DIAGNOSIS VIA CAUSAL REASONING: PATHS OF INTERACTION AND THE LOCALITY PRINCIPLE*

RANDALL DAVIS

The Artificial Intelligence Laboratory
Massachusetts Institute of Technology
545 Technology Square
Cambridge, MA 02139

Abstract

Interest has grown recently in developing expert systems that reason "from first principles", i.e., capable of the kind of problem solving exhibited by an engineer who can diagnose a malfunctioning device by reference to its schematics, even though he may never have seen that device before. In developing such a system for troubleshooting digital electronics, we have argued for the importance of pathways of causal interaction as a key concept. We have also suggested using a layered set of interaction paths as a way of constraining and guiding the diagnostic process.

We report here on the implementation and use of these ideas. We show how they make it possible for our system to generate a few sharply constrained hypotheses in diagnosing a bridge fault.

Abstracting from this example, we find a number of interesting general principles at work. We suggest that diagnosis can be viewed as *the interaction of simulation and inference* and we find that the concept of *locality* proves to be extremely useful in understanding why bridge faults are difficult to diagnose and why multiple representations are useful.

1. INTRODUCTION

Interest has grown recently in the development of expert systems that reason "from first principles", i.e., from an understanding of the structure and function of the devices they are examining. This approach has been explored in a number of domains, with the "devices" ranging from the gastro-intestinal tract [6], to transistors [1] and digital logic components like adders or multiplexors [3,5]. Our work has focused on the last of these, attempting to build a troubleshooter for digital electronic hardware.

By reasoning from first principles, we mean the kind of skill exhibited by an engineer who can troubleshoot a device by reference to its schematics, even though he may never have seen that particular device before. To do this we require something more than a collection of empirical associations specific to a given machine. We will see that the alternative mechanism has a degree of machine independence and is revealing for what it indicates about the nature of the diagnostic process.

We have previously proposed the use of a layered set of models as a mechanism for guiding diagnosis [2,3]. Here we describe the implementation of that idea and demonstrate its utility in diagnosing a bridge fault. We then abstract from this example to consider why bridge faults are difficult to diagnose and why multiple representations are useful. This results in a number of observations about the nature of diagnostic reasoning and the selection and design of representations.

* This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's Artificial Intelligence research on electronic troubleshooting is provided in part by a grant from the Digital Equipment Corporation.

2. CENTRAL CONCERNS

Four issues are of central concern in this paper. We describe them here briefly, enlarging on them in the remainder of the paper.

† *Diagnosis can be accomplished via the interaction of simulation and inference.*

Given knowledge of the inputs to a device and an understanding of how it is supposed to work, we can generate expectations about its intended behavior. Given observations about its outputs, we can generate conclusions about its actual behavior. Comparison of these two, in particular differences between them, provides the foundation for our troubleshooting.

† *Paths of causal interaction play a central role in diagnosis.*

An important part of the knowledge about a domain is understanding the mechanisms and pathways by which one component can affect another. We argue that such models of interaction are more fundamental than traditional fault models.

† *One technique for dealing with the complexity of diagnosis is layering the paths of interaction.*

To be good at hardware diagnosis, we need to handle many different kinds of paths of interaction. But this presents a problem: including all of them destroys our ability to discriminate among potential candidates, yet omitting any one of them makes it impossible to diagnose an entire class of faults. In response, we suggest the simple expedient of layering the models, using the most restrictive first and falling back on less restrictive models only in the face of contradictions.

† *The concept of* locality *proves to be a useful principle in both diagnosis and the selection of representations.*

We find that the concept of locality, or adjacency, helps to explain why bridge faults are difficult to diagnose: changes small and local in one representation are not necessarily small and local in another. We discover that locality can be defined by reference to the paths of interaction and find that the utility of multiple representations arises in part from the different definitions of locality they offer.

3. BACKGROUND

If we wish to reason from knowledge of structure and behavior, we need a way of describing both. We have developed representations for each of these, described in more detail elsewhere [3,4]. We limit our description here to reviewing only those characteristics of our representations important for understanding the example in Section 4.

The basic unit of description is a *module*, similar in spirit to the notion of a black box. Modules have *ports*, the places through which information enters and leaves the module.

