USING BELIEF REVISION TO DETECT FAULTS IN CIRCUITS[*]

VMES Technical Note No. 7
SNeRG Technical Note No. 15

Scott S. Campbell and Stuart C. Shapiro

Department of Computer Science
State University of New York at Buffalo
Buffalo, New York 14260, U.S.A.

campbl%buffalo@csnet-relay
shapiro%buffalo@csnet-relay

ABSTRACT

To detect faults in electrical circuits, programs must be able to reason whether the observed inputs and outputs are consistent with the desired function of the circuit. The SNePS Belief Revision System (SNeBR) is designed to reason the consistency of rules and hypotheses defined within a particular context or belief space. This paper shows how belief revision can be applied to fault detection in circuits, and so leads to a unification of the fields of belief revision (also known as truth maintenance) and fault detection.

[*] This work was conducted at the State University of New York at Buffalo, Department of Computer Science, by the Versatile Maintenance Expert System (VMES) Research Group. Support for this project was provided in part by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Bolling AFB DC 20332 under Contract No. F30602-85-C-0008, which supports the Northeast Artificial Intelligence Consortium (NAIC).

1. Introduction

Belief revision systems maintain records of hypotheses and justifications that are responsible for conclusions drawn by a reasoning system (Martins, forthcoming). If a *contradiction* is derived while reasoning, then the set of hypotheses responsible for the contradiction is declared to be inconsistent. A decision is then made about which hypotheses to discard in order to resolve the conflict so that inference can continue in a consistent belief space.

The Versatile Maintenance Expert System (VMES) Research Group is developing a versatile maintenance expert system (Shapiro, et al., 1985, 1986). We are striving for a system whose inputs would be the functional descriptions of an electrical device's components (from a component library), a description of how the components are connected to one another in that device, and its current observed inputs and outputs. From there, the system would analyze the device, isolate the fault to as few components as possible, and report its findings to a repairperson.

It has been suggested that belief revision (or truth maintenance, as it is sometimes called) may be particularly applicable to the field of fault detection (Doyle, 1979) (Martins, 1983) (Davis, 1984). It is apparent (Martins, forthcoming) that belief revision, in part, actually evolved from systems for circuit analysis such as EL (Stallman & Sussman, 1977), in which a record of all dependencies of each proposition was maintained. Our work was motivated, in part, by DART (Genesereth, 1984), a system that is specifically designed for isolating faults in circuits using design descriptions rather than MYCIN-like symptom fault rules. We were futher motivated by Ginsberg's work on counterfactuals (Ginsberg, 1985). His method for detecting faults involves asserting observed output values as counterfactual assumptions. In contrast, we use belief revision to attempt to derive values contrary to those observed, and our approach has been implemented using the

SNePS Belief Revision System (SNeBR) (Martins, 1983, forthcoming) (Martins and Shapiro, 1983). As far as we know, this is the first description of a running *domain-independent* belief revision system being applied to fault detection.

Our intent is to show that fault detection is a specific application of belief revision. In so doing, we are striving towards a unification of the two fields of belief revision and fault detection.

2. SNeBR Overview

2.1. Belief Revision Terminology[1]

SNeBR is a belief revision system based on the Semantic Network Processing System (SNePS) (Shapiro, 1979a). Information is added to the system as assumptions or hypotheses — including both facts and rules. A *context* is defined as any set of such hypotheses. The set of hypotheses that are believed to be true at any given time is referred to as the *current context*. Subsequently, inferences the system makes are confined to using only those hypotheses that constitute the current context and other propositions derived from them.

A *belief space* is the union of a context and all of the propositions derived within that context. A-belief space that contains both a statement and that statement's negation is said to contain a *contradiction*.

Information in SNeBR is represented by nodes. Each node, either hypothesized or derived, has three pieces of information associated to it. The first item is an *origin tag* which describes how the node was created. If the node was entered as a hypothesis, then

[1]For further discussion of the terms presented in this section, the reader is referred to (Martins, 1983) (Martins and Shapiro, 1983, 1984).

the origin tag is **hyp**. If the node was derived during execution, then the origin tag is either **der** or **ext**[2]. The second item associated with a node is its *origin set*. This is the set of all hypotheses that were used to derive a particular node. If the node in question is a hypothesis, then its origin set consists of only the node itself. The last item is a *restriction set*. This is a set of sets of nodes where each set is *known* to be inconsistent with that node's own origin set.

