N90-27280

Dynamic Test Input Generation for Multiple-Fault Isolation

Phil Schaefer
Martin Marietta Advanced Computing Technology
P.O. Box 179, M.S. 4372
Denver, CO 80201

Abstract

Recent work in Causal Reasoning has provided practical techniques for multiple fault diagnosis. These techniques provide for a hypothesis/measurement diagnosis cycle. Using probabilistic methods, they choose the best measurements to make, then update fault hypotheses in response.

For many applications such as computers and spacecraft, few measurement points may be accessible, or values may change quickly as the system under diagnosis operates. In these cases, a hypothesis/measurement cycle is insufficient. This paper presents a technique for a hypothesis/test-input/measurement diagnosis cycle. In contrast to generating tests a priori for determining device functionality, it dynamically generates tests in response to current knowledge about fault probabilities. The paper shows how the mathematics previously used for measurement specification can be applied to the test input generation process. An example from an efficient implementation called MPC is presented.

I. Introduction

In recent years, AI techniques have proven useful for constructing fault diagnosis tools. A particularly interesting subset of these techniques is based on Causal Reasoning. Causal reasoning tools use a model of how the unit should behave, assuming no faults. This model can then be used to infer possible faults by comparing the observed behavior to the behavior predicted by the model.

Recent work has yielded techniques for multiple fault diagnosis using such techniques. The approach of (de Kleer and Williams), for example, provides an efficient framework for hypothesizing faults, given a set of measurements. At each step in diagnosis, it determines the most helpful measurement to make next. This provides a hypothesis/measurement diagnosis cycle.

A fault-isolation procedure based solely on a hypothesis/measurement cycle is often insufficient. Many complex systems such as computers and spacecraft are packaged in a way that makes measurement of most internal points time-consuming and expensive. Additionally, values within the system may not be static. For example, in a microprocessor-based system, data is sent over the databus to peripheral devices. Without sophisticated test equipment, the actual databus value cannot be measured directly, because it is present for only a fraction of a microsecond. For these reasons, it is often preferable to restrict measurement, when possible, to a few easily-accessible points. In this style of diagnosis, multiple test inputs are generated to observe the system in multiple states, rather than multiple measurements taken with the system in a single state. Recently, researchers have introduced off-line techniques for purposes such as post-assembly checkout (Shirley). However, little has been presented about dynamically generating tests for isolating multiple faults.

The following sections present a technique for adapting the mathematics of hypothesis/measurement techniques for performing dynamic test input generation for multiple-fault isolation. Section II describes the causal models for which this technique applies. Section III presents the approach to the test generation and selection processes, and the techniques adopted for deducing the most probable faults. Finally, Section IV presents an example diagnosis using an implemented test-generation/fault-isolation system.

II. Causal Models for Multiple-Fault Diagnosis

Central to the causal-reasoning scheme is a causal model describing how the system under diagnosis properly functions. The causal model contains a description of the important points within the system, referred to as *elements,* and how the values of these elements cause and effect each other. Consider the system shown in Figure 1. It is a modified portion of an experiment control electronics subsystem concept being developed by Martin Marietta. This subsystem will be used as an example in the following sections. In the model, elements correspond to inputs and outputs of modules, as well as to a few module-internal points. Such a system can be modeled with our DEFCAUSAL syntax. For example, one operation of the Remote Interface Unit (RIU) is to write data to port EXP-CMD-3 when a :write-3