3.1 Functional Organization, Physical Organization

By structure we mean information about the interconnection of modules. Roughly speaking, it is the information that would remain after removing all the textual annotation from a schematic.

Two different ways of organizing this information are particularly relevant to machine diagnosis: the *functional* view gives us the machine organized according to how the modules interact; the *physical* view tells us how it is packaged. We thus prefer to replace the somewhat vague term "structure" by the more precise terms *functional organization* and *physical organization*. In our system every device is described from both perspectives, producing two distinct (but interconnected) descriptions.

Both descriptions are hierarchical in the usual sense: modules at any level may have substructure. An adder, for example, can be described by a functional hierarchy (adder, individual bit slices, half-adders, primitive gates) and a physical hierarchy (cabinet, board, chip). The two hierarchies are interconnected, since every primitive module appears in both: a single xor-gate for example, might be both functionally part of a half-adder, which is functionally part of a single bitslice of an adder, etc., and physically part of chip E67, which is physically part of board 5, etc. Cross-link information for primitive modules is supplied by the schematic; additional cross-links can be inferred by intersection (e.g., the adder can be said to be on board 3 because all of its primitive components are in chips on board 3).

3.2 Describing Behavior

We define behavior in terms of the relationship between the information entering and leaving a module, and describe it by writing a set of rules. A complete specification of a module, then, includes its structural description as outlined above and a behavior description in the form of rules interrelating the information at its ports.

As we have noted elsewhere [3,4], we use rules that capture two distinctly different forms of knowledge: *simulation rules* model the electrical behavior of a device, while *inference rules* capture the reasoning we can do about it.

As a simple example, consider the behavior of an OR gate. The device simulation rule is[1]

```
If either input is a 1, then the output is 1, else
the output is 0
```

One of the device inference rules is

```
If the output is 0, then both inputs must have been
0.
```

Since the device is electrically unidirectional, it is clear that only the first rule can be modeling physical causality. The second rule, and the inference rules in general, capture conclusions we can make about the inputs of the device given its output.

This approach to describing behavior is very simple, but has nevertheless provided a good starting point for our work.

3.3 Troubleshooting

In previous papers [2,3] we outlined a progression of techniques that have been used in automated reasoning about circuits. We discussed test generation and argued that it handles only part of the problem, because it requires that we choose a part to test and specify how it might be failing. We then described discrepancy detection, showing how it offered important advantages.

But in examining cases involving a bridge fault or power failure, we discovered that straightforward use of discrepancy detection seemed unable to generate the appropriate candidates. We argued that the problem lay in distinguishing carefully between the machinery we use for solving problems and the knowledge that we give that machinery to work with.

3.3.1 Discrepancy Detection and Candidate Generation

Since understanding both the strengths and limitations of discrepancy detection is important in the remainder of this paper, we review the technique briefly. Consider the simple example shown in Figure 1.

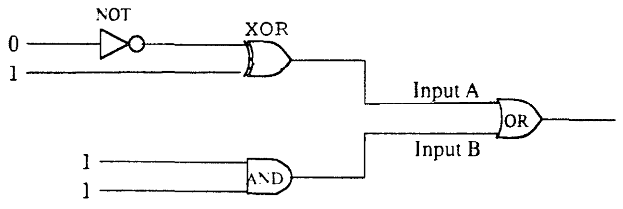

Figure 1 - Simple troubleshooting example

Assume that the actual device yields a 0, producing a discrepancy between what our simulation rules predicted and what the device produced.

We begin the process of generating plausible candidates --- devices whose misbehavior can explain the symptoms --- by asking why we expected a 1 at the output. There are three reasons: we expected that the OR gate was working, we expected `INPUT-A` to be 0, and `INPUT-B` to be 1. Assuming that there is a single point of failure, one of these expectations must be incorrect.

If the first expectation is incorrect, then the OR gate is failing, hence we can add that to our candidate list.

If one of the other expectations is incorrect, the OR gate is working and the problem lies further back. But if the OR gate is working, the inference rules about it are valid. In this case the inference rule shown earlier would indicate that both inputs must have been 0.

