HIGHER-LEVEL CAUSAL REASONING FOR DIAGNOSIS

Philip Schaefer
Martin Marietta Astronautics Group
P.O. Box 179, M.S. 4443
Denver, CO 80201 USA

ABSTRACT

Causal Reasoning provides a useful paradigm for diagnosis, as it provides a degree of flexibility not available from many conventional AI techniques. Most causal-reasoning approaches, however, reason at a detailed "first principles" level, although module-level diagnosis is often all that is required. This paper presents a higher-level technique for causal reasoning and diagnosis. Modules are the basic components in the resulting representation. Considerations unique to causal reasoning at a high level are discussed, and an algorithm and examples encorporating the resulting characteristics are presented.

I. Introduction

In recent years, the usefulness of Causal Reasoning for system analysis and diagnosis has been promoted by researchers in the AI field. Rather than providing traditional expert systems which deal with the target system in terms of "association rules" about symptoms, faults, and repairs, the Causal Reasoning approach provides a tool for analyzing the implications of the cause-and-effect structure of the system. Using domain-independent diagnosis techniques, it is possible to reason about which faults could cause the actual behavior to deviate from that predicted by the Causal model. This "deep reasoning" approach adds robustness and flexibility to the diagnosis task.

For many industrial applications, the causal-reasoning approach is particularly attractive[1]. For example, if the design of a system is changed, only the model reflecting the design change need be modified, rather than modifying multiple parts of a knowledge-engineered diagnosis task. For systems which have many possible configurations, such as an industrial controller with many optional modules, additional advantages obtain. Because the particular unit under test can be diagnosed simply by including in the causal model the parts actually present in the field and the test equipment available, separate diagnostics for each of the possible configurations and test hardware need not be prepared beforehand.

Most Causal-Reasoning systems reason at the "first principles" level, e.g., with electronic circuits, at the transistor or gate level[2,3]. While theoretically sound, reasoning about a system at that level is often excessively tedious when dealing with complex, real-world industrial equipment. This is especially true when module-level diagnosis is all that is required. An alternative method is to combine the best of the two diagnosis approaches- to integrate the robustness of a Causal-Reasoning approach with a higher-level view of the target system. The technique described in this paper makes this integration, providing Causal Reasoning on Knowledge-Engineered models of the target system modules. Using models provided by domain experts, this technique accepts a description of the observed symptom, derives test inputs to determine the nature of the anomaly, and generates further tests to be performed as the symptom is narrowed down to suspect modules. An expert system, FEXPR (Flexible EXPectation-based Reasoner)[1], has been constructed to demonstrate this technology.

II. Constructing a High-level Model

Building a Causal Model is accomplished by considering the target system, at the level at which diagnosis is desired, as a network of modules . An example network of the modules for a simple microprocessor-based system is shown in Figure 1. With the aid of an expert, such as a design engineer, the inputs and outputs (referred to as "external" signals) of each module are enumerated. Additionally, various "internal" signals are also defined, as required for the expert to explain the

Initial phases of this work were done at the Center for Automation and Intelligent Systems Research, Case Western Reserve University.

88CH2529-6/88/0000-0033 $1.00 © 1988 IEEE

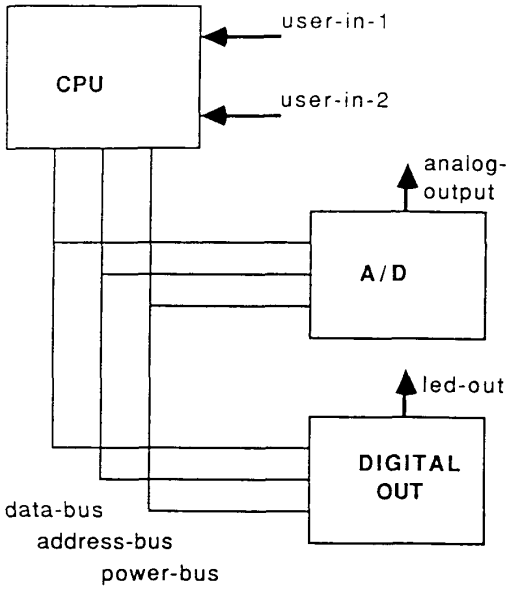

Figure 1. Microprocessor-based example.

