Model-based diagnosis in SOPHIE III[1]

Johan de Kleer and John Seely Brown
Xerox Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto CA 94304 USA
Email: dekleer, brown@parc.xerox.com

April 24, 1992

Abstract

This paper describes the model-based diagnostician used in SOPHIE III. This diagnostician is capable of localizing faults in analog circuits such as DC power supplies. Although it presumes that the circuit contains a single fault, it introduces a variety of important model-based techniques: (1) it uses models of component behaviors to perform first-principles reasoning to detect conflicts and corroborations with observations, (2) it considers a component to be faulted if its removal would eliminate all conflicts yet preserve all corroborations, (3) it uses propagation of constraints with numerical ranges, (4) it uses fault mode information to eliminate components from consideration, and (5) it provides a smooth integration with rule-based diagnostic rules when first-principles reasoning is inadequate to perform the diagnosis.

1 Introduction and history of the SOPHIE project

The research described in this paper took place over a five-year period and centered around three different SOPHIE systems (known as SOPHIE I, SOPHIE II and SOPHIE III). Work began on what was to become SOPHIE I in early 1973 at the University of California at Irvine and was completed in 1977 at Bolt Baranek and Newman in Massachusetts. SOPHIE was a large project which produced results in diverse domains. This paper focuses on one small part of this project: the inferential machinery for drawing diagnostic inferences in SOPHIE III. The following is a brief history of the SOPHIE project. For a fuller overview of the project see [1].

[1]This paper integrates excerpts from a number of earlier papers. Two of these are: "Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III" by J.S. Brown, R. R. Burton and J. de Kleer which appeared in *Intelligent Tutoring Systems* edited by D. Sleeman and J.S. Brown published by Academic Press (1982) [1]; "Local methods of localizing faults in electronic circuits," by J. de Kleer published as M.I.T. Artificial Intelligence Laboratory Memo AIM-394 (1976) [2]. This paper was retypeset with a modern Latex on May 14, 2019.

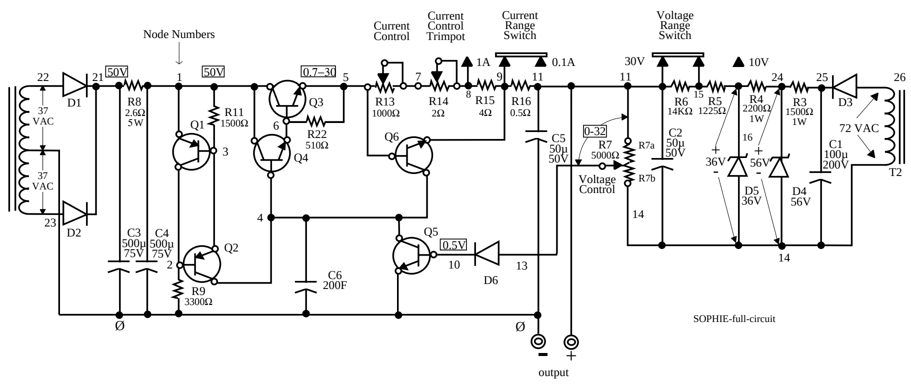


Figure 1: Schematic of the IP-28

The Air Force had expressed in interest in using computers in their advanced electronic-troubleshooting course, particularly in the laboratory section; and we were interested in exploring interactive learning environments which encouraged explicit development of hypotheses during fault solving by facilitating the communication of the student's ideas to the machine and enabling the machine to critique them. From this motivation came the original SOPHIE I which provided the student an electronic troubleshooting environment. The instructor can insert faults into the simulated circuit (using SPICE [6]) and the student, through a natural language interface, make measurements to pinpoint the faulty component. The SOPHIE project focused on troubleshooting DC power supplies (an example of which is illustrated in Figure 1). The student was not restricted to a few predetermined measurement points, but rather could make any measurement he/she wanted. A fundamental advantage of using a simulated over an actual circuit is that it makes it easy for the instructor to insert any conceivable fault, not just those that are easy to insert in a physical circuit, or those which do not destroy the entire circuit. SOPHIE I also includes extensive pedagogical machinery and course material with which we do not concern ourselves here.

In SOPHIE II we began to explore the idea of a computer-based troubleshooting coach which could advise and guide the student's actions with the objective of improving their troubleshooting skills. Of the many ideas explored, three bear on the current issue of diagnostic inference we are focusing on here. First, we provided a module which was capable of expert-level troubleshooting of any fault; thus the student could observe an expert diagnosing a fault. Second, we provided a module which would tell the student which faults could explain the measurements made so far. Finally, we provided a module which critiqued the student's actual measurements. Although all of these capabilities, on the surface, appear to require substantial inferential machinery, they were all achieved by very simple underlying mechanisms.

The expert troubleshooting of SOPHIE II module is based on an underlying decision tree. However, the decision tree is difficult to construct and must be modified for each new circuit. The decision tree typically cannot accommodate the measurements the student has already made. It cannot tell the student what should be done next, but only what an expert would have done starting from the outset. Finally, the decision tree cannot provide pedagogically useful rationale for its decisions.

The fault identification module of SOPHIE II is more interesting. It is completely general, but is computationally intractable and therefore of limited use. It uses SPICE in a novel way by adding all the observations to the equation set being solved and systematically leaving each of the component parameters unspecified. By employing SPICE in a root-finding loop SOPHIE II can identify what component parameter shifts would explain the symptoms. For example, R3 has a nominal value of 1500 ohms. By iterating SPICE runs we might determine that if R3 were 50 ohms, then the results would be consistent with the observations. Thus, R3 is a possible fault. On the other hand, if no resistance value for R3 can be found that is consistent with the observations, then no fault in R3 alone can explain the symptoms. This approach has two major disadvantages. First, the technique is extremely slow (even for 1992 machines) and does not scale well — the student doesn't want to wait forever to have his question answered. Second, this approach is incapable of explaining why a component is or is not faulted in terms the student is familiar with. Thus, it is of limited pedagogical value.

The measurement critiquing module is built on the fault identification module. If a measurement did not significantly reduce the number of possible faults, then the measurement is a poor one (but see discussion in Section 2.4). This module suffered from similar problems as the fault identification module. First, it is too computationally intensive. Second, it is incapable of explaining why a particular measurement is good or bad. Third, if the measurement was bad, then it is incapable of proposing a better one for the student to make.

1.1 SOPHIE III

In building SOPHIE III we decided to adopt a completely different approach to achieving the required diagnostic inferences. Our approach was to base its inference techniques on those that we observed experts and students using. This allows SOPHIE III to provide explanations in terms the student is familiar with. By making SOPHIE III's inferencing strategies more akin to those used by the student, we could begin to determine which deductions the student was using, construct a model of his abilities, then use this model to generate explanations in familiar terms. For example, SOPHIE III might discover that the student repeatedly made measurements of both the current and the voltage through resistors; this would be evidence that the student did not completely understand Ohm's Law.

We also needed a fundamentally different and more human-oriented inference scheme because we wanted to investigate using SOPHIE III as a computer-based consultant for on-the-job training as an intelligent job-performance aid. We thus wanted SOPHIE III to be able to work from measurements being performed on *real, physical equipment*, completely independently of any kind of circuit simulator.