This matches our second expectation (`INPUT-A = 0`), so there is no discrepancy and thus no need to explore this expectation further. That is, the devices "upstream" of `INPUT-A` may or may not be completely free of faults, but under the current set of assumptions (made explicit below), none of them can be responsible for the observed misbehavior.

There is a discrepancy between our inference and the third expectation, since we expected a 1 from the AND gate. We proceed now with the AND gate just as we did with the OR gate, asking why we expected a 1, adding the gate to our list of candidates and pushing the inferred values yet further back in the circuit.

We describe this style of diagnostic reasoning as the *interaction of simulation and inference*. Simulation generates expectations about correct behavior based on inputs and knowing how devices work (the device simulation rules). Inference

1. This has been rendered in English to make it clear; for an example of the internal syntax see [3].

generates conclusions about actual behavior based on observed outputs and device inference rules. The comparison of these two, in particular differences between them, provides the foundation for our troubleshooting and has produced a system with a number of advantages.

It is, first of all, fundamentally a diagnostic technique, since it allows systematic isolation of possibly faulty devices. Second, since it defines failure functionally, i.e., as anything that doesn't match the expected behavior, it can deal with a wide range of faults, including any systematic misbehavior. Third, while we have illustrated it here at the gate level, the approach also allows natural use of hierarchical descriptions, a marked advantage for dealing with complex structures (see, e.g., [3]). Finally, the technique also yields symptom information about the malfunction. For example, if the OR gate is indeed the culprit, then we know a little about how it is misbehaving: it is receiving 0 and 1 and producing 0. This utility of this information is demonstrated below.

3.3.2 Mechanism and Knowledge

While this mechanism --- the interaction of simulation and inference --- is very useful, it is only as powerful as the knowledge we supply. Recall that in the example above, when exploring the cause of the discrepancy on `INPUT-B`, we looked only at the AND gate. Why didn't we think that some other module, like the inverter, could have produced the problem there? The answer of course is that there is no apparent connection between them, hence no reason to believe one might affect the other.

Note carefully the character of this assumption: it concerns the *existence of causal pathways, the applicability of a particular model of interaction*. We saw no way in which the inverter could affect `INPUT-B`, yet a pathway is clearly plausible --- a bridge fault, for example. We were implicitly assuming that there was no such pathway.

We believe that the important focus in this work is understanding such assumptions and the nature and character of the pathways. This understanding is crucial to candidate generation: given a discrepancy noticed at some point in the device, candidate generation attempts to determine which modules could have caused the problem. To answer the question we must know by what mechanisms and pathways modules can interact. Without some notion of how modules can affect one another, we can make no choice, we have no basis for selecting any one module over another.

In this domain the obvious answer is "wires": modules interact because they're explicitly wired together. But that's not the only possibility. As we saw, bridges are one exception; they are "wires" that aren't supposed to be there. But we also might consider thermal interactions, capacitive coupling, transmission line effects, etc.

Generating candidates, then, is not done by tracing wires, it is done by *tracing paths of causality*. Wires are only the most obvious pathway. In fact, given the wide variety of faults want to deal with, we need to consider many different pathways of interaction.

And that leaves us on the horns of a classic dilemma. If we include every interaction path, candidate generation becomes *indiscriminate* --- there will be some (possibly convoluted) pathway by which every module could conceivably be to blame. Yet if we omit any pathway, there will be whole classes of faults we will *never* be able to diagnose.

The key appears to lie in the models of interaction: we suggest that the difficult and important work is their enumeration and careful organization. We get a hint about organization from what a good engineer might do when faced with the dilemma above: make a number assumptions to simplify the problem, making it tractable, but be prepared to discover that some of those assumptions are incorrect. In that case, surrender them and solve the problem again with fewer simplifications.

This leads to the suggestion of layering the models. We start the diagnosis with the most restrictive model, the one that considers the fewest paths of interaction, and only use less restrictive models if this one fails. By "fail" we mean that we reach an intractable contradiction: given the current model and set of assumptions, there is no way to account for the observed behavior. This approach permits us to simplify the problem in order to get started, but does not prevent us from exploring more complex hypotheses.