behavior of the module. For example, the Analog-to-Digital (A/D) card of Figure 2. has the external signals of POWER, ADDRESS, BOARD-POSITION, and DATA-BUS. Additionally, the internal LATCH-VALUE signal was thought necessary by the expert to discuss the functionality of the card. In the FEXPR syntax, this module would be described as

(defmodule A/D
 :inputs (power address board-position data)
 :outputs (analog-output data)
 :internals (latch-value))

The next step in model construction is the specification of causal rules to describe how the appearance of values on certain sets of signals (presented in the "cause" parts of the rule) causes values to appear on others (presented in the "effects" parts of the rules). An example A/D causal rule is:

(defcausal A/D-1
 (power on) and
 (address X) and
 (board-position X) and
 (data Y) causes
 (latch-value Y)).

Figure 3. contains the set of modules and causal rules to describe the example system of Figure 1.

When the set of module definitions and causal rules is complete, the expert's picture of how each module functions in isolation (say, on a test bench) has been

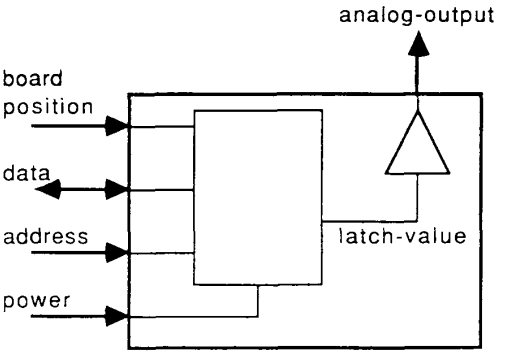

Figure 2. The A/D module with signals.

(defmodule A/D
 :inputs (power address board-position data)
 :outputs (analog-output data)
 :internals (latch-value))

(defcausal A/D-1
 (power on) and
 (address A) and
 (data D) and
 (board-position A) causes
 (latch-value D))

(defcausal A/D-2
 (power on) and
 (latch-value L) causes
 (analog-output (/ L 100)))

(defmodule CPU
 :inputs (cpu-power user-in-1 user-in-2)
 :outputs (data-lines address-lines))

(defcausal CPU-1
 (cpu-power on) and
 (user-in-1 A) and
 (user-in-2 D) causes
 (data-lines D)
 (address-lines A))

(defmodule DIGITAL-OUT
 :inputs (dig-pwr board-position address data)
 :outputs (led-out))

(defcausal DIGITAL-OUT-1
 (dig-pwr on) and
 (board-position B) and
 (address B) and
 (data D) causes
 (led-out D))

Figure 3. Higher-level causal model for a simple microprocessor-based system.

modeled. The interconnection between modules must also be specified to be able to reason about the functioning of the entire system. This is achieved by indicating the relevant "connection points" for each external signal of each module. Interactions of a module with other modules will take place via the connection points which they have in common. An example network of connection points is:

```
(defnetwork
 (cpu-board
  (user-in keyboard)
  (cpu-pwr power-bus)
  (data-lines data-bus)
  (address-lines address-bus))
 (A/D
  (power power-bus)
  (address address-bus)
  (board-position (set-to 6))
  (analog-out nil)
  (data data-bus))
 (digital-out
  (power power-bus)
  (address address-bus)
  (board-position (set-to 4))
  (led-out nil)))
```

where ADDRESS-BUS, POWER-BUS, DATA-BUS, etc. are the connection points. From this network description, we know that the "address" input of the A/D module is connected to the "address-lines" output of the CPU-board, for example.