It is important to note that this paper is not about pedagogy or cognitive science. Although we constantly refer to explanation and critiquing capabilities, this does not at all suggest that the student should be immediately critiqued or interrupted — that decision is the job of the coach. What we want to provide is a general inferential framework which is powerful enough to make the necessary inferences and critiques — some other module based on some other theory makes the ultimate decisions on whether the student should or should not be interrupted and what should be said to the student with relatively few measurements.

The remainder of this paper describes the inferential machinery necessary to achieve the diagnostic objectives. The inferential machinery should:

- determine what other voltages and currents follow from the measurements and be able to explain these conclusions in terms familiar to the student.
- determine the diagnostic consequences of a measurement (i.e., the component faults that explain the symptoms) and to provide explanations for these conclusions.
- given whatever the measurements the student has already made, rate the measurements that the student can now make and provide explanations for these ratings. Notice that by repeatedly picking the best measurement the system can diagnose the circuit without the student.
- be as circuit independent as possible. We don't want to face a large knowledge engineering task every time we want to troubleshoot a new circuit.
- be efficient and be able to provide its results to the coach in a timely manner.

Although the concepts apply to both AC and DC circuits, at present SOPHIE III only models DC behavior. This is adequate for modeling simple power supplies since they (except for switching regulators) can be understood almost entirely from the quiescent point of view. The circuits consist of resistors, diodes, zener diodes, capacitors, transistors, switches, potentiometers and DC voltage sources.

There is an important class of inferences we were not able to incorporate in SOPHIE III. In some situations, it is more informative to change the front panel settings and then to make a measurement, than it is to make another measurement under the current control panel settings. A student is always free to change the front panel controls of the IP-28. SOPHIE III's inferential powers are sufficient to draw the conclusions from the information garnered from various control settings. However, we did not develop good inferential techniques for initiating changes to the front panel controls.

The typical knowledge engineered system has the advantage that it controls the initiative (or it knows a priori what data will be given to it) on what strategies and actions to take. The major question for such systems is whether sufficient knowledge has been codified to handle all the cases that arise, while the question for SOPHIE III is whether a sufficient amount of knowledge has been codified to track any student. The student may make some very bizarre measurements and SOPHIE III has to be prepared to comment on them and make suggestions based on the information obtained by that potential poor measurement even though it would have never made that measurement if it had the initiative. Because the system does not have control over what information it must reason with, it needs a deep understanding of electronics and troubleshooting.

1.2 The architecture of the electronics expert

Symptomatic circuit behavior is caused by some component(s) failing to behave as it was designed to. The task of troubleshooting is to identify the failing component(s). The difficulty is that we usually cannot directly examine components to see whether they are faulted, and instead must reason indirectly about their behaviors. Troubleshooting proceeds by making measurements in the faulted circuit and garnering as much information as possible from the results. The local propagator (LOCAL) forms the basis for the electronics expert. It uses general knowledge of circuit laws to determine what further voltages and currents can be determined from the measurements that have been made. Because the local propagator knows, for example, Ohm's Law, it can deduce given the current through a resistor what the voltage across it must be. In order for a measurement to provide useful diagnostic information we must have some expectation about its value. If nothing is known about the value, then no information is gained by the measurement.

Consider the Ohm's Law example again. Having measured the current through the resistor, Ohm's Law tells us what the voltage across it should be. Having made this calculation it now makes some sense to make the measurement of the voltage across the resistor. If this voltage is what we expected, then we can be reasonably confident that there is nothing wrong with the resistor. If this voltage is different than expected, then we know for certain that something is wrong with the resistor. SOPHIE III's electronic expert is designed around this fundamental idea that confirmations and refutations of predictions pinpoint the faulty components. LOCAL consists of two interacting modules: (1) a prediction module which predicts what other circuit quantities are known, given the observations so far, and (2) an interpretation module which determines (using local propagation of constraints — hence the name LOCAL) of the diagnostic consequences of confirmations or violations of predictions.

If some component is not behaving as it should, then the prediction module will eventually encounter a irreconcilable contradiction since its model of the circuit will differ from the actual faulty instance that is being debugged. Every inference the prediction module makes implicitly involves some assumption (the component is behaving as specified by its manufacturer) about a component or piece of circuit wiring. In troubleshooting, these assumptions must be made explicit. For example, given the voltage across a resistor, the current through it can be easily deduced by assuming that its resistance is as specified—that is not faulted. A contradiction is an informative event since it indicates which assumptions are violated, narrowing the field of possible faulty components.

The cost incurred by this elegant scheme is that the inference mechanism of the prediction module must always supply accurate and exhaustive justifications and assumptions for all of its deductions. A single exception could render the troubleshooter impotent for some classes of faults. To some extent this same organizational cleanliness is also demanded for generating coherent explanations, but the added constraint of this troubleshooting scheme now necessitates that the prediction module's deductions be void of any hidden presuppositions.

A major complication stems from the fact that electronics (and electronic troubleshooting) is a complex problem domain, part of which has been formalized, part of which has not, especially in terms of the causal calculii tacitly used by human experts. Because of the complexity, we were faced with either restricting ourselves to formalized aspects of the domain or working out a framework to systematically include the collection of ad hoc rules and inference mechanisms needed for the poorly understood part of the domain. We chose the latter course, and our strategy for building SOPHIE III was to encode as much of the knowledge in the most general form possible.

The circuit-specific rules bridge between the prediction module and the interpretation module. For example, one rule is: "If the output voltage is low, then one of ... must be faulted." The explanations of the circuit-specific knowledge are supplied by the electrical engineer and are attached to each rule. In contrast, explanations for deductions made from the generic electronic laws do not presume any particular circuit and thus can be constructed from the steps taken by the inference mechanism used to make the deduction.

The circuit-specific knowledge is aimed directly at those situations in which the general knowledge fails. Because of its known limited context, circuit-specific knowledge can be put in a canonical form, the number of ad hoc rules can be minimized and the determination of whether enough circuit-specific knowledge has been included to succeed is made much easier.

The circuit-specific knowledge is organized around a structural decomposition of the circuit. A circuit is a designed artifact, consisting of a collection of weakly interacting modules. Each module is, itself, a circuit in its own right and can be likewise decomposed. The modules behave cooperatively to produce the behavior of the overall circuit. The key to the circuit-specific knowledge is having terms to express the behavior of these modules. The circuit-specific knowledge consists of rules about how the behaviors of modules affect the behaviors of other modules: Neighboring modules affect each other as well a being influenced by the behavior of their lower-level, constituent modules.

Our objective in designing and using the circuit-specific rule system was *not* to facilitate building rule bases for new circuits — although that was certainly a subgoal. Our overall objective was to develop a thorough enough theory of diagnostic reasoning to make LOCAL powerful enough to diagnose circuits without using *any* circuit-specific rules at all. Therefore, we added rules only when absolutely necessary.

We obtained a test suite of about 1000 troubleshooting scenarios of students using SOPHIE I and SOPHIE II. For each fault and at each step of each of the scenarios we used the brute force algorithm of SOPHIE II to identify the faults which explained the symptoms. Using this database as a gold-standard, we ensured that SOPHIE III was able to achieve the same, perfect, diagnostic precision. For every one of the 50 rules required to perform the necessary diagnostic inferences on the IP-28, there exists a scenario for which a fault mode will fail to be eliminated without its presence.