2.2. Example of Belief Revision

Consider the set of hypotheses[3] described in Figure 1. With these in the current context, SNeBR can be asked the question:

mortal(Socrates)?

wff3: $\forall(v1)\ {}_2X_1^1\ \{\text{mortal}(v1), \text{philosopher}(v1)\}$	Anything is either a philosopher or a mortal.
wff6: $\forall(v1)\ [\text{man}(v1) \rightarrow \text{mortal}(v1)]$	All men are mortal.
wff9: $\forall(v1)\ [\text{man}(v1) \rightarrow \text{rational-animal}(v1)]$	All men are rational animals.
wff12: $\forall(v1)\ [\text{philosopher}(v1) \rightarrow \text{prophet}(v1)]$	All philosophers are prophets.
wff13: philosopher(Socrates)	Socrates is a philosopher.
wff14: man(Socrates)	Socrates is a man.

Figure 1. Set of hypotheses.

[2]A node's derivation effectively depends on *every* node in its origin set. A der origin tag identifies normally derived nodes, whereas a ext origin tag identifies a special node that has an *extended* origin set. For further discussion see (Martins, 1983) (Martins and Shapiro, 1984, 1986).

[3]All descriptions of hypotheses and other nodes in this paper have been rewritten as well-formed formulas, without loss of meaning, to be more readable.

SNeBR finds that there is a contradiction in the current context, and displays Figure 2. This figure contains the node in which the contradiction occurred, followed by a prompt asking the user how he/she wishes to resolve that contradiction. The design of SNeBR *guarantees* that the members of the set {wff14 wff6 wff13 wff3} are the *only* hypotheses in the current context that can be contributing to the contradiction (Martins, 1983) (Martins and Shapiro, 1984, 1986). Since wff9 and wff12, although in the current context, are in no way contributors to the contradiction, they are not mentioned. This contradiction can be resolved by removing *any* of the wffs in the set {wff14 wff6 wff13 wff3} from the current context. However, this does not eliminate the possibility of

```
A contradiction was detected within context (wff14 wff13 wff12 wff9
                                             wff6 wff3). *
The contradiction involves the node

¬(mortal(Socrates) | der, {wff14 wff6}, {{wff3 wff13}})
                     | der, {wff13 wff3}, {{wff6 wff14}}
You have the following options:
  1. Continue anyway, knowing that a contradiction is derivable;
  2. Re-start the exact same run in a different context which is
     not inconsistent;
  3. Drop the run altogether.
Do you want to continue anyway?
=><= n
Do you want to re-start the run in a new context?
=><= y
In order to make the context consistent you must delete some hypotheses
from the set (wff14 wff6 wff13 wff3)
You are now entering a package that will enable you to delete some
hypotheses from this set.
Do you need guidance about how to use the package?
=><= n
```

Figure 2. A contradiction has been found.

* The connective $_{m}X_{i}^{j}$ is an and-or where, **m** is the total number of propositional arguments, **i** is the minimum number that must be true, and **j** is the maximum number that can be true. For additional information see (Shapiro, 1979a, 1979b).

deriving the same contradiction in a different way.

The node returned as the source of the contradiction contains several items describing the contradiction. First, mortal(Socrates) is a derived node with origin set = {wff14 wff6} and restriction set = {{wff3 wff13}}. Also notice that ¬(mortal(Socrates)) has also been derived within the current context, with origin set = {wff13 wff3} and restriction set = {{wff6 wff14}}. The derivation of both of these nodes within the current context *is* the contradiction, and the union of their origin sets is the smallest set of suspected hypotheses.

Once the contradiction has been identified, we need to decide what to do so that the conflict can be resolved. Therefore, we must examine each of the nodes in the conflict set until a culprit can be identified and discarded from the current context.

The first node to be examined in Figure 3a is wff14. We speculate that this hypothesis, which means *Socrates is a man*, is still valid. Consequently, we will keep this node in the current context. As we look at wff6 and wff13 in Figure 3b, we notice that these wffs should also be kept for similar reasons.

Finally, we come to wff3 described in Figure 4. Here, we re-evaluate our hypothesis that all philosophers are immortal, and choose to discard this wff from the current context.