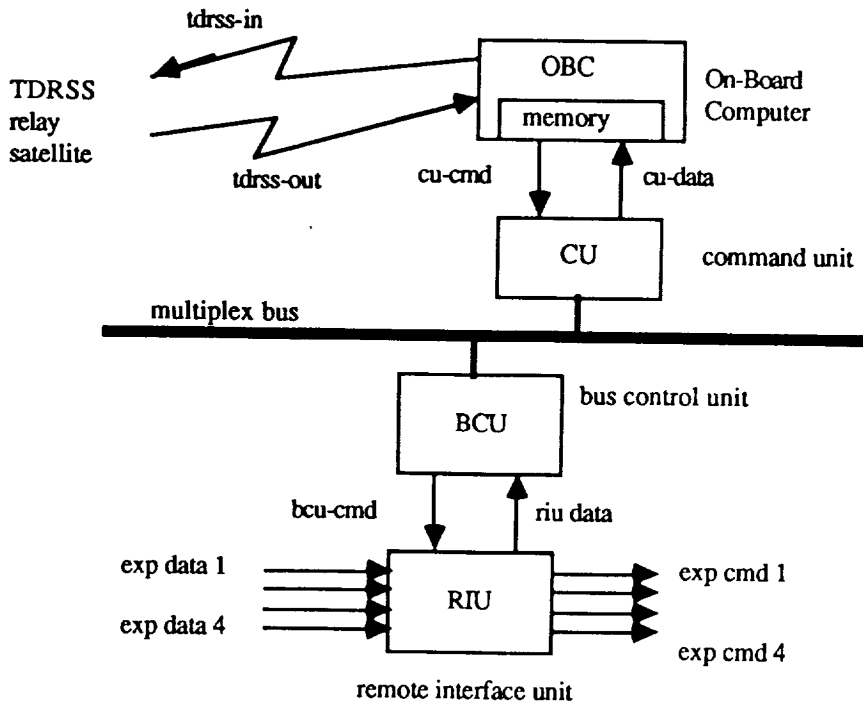

Figure 1. Example system for diagnosis Part of experiment control electronics subsystem. Commands sent and data gathered via the TDRSS relay satellite.

command is received on BCU-CMD. This relation is shown below (values starting with a "?" are variables).

```
(defcausal RIU
  (BCU-CMD (:write-3 ?data) :momentary) causes
  (EXP-CMD-3 ?data :continuing))
```

Another causal relation describes how the On-Board Computer (OBC), when in the :experiment mode, sends commands on the Command Unit (CU) line. The commands are sent from the OBC command-sequence memory:

```
(defcausal OBC
  (OBC_MEM ?memory)
  (MODE :experiment)
  (ELAPSED-TIME ?t) causes
  (CU-CMD (address ?t ?memory) :momentary)).
```

The :momentary and :continuing flags indicate the temporal relations of the causes and effects.

III. Dynamic Test Input Generation

An important part of diagnosis is to gain knowledge about the internal state of the malfunctioning system. This knowledge helps to decide which among several fault hypotheses is actually correct. In systems with many measurable internal points, this knowledge is gained by direct measurement, e.g., with voltmeters, logic probes, etc. When internal points are inaccessible, we must adopt a different means of obtaining this information. Our approach is based on a concept of *path generation*. In this approach, paths through the system are generated, which, if tested, will yield information about the internal system state. Our approach incorporates this idea with the following processes: (1) A Candidate Generator, which uses the measurements resulting from tests to produce fault hypotheses and their associated probabilities; (2) a Path Constructor, which suggests test paths through the model to gain information about the hypothesized faults; (3) a Path Selector, which chooses the path most helpful in discriminating between fault hypotheses; and (4) a Causal Planner, which produces a test input sequence to activate the selected path. Figure 2. presents a diagram of how these processes interact in a test-generation/fault-isolation system.

Candidate Generation

The Candidate Generator derives the fault hypotheses (candidates) implied by observed symptoms. It assigns probability estimates to each candidate.

Upon input of a symptom (an unexpected value at an observed element), the Causal Planner determines which causal relations imply the correct value, rather than the symptom value, was to be expected. This set is known as the *active relations set*. The Causal Planner produces this set as follows: given the correct outputs as goals and the inputs which were present when the symptom was observed