A plausible guess at an ordering for the models might be[2]

* localized failure of function (e.g., stuck-at on a wire, failure of a RAM cell)
* bridges
* unexpected direction (inputs acting as outputs and driving lines)
* multiple point of failure
* timing errors
* assembly error
* design error

In terms of the dilemma noted above, the models serve as a set of *filters*. They restrict the categories of paths of interaction we are willing to consider, thereby preventing the candidate generation from becoming indiscriminate. But they are filters that we have carefully ordered and consciously put in place. If we cannot account for the observed behavior with the current filter in place, we remove it and replace it with one that is less restrictive, allowing us to consider additional categories of interaction paths.

4. LAYERS OF INTERACTION EXAMPLE: DIAGNOSING A BRIDGE FAULT

In this section we show how our system diagnoses a bridge fault, illustrating the utility of layering the interaction models.

There is, alas, a large amount of detail involved in working through this example. Where possible we have abstracted out much of it, but patience and a willingness to read closely will still be useful. A simple roadmap of the example will help make clear where we're going.

> The device is a 6-bit adder that displays an incorrect result in Test T1. The candidate generation mechanism outlined earlier produces a set S1 of three sub-components of the adder that can account for the misbehavior.
>
> A second test T2 is run to distinguish among the three possibilities in S1. Candidate generation produces a set S2 of two candidates capable of explaining the results of T2.
>
> Surprisingly, the intersection of S1 and S2 is null. We have reached a contradiction: no single component is capable of explaining all the data.

2. For the rationale behind this ordering see [2].

Put slightly differently, we have a contradiction *under the current set of assumptions and interaction models.* We therefore have to surrender one of our assumptions and use a less restrictive model.

The next model in the list --- bridge faults --- surrenders the assumption that the structure is as shown in the schematic and considers one additional interaction path: wires between adjacent pins.

Surrendering the assumption that the schematic is correct only indicates that we know what the structure is *not*; the difficult problem is generating plausible hypotheses about what it *is*.

Knowledge of electronics offers insight into how the physical modification --- adding a wire --- manifests itself functionally. This provides us with a behavior pattern characteristic of bridges that can be used to hypothesize their location.

Physical adjacency then provides a strong additional constraint on the set of connections which might be plausible bridges. The combined requirement of functional and physical plausibility results in the generation of only a very few carefully chosen bridge hypotheses.

The first attempt to apply these ideas produces two hypotheses that are plausible functionally, but prove to be implausible physically.

Dropping down a level of detail in our description reveals additional bridge candidates, two of which prove to be physically plausible as well. Further tests determine that one of them is in fact the error.

4.1 The Example

Consider the six bit adder shown in Fig. 2. Assume that the attempt to add 21 and 19 produces 36 rather than the expected value of 40. Invoking the candidate generation process described above, we would find that there are three devices whose individual malfunction can explain the behavior (SLICE-1, A2 and SLICE-2).[3]

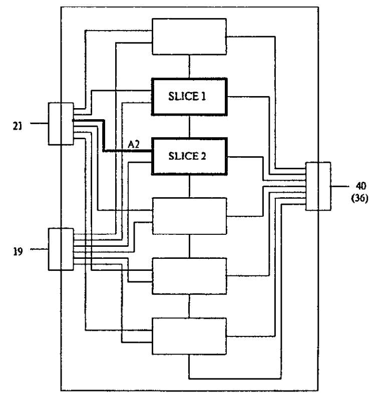

Figure 2 - Six bit adder constructed from single bit slices. Heavy lines indicate components implicated as possibly faulty.

3. The example has been simplified slightly for presentation.

A good strategy when faced with several candidates is to devise a test that can cut the space of possibilities in half. In this case changing the first input (21) to 1 will be informative: if the output of SLICE-2 does not change (to a 0) when we add 1 and 19, then the error must be in either A2 or SLICE-2.[4]

As it turns out, the result of adding 1 and 19 is 4 rather than 20. Since the output of SLICE-2 has not changed, it appears that the error must be in either A2 or SLICE-2.

But if we invoke the candidate generator, we discover an oddity: the only way to account for the behavior in which adding 1 and 19 produces a 4 is if one of the two candidates highlighted in Fig. 3 (B4 and SLICE-4) is at fault.