Over the course of the project the number of circuit-specific rules both increased and decreased. We constantly analyzed the set of circuit-specific rules for patterns of inference which could be supported by extending LOCAL. Whenever that occurred, the number of circuit-specific rules dropped dramatically. Other times we would discover some pathological measurement order that some student used which defeated LOCAL's inferential capacity. Then additional circuit-specific rules would be added.

If the goal of SOPHIE III had solely been to build a system to diagnose the IP-28, then the rule set could be reduced to about a dozen. The additional rules are required to glean information from sequences of measurements that an optimal troubleshooter would never make (and therefore not needed in the core dozen). Thus, if SOPHIE III were able to control which measurements to make, then these rules would have been unnecessary.

In order to make it simpler to draw diagnostic inferences, SOPHIE III makes a number of presuppositions:

1. The circuit contains a single fault.

2. Faults only occur in components; not in circuit topology.

3. All component fault modes are known.

4. The circuit is non-intermittent.

5. If a global symptom is observed (e.g., IP-28 output is lower than its front panel controls indicate), then this is caused by some component *presently* manifesting a symptom.

6. Only DC behavior is important.

7. All faults are equally likely.

8. All measurements are equally easy to make.

9. All fault modes are equally likely.

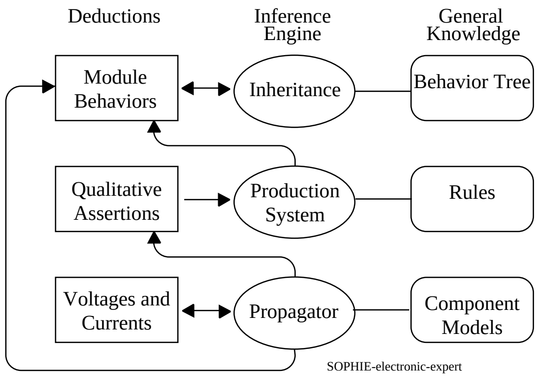


Figure 2: Electronics Expert

As SOPHIE III was primarily used in a simulated laboratory setting, we could control the situation such that these presuppositions are almost never violated. In real circuits, these presuppositions are easily violated. Some of these presuppositions are re-examined in more detail in Section 6.1.

Three types of reasoning are involved in the electronics expert each with its own knowledge structure, complete with inference mechanism and database (see Figure 2). The propagation database contains quantitative voltage and currents (e.g., output voltage is 30 volts) operating on by the propagator using the models of the components. The qualitative database contains assertions about the operating regions of the components, voltages, and currents (e.g., output current is low, transistor Q5 is off). The database for the third knowledge structure, the behavior tree, consists of the possible behavioral modes of components and circuit modules (e.g., R5 is open, the current source is anemic).

The fundamental problem of intercommunication between different reasoning types is elegantly solved in SOPHIE III with a common language of justifications and assumptions: Each deduction made by any of the reasoning types simply records the reasons for and assumptions under which the deduction was made. This justification/assumption database, just as in a general-purpose truth-maintenance system, can be oblivious to the different kinds of reasoning that underlie each deduction step.

2 The local propagator

The local propagator may appear simple, but the variety of subtle problems with which it must grapple make it quite complex. The biggest obstacle is the necessity for generality — it must deal with arbitrary measurements in arbitrary circuits. But the profit we gain is great; because it is the only part of SOPHIE III that has to reason upon the measurements and circuit topology directly.

The propagations caused by a single measurement can be quite deep. Con-

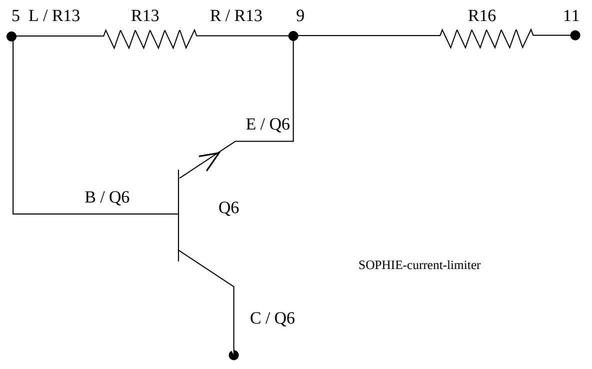


Figure 3: A Current Limiter

sider the circuit described by Figure 3. Suppose we measured the current in R13 to be 1 milliampere. Since the resistance of R13 is set at 100 ohms, the voltage across it must be 0.1 volts. But the voltage across R13 is the same as the voltage across the base-emitter junction of Q6. We know that a silicon transistor conducts no current when the base-emitter voltage is less than .6 volts so the current flowing in each of the transistor terminals must be zero. Now consider node N9 (we precede every node by "N", so the node labeled "9" on schematics is referred to as N9) which connects to R13, Q6 and R16. The current flowing out of R13 and Q6 must be flowing into resistor R16 so the current flowing through R16 is 1 ma. As just shown, a single measurement might lead to the calculation of many other circuit values. LOCAL formalizes this "If $A = x$, then $B = y$" kind of reasoning. It keeps records of each deduction so that explanations can be generated and examined when troubleshooting.

Each component has an expert associated with it which constantly checks to see whether the currents or voltages on its terminals are measured or calculated; as soon as a value is discovered for one, it is used to deduce voltages and currents on the others. Since the terminals and nodes are shared with other components, these new currents and voltages in turn trigger other component experts. The process continues until no new information can be determined.

Each propagation is recorded by constructing a justification which lists the antecedents and a detailed description of the propagation itself. These descriptions are kept in the following form:

```
(<type> <location> <reason> <assumptions>) = <value>.
```

`<type>` is either "`V`" or "`I`." `<location>` (for location) is a pair of nodes for a voltage, a terminal for a current. The circuit consists of components whose terminals are joined at nodes. Since terminals, unlike nodes, are always attached to components we adopt the convention of labeling them by `<terminal-type>/<component>`. Currents are normally associated with terminals, voltages with nodes. In Figure 3, L/R13, R/R13, B/Q6, E/Q6, and C/Q6 are terminals, and N5 and N9 are nodes. `<reason>` describes how a component expert computed the propagation and its form varies by component type. `<assumptions>` is a list of components which must be working correctly to produce this propagation. This list is constructed by adding the component (if any) to the union of the assumptions of all the antecedents to the propagation.

The resistor, one of the simplest component types, obeys Ohm's Law. If LOCAL determines the voltage across the resistor from the current through it (via $v = iR$), then the `<reason>` for the propagation is the pair `(RESISTORI <resistor>)`. Conversely, if the current is determined from the voltage (via $i = \frac{v}{R}$), then the pair is `(RESISTORV <resistor>)`.

Our example (Figure 3) proceeds as follows:

```
(I R/13 (MEASUREMENT) ()) = .001
```

The current in terminal R/R13 is measured to be 1 milliampere.

```
(V (N5 N9) (RESISTORI R13) (R13))=.1
```

The voltage across the resistor is determined from the current through it by an instance of Ohm's Law (abbreviated RESISTORI). This propagation assumes the resistance of the resistor (set at 100 ohms) has not changed and thus the assumption R13 is added to the `<assumptions>`.

The next propagation is (we discuss transistors in more detail later):

```
(I E/Q6 (TRANOFF Q6) (Q1 R13)) = 0
```

The voltage across the base-emitter junction of Q1 indicates it is off and therefore conducting no current.