Now that the current context has been modified, and hence no longer *known* to be inconsistent, we may once again consider Socrates' mortality. In very little time, we receive the answer:

mortal(Socrates) | der, {wff14, wff6}, {{wff3, wff13}}

3. Fault Detection

We may use SNeBR's ability to identify a subset of elements of the current context as suspects for a contradiction to reason about the function of a circuit. The current context is, in this case, a set of hypotheses describing how several electronic components are

```
Do you want to take a look at hypothesis wff14 ?
=><= y

man(Socrates) | hyp, {wff14}, {{wff3 wff13 wff6}}
There are 1 nodes depending on hypothesis wff14 :
(wff15).
Do you want to look at [a]ll of them, [s]ome of them, or [n]one?
(please type a, s or n)
=><= n
What do you want to do with hypothesis wff14 ?
[d]iscard from the context, [k]eep in the context,
[u]ndecided, [q]uit this package
(please type d, k or u)
=><= k
```

Figure 3a. A valid hypothesis is kept.

```
Do you want to take a look at hypothesis wff6 ?
=><= y

∀(v1)[man(v1) → mortal(v1)] | hyp, {wff6}, {{wff3 wff13 wff14}}
There are 1 nodes depending on hypothesis wff6 :
(wff15).
Do you want to look at [a]ll of them, [s]ome of them, or [n]one?
(please type a, s or n)
=><= n
What do you want to do with hypothesis wff6 ?
[d]iscard from the context, [k]eep in the context,
[u]ndecided, [q]uit this package
(please type d, k or u)
=><= k
Do you want to take a look at hypothesis wff13 ?
=><= y

philosopher(Socrates) | hyp, {wff13}, {{wff3, wff6, wff14}}
There are 1 nodes depending on hypothesis wff13 :
(wff16).
Do you want to look at [a]ll of them, [s]ome of them, or [n]one?
(please type a, s or n)
=><= n
What do you want to do with hypothesis wff13 ?
[d]iscard from the context, [k]eep in the context,
[u]ndecided, [q]uit this package
(please type d, k or u)
=><= k
```

Figure 3b. More valid hypotheses are kept.

```
Do you want to take a look at hypothesis wff3 ?
=><= y

∀(v1) _2X_1^1 {mortal(v1), philosopher(v1)} | hyp, {wff3}, {{wff13, wff6, wff14}}
There are 1 nodes depending on hypothesis wff3 :
(wff16) .
Do you want to look at [a]ll of them, [s]ome of them, or [n]one?
(please type a, s or n)
=><= n
What do you want to do with hypothesis wff3 ?
[d]iscard from the context, [k]eep in the context,
[u]ndecided, [q]uit this package
(please type d, k or u)
=><= d

The following (not known to be inconsistent) set of
hypotheses was also part of the context where the
contradiction was derived:
(wff12 wff9)
Do you want to inspect or discard some of them?
=><= n
Do you want to add some new hypotheses?
=><= n
```

Figure 4. A questionable hypothesis is discarded.

connected to create a working device. Based on these and additional hypotheses of the observed inputs, observed outputs, and functional knowledge of the device, SNeBR can reason what the output of the device should be — given that it is working as described. If SNeBR finds that there is an inconsistency between the predicted output values and observed output values, then a contradiction is raised and the list of hypotheses underlying the contradiction, representing only those components that can possibly contribute to the fault, is returned.

3.1. Example 1

The example that we will use here is a simple full adder as diagrammed in Figure 5. Each pin (X, Y, Z, S, C), gate (X1, X2, A1, A2, O1), and all of the connections between

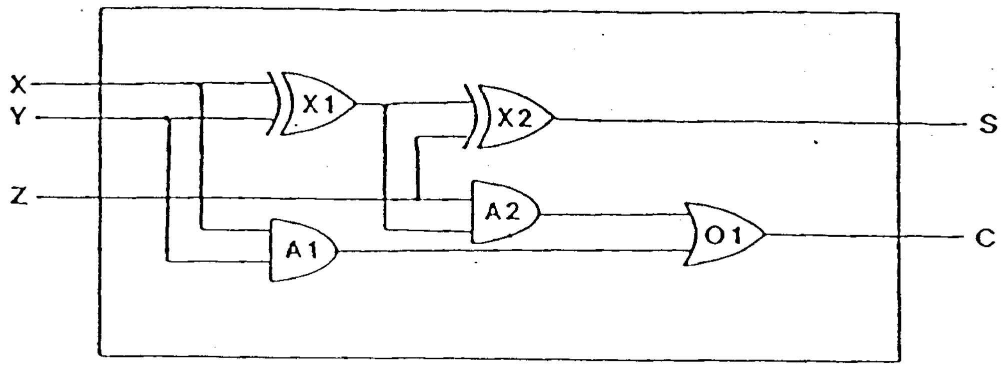

Figure 5. Full Adder.