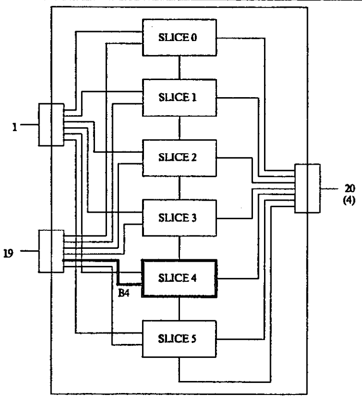

Figure 3 - Components indicated as possibly faulty by the second test.

Therein lies our contradiction. The only candidates that account for the behavior of the first test are those in Fig. 2; the only candidates that account for the second test are those in Fig. 3. There is no overlap, so there is no single candidate that accounts for all the observed behavior.

Our current model --- the localized failure of function --- has thus led us to a contradiction.[5] We therefore surrender it and consider the next model, one that allows us to consider an *additional kind of interaction path* --- bridging faults. The problem now is to see if there is some way to unify the test results, some way to generate a single bridge fault candidate that accounts for all the observations.

Much of the difficulty in dealing with bridges arises because they violate the rather basic assumption that the structure of the device is in fact as shown in the schematic. But admitting that the structure may not be as pictured says only that we know what the structure *isn't*. Saying that we may have a bridge fault narrows it to a particular class of modifications to consider, but the real problem here remains one of *making a few plausible conjectures about modifications to the structure.* Between which two points can we insert a wire and produce the behavior observed?

4. The generation of tests in this paper is currently done by hand; everything else is implemented. Work on automating test generation is in progress [7].

The logic behind this test is as follows: if the malfunctioning component really were SLICE-1, then the both A2 and SLICE-2 would be fault-free (the single fault assumption). Hence the output of SLICE-2 would have to change when we changed one of its inputs. (Notice, however, if the output actually does change, we don't have any clear indication about the error location: SLICE-2, for example, might still be faulty.)

5. Note that dropping down another level of detail in the functional description cannot help resolve the contradiction, because our functional description is a tree rather than a graph: in our work to date, at least, no component is used in more than one way.

To understand how we answer that question, consider what we have and what we need. We have test results, i.e., *behavior*, and we want conjectures about modification to *structure*. The link from behavior to structure is provided by knowledge of electronics: in TTL, a bridge fault acts like an and-gate, with ground dominating.[6]

From this fact we can derive a simple pattern of behavior indicative of bridges. Consider the simple example of Fig. 4 and assume that we ran two tests. Test 1 produced one candidate, module A, which should have produced a 1 but yielded a 0 (the zero is underlined to show that it is an incorrect output). Module B was working correctly and produced a 0 as expected. In Test 2 this situation is exactly reversed, A was performing as expected and B failed.

The pattern displayed in these two tests makes it plausible that there is a bridge linking the outputs of A and B: in the first test the output of A was dragged low by B, in the second test the output of B was dragged low by A.

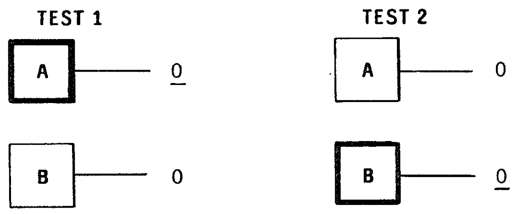

Figure 4 - Pattern of values indicative of a bridge. Heavy lines indicate candidates.

We have thus turned the insight from electronics into a pattern of values on the candidates. It is plausible to hypothesize a bridge fault between two modules A and B from two different tests if: in test 1, A produced an erroneous 0 and B produced a valid 0, and in test 2, A produced a valid 0 while B produced an erroneous 0. Note that this can resolve the contradiction of non-overlapping candidate sets: it hypothesizes one fault that involves a member of each set and accounts for all the test data.

Thus, if we want to account for all of the test data in the original problem with a single bridge fault, we need a bridge that links one of the candidates from the first test (SLICE-1, A2, SLICE-2) with one of the candidates from the second test (B4, SLICE-4), and that mimics the pattern shown in Fig. 4.