The simplest propagations are those involving Kirchoff voltage and current laws. Kirchoff's current law states that the sum of all the currents flowing into a node must be zero. Therefore, if all but one of the terminal currents of a component or node are known, then the remaining one can be deduced. Therefore, in our example we can now propagate (Kirchoff's voltage law propagations are given the `<reason>` `(KCL <component-or-node>)`):

```
(I L/R9 (KCL N9) (R13 Q6)) = -.001
```

The current in terminal L/R9 is determined to be -.001 milliamperes by using Kirchoff's current law (abbreviated KCL*) to node N9.*

```
(V (N9 N11) (RESISTORI R16) (R16 R13 Q6)) = .0005
```

Kirchoff's voltage law states that the sum of the voltage differences around a collection of nodes must be zero. This allows the deduction that if two voltages are known relative to a common point, the voltage between the other nodes can be computed. Kirchoff's voltage law propagations are given the `<reason>` `(KVL <node1> <node2> <node3>)`. For example:

```
(V (N5 N11) (KVL N5 N9 N11) ()) = .1005
```

The voltage from node N5 to node N11 is calculated by Kirchoff's voltage law (abbreviated KVL*) by summing the voltage from N5 to N9 with the voltage from N9 to N11.*

LOCAL's propagation machinery is implemented by a collection of experts one for each component type as well as ones for KCL and KVL. Every propagation step which introduces no new assumptions is executed immediately. Thus,

Component	**Operating regions**
Diode	ON, OFF
Zener Diode	ON (at breakdown), OFF
Transistor	ON, OFF, SATURATED-ON, SATURATED-OFF

Table 1: Component operating regions.

KVL is immediately executed whenever any new voltage is discovered. Likewise, KCL is immediately executed whenever any new current is determined. The component experts are invoked in a breadth-first manner with the use of a queue. Thus, the response to a new measurement is as follows. First, KCL or KVL is run on the new measurement. Then, the new measurement and any deductions which follow from KVL and KCL form the initial queue. At each cycle of the propagation, LOCAL removes a propagation from the queue and invokes the applicable component experts on it. If any of these experts produce new currents or voltages, then KCL and KVL are immediately invoked and the propagations placed on the queue. This propagation cycle continues until the queue is empty[2]

2.1 Component experts

LOCAL has an expert for each type of component it models. These experts are based on the mathematical electrical engineering models, but are implemented with the propagation scheme outlined in the previous section. Each component of a given type must be modeled in the same way. If, for example, one transistor in the circuit were modeled differently from the others without there being some significant physical difference in the transistors, then the modeling would have presumed the functionality of the overall circuit. Presuming the functionality of a circuit is particularly dangerous for a troubleshooting system because a fault may so modify the circuit as to significantly change its functionality. Presupposing functionality would also bias LOCAL's deductions to assume the circuit functioned correctly. Of course, different components will be connected to different nodes and thus respond to different voltages and currents, but given these differences and individual differences in component parameters (e.g., the resistance of a resistor) each component must modeled by the same prototypical expert.

Transistors, diodes, and zener diodes have multiple operating regions. Each region is characterized by a distinct behavior and the region usually needs to be identified before any values can be propagated. Table 1 enumerates the possible

[2]The actual implementation needs to be careful to avoid duplicate propagations. For example, any propagation produced by a component expert should not immediately be applied to the same component as the result is guaranteed to be redundant. LOCAL enforces a more general version of this condition by not allowing any component expert to run if any of the triggering propagations depend on the component already (this is easily checked through a membership test on the `<assumptions>`).

component regions. Note that this model of a transistor does not completely conform to the traditional one which has only one SATURATED mode, but it makes certain deductions more convenient. ON and OFF are determined from collector current only. SATURATED is determined from the collector-base voltage only. While ON and OFF are exclusive states, a transistor can be SATURATED and ON, or SATURATED and OFF.

The diode is the simplest kind of semiconductor device. Its model is very simple: when it is reverse-biased (current can flow in only one direction through a diode), the current through it must be zero:

```
(I D (DIODEV) (D)) = 0.
```

From the zener diode we know that if the current through it is greater than some threshold, then the voltage across it must be at its breakdown voltage:

```
(V Z (ZENERI) (Z)) = ...
```

Conversely if the voltage across the zener diode is at less than its breakdown voltage, then the current through the diode must be zero:

```
(I Z (ZENERV) (Z)) = 0.
```

The transistor is the most difficult component to model. This is both because it has the discontinuous characteristics of a semiconductor component, and because it is a three-terminal component. If the current through any one of the terminals is known, then the current through the other two can be determined using the gain characteristics of the component:

```
(I C/Q1 (BETA Q1 B/Q1) (Q1)) = ...
```

The collector current of Q1 is deduced by applying its gain to the base current ($i_C = \beta i_B$)

Furthermore, if the voltage across the base-emitter junction is less than some threshold (.55 volts for silicon transistors), then the current flowing through any of its terminals should be zero:

```
(I C/Q1 (TRANOFF Q1) (Q1)) = ...
```

Consider the fragment of the IP-28 illustrated in Figure 4. Suppose voltage measurements at the output and across D5 have just been made:

```
(V (N15 N14)  (MEASUREMENT) ()) =            30
(V (N16 N14)  (MEASUREMENT) ()) =            34
```

The following propagations ensue. For brevity, currents in two-terminal components are simplified to refer to the component instead of the terminal.

```
(V (N16 N15)  (KVL N16 N14 N15) ())   =      4
(I R5         (RESISTORV R5)    (R5)) =      0.003
(I D5         (ZENERV D5)       (D5)) =      0.
```

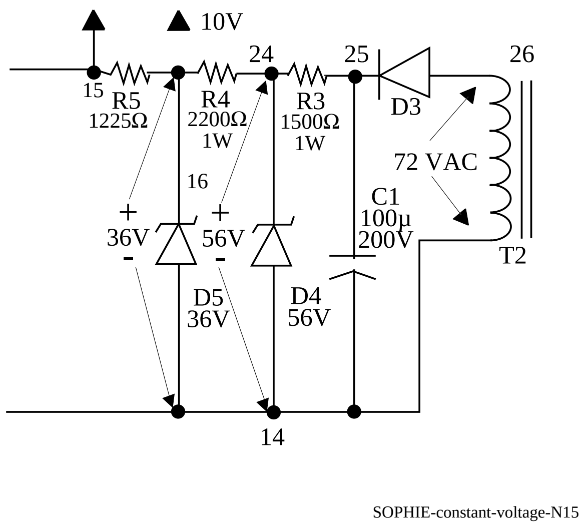


Figure 4: Constant Voltage Reference

The voltage across the zener D5 is less than its breakdown, therefore the current through it must be zero.

```
(I R4       (KCL N16)        (R5 D5))      =0.003
(V (N24 N16)(RESISTORI R4)   (R4 R5 D5))   =7.18
(V (N24 N14)(KVL N24 N16 N14)(R4 R5 D5))   =41.18
(V (N24 N15)(KVL N24 N16 N15)(R4 R5 D5))   =11.18
(I D4       (ZENERV D4)      (D4 R4 R5 D5))=0.
```

The voltage across zener D4 is less than its breakdown.

```
(I R3       (KCL N24)        (D4 R4 R5 D5))   =0.003
(V (N24 N25)(RESISTORI R3)   (R3 D4 R4 R5 D5))=4.90
(V (N25 N14)(KVL N25 N24 N14)(R3 D4 R4 R5 D5))=46.1
(V (N25 N16)(KVL N25 N24 N16)(R3 D4 R4 R5 D5))=12.01
(V (N25 N15)(KVL N25 N24 N15)(R3 D4 R4 R5 D5))=16.01.
```

More will be said concerning the component experts later.