III. The Diagnosis Algorithm

The fundamental notion in a higher-level diagnosis scheme is that faults in a target system will correspond to a discrepancy between the signal values derived from causal rules and the actual signal values. Fault diagnosis, therefore, is a process of finding the modules whose causal rules are violated.

One frequent approach to diagnosis is the use of "faulted component" models[4]. In this approach, the various known faulty states of components are modeled, and the fault models are inserted into the causal network to test hypotheses about possible faults. Because high-level rules rather than first-principles relations are used as models in the diagnosis task, however, it would not generally be feasible to apply this technique. This is because the faults in the target system will occur at the "physics" level. Each causal rule or module may correspond to many diverse physical processes, making the modeling of all possible faulted states very difficult indeed. Rather, it appears more appropriate to use a technique similar in spirit to [3] in which faults are isolated by finding causal relationships which appear not to hold. In the case of higher-level reasoning, this would mean locating the sets of causal rules whose "cause" parts are all true, but whose "effects" parts are not all true.

Another issue is the number of faults which should be assumed during the diagnosis task. With higher-level models, each causal relationship may correspond to many physical processes. Conversely, a single physical process may take part in many higher-level causal relationships. For example, a single physical fault in a microprocessor could affect many of the higher-level relationships describing the behavior of software running on that microprocessor. Therefore, it is clear that a "single-fault" assumption is not appropriate for this diagnosis task.

Some approaches use an "N-faults" technique, wherein N causal relationships at a time are suspended, to see if the rest of the system appears to follow its expected causal behavior[3]. In a first-principles approach, this is quite appropriate, because one can expect a good mapping between faults and causal relationships, if the correct physical perspective of the system is taken [3]. However, with the more abstract models at hand, such a mapping cannot be assumed, making the choice of how many faulty rules to hypothesize unclear. Additionally, suspending the models of all sets of n causal rules becomes computationally impractical for large values of N. Alternately, one could adopt the technique of suspending all rules for N modules at a time, but with a loss of efficacy, because partial functionality of a module often yields the most useful information about its faults. The solution presented here, therefore, is to assume that multiple faults may be present, but to make no commitment as to how many may exist. The resulting diagnosis algorithm is:

1. Determine the set of causal rules which show how the target system should have yielded the correct outputs for the inputs which were applied.

2. For each of the rules from 1), look for evidence that the "cause" parts of the rule were true, but that the "effect" parts were not all true.

3. When a set of rules meeting the criterion of 2) are found, the fault has been diagnosed.

4. If a complete diagnosis has not been achieved in 3), Gather the set of modules for which not enough evidence was obtained either to confirm or refute expected signal values. These are the suspected modules, which, for example, may be individually tested.

Successful diagnosis in step 3) is easiest for a system with many modules connecting to relatively few common points, such as a bus-oriented system. Systems with many modules connected in serial will be the most difficult to diagnose with this algorithm.

The following subsections discuss the details of the required operations, along with examples from the system of Figure 1.

III.A. Finding Causal Rules Relevant to a Symptom

A symptom requiring diagnosis will be defined as a set of output signal values which violate expectations, the expected values for those outputs, and the input signal values which failed to produce the expected results. Consider the example symptom, stated in English, "With USER-INPUT-2 of 52, I expected .52 volts at ANALOG-OUTPUT, but observed 1.3 volts." A malfunction symptom indicates that some causal relations which should hold in the system have been violated. The violated relations will be among those which should have contributed to the expected output signal values. Therefore, to find which causal rules could contribute to the expectation violation, those rules which show a mapping from the inputs to the expected, but violated outputs need to be found.

One way to find these relevant rules is to perform a backward-chaining search from the violated output values to the inputs, with the constraint that the derived inputs must include the values of the symptom input set. In the example, to have .52 volts at the ANALOG-OUTPUT point, rules A/D-1 A/D-2 would be matched to find the need for the POWER-BUS to have a value of ON, the ADDRESS-BUS to have a values of 6, and a DATA-BUS value of 52. Continuing, rules from the CPU-board module would similarly be used to find inputs USER-INPUT-1 of 52, and USER-INPUT-2 of 6. The set of relevant rules would therefore be A/D-1, A/D-2, and CPU-1.

