SNePS [17] incorporates a belief revision system SNeBR which reasons about the consistency of hypotheses. This belief revision system has been [6] applied to fault detection in circuits. Some of the basic concepts used in the SNePS approach are similar to those used in GDE—both use an assumption-based belief revision system (see [9] for a comparison). As described in [6] SNePS is weaker than either GDE or Reiter's approach. In GDE's terminology, SNePS operates in a single environment, detecting symptoms (and hence conflicts) for only those variables about which a user query is made. The system provides no mechanisms to detect all symptoms (and thus all conflicts), to identify candidates, to propose measurements, or to exploit probabilistic information. However, there seems no reason, in principle that the methods of GDE cannot be incorporated into their system.

### 6.5. IN-ATE and FIS

IN-ATE [7] and FIS [24] are expert system shells specifically designed for fault diagnosis. Although neither is purely model-based (i.e., not purely within the structure-function paradigm), they are the only other electronic diagnosis systems we are aware of that propose measurements, and thus it is instructive to compare them with GDE.

IN-ATE incorporates two formal criteria to determine the best next measurement: minimum entropy (as in GDE) and gamma miniaverage heuristic search [32]. We chose minimum entropy (as FIS does) in GDE because it was the best evaluation function which is efficiently computable (i.e., it is based on one-step lookahead). The gamma miniaverage heuristic search is a multi-step lookahead procedure and hence computationally expensive to apply. Nevertheless, there is no reason, in principle, it could not be incorporated into GDE.

Unlike GDE, IN-ATE and FIS can exploit a variety of kinds of knowledge (e.g., fault trees, expert-supplied rules, hierarchy) in the diagnostic process. However, viewed from the pure model-based approach, they are limited in predictive power and as a consequence can only estimate the probabilities for their predictions.

As IN-ATE is not model-based, it does not construct explicit value predictions as does GDE. Instead, it predicts whether a particular measurement outcome will be "good" or "bad." (This, in itself, presents difficulties because a measurement being "good" or "bad" is relative to the observations being compared to. However, IN-ATE takes "good" and "bad" as absolute.) These predictions are computed by analyzing expert-supplied empirical associations based on "good" and "bad." IN-ATE incorporates a heuristic rule generator which exploits connectivity information, but it, at a minimum, must be provided with the definitions of "good" and "bad" for the test points. In this approach, the number of test points is limited to those explicitly mentioned in rules (or for which "good" and "bad" are externally defined). With this limited set of test points full search (i.e., gamma miniaverage search) becomes plausible. FIS exploits more model knowledge, propagating qualitative values such as {hi, ok, lo} through the circuit topology via expert-supplied causal rules. The propagated values are always relative to fixed expert-supplied values. Thus neither FIS nor IN-ATE can analyze hypothetical faults.

All three systems must compute probabilities for their predictions. GDE directly computes the probability of the predictions from the probabilities of the candidates involved. IN-ATE has a much more difficult time determining good/bad probabilities because it cannot determine the candidate space and their associated probabilities. Instead it uses the expert-supplied rules to propagate probabilities (much like MYCIN) and Dempster–Shafer to combine the resulting evidence. FIS' definition of candidate probability is similar to GDE's, however, as it cannot perform hypothetical reasoning it can only estimate the probabilities of its predictions.

### 6.6. Medical diagnosis

There has been extensive research on the diagnostic task in the medical decision making community. Although most of this research is not model-based, its concern with identifying the diseases causing symptoms and subsequently proposing information-gathering queries is analogous to ours.

Like Gorry and Barnett [14] GDE solves a sequential diagnostic task. The fundamental equation (Bayes' rule) [35, formula (2)] states that the posterior probability of hypothesis $H_i$ being the true state $H_T$ given evidence $Q_i$ is:

$$p(H_j|Q_i) = \frac{p(Q_i|H_j)p(H_j)}{\sum_{k=1}^{n} p(Q_i|H_k)p(H_k)} ,$$

where $p(Q_i|H_j)$ is the conditional probability of evidence $Q_i$ while $p(H_j)$ is the a priori probability of $H_j$. Given these probabilities it is possible to determine the posterior probability distribution, and, in addition, to determine the hypothetical probability distribution resulting from proposed tests. Thus, this provides complete information to evaluate the expected gain of a test [14] and to select the best test to make next.