2.2 Coincidences, conflicts and corroborations

In addition to simple propagation, two other activities are required to troubleshoot. *Passive troubleshooting* interprets the results of the propagations to draw conclusions about the correctness of components. *Active troubleshooting* chooses new measurements to make. Passive troubleshooting knowledge is incorporated into LOCAL as an extension of the propagator. Active troubleshooting is discussed in Section 2.4.

The discovery of a value for a circuit quantity for which we already know a predicted, propagated value is called a *coincidence*. When the two propagations have the same values, we call the coincidence a *corroboration*; when they differ we call it a *conflict.* Coincidences provide information about the assumptions made in the propagation: Corroborations verify them and conflicts indicate at least one of them is in error.

Continuing the example of the previous section, consider what we would learn if we measured the voltage between nodes N15 and N25 to be 20 volts instead of the predicted 16.01 volts. This conflict indicates that one of the assumptions underlying the prediction is violated: One of (R3 D4 R4 R5 D5) is faulted. (Sets of assumptions underlying conflicts are called *nogoods.*) Suppose we instead measured the voltage between N15 and N25 to be the predicted 16.01 volts. In this case we can be assured that R3, D4, R4, R5 and D5 are working correctly. (Sets of assumptions underlying corroborations are called *goods.*)

Although a component is faulted, the overall system may still be functioning correctly. (The fault may only manifest itself under certain load conditions, for example.) Therefore a corroboration provides no information if the system is not manifesting a symptom under the specified external conditions (e.g., load and control settings). To facilitate this deduction, LOCAL maintains a special flag which indicates whether the circuit is violating its overall function.

Any troubleshooting scenario produces a number of conflicts and coincidences. Each of the conflicts leads to a nogood, and each of the coincidences leads to a good. Under the presupposition that the circuit contains at most one fault, each component not mentioned by any nogood must be unfaulted. This observation reduces the set of possibly faulty components to the intersection of all the nogoods. In addition, any components appearing in any corroboration must also be removed from this final set. Equivalently, the set of possibly faulty components is the intersection of all the nogoods and the complements of all the goods. Intuitively, a component is possibly faulted if its removal from the circuit would remove every conflict while maintaining every corroboration. When there are multiple faults these concepts must be generalized. Although a nogood indicates that at least one of its assumptions is faulted, the other faults may lie outside the nogood and cannot be immediately verified. Corroborations involving only one unverified component always guarantee the component is ok. However, corroborations involving more than one component may be caused by multiple faults canceling each other.

2.3 Representing propagations

So far we have been describing LOCAL as if every circuit quantity has only one propagation associated with it. However, LOCAL maintains a set of propagations for each circuit quantity. These propagations are distinguished by their value and their underlying assumptions. LOCAL does not select out one of the values to propagate further, but propagates all values. Two conventions help control the proliferation of propagated values. First, if the assumptions underlying a propagation are a superset of a nogood, then that propagation's value is known to be invalid and is discarded. Second, if two propagations have identical values and the assumptions of one propagation is a proper subset of the other, then the propagation with the larger assumption set is discarded.

One of the consequences of these observations is that there can be relatively many coincidences. Suppose some measurement produces a conflict with a long propagation chain. After the conflict has been processed, the measurement will likely propagate to produce another conflict earlier in the long propagation chain. These coincidences between propagations are usually redundant and produce no new information. However, there are a few important circumstances where they do, so they all have to be processed. Because LOCAL does not use symbolic algebra but only propagates numeric quantities it is not logically complete. As a consequence of this incompleteness, a new measurement may not produce an initial coincidence, but propagations caused by it do (this is illustrated later in Table 3).

In the worst case, if a circuit contains n possible faults, then each circuit quantity can have up to n distinct propagations associated with it. For example, if the only possible faults are A, B, C and D, then a quantity could have 4 propagations with different values depending on the assumption sets: $\{A, B, C\}$, $\{A, B, D\}$, $\{A, C, D\}$ and $\{B, C, D\}$. The nogood produced by the union of any two of these sets is always the set of all faulty components, $\{A, B, C, D\}$, so provide no new information. Fortunately, this case does not commonly arise in practice. Usually there are no more than two propagations associated with each circuit quantity.

The effect of conflicts and corroborations between propagations is as follows. Suppose coinciding propagations A and B have underlying assumption sets P_A and P_B (e.g., the two propagations predict the same value for the same circuit quantity under differing assumptions). A conflict yields a nogood $P_A \cup P_B$. A corroboration yields the good $P_A \cup P_B - [P_A \cap P_B]$. The intuition is as follows. A fault lying in the intersection of the antecedent assumptions could result in both propagations being incorrect yet corroboratory so these cannot be verified. Consider the remaining assumptions. Suppose that some component $C \in P_A - P_B$ is faulted. Propagation B must be correct because it does not depend on C. On the other hand as C contributes significantly to propagation A, therefore its value must deviate significantly from the correct value. As the coincidence was a corroboration and not a conflict this cannot occur. Hence C is unfaulted. Notice that the combined effect of a corroboration between two propagations is to remove from either propagation any assumptions not in common. For example, if $\{X, Y\}$ underlies propagation A, and $\{Y, Z\}$ underlies corroborating propagation B, then X and Y are eliminated with a result that both A and B become essentially identical and depend on Y alone.

In the course of troubleshooting the student can change the front panel settings. As we presume the circuit is non-intermittent and the fault doesn't change, the nogoods and goods obtained from different control settings can be combined as usual. However, it is important to keep the propagations distinct. Therefore, for every front panel settings, LOCAL creates a dummy assumption which is added to every measurement made under those settings. Coincidences between propagations obtained under different settings are ignored, and dummy assumptions are removed from all goods and nogoods. However, for simplicity,

Measurement	Conflict Eliminates	Corroboration Eliminates	Score
`(V (N16 N15))`	impossible		0
`(I R5)`	R3,D4,R4,D5	R5	1.6
`(I D5)`	R3,D4,R4,R5	D5	1.6
`(I R4)`	R3,D4,R4	R5,D5	2.4
`(V (N24 N16))`	R3,D4	R4,R5,D5	2.4
`(V (N24 N14))`	R3,D4	R4,R5,D5	2.4
`(V (N24 N15))`	R3,D4	R4,R5,D5	2.4
`(I D4)`	D4,R4,R5,D5	R3	1.6
`(I R3)`	D4,R4,R5,D5	R3	1.6
`(V (N24 N25))`		impossible	0
`(V (N25 N14))`		impossible	0
`(V (N25 N16))`		impossible	0
`(V (N25 N15))`		impossible	0

Table 2: Proposed measurements

we do not discuss these dummy assumptions further in this paper.

2.4 Active troubleshooting

The goal of troubleshooting is to remove as many components from suspicion as efficiently as possible. Thus, the quality (score) of a potential measurement is the *expected* number of components that it would remove from suspicion. The higher the score, the better the measurement. LOCAL computes these scores for every circuit quantity having a propagated value. Suppose a circuit quantity has exactly one propagation. The expected value is the sum of two products: the number of components that would be verified by a corroboration times the probability of a corroboration, plus the number of components that would be verified by a conflict times the probability of a conflict.