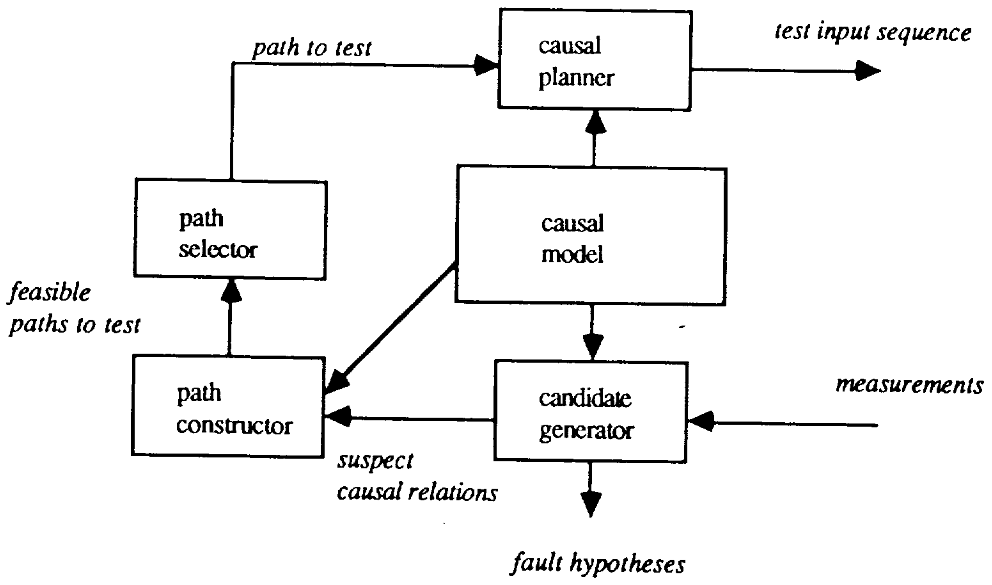

Figure 2. Test generation/ Diagnosis architecture.

as constraints on the solution, it generates a plan indicating the required inputs. During planning, the planner selects a causal relation to achieve each subgoal. When a solution is found, the set of relations selected in planning comprises the active relations set. This set can be described as a directed graph, relating input values, through intermediate values, to the desired output value. Assume, for example, that a value of 35 is desired on the TDRSS-IN line of the system of Figure 1. The resulting active relations set is shown in Figure 3. The active relations set describes the mechanisms which if functioning properly, would provide the correct behavior. When a failure symptom is observed, therefore, at least one of the active relations must not be functioning as specified. Conversely, if the correct output results, there is evidence that the active relations set is without fault.

One useful technique for generating candidates based on causal information has been described by (de Kleer and Williams). The technique maintains a set of *minimal candidates*, each of which must be able to account for all observed symptoms. Probability values are assigned to each candidate, using Bayesian probability concepts. Candidate generation, in our approach describing faults in terms of faulty active relations, is taken directly from (de Kleer and Williams), so will not be repeated here. A slight modification of the probability assignment approach is presented here.

A test *passes* if the value predicted by the model is observed at the test point. After each test is run, the probabilities of all candidates are updated, based on whether the test passes or fails. The probability that all of the relations in the candidate are faulty is assigned to each candidate. The probability of candidate Cj, with respect to its previous probability P(Cj) is, by Bayes' rule (if the test fails)

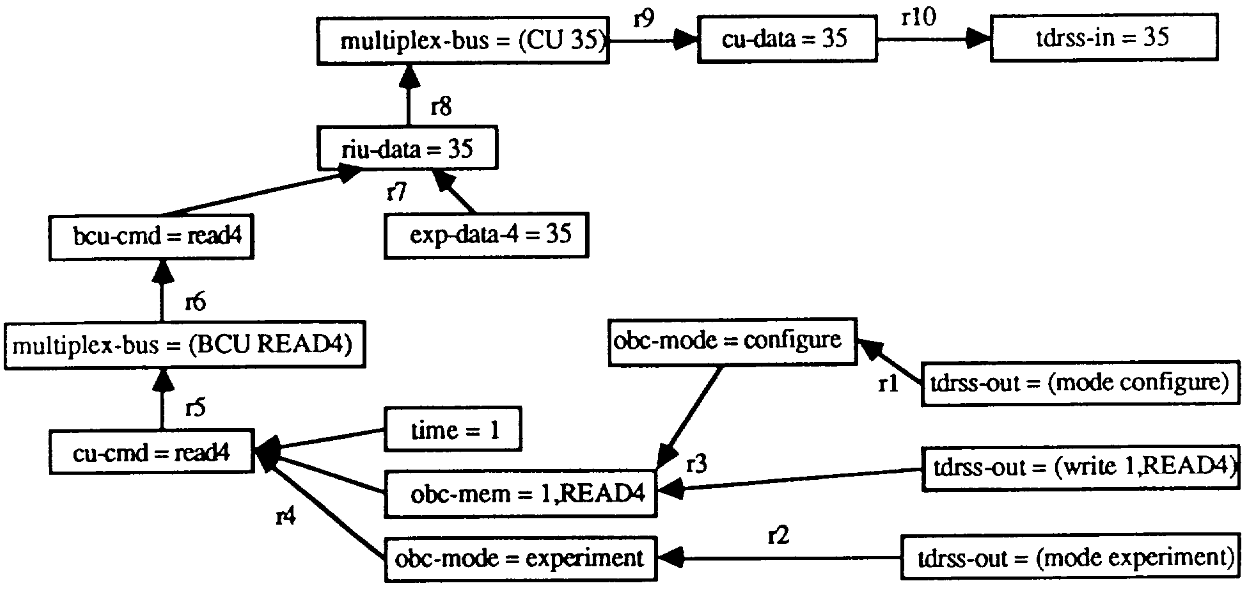

Figure 3. Active causal relations (ri) indicating why TDRSS-IN should equal 35. OBC memory is loaded with "read-4" in location 1. Then, in the "experiment" mode, when time=1, the OBC sequencer sends the command, which sends the result to TDRSS.