these components are entered as hypotheses indicating their respective component type (i.e., we are hypothesizing that each of these items is indeed what it appears to be). The following five hypotheses represent each of the five gates in the adder.

wff94: XORG(X1)

wff95: XORG(X2)

wff96: ANDG(A1)

wff97: ANDG(A2)

wff98: ORG(O1)

In order to give the reader an understanding of the rules used in this example, consider the functional description of an exclusive-or, gate given in Figure 6a. This rule simply states that for all exclusive-or gates, *v*1, if *v*1's inputs are different, then *v*1 has the value 1; and if the inputs are the same, then *v*1 has the value 0. Figure 6b, on the other hand, states that for all objects *v*1 , that object has either a value of 1 or 0, and not both.

Consider a situation in which the observed inputs for the adder are X= 1, Y= 0, and Z= 1, and the observed outputs are S= 1 and C= 1, corresponding to S and C. The following hypotheses would then be added to the system to represent the observed values of each

$$\forall(v1)\,[\mathrm{XORG}(v1) \rightarrow {}_{2}\mathbb{X}_{2}^{2}\,\{(\mathrm{eq}(\mathrm{in1}(v1),1) \wedge \mathrm{eq}(\mathrm{in2}(v1),0) \vee \mathrm{eq}(\mathrm{in1}(v1),0) \wedge \mathrm{eq}(\mathrm{in2}(v1),1)) \rightarrow \mathrm{eq}(\mathrm{val}(v1),1)), (\mathrm{eq}(\mathrm{in1}(v1),1) \wedge \mathrm{eq}(\mathrm{in2}(v1),1) \vee \mathrm{eq}(\mathrm{in1}(v1),0) \wedge \mathrm{eq}(\mathrm{in2}(v1),0)) \rightarrow \mathrm{eq}(\mathrm{val}(v1),0))\}]$$

Figure 6a. Operational Description of an Exclusive-or Gate.

$$\forall(v1)\ {}_{2}\mathbb{X}_{1}^{1}\ \{\mathrm{eq}(\mathrm{val}(v1),1),\ \mathrm{eq}(\mathrm{val}(v1),0)\}$$

Figure 6b. All objects either have the value 1 or 0, and not both.

of these five pins.

eq(val(X), 1)
eq(val(Y), 0)
eq(val(Z), 1)
eq(val(S), 1)
eq(val(C), 1)

Having hypothesized the functional descriptions, connectivity, and observed values of the device, we can ask the question:

$${}_{2}\mathbb{X}_{2}^{2}\ \{\mathrm{eq}(\mathrm{val}(S),0),\ \mathrm{eq}(\mathrm{val}(C),0)\}?$$

This question amounts to asking whether or not it is possible to derive values for S and C that are different from those that were observed. Since the component being tested is a full adder, the only possible value different from the observed 1 is 0. If it is possible to derive a predicted value of 0 for either output, then this constitutes a contradiction of the observed value, and SNeBR will notify us of this. Figure 7 shows SNeBR's report that S both has and doesn't have the value 0.