In addition to backward chaining to find inputs, a forward-chaining operation needs to be performed to guard against undesired side effects. This could occur, for example, if a causal rule had more "effect" parts than the one matched in the backward-chaining operation. Such side effects could combine to cause other side effects, eventually resulting in a goal conflict. In the example, however, such side effects do not occur.

III.B. Inferring Signal Values

In order to find causal rules whose "cause" parts are all true, but whose "effect" parts are not all true, it is necessary to infer the actual values of the signals referenced by the relevant causal rules that have been found. For very simple systems, it would be practical to determine all such signal values by direct measurement- e.g., with a voltmeter. However, for many applications, some signals are much easier to measure than others. In the example system, the LED values would be very easy, the ANALOG-OUTPUT more difficult, and the DATA-BUS values very difficult. And, because the causal rules are a high level description of how the device functions, it is even possible to have abstract signals which do not correspond to physical points, and hence are not, in principle, possible to measure. Fortunately, techniques have been developed which allow much more efficient value inference, by finding indirect measurements, when appropriate, to obtain the necessary evidence.

III.B.1 Evidence Rules

To obtain indirect evidence about signal values, knowledge of the structure and relationships between signals is necessary. One of the interesting complications of the high-level causal algorithm, however, is that while evidence about the actual signal values for one suspected rule is being sought, one cannot assume that the other rules do hold, because of the "no N-fault assumption" property discussed previously. Because of this complication, causal rules cannot directly be used to make inferences about signal values. Even if the N-fault technique were used at the module level, and evidence about signals were restricted to come from other modules sharing common signals, problems would arise. This idea would work for systems where complete models for components were available [2,3,5]. However, with higher-level modeling, incomplete models may be commonplace. In the example system, the only explanation (from causal rules) for 0 volts on ANALOG-OUTPUT is a 0 on the DATA-BUS. In practice, though, the absence of power would also yield 0 volts. In modeling the A/D board, one can see, the expert did not consider an unpowered board as a useful thing to describe.

The solution adopted here is to knowledge-engineer "evidence" rules to augment the causal rules. Evidence rules are also higher-level rules, but indicate sets of signal values in their "known" (antecedant) parts which give reasonable

confidence about the signal values in their "inferrable" (consequent) parts. If the "known" parts are believed to be true, from measurement or from other evidence rules, the "inferrable" parts will be believed to be true. An important feature is that, unlike causal rules, the evidence rules do not assume a correctly-functioning module. The underlying notion is to capture relations which would be highly unlikely to be coincidences. To this end, evidence rules may reference the expected values of signals (as determined by causal rules for a correctly-functioning device). Examples of evidence rules in the FEXPR syntax are shown in Figure 4.

```
(defevidence digital-out-1
  (led-out X) implies
  (power on))

(defevidence digital-out-2
  (board-position P) and
  (led-out L) with-conditions
  (expected-that data L) and
  (expected-that address P) implies
  (data L))

(defevidence A/D-1
  (analog-output A) with-conditions
  (<> A 0) implies
  (power on)).
```

Figure 4. Example evidence rules for the system shown in Figure 1.

Consider the phase of diagnosis in which the signals of causal rule A/D-2 are being checked. The first check is for the "cause" part (power on). One way to check this would be to request measurement of the voltage of the POWER-BUS. However, before asking for a new measurement, the algorithm looks for evidence rules about that signal. Evidence rules digital-out-1 and A/D-1 are found. Because the symptom stated that .13 volts was measured at ANALOG-OUTPUT, rule A/D-1 gives indication that (power on) is actually true. If the evidence about ANALOG-OUTPUT were not available, the system would have used the digital-out-1 rule to ask

> Is there an output on LED-OUT?

which, in that case, would have been the easiest way to find evidence about the power signal.