$$P(Cj|\text{ fails}) = \frac{P(\text{fails}|\ Cj)\ P(Cj)}{P(\text{fails})}$$

where

$$P(\text{fails}|\ Cj) = \begin{cases} 0 & \text{if } Cj \not\cap \text{active-relations} \\ 1-r & \text{if } Cj \cap \text{active-relations} \end{cases}$$

$$P(\text{fails}) = \sum_{j}^{Cj \cap \text{ active relations}} P(cj)(1-r)$$

and, if the test passes,

$$P(Cj|\text{ passes}) = \frac{P(\text{passes}|\ Cj)\ P(Cj)}{P(\text{passes})}$$

where

$$P(\text{passes}|\ Cj) = \begin{cases} 1 & \text{if } Cj \not\cap \text{active-relations} \\ r & \text{if } Cj \cap \text{active-relations} \end{cases}$$

$$P(\text{passes}) = \sum_{j}^{Cj \cap \text{ active relations}} P(Cj)\ r \quad + \sum_{j}^{Cj \not\cap \text{ active relations}} P(Cj)$$

As reflected above, even if a candidate is the fault, the correct value will in some cases result at the test point under the conditions of the test. This effect is approximated above with a constant, r, which indicates the probability that a particular relation will so behave.

Path Construction

Path construction is the basis of our test generation approach. The purpose of path construction is to generate a path using causal relations from a point internal to the system to a measurable point. By measuring the result at the output point, evidence about the internal state will be obtained.

When generating a test, we wish to obtain knowledge about which active relations have failed. The test path will therefore traverse causal relations from the active relations set. For example, given the active relations of Figure 3, each test path would pass through at least one of the relations r1 through r10. If an incorrect value results at the end of a path, there exist faults within the relations traversed by the path. Otherwise, evidence indicates faults probably do not exist in the path.

The process of path construction is depicted in Figure 4. The causal relation ri takes the values of the elements *ei* and causes the value at elements *ei+1*. Assume that the relations r' are not within the active relations set. If *ei* has the expected

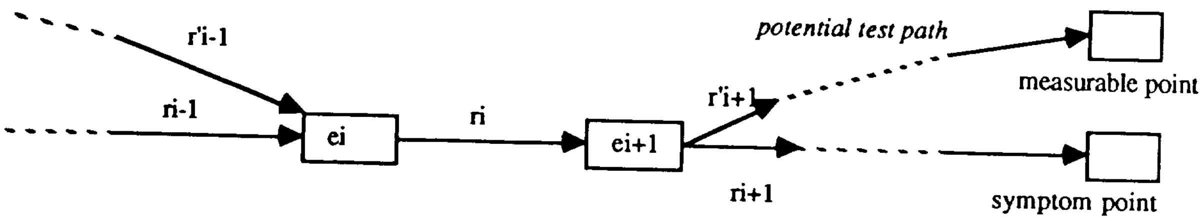

Figure 4. Path Construction for isolating possible fault of relation ri (see text).

value, but *ei+1* does not, *ri* must be faulty. To discriminate between *ri* and the relations *rj>i*, paths not passing through *rj* are needed. Paths through *ri* and r'*i+1* will therefore be constructed. Similarly, to discriminate between *ri* and *rj<i*, paths through r'*i-1* and *ri* are constructed. All of the constructed paths must terminate at a measurable element, so that measured values can be used to gain the required information.

To this end, the path constructor generates all paths through relations *ri* of the active relations set, with and without alternative relations r', terminating at measurable points. When a path is activated by a test input, the value at its measurable point is the test result.

Path Selection

The Path Selector chooses the path most useful to test, and gives it to the Causal Planner as a test goal.