Notice that the set of wffs that needs to be modified to make the context consistent includes, **wff94** and **wff95** (shown in bold in Figure 7) representing the two gates, X1

```
A contradiction was detected within context (wff103 wff102 wff101
                    wff100 wff99 wff98 wff97 wff96 wff95 wff94 wff93
                    wff92 wff91 wff90 wff89 wff88 wff87 wff86 wff85 wff84
                    wff83 wff82 wff81 wff80 wff79 wff78 wff77 wff76 wff71
                    wff67 wff63 wff60 wff53 wff40 wff27 wff20 wff10).
The contradiction involves the node

¬(eq(val(S), 0) | ext, {wff92 wff80 wff101 wff87 wff86 wff82 wff99
                    wff100 wff83 wff71 wff10 wff94 wff67 wff20 wff95
                    wff76}, {{wff63 wff102}})
                             | der, {wff102 wff63}, {{wff76 wff95 wff20
                                       wff67 wff94 wff10 wff71 wff83 wff100
                                       wff99 wff82 wff86 wff87 wff101 wff80
                                       wff92}}
You have the following options:
  1. Continue anyway, knowing that a contradiction is derivable;
  2. Re-start the exact same run in a different context which is
     not inconsistent;
  3. Drop the run altogether.
Do you want to continue anyway?
=><= n
Do you want to re-start the run in a new context?
=><= y
In order to make the context consistent you must delete some
hypotheses from the set (wff92 wff80 wff101 wff87 wff86 wff82 wff99
                              wff100 wff83 wff71 wff10  wff94  wff67 wff20
                              wff95  wff76 wff102 wff63)
You are now entering a package that will enable you to delete some
hypotheses from this set.
Do you need guidance about how to use the package?
=><= n
```

Figure 7. Contradiction involving two gates.

and X2, from the full adder that are possibly faulty. The rest of the wffs in that set represent all of the connections and pins in the adder that could possibly be contributing to the fault (highlighted in Figure 8) as well as the observation that S= 1, and the rules describing the function of the gates. The set *does not contain* wff96, wff97, or wff98, which represent the other three gates in the full adder, indicating that they are not responsible for the fault. This indicates that somewhere along the path of highlighted components in Figure 8 is a part that is not working as described or hypothesized. This example shows that SNeBR can identify a subset of the components as suspect for a fault in a circuit. The members of the set of suspects are known to be the *only* components that can possibly be contributing to the fault — due to SNeBR's design (Martins, 1983) (Martins and Shapiro, 1984, 1986). Hence, one of these suspects *must* be the faulty component.

3.2. Example 2

This next example shows that the set of suspects SNeBR identifies does, indeed, depend on the particular fault.

Consider the full adder used in example 1; however, in this instance the observed outputs are both 0's. In this case, the following set of observed values for the device are hypothesized, and again we try to derive different values.

eq(val(X), 1)
eq(val(Y), 0)
eq(val(Z), 1)
eq(val(S), 0)
eq(val(C), 0)

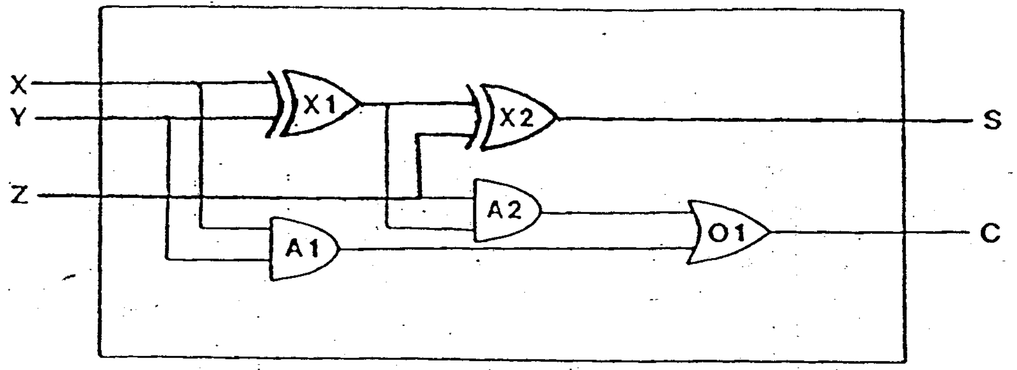

Figure 8. Possible faulty parts of the adder.

$_2X_2^2$ {eq(val(S), 1), eq(val(C), 1)}?