III.B.2. Test Input Generation

In some cases, finding evidence requires more than measurement of signal values while the target system is in the state in which the symptom occurred. This could happen when the only evidence rules for a signal have "known" parts which are not currently true, but which could be true with appropriate new user inputs. For example, consider the case where it is necessary to test for the (data 52) "cause" part of causal rule A/D-1. The only evidence rule about this signal is digital-out-2, but it has (expected-that address 4) as a "known" part. The following algorithm is used to handle such cases, by generating inputs to meet the evidence rule requirements.

When looking for evidence about signal S, given the set of causal rules C which should have yielded the desired value of S and evidence rule E which potentially could yield evidence about S,

1) Set up the "known" parts of the evidence rule E as goals to be achieved.

2) Use the backward-chaining algorithm described above to find the appropriate user inputs to achieve the goals found in 1). Be sure to use the set C of causal rules in the solution to substantiate that S will have the desired value iff it did before.

3) Find evidence about the "known" parts of the evidence rule E to determine if evidence about S exists.

Note that this procedure may be called recursively, such that multiple levels of test input may be required to find evidence about particularly "well-buried" signals.

For the example discussed above, the signal S would be (data-bus 52), C would be causal rules A/D-1 and CPU-1, and E would be evidence rule digital-out-2. The signal goal would then be (led-out 52). The backward-chaining process would culminate with the input request

> Please enter the following:

Set USER-INPUT-1 to 52.
Set USER-INPUT-2 to 4.

which would be followed by measurement request

> What is the value on LED-OUT?

If it were 52, it could be inferred that the DATA-BUS value was indeed 52. By similar techniques, if it were determined that the ADDRESS-BUS value was indeed originally 6, all of the "cause" parts of the A/D module rules which should have resulted in the ANALOG-OUTPUT being .52, would have been verified. Therefore, diagnosis would indicate a fault with the A/D board, whose causal rules did not hold.

IV. Conclusions

Using Higher-level Causal Reasoning has shown to be a useful technique for introducing the flexibility of causal reasoning to complex systems for which module-level diagnosis is desired. The use of higher-level reasoning, however, introduces unique additional considerations. Methods of assuming n faults to be present fail due to the many-to-many mapping between physical processes and high-level causal relationships. Additionally, incompleteness in knowledge-engineered causal rules prevents their use for accurate inference about the internal state of the system under diagnosis. For these reasons, "evidence" rules were introduced to complement the causal rules.

A prototype system, FEXPR, has been successfully implemented to demonstrate the higher-level causal diagnosis approach. Current work includes the incorporation of temporal reasoning capabilities to allow more sophisticated diagnosis and test input generation. Additionally, techniques for a more automatic generation of evidence rules, based on causal rules and properties of classes of modules and their consitituent signals are being considered.

REFERENCES

[1] P. Schaefer and H.A. Guvenir, "Using Expert Systems to Troubleshoot Microprocessor Based Control Systems: An Expectation Based Approach," **IFAC Symp. Microcomputer Application in Process Control**, 1986.

[2] J. DeKleer and J.S. Brown, "A Qualitative Physics based on Confluences," in **Qualitative Reasoning about Physical Systems**, Elsevier Science Publishers, 1984.

[3] Randall Davis, **Diagnostic Reasoning Based on Structure and Behavior**, A.I.Memo 739, M.I.T., June 1984.

[4] Yung-Choa Pan, **Qualitative Reasonings with Deep-Level Mechanisms for Diagnosis of Dependent Failures**, Ph.D. Dissertation, Univ. Illinois at Urbana-Champaign, 1983.

[5] Y. Iwasaki and H.A. Simon, "Causality in Device Behavior," **Artificial Intelligence 29**, pp. 3-32, 1986.