Fig. 5 shows the candidate generation results from both tests in somewhat more detail.[7] In that data there are two pairs of devices that match the desired pattern, yielding two functionally plausible bridge hypotheses:

Dotted line X, bridging wire A2 to the sum output of SLICE-4;

Dotted line Y, bridging the carry output of SLICE-2 to the sum output of SLICE-4.

6. This is in fact an oversimplification, but accurate enough to be useful. In any case, the point here is how the information is used; a more complex model could be substituted and carried through the rest of the problem.

7. As indicated earlier, the candidate generation procedure can indicate for each candidate the values that would have to exist at its ports for that candidate to be the broken one. For example, for SLICE-1 to be at fault in test 1, it would have to have the three inputs shown, with its sum output a zero (as expected) and its carry output also a zero (the manifestation of the error, underlined).

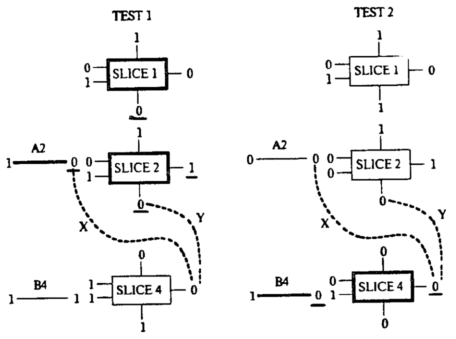

Figure 5 - Candidates and values at their ports.

But the faults have to be physically plausible as well. For the sake of simplicity, we assume that bridge faults result only from solder splashes at the pins of chips.[8] To check physical plausibility, we switch to our physical representation, Fig. 6. Wire A2 is connected to chip E1 at pin 4 and chip E3 at pin 4; the sum output of SLICE-4 emerges at chip E2, pin 13. Since they are not adjacent, the first hypothesis is not physically reasonable. Similar reasoning rules out Y, the hypothesized bridge between the carry-out of SLICE-2 and the sum output of SLICE-4.

I - end of A2
II - sum output of SLICE-4
III- carry out of SLICE-2

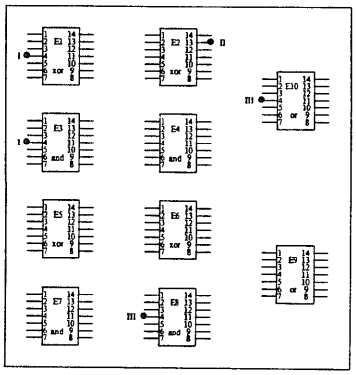

Figure 6 - Physical layout of the board with first bridge hypotheses indicated. (Slices 0, 2, and 4 are in the upper 5 chips, slices 1, 3, and 5 are in the lower 5.)

So far we have considered only the top level of functional organization. We can run the candidate generator at the next lower level of detail in each of the non-primitive components in Fig. 5. (Dropping down a level of detail proves useful here because additional substructure becomes visible, effectively revealing new places that might be bridged.)

We obtain the components and values shown in Fig. 7. Checking here for the desired pattern, we find that either of the two wires labeled A2 and S2 could be bridged to either of the two wires labeled S4 and C4, generating four functionally plausible bridge faults.

8. Again this is correct but oversimplified (e.g., backplane pins can be bent or bridged), but as above we can introduce a more complex model if necessary.

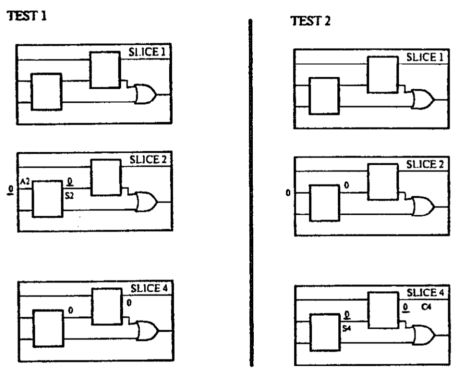

Figure 7 - Candidates at the next level of functional description. Each single bit adder is built from two "half-adders" and an OR gate. (To simplify the figure, only the relevant values are shown.)