The practical obstacle to employing Bayes' rule is the unavailability of the conditional probabilities. These probabilities are both hard to estimate and extremely numerous. For even small numbers of possible symptoms and diseases the number of probabilities required is extremely large (given just 10 hypotheses and 5 possible binary tests, 2420 conditional probabilities are required [35]).

The engineering domain differs from the medical domain in that very accurate models exist based on the structure of the (faulty) system. Thus, exploiting model-based reasoning, GDE computes these conditional probabilities directly rather than depending on empirical results. Another advantage of the model-based approach for engineering is that the potential tests are computed in the course of the analysis instead of having them supplied a priori.

Early research [14] presumed that the patient had only one disease. This is analogous to the single-fault assumption in circuit diagnosis. More recent research allows for multiple simultaneous diseases. Bayes' rule can also be applied to this multimembership classification problem [2, 3] but the number of conditional probabilities required becomes exponentially larger than the single-disease case (which already is a large number). Again, GDE computes these conditional probabilities by model-based reasoning.

The concept of set covering used in [21, 22, 23, 25, 27, 28] is similar to the candidate generation phase of our diagnostic approach. The irredundant set cover of the general set covering theory (GSC) of Reggia corresponds directly to our notion of minimal candidates. The primary difference is that we consider, in theory, every possible cover to be an explanation for the symptoms while GSC considers only the minimal candidates as explanations. Although GSC incorporates a crude heuristic strategy for proposing new measurements, as the conditional probabilities are available to GDE, the preferred approach, minimum entropy, can be applied.

## ACKNOWLEDGMENT

Daniel G. Bobrow, Randy Davis, Kenneth Forbus, Matthew Ginsberg, Frank Halasz, Walter Hamscher, Tad Hogg, Ramesh Patil, Leah Ruby, Mark Shirley and Jeff Van Baalen, provided useful insights. We especially thank Ray Reiter for his clear perspective and many productive interactions. Lenore Johnson and Denise Pawson helped edit drafts and prepare figures.

## REFERENCES

1. Ben-Bassat, M., Myopic policies in sequential classification, *IEEE Trans. Comput.* **27** (1978) 170–178.
2. Ben-Bassat, M., Multimembership and multiperspective classification: Introduction, applications, and a Bayesian model, *IEEE Trans. Syst. Man Cybern.* **6** (1980) 331–336.
3. Ben-Bassat, M. Campbell, D.B., Macneil A.R. and Weil, M.H., Pattern-based interactive diagnosis of multiple disorders: The MEDAS system, *IEEE Trans. Pattern Anal. Mach. Intell.* **2** (1980) 148–160.
4. 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.
5. Davis, R., Diagnostic reasoning based on structure and behavior, *Artifical Intelligence* **24** (1984) 347–410.
6. Campbell, S.S. and Shapiro, S.C., Using belief revision to detect faults in circuits, SNeRG Tech. Note No. 15, Department of Computer Science, SUNY at Buffalo, NY, 1986.