The results of a particular test will give evidence about whether a fault exists within the active relations of the test. Therefore, to determine the usefulness of a path, it is necessary to know which causal relations need to be active to run the test. Figure 5 depicts this situation. The large triangle of Figure 5 represents the active relations which should have caused a correct value at the symptom-point. The small triangle represents the relations contributing to the expected values needed for the relation *ri-1*. If the path r' were to be tested, the relations represented by the large dashed triangle of Figure 5 would be under test, because they are all needed to activate the path r'. The way to determine this set of relations is to run the Causal Planner, as described in "Candidate Generation." Given a large model and a large set of potential paths (a common occurrence), the time to evaluate all the options would be prohibitive (In the example model, the planner runs for about 5 seconds for each plan). An approximation is therefore used. The relations within the small triangle are already known, because they were determined

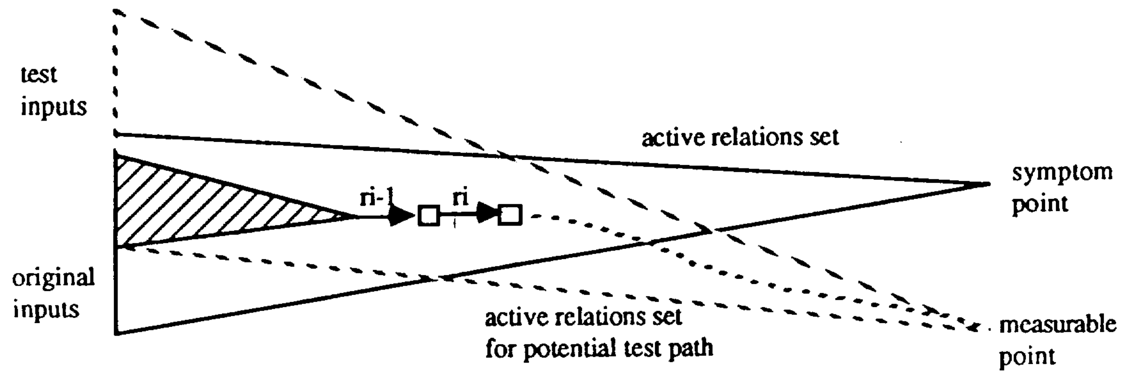

Figure 5. The active relations set for the test path is approximated by the relations comprising the small triangle union the relations forming the potential test path.

in finding the active relations leading to the original symptom. The union of that set and the set of relations used to construct the potential path can therefore approximate the active relations set. This approximation may skew path selection, by affecting the probability of the test passing, as indicated in the equations above. Fortunately, it will not affect accuracy of the test results, because the planner will later be run for the selected path to generate the test input. This will determine the exact active relations set.

At this point, the potential paths, each with the set of causal relations it tests, have been determined. The final step in path selection is to choose the set of relations most useful to test. Techniques similar to those presented in (de Kleer and Williams) apply to path selection. The essentials of the process are discussed here.

The best test is defined as that which minimizes the expected entropy of the candidate set, using the Information Theory definition of entropy

$$H = - \sum_{j}^{\text{candidates}} P(Cj) \log P(Cj).$$

As the probabilities move toward 0 or 1, this sum is minimized. The expected entropy resulting from a given test is

$$\hat{H} = \hat{H}(\text{passes})\, p(\text{passes}) + \hat{H}(\text{fails})\, p(\text{fails}).$$

In terms of the definition of entropy,

$$\hat{H}(\text{passes}) = -\sum_{j}^{\text{candidates}} P(Cj \mid \text{passes}) \log P(Cj \mid \text{passes})$$

$$\hat{H}(\text{fails}) = -\sum_{j}^{\text{candidates}} P(Cj \mid \text{fails}) \log P(Cj \mid \text{fails}).$$

The conditional probabilities of each candidate are as given in "Candidate Generation." The test which minimizes H is selected as the best test to run next.

The final part of test generation is implemented by giving the Causal Planner the desired value of the measurable point (the output of the selected path) as a goal. The constraints on the plan are that the approximated active relations set is included in the solution. When the planner terminates, the plan produced is the test input.

IV. MPC- An Implementation of Test Input Generation

A computer program implementing the test-generation/fault-isolation architecture of Figure 2. has been implemented as part of the MPC (Multi-Purpose Causal) tool. It is implemented in Lisp on a Symbolics 3670. MPC accepts models described in the DEFCAUSAL syntax and currently has an interface requesting tests and accepting measurement results. It has been tested on several models, including an expanded version of the example presented here.