Similarly to the previous example, a contradiction is found in Figure 9. This time, the contradiction involves the derivation of both eq(val(C), 1) and ¬eq(val(C), 1) within the current context. As a result of this derivation, the set of questionable components

```
A contradiction was detected within context (wff103 wff102 wff101
                wff100 wff99 wff98 wff97 wff96 wff95 wff94 wff93
                wff92 wff91 wff90 wff89 wff88 wff87 wff86 wff85 wff84
                wff83 wff82 wff81 wff80 wff79 wff78 wff77 wff76 wff71
                wff67 wff63 wff60 wff53 wff40 wff27 wff20 wff10).
The contradiction involves the node

¬(eq(val(C), 1) | ext, {wff93 wff81 wff91 wff101 wff89 wff88 wff82
                wff83 wff10 wff94 wff27 wff97 wff90 wff99 wff84 wff100
                wff85 wff71 wff46 wff96 wff67 wff53 wff76}, {{wff63
                wff103}})
                    | der, {wff103 wff63}, {{wff76 wff98 wff53 wff67
                        wff96 wff40 wff71 wff85 wff100 wff84
                        wff99 wff90 wff97 wff27 wff94 wff10
                        wff83 wff82 wff88 wff89 wff101 wff91
                        wff81 wff93}}
You have the following options:
 1. Continue anyway, knowing that a contradiction is derivable;
 2. Re-start the exact same run in a different context which is
    not inconsistent;
 3. Drop the run altogether.
Do you want to continue anyway?
=><= n
Do you want to re-start the run in a new context?
=><= y
In order to make the context consistent you must delete some
hypotheses from the set (wff93 wff81 wff 91 wff101 wff89 wff88 wff82
                    wff83 wff10  wff94  wff27  wff97  wff90 wff99
                    wff84 wff100 wff85 wff71 wff40  wff96  wff67 wff53
                     wff98  wff76 wff103 wff63)
You are now entering a package that will enable you to delete some
hypotheses from this set.
Do you need guidance about how to use the package?
=><= n
```

Figure 9. Contradiction involving four gates.

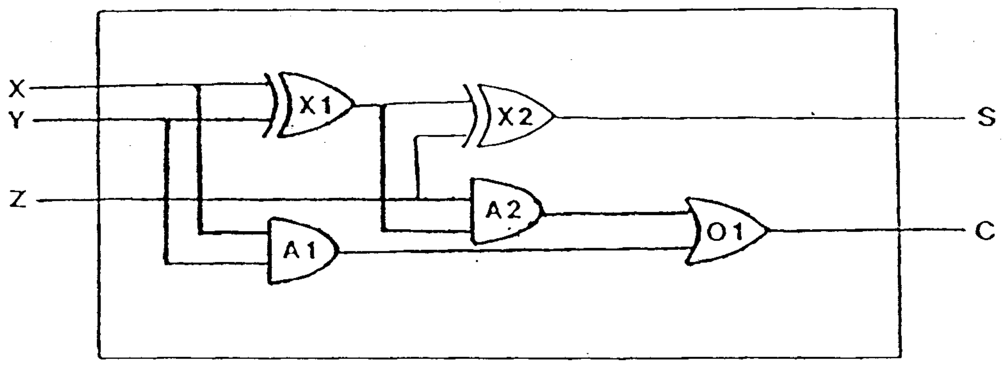

Figure 10. Possibly faulty components of adder.

includes wff94, wff96, wff97, and wff98 — all the gates except X2. As indicated, the exclusive-or gate X2 and nearby connections were not selected as possible culprits as shown in Figure 10.

4. Scaling-up the Approach

The approach of using belief revision to detect faults appears to be practical for a device such as a full adder. However, for a complex device, it may be impractical to describe its function at the level of detail used in presenting the full adder due to the large number of components that would be involved. To avoid this problem a component library is consulted which contains the ***input/output behavior*** of every component.

When a device is diagnosed, a description of the device, containing the input/output behavior and connectivity of its components, is used to identify which components can possibly be at fault. Each of the suspected components is then diagnosed in a similar fashion until we reach the level of the replaceable component. At that point we decide that a component is bad and physically replace it in the device.

5. Future Work

In this paper we discussed fault detection of objects whose values could either be a 1 or a 0. With more complex devices, it may be necessary to represent objects that can have one of several values on its wires, rather than just a 1 or 0. In a future paper, we will show how this method of fault detection can be upgraded to work on such circuits as those described in (Taie, et al., 1986), by introducing *numerical quantifiers* (Shapiro, 1979c) into the system.

The result of this work using SNeBR is the first step in actually *isolating* the fault(s) of a circuit, by removing all of those components that *cannot* be faulty. The second step is to examine all of the suspect components, and narrow that set down to as few components as possible (hopefully just one). However, this narrowing is not just applicable to fault isolation. In the general belief revision problem, it amounts to identifying the *particular* suspect hypothesis that *should* be removed from the current context. To achieve this, we need to examine the suspects in some order, by *assigning* some value of blame to each suspect reflecting the possibility that it is at fault. Then, each hypothesis must be tested, from the most blameworthy to the least, until a culprit for the fault is found.