In the following analysis we presume that we have determined the circuit has a symptom and that the faulty behavior is due to some, as yet undetermined, single fault. Suppose there are n components under suspicion and p unverified components underlying a particular propagation. We assume that every component is equally likely to fail. Therefore, the probability that the propagation is faulty is $\frac{p}{n}$ which will produce a nogood of size p thereby verifying the remaining $n-p$ components. The probability that the propagation is correct is $1-\frac{p}{n}$ which will produce a good of size p. Combining both contributions we find that the expected number of components verified by the measurement is:

$$\frac{p}{n} \times (n-p) + (1-\frac{p}{n}) \times p,$$

which simplifies to:

$$\frac{2p(n-p)}{n}.$$

The best possible score is always half the number of components currently under suspicion.

Suppose we have measured the current and voltage of resistors in series and determined that one of the resistors is faulted. The familiar half-split method would propose a measurement midway within the resistor network. Our scoring function, which is maximum at $p = \frac{n}{2}$, proposes the same measurement. Moreover, our scoring function applies even when the circuit has a complex topology for which it is not obvious how to apply the conventional half-split method. Notice that if there are always measurements available with $p = \frac{n}{2}$, then the fault is always isolatable in $log_2\ n$ measurements.

Consider the example of the previous section (Figure 4). The following is an simplified example of SOPHIE III troubleshooting this circuit fragment. Suppose that the only possible faults are R3, D4, R4, R5 and D5 and that the actual fault is that D4 has its breakdown voltage too low and thus is drawing a great deal of current. The possible points to measure and their scores are listed in Table 2.

The table shows the best measurement to make is one of `(I R4)`, `(V (N24 N16))`, `(V (N24 N14))` or `(V (N24 N15))`. The current through R4 is measured and produces a corroboration. Therefore R5 and D5 are verified to be correct and one of R3, D4 or R4 must be faulted. This leaves 5 interesting places to measure: `(V (N24 N16))`, `(V (N24 N14))`, `(V (N24 N15))`, `(I D4))`, or `(I R3)`. (For brevity we do not show to revised scores.) Suppose we measure the current in D4. As D4 is shorted it draws a great deal of current. Thus, this produces a conflict whose nogood is `(R4 D4)`. Thus R3 is verified. The two possible faults are now that R4 could be faulted high or D4 faulted low. Any of the measurements `(V (N24 N16))`, `(V (N24 N14))` or `(V (N24 N15))` isolate the fault to D4.

The objective is to localize the fault as quickly as possible *on average*. Notice that the student can make a very poor measurement which yields a lot of information. Consider an extreme example. Suppose there are 40 unverified components. Suppose one quantity has a propagation with 20 underlying assumptions and another quantity has a propagation with 1. Suppose the student makes the second measurement which turns out to be a conflict and thus instantly identifies the fault. The student should be critiqued for his poor measurement because it was very improbable that the outcome of this measurement would produce a conflict — far more likely it would have produced a corroboration which would only have eliminated one component. The student was just lucky, since the expected value of that measurement was low. The expected number of components eliminated by the first measurement is 20, while the expected number of components eliminated by the second measurement is 1.95.

The scoring function may yield a large number of good places to measure, many of which are topologically distant from the suspect components. This occurs because there can be long propagation chains through components which have been verified. Therefore, when SOPHIE III is asked by the student to propose a next measurement, it returns a measurement with highest score but which also depends on the fewest number of verified assumptions. The fewer the number of verified components the propagation depends on, the closer it is topologically to the suspect components and therefore more intuitive to the student.

A circuit quantity may have many propagations associated with it. The more propagations a circuit quantity has the better its score can be. Reconsider the previous example where a quantity has 4 conflicting propagations with assumption sets: $\{A, B, C\}$, $\{A, B, D\}$, $\{A, C, D\}$ and $\{B, C, D\}$. Measuring this quantity will immediately identify the faulty component as there is a one-to-one correspondence between faulty components and values. In general, in order to determine the score of a multi-propagation quantity, careful analysis of the assumptions underlying the propagations must be made to determine the degree to which the propagations are independent. However, there are many other effects which influence the score of a tentative measurement (see Section 4.2).

2.5 Primary and secondary assumptions

This strategy of exploiting conflicts and corroborations to localize the faulty component forms the essential core of LOCAL's troubleshooting strategy. Unfortunately, in practice, we have to make an additional caveat as this idealistic framework is inadequate.

The circuit may be exhibiting a symptom, but the way the faulted component causing the symptom is used in a particular propagation may not be significant. A component contributes significantly to a propagation if a fault in the component would produce a value sufficiently different from the one propagated that it would be detected. Therefore, LOCAL partitions the set of assumptions associated with each propagation into a *primary* and a *secondary* set. Primary assumptions contribute significantly, while secondary assumptions may or may not contribute significantly. Thus, when a corroboration occurs only the components named by the primary assumptions are eliminated. The components of the secondary assumptions cannot be eliminated because we cannot guarantee the value is sensitive to a fault in them. Therefore, LOCAL must ensure that assumptions are primary only if they significantly impact the propagation's value.

Unless overridden by some exception, every component expert's assumption is primary and the secondary(primary) assumptions of the propagation include the union of all the secondary(primary) assumptions of the antecedents. Some of the exceptions to this general rule are:

- When a large quantity is added to a small quantity the smaller quantity may vary significantly without affecting the sum. The primary assumptions of the smaller quantity are secondary in the sum.

- When a quantity is multiplied by zero, the assumptions of the non-zero quantity become secondary.

- In a propagation which uses the same component twice, two effects may cancel each other out. Therefore, whenever two antecedents share a common assumption, this assumption is made secondary for the ensuing propagation.

- The assumptions underlying a comparison must become secondary. If the base-emitter voltage of a transistor is below 0.55 volts, then the collector current is 0 amps. If the collector current is measured to be 0, this does not guarantee that the base-emitter voltage was correct because it could be wildly incorrect but just less than 0.55 volts. As we need to err on the conservative side, the assumptions must therefore become secondary.

In all cases the primary and secondary assumptions cannot overlap and all secondary assumptions are always removed from the primary set.

Consider a coincidence between a measurement and propagation. If the coincidence is a conflict, then the fault must lie in the union of the primary and secondary assumptions (the nogood set). If the coincidence is a corroboration, then all the primary assumptions are eliminated (the good set).

The general case is as follows. Suppose the primary and secondary assumptions underlying the two coinciding propagations are P_A, P_B, S_A and S_B. If the coincidence is a conflict, then we have a new nogood: $P_A \cup P_B \cup S_A \cup S_B$. If the coincidence is a corroboration, then we have a new good: $P_A \cup P_B - [S_A \cup S_B \cup (P_A \cap P_B)]$. The reasoning is analogous to that of the previous sections, except that secondary assumptions should never be verified by a corroboration. As a consequence of the newly discovered good, each of the corroborating propagations will propagate onward with their primary non-intersecting assumptions eliminated (just as before in Section section:Propagations).