An example diagnosis session using MPC will now be described, indicating the operation of the various test-generation subsystems. Assume that the sequence of TDRSS commands (mode :configure), (write 1,(write-3 35)), and (mode :experiment) were sent to the system of Figure 1. As shown in Figure 3, a value of 35 on the EXP-CMD-3 control line would be expected. Assume that this value was not observed. MPC is therefore given EXP-CMD-3 as the initial symptom point. Ten candidates are generated, one for each of the potentially faulty relations shown in Figure 3. The candidate probabilities are as follows:

{r1} = .100, {r2} = .100, {r3} = .100, {r4} = .100, {r5} = .100,
{r6} = .100, {r7} = .100, {r8} = .100, {r9} = .100, {r10} = .100.

Paths from the points associated with these candidates to measurable outputs (EXP-CMD-1 - EXP-CMD-4 and TDRSS-IN) are generated. The most useful path, according to the entropy-measurement equations described in "Path Selection," is the path from the BCU-CMD element to the measurable point EXP-CMD-1. This selection corresponds to the intuitive "divide the problem in half" approach often used by technicians. To generate a test of this path, the causal planner is invoked, and is constrained to use the causal relations r1 through r6 in the plan, as they were used to cause the point of interest BCU-CMD, as can be seen in Figure 3. The resulting plan is

```
TDRSS-OUT = (mode :configure)
TDRSS-OUT = (write 1, (write-1 35))
TDRSS-OUT = (mode :experiment).
```

MPC then prompts

```
Is the value at EXP-CMD-1 equal to 35?
```

Assume that the answer is "yes." The updated candidate probabilities are

{r1} = .039, {r2} = .039, {r3} = .039, {r4} = .039, {r5} = .039,
{r6} = .039, {r7} = .192, {r8} = .192, {r9} = .192, {r10} = .192,

indicating that the candidates describing the relations on the path from the experiment back to the TDRSS are most suspect. The paths from the active relations are once again evaluated. Based on the new candidate probabilities, however, the most useful path is from BCU-CMD to TDRSS-IN, but using a different causal relation from BCU-CMD. The path selected goes through the relation r11, using EXP-DATA-1, rather than the original EXP-DATA-4. The resulting plan is

```
TDRSS-OUT = (mode configure)
TDRSS-OUT = (write 1, read-1)
EXP-DATA-1 = 35
TDRSS-OUT = (mode experiment),
```

followed by the prompt

```
Is the value of TDRSS-IN equal to 35?
```

If the answer to the test is "yes," the probabilities indicate a strong preference for a single candidate, indicating that r7 is the faulty mechanism:

{r1} = .022, {r2} = .022, {r3} = .022, {r4} = .022, {r5} = .022, {r6} = .022,
{r7} = .543, {r8} = .109, {r9} = .109, {r10} = .109.

If this amount of convergence is sufficient to terminate testing, MPC reports its findings. Because R7 is implemented in the RIU module, RIU is reported as the suspect module. These results were obtained by using a value of .2 for r in the probability equations. With a smaller value, the convergence on the candidate {r7} would have been faster.

Conclusions

The test generation architecture implemented in the MPC system contributes a new tool to the set of causal reasoning capabilities now available. For systems in which few points are accessible, or in which transient effects are important, it provides a means to dynamically generate tests in response to observed symptoms.

The MPC approach is an extension to several other causal-reasoning efforts. Sharing some of the techniques of (Shirley), it generates tests to narrow down fault hypotheses, rather than to test specific components. It makes use of the probabilistic hypothesis generation and belief ideas of (de Kleer and Williams), but for test-generation purposes. This use of probability avoids the need for the "evidence model" required in the approach described in (Schaefer).

The current approach assumes that when a causal relation fails, the physical failure is in the device designed to implement the relation. Occasionally, however, another device may have a failure, such as a short circuit, which interferes to cause the relation to fail. Using the model to explore these possibilities, making use of "Pathways of Interactions" techniques similar to (Davis), is a topic of ongoing research. Other extensions include more sophisticated techniques for explaining the significance of test results to the user.

References

1. De Kleer, J., and B.C. Williams, "Diagnosing Multiple Faults," **Artificial Intelligence**, vol. 32, nr. 1, pp. 97-130, 1987.

2. Shirley, M. H., "Generating Tests by Exploiting Designed Behavior," **Proc. AAAI-86**, pp. 884-890, 1986.

3. Schaefer, P. R., "Higher Level Causal Reasoning for Diagnosis," **Proc. IEEE Int'l Workshop on AI for Industrial Appl.**, pp. 33-38, 1988.

4. Davis, Randall, **Diagnostic Reasoning Based on Structure and Behavior**, AI Memo 739, Massachusetts Institute of Technology, 1984.