Once again we check physical plausibility by examining the actual locations of A2, S2, S4, and C4, Fig. 8. [9] As illustrated there, two of the possibilities are physically plausible as well: A2-S4 on chip E1 and S2-S4 on chip E2.

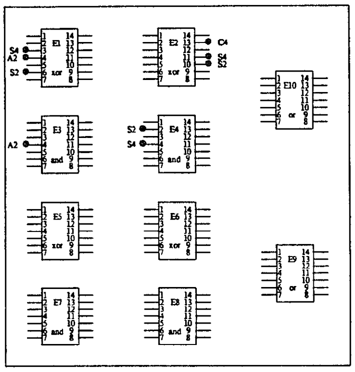

Figure 8 - Second set of bridge hypotheses located on physical layout.

Switching back to our functional organization once more, Fig. 9, we see that the two possibilities correspond to (X) an output-to-input bridge between the xor gates in the rear half-adders of SLICE-2 and SLICE-4, and (Y) a bridge between two inputs of the xor in the forward half-adders of slices 2 and 4.

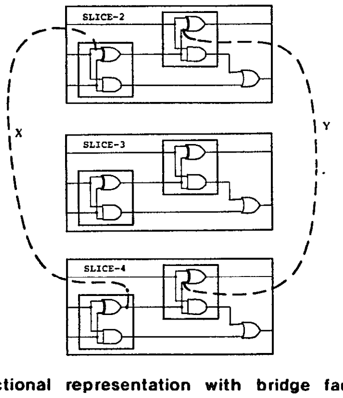

Figure 9 - Functional representation with bridge fault hypotheses illustrated.

It is easy to find a test that distinguishes between these two possibilities[10] : adding 0 and 4 means that the inputs of SLICE-2 will be 1 and 0, with a carry-in of 0, while the inputs of SLICE-4 will both be 0, with a carry-in of 0. This set of values will show the effects of bridge Y, if it in fact exists: the sum output of SLICE-2 will be 0 if it does exist and a 1 otherwise. When we perform this test the result is 1, hence bridge Y is not in fact the problem.

Bridge X becomes the likely answer, but we should still test for it directly. Adding 4 and 0 (i.e., just switching the order of the inputs), is informative: if bridge X exists the result will be 0 and 1 otherwise. In this case the result is 0, hence the bridge labeled X is in fact the problem.[11]

5. PATHS OF INTERACTION; THE LOCALITY PRINCIPLE

Two interesting questions are raised by the problem solving used just above.

Why are bridge faults difficult to diagnose?

Why does the physical representation prove to be so useful?

To see the answer, we start with the trivial observation that all faults are the result of some difference between the device as it is and as it should be. With bridge faults the difference is the addition of a wire between two physically adjacent points.

Now recall the nature of our task: we are presented with a device that misbehaves, not one with obvious structural damage. Hence we reason from behavior, i.e., from the functional representation. And the important point is that for a bridge fault, the difference in question --- the addition of a single wire --- is not *local* in that representation. As the comparison of Figs. 8 and 9 makes clear, the new wire connects two points that are adjacent in the physical representation but widely separated in the functional representation.

The difference is also not as *simple* in that representation: if we include in our functional diagram the AND gate implicitly produced by bridge X, we see that a single added wire in the physical representation maps into an AND gate and a fanout in the functional representation (Fig. 10).

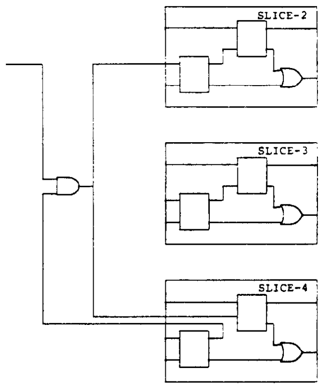

Figure 10 - Full functional representation of bridge fault X.

9. Note that the erroneous 0 on wire S2 can be in any of three physical locations, because S2 fans out (inside the module it enters on its right).

10. As above, tests are generated by hand.

11. Had both been ruled out by direct test, then we would once again have had a contradiction on our hands and would have had to drop back to consider yet a more elaborate model with additional paths of interaction.