It is important to control the proliferation of propagations as the introduction of the distinction between primary and secondary assumptions causes additional propagations. Consider a corroboration in which $P_A = \{X\}, S_A = \{Y, Z\}, P_B = \{Z\}, S_B = \{X, Y\}$. Without the distinction between primary and secondary assumptions one of these propagations would be discarded. Now, both must be propagated. To control the proliferation of propagations, LOCAL removes all propagations subsumed by others. For example, if $P_A \subset P_B \cup S_B$ and $S_A \subset S_B$, then propagation B is discarded because little additional diagnostic information can be obtained by propagating it further.

3 Ranges

All measurements in the circuit and all circuit parameters have some degree of error. The errors in circuit parameters originate from the fact that manufacturers cannot make perfect components and instead guarantee their specifications to within certain tolerances. (Typically the resistance of a resistor is only within 10 % of its specified value.) Similarly, the meter used to measure circuit quantities can only measure voltages and currents to certain accuracies (typically 2

```
  Location   Reason           Primary       Secondary
                              Assumptions   Assumptions
  (I C/Q2    (MEASUREMENT)    ()            ())          = [.00017 .00019]
  (I B/Q2    (BETA Q2 C/Q2)   (Q2)          ())          = [1.1E-6 3.8E-6]
  (I E/Q2    (BETA Q2 C/Q2)   (Q2)          ())          = [-.00019 -.00017]
  (V (N2 G)  (MEASUREMENT)    ()            ())          = [45 49]
  (I R9      (RESISTORV R9)   (R9)          ())          = [.012 .017]
->(I C/Q1    (KCL N2)         (R9)          (Q2))        = [.012 .017]
  (I B/Q1    (BETA Q1 C/Q1)   (Q1 R9)       (Q2))        = [8.1E-5 33E-5]
  (I E/Q1    (BETA Q1 C/Q1)   (Q1 R9)       (Q2))        = [-.017 -.012]
  (I R11     (KCL N3)         (Q1 R9)       (Q2))        = [-.00015 .00011]
  (V (N1 N3) (RESISTORI R11)  (Q1 R9 R11)   (Q2))        = [-.26 .18]
->(I C/Q1    (TRANOFF Q1)     (R11 Q1 R9)   (Q2))        = [-1.E-6 4.0E-5]
```

Table 3: Propagation trace for R11 high. The conflicting propagations are highlighted.

% error). A further limitation is that the meter has a minimum range; SOPHIE's meter cannot accurately measure below .1 volt or below 1 μA. These effects, though artificially introduced into SOPHIE, are representative of what is encountered in real circuits. Because the numerical computations performed by the component experts introduce truncation and roundoff errors, it makes more sense to propagate either values *and* their tolerances, or ranges of values. Consequently, all the basic arithmetic operations performed by LOCAL are modified to accommodate the tolerances associated with each circuit quantity.

LOCAL represents each quantity by a range indicating its extreme values (which may be $+\infty$ or $-\infty$). A quantity Q is described by $[Q_L, Q_H]$ which indicates $Q_L \leq Q \leq Q_H$. No probability distributions are maintained. Consider an example. If we know the current through a resistor is between 2.9 amperes and 3.1 amperes ($I = [2.9, 3.1]$), and its resistance is 100 ohms with a 10% tolerance ($R = [90, 110]$), then Ohm's Law tells us that the voltage across the resistor must be between 261 and 310 volts ($V = IR = [2.9, 3.1] \times [90, 110] = [261, 341]$).

Suppose R11 of the IP-28 has a higher resistance than specified. The resulting propagation trace is illustrated in Table 3. Each propagation includes two lists of assumptions; the first are the primary assumptions and the second the secondary assumptions. This propagation sequence illustrates two additional interesting properties. First, a coincidence arises between two propagations neither of which is a measurement. Second, an assumption becomes secondary because a small range is added to a large one. Careful analysis of the trace illustrates that this propagation would be much simpler if we added information about maximum base currents. We discuss this issue next.

By using range notation, a great deal of additional knowledge can be included in the component models. For example, knowledge that the current through a diode is always positive can be expressed by the range $[0, \infty]$. A silicon transistor in the ON state must have a base-emitter voltage of [.55 , 1]. Components have certain maximum limits which cannot be exceeded without destruction. For example, a current of 2 amperes through diode D6 well exceeds its rating, indicating that either it is shorted, or the 2 ampere propagation is incorrect and one of the underlying assumption used in its derivation has been violated. Suppose the base-emitter of a two-transistor darlington pair (such as Q3 and Q4 of the IP-28) measures greater than 2 volts. Neither of the individual base-emitter drops are known. If the database is initialized with a $[-\infty, .7]$ drop for each base-emitter junction the 2 volt measurement would conflict with the $[-\infty, 1.4]$ volt propagation and the problem will have been localized to the darlington. Before any measurement is ever made, LOCAL propagates all minimums and maximums. Thus, when a circuit quantity is discovered to exceed some limit, a conflict is triggered.

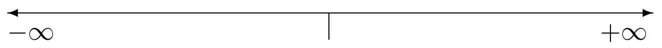
Ranges also admit a third kind of innovation. Sometimes we would like a component model to propagate a range with different explanations for each end of the range. This can be accomplished as follows. Instead of propagating $[a,b]$, we propagate two ranges: $[-\infty, b]$ and $[a, +\infty]$ each with its own justification.

If all voltage and current sources in a DC circuit are known, then absolute limits for all voltages and currents can also be determined from basic principles. For example, suppose the current through a 2 MΩ resistor is propagated to be .5 ma. The resistor model propagates the voltage across it as 1000 volts by Ohm's Law. No problem is detected since the resistor can operate at 1000 volts and the .5 watts that it is dissipating falls within its rating. If this resistor were part of a low-voltage power-supply such as the IP-28, then there could be no possibility that any voltage inside of it was 1000 volts. If this measurement turned up either the resistor would be shorted or the propagation of the .5 ma would be incorrect.

Primary and secondary assumptions play the same role as before. The basic intuition is that an assumption p is primary to a propagated value with range v, if every fault in p causes the actual circuit value to be outside of v. As we see in a moment, this distinction is important because if a later measurement is within v, then we can be guaranteed that p cannot be faulted. Thus, an assumption is primary if $|\frac{\partial v}{\partial p}| > \frac{w_v}{w_p}$ where w_v is the width of v's range and w_p is the tolerance of the component. As this condition is difficult to guarantee, LOCAL always errs to the conservative side, making assumptions secondary unless it can be definitely determined that they are primary.

3.1 Consequences of ranges on coincidences

A comparison between ranges A and B can have one of five outcomes (see Figure 5): (1) values conflict, (2) values corroborate, (3) A splits B, (4) B splits A, and (5) no comparison possible. The comparison procedure first attempts to determine whether the ranges corroborate. A tolerance for the comparison is computed by choosing the minimum width if the widths are very different and choosing half the width if the widths are approximately the same. If the width of either range is ∞, then the coincidence cannot be a corroboration. Depending on the circuit and whether the coincidence is between voltages or currents, a minimum tolerance is chosen. The minimum tolerance for a typical circuit is .1 μA and .1 volts. Then the differences between the corresponding



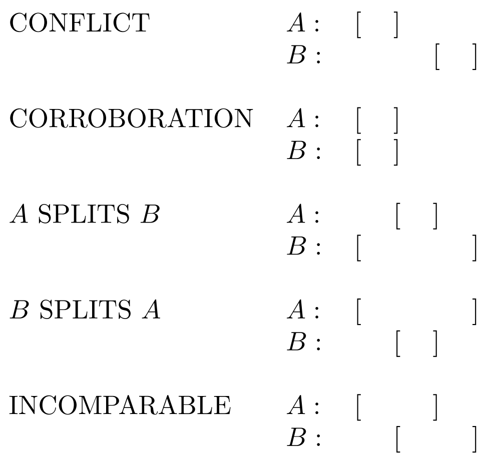


Figure 5: Range Comparisons

ends of the ranges are determined. If both differ within the tolerance, then the values are determined to be corroboratory. For example, [.1 , .2] volts and [.15 , .3] volts are judged to be corroboratory. If only one side is within tolerance the tolerance is relaxed by 50% and the failing side is checked again. This ad hoc rule worked well for all our examples.