Perhaps attaching blame to suspect components involves identifying the hypotheses in the conflict set that supports the observed outputs of other components. Consider the diagrams in Figures 8 and 10. In these Figures, X, Y, Z, and X1 contribute to deriving values for C (in Fig. 8) and S (in Fig. 10) that are consistent with the observed values. Hence, when isolating a fault, it appears that this kind of information should be taken into account.

6. Conclusion

We have shown that domain-independent belief revision systems *can* be used to detect faults in electrical circuits. By hypothesizing the observed inputs, observed outputs, connectivity of the device, and the function of its components, we can reason how the observed values compare to the particular value of the device. If they are inconsistent, then a list of suspect hypotheses is returned, and this provides the possibly faulty components with *no* mention of any component that cannot be a contributor to the fault. Isolating the single faulty component amounts to finding the most blameworthy hypothesis. It is our contention that fault detection is really an application of belief revision. This unification of belief revision and fault detection has been illustrated by applying SNeBR, a running domain-independent belief revision system, to a problem of fault detection in a full adder.

Acknowledgements

We thank James Geller, Sargur N. Srihari, and members of the SNePS Research Group for their invaluable support and comments — especially William Rapaport and Ming-Ruey Taie for their comments and suggestions.

We also thank both Dale Richards and Norman J. Sturdevant of the Rome Air Development Center, Griffiss AFB., for their continuing support and cooperation.

References

Davis R., "Diagnostic Reasoning Based on Structure and Behavior," *Artificial Intelligence* Vol. 24, No. 1-3, 1984.

Doyle J., "A Truth Maintenance System", *Artificial Intelligence*, Vol. 12, No. 3, 1979, pp. 231-272.

Genesereth M. R., "The Use of Design Descriptions in Automated Diagnosis", *Artificial Intelligence* Vol. 24, No. 1-3, 1984, pp. 411-436.

Ginsberg M. L., "Counterfactuals", *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, 1985.

Martins J. P., "Reasoning in Multiple Belief Spaces", Ph.D. Dissertation, Department of Computer Science, SUNY at Buffalo, May 1983.

Martins J. P., "Belief Revision", in *The Encyclopedia of Artificial Intelligence*, ed. S. C. Shapiro, Wiley & Sons, New York, forthcoming.

Martins J. P. and Shapiro S. C., "Reasoning in Multiple Belief Spaces", *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, 1983.

Martins J. P. and Shapiro S. C., "A Model for Belief Revision", *Non-Monotonic Reasoning Workshop*, The American Association of Artificial Intelligence, 1984, pp. 241-294.

Martins J. P. and Shapiro S. C., "Theoretical Foundations for Belief Revision", in *Theoretical Aspects of Reasoning About Knowledge*, ed. J. Y. Halpern, Morgan Kaufmann Pub., Monterey, 1986, pp. 383-398.

Shapiro S. C., "The SNePS Semantic Network Processing System", in *Associative Networks*, ed. N. V. Findler, Academic Press, New York, 1979a, pp. 179-203.

Shapiro S. C., "Using Non-standard Connectives and Quantifiers for Representing Deduction Rules in a Semantic Network", presented at "Current Aspects of AI Research", a seminar held at the Electrotechnical Laboratory, Tokyo, August 27-28, 1979b.

Shapiro S. C., "Numerical Quantifiers and Their Use in Reasoning With Negative Information", *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, 1979c, pp. 791-796.

Shapiro S. C., Srihari S. N., Taie M-R., Geller J., "Development of an Intelligent Maintenance Assistant", *SIGART Newsletter*, (92) April 1985, pp. 48-49.

Shapiro S. C., Srihari S. N., Taie M-R., Geller J., "VMES: A Network based Versatile Maintenance Expert System", *Applications of AI to Engineering Problems*, Southampton Univ., U.K., 1986.

Stallman R. and Sussman G., "Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis," *Artificial Intelligence*, Vol. 9, 1977, pp. 135-196.

Taie M. R., Srihari S. N., Geller J., Shapiro S. C., "Device Representation Using Instantiation Rules and Structural Templates," *Proceedings of CSCSI-86*, Montreal, Canada, 1986.