7. Cantone, R.R., Lander, W.B., Marrone, M.P. and Gaynor, M.W., IN-ATE: Fault diagnosis as expert system guided search, in: L. Bolc and M.J. Coombs (Eds.), *Computer Expert Systems* (Springer, New York, 1986).
8. de Kleer, J., An assumption-based TMS, *Artifical Intelligence* **28** (1986) 127–162.
9. de Kleer, J., Problem solving with the ATMS, *Artifical Intelligence* **28** (1986) 197–224.
10. de Kleer, J., Local methods of localizing faults in electronic circuits, Artificial Intelligence Laboratory, AI Memo 394, MIT, Cambridge, MA, 1976.
11. Doyle, J., A truth maintenance system, *Artificial Intelligence* **12** (1979) 231–272.
12. Genesereth, M.R., The use of design descriptions in automated diagnosis, *Artificial Intelligence* **24** (1984) 411–436.
13. Ginsberg, M.L., Counterfactuals, *Artificial Intelligence* **30** (1986) 35–79.
14. Gorry, G.A. and Barnett, G.O., Experience with a model of sequential diagnosis, *Comput. Biomedical Res.* **1** (1968) 490–507.
15. Hamscher, W. and Davis, R., Diagnosing circuits with state: An inherently underconstrained problem, in: *Proceedings AAAI-84*, Austin, TX (1984) 142–147.
16. Hamscher, W. and Davis, R., Issues in diagnosis from first principles, Artificial Intelligence Laboratory, AI Memo 394, MIT, Cambridge, MA, 1986.
17. Martins, J.P. and Shapiro, S.C., Reasoning in multiple belief spaces, in: *Proceedings IJCAI-83*, Karlsruhe, F.R.G., 1983.
18. McCarthy, J., Applications of circumscription to formalizing common-sense knowledge, *Artificial Intelligence* **28** (1986) 89–116.
19. Mitchell, T., Version spaces: An approach to concept learning, Computer Science Department, STAN-CA-78-711, Stanford University, Palo Alto, CA, 1970.
20. Pearl, J., Entropy, information and rational decision, *Policy Anal. Inf. Syst.* **3** (1979) 93–109.
21. Peng, Y.P. and Reggia, J.A., Plausibility of diagnostic hypotheses: the nature of simplicity, in: *Proceedings AAAI-86*, Philadelphia, PA (1986) 140–145.
22. Peng, Y.P. and Reggia, J.A., A probabilistic causal model for diagnostic problem-solving; Part 1: Integrating symbolic causal inference with numeric probabilistic inference, Department of Computer Science, University of Maryland, College Park, MD, 1986.
23. Peng, Y.P. and Reggia, J.A., A probabilistic causal model for diagnostic problem-solving; Part 2: Diagnostic strategy, Department of Computer Science, University of Maryland, College Park, MD, 1986.
24. Pipitone, F., The FIS electronics troubleshooting system, *IEEE Computer* (1986) 68–75.
25. Pople, H., The formation of composite hypotheses in diagnostic problem solving: An exercise in synthetic reasoning, in: *Proceedings IJCAI-77*, Cambridge, MA (1977) 1030–1037.
26. Quinlan, J.R., Learning efficient classification procedures and their application to chess end games, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell, (Eds.), *Machine Learning* (Tioga, Palo Alto, CA, 1983) 463–482.
27. Reggia, J.A. and Nau, D.S., An abductive non-monotonic logic, in: *Workshop on Nonmonotonic Reasoning*, New Paltz, NY, 1984.
28. J.A. Reggia, Nau, D.S., and Wang, P.Y., Diagnostic expert systems based on a set covering model, *Int. J. Man-Mach. Stud.* **19** (1983) 437–460.
29. Reiter, R., A theory of diagnosis from first principles, *Artificial Intelligence* **32** (1987) 57–95 (this issue).
30. Shannon, C.E., A mathematical theory of communication, *Bell Syst. Tech. J.* **27** (1948) 379–623.
31. Shirley, M.H. and Davis R., Generating distinguishing tests based on hierarchical models and symptom information, in: *Proceedings IEEE International Conference on Computer Design*, Rye, NY, 1983.
32. Slagle, J.R. and Lee, R.C.T., Application of game tree searching techniques to sequential pattern recognition, *Commun. ACM* **14** (1971) 103–110.

33. Steele, G.L., The definition and implementation of a computer programming language based on constraints, AI Tech. Rept. 595, MIT, Cambridge, MA, 1979.

34. Sussman, G.J. and Steele, G.L., CONSTRAINTS: A language for expressing almost-hierarchical descriptions, *Artificial Intelligence* **14** (1980) 1–39.

35. Szolovitz, P. and Pauker, S.G., Categorical and probabilistic reasoning in medical diagnosis, *Artificial Intelligence* **11** (1978) 115–144.

36. Williams, B.C., Doing time: Putting qualitative reasoning on firmer ground, in: *Proceedings AAAI-86*, Philadelphia, PA (1986) 105–112.

*Received September 1985; revised version received July 1986*