If this still does not match, then we cannot really claim a corroboration; instead we can only say that one value splits the other. For example, [0 , 1] splits [0 , 10]. The two remaining cases occur when the values are completely disjoint (e.g., [0 , 1] and [3 , 4]) and when one contains the other (e.g., [0 , 6] and [3 , 4]). The containment case is treated as a split. Ranges are considered disjoint only if they differ by greater than the tolerance. If none of these conditions are met, then the coincidence is neither a corroboration nor a conflict. For example, [0 , .1] volts and [.2 , .3] neither contradict nor corroborate.

Conflicts and corroborations produce the same goods and nogoods as in the non-range case. We can also draw useful conclusions from the splitting case. Let P_A, P_B, S_A, S_B the primary and secondary assumptions of A and B. If A splits B, then we have the new good $P_B - [P_A \cup S_A]$. The rationale is as follows. If the range for A splits the range for B and therefore are subsets, then if A is valid (i.e., must include the actual circuit value), then B must be valid. For example, since A:[3 , 4] splits B:[0 , 10], the validity of A implies the validity of B. Consider some possibly faulted component of P_B. By definition, this will cause the actual value of B to be outside of its current range, and of necessity, conflict with A's current range. If C doesn't contribute A, then by the single-fault assumption A's range is correct. Therefore, any failure in C would have caused a conflict and not a split and therefore C cannot be faulted. Notice that the notion of corroboration is now redundant — a corroboration of A with B as the same effect as simultaneously splitting A with B and B with A.

When the coinciding values are incomparable, LOCAL synthesizes a new propagation which combines them. The range of the synthesized propagation is the intersection of the two ranges (if the intersection is empty, the new propagation is not constructed). The primary assumptions of the syntesized propagation are the union of the primary assumptions of the antecedents. The secondary assumptions of the synthesized propagation are the union of the secondary assumptions of the antecedents, with the primary assumptions removed. This synthesized propagation is usually propagates further than its antecedents as it has a smaller width.

As ranges allow one to express far more information than before, there are now typically many propagations per circuit quantity. LOCAL incorporates a variety of tactics to reduce the number of propagations. Propagations with very wide ranges are arbitrarily stopped. New values which are only marginally better (only marginally narrower ranges) than old ones are ignored since they probably are the result of iteration. Kirchoff's Voltage Law produces so many propagations and voltages between nodes of no interest that all KVL deductions are performed by a separate module which in one step produces the best value for all interesting voltages. Finally, subsumed propagations are discarded. Propagation A subsumes propagation B if the primary and secondary assumptions subsume each other as in the non-range case, and if the range of propagation A corroborates or splits the range of propagation B. Subsumed propagations are removed after all the fault information has been extracted from the corroboration.

3.2 The model for a diode

Although every component of a given type behaves in the same basic way, the fine details of its behavior are controlled by its parameters. These parameters are determined by the manufacturer, not by the component's use in the circuit. For example, every resistor obeys Ohm's Law, but with varying resistance. Each of LOCAL's component models is complicated and lengthy and there is no point in giving every one of them in this paper. Instead, we only describe the diode as it is the simplest non-linear device. The transistor model, in contrast, is dramatically more complicated.

The model for the diode is specified by eight parameters, each parameter is listed with its value for D6 of the IP-28:

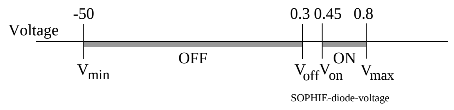


Figure 6: Diode Voltage Parameters

I_{MIN} : maximum reverse current flow, $-1\ \mu$A.
I_{MAX} : maximum forward current flow, 1 A.
V_{MIN} : minimum voltage across the diode, -50 V.
V_{MAX} : maximum voltage across the diode, .8 V.
I_{OFF} : defines the diode OFF state, 1 μA.
I_{ON} : defines the diode ON state, 2 μA.
V_{OFF} : defines the diode OFF state, .3 V.
V_{ON} : defines the diode ON state, .45 V.

$I_{MIN}, I_{MAX}, V_{MIN}, V_{MAX}$ are all limits beyond which D6 would be destroyed. $I_{OFF}, I_{ON}, V_{OFF}, V_{ON}$ help specify the operating regions of the diode.

The maximum limits are propagated before any measurement is ever made. Thus, LOCAL immediately propagates $[I_{MIN}, I_{MAX}]$ with secondary assumption D6. If the current is propagated (or measured) to lie outside this range, then a conflict is triggered. For example, this allows LOCAL to deal with diodes in series. If the voltage across two diodes is measured to be greater than $2V_{MAX}$, for example, LOCAL immediately detects that one of the two diodes is open. Note that a corroboration can never verify D6, both because it is only a secondary assumption of the initial propagation and the range widths are too wide.

If a newly discovered voltage is less than the threshold required to turn on the diode, then the diode cannot be conducting much current; on the other hand, if the new voltage indicates the diode is on, then the diode must be conducting a significant amount of current (see Figure 6). As the maximum and minimum limits imposed by the diode are propagated before any measurement is taken, we do not include them in these models:

A new voltage $V = [V_L, V_H]$ causes the propagation: If $V_H \leq V_{OFF}$, then propagate the range $I = [-\infty, I_{ON}]$, otherwise if $V_L \geq V_{OFF}$, then propagate the range $I = [I_{ON}, +\infty]$.

(Note that we could not even express this rule without ranges.)

The propagations which result from a new current are analogous (see Figure 7):

A new current $I = [I_L, I_H]$ causes the propagation: If $I_H \leq I_{OFF}$, then propagate the range $V = [-\infty, V_{OFF}]$, otherwise if $I_L \geq I_{ON}$, then propagate the range $V = [V_{ON}, +\infty]$.

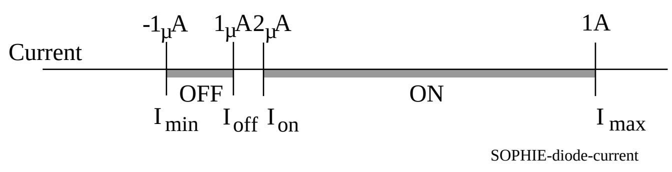


Figure 7: Diode Current Parameters

Type	**Fault Modes**
RESISTOR	open, shorted, high or low
CAPACITOR	shorted or leaky
DIODE	open or shorted
ZENER DIODE	breakdownhigh or breakdownlow
TRANSISTOR	ok/op/op, op/op/ok, op/ok/op, op/op/op