This view helps to explain why bridge faults produce behavior that is difficult to envision and diagnose. Bridge faults are modifications that are simple and local in the physical description, but our diagnosis is done using the functional description. Hence the dilemma: *The desire to reason from behavior requires us to use a representation that does not necessarily provide a compact description of the fault.*

This non-locality and complexity should not be surprising, since devices physically adjacent are not necessarily functionally related. Hence there is no guarantee that a change that is small and local in one will produce a change that is small and local in the other. More generally, *changes local in one representation are not necessarily local in another.*

We can turn this around to put it to work for us:

Part of the art of choosing the right representation(s) for diagnostic reasoning is finding one in which the change in question is local.

This explains the utility of the physical representation: it's the "right" one because it's the one in which the change is local.

But why is locality the relevant organizing principle? We believe the answer follows from two facts: (a) devices interact through physical processes (voltage on a wire, thermal radiation, etc.) and (b) physical processes occur locally, or more generally, causality proceeds locally: there is no action at a distance. To make this useful, we turn it around:

The mechanisms (paths) of interaction define locality for us. That is, each kind of interaction path can define a representation.

Bridge faults arise from *physical adjacency* and hence are local in the physical representation. The notion of *thermal adjacency* would be useful in dealing with faults resulting from heat conduction or radiation, *electromagnetic adjacency* would help with faults dealing with transmission line effects, etc.

Each of these produces a different representation, different in its definition of locality. And each will be useful for understanding and diagnosing a category of fault.

There is still substantial work to do in enumerating the pathways of interaction, but we seem at least to be asking the right question. It seems to make sense for a wide range of faults and appears to be applicable to other domains as well. When debugging software, for example, the pathways of interaction differ (e.g., procedure call, mutation of data structures), but the resulting perspectives appear to make sense and there are some interesting analogies (e.g., unintended side effects in software are in some ways like bridge faults; there are even faults where the notion of "physical adjacency" is useful in understanding the bug, as in out of bounds array addressing).

6. SUMMARY

We seek to build a system that reasons from first principles in diagnosing hardware failures. We view diagnosis as the interaction of simulation and inference, with discrepancies between them driving the generation of candidates. In exploring this interaction, we find that the concept of paths of causal interaction plays a key role, supplying the knowledge that makes the diagnostic machinery work. But the desire to deal with a wide range of faults seems to force us to choose between an inability to discriminate among candidates and the inability to deal with some classes of faults.

In response, we suggest layering the interaction models, using the most restrictive first and hence considering the fewest paths of interaction initially. If this fails to generate a consistent hypothesis, we use the next model in the sequence, one which allows consideration of an additional pathway.

We illustrated this approach by diagnosing a bridge fault, sharply constraining the generation of hypotheses by using the physical representation as well as the functional. Finally, we found this to be one example of an important general principle --- locality --- and discovered that one useful definition of locality is given by the pathways of interaction.

Acknowledgments

Contributions to this work we made by all of the members of the Hardware Troubleshooting project at MIT, including: Howie Shrobe, Walter Hamscher, Karen Wieckert, Mark Shirley, Harold Haig, Art Mellor, John Pitrelli, and Steve Polit.

Bruce Buchanan and Patrick Winston offered a number of very useful comments on earlier drafts.

REFERENCES

[1] Brown J S, Burton R, deKleer J, Pedagogical and knowledge engineering techniques in the SOPHIE systems, Xerox Report CIS-14, 1981.

[2] Davis R, Reasoning from first principles n hardware troubleshooting, *Intl Journal of Man-machine Studies*, to appear, 1983.

[3] Davis R, et al., Diagnosis based on structure and function. *Proc AAAI 1982*, pp 137-142, August 1982.

[4] Davis R, Shrobe H E, Representing structure and function, *IEEE Computer*, to appear Sept 1983.

[5] Genesereth M, The use of hierarchical models in the automated diagnosis of computer systems, Stanford HPP memo 81-20, December 1981.

[6] Patil R, Szolovits P, Schwartz W, Causal understanding of patient illness in medical diagnosis, *Proc IJCAI-81*, August 1981, pp 893-899.

[7] Shirley W, Davis R, Digital test generation from symptom information, *IEEE 1983 VLSI Workshop*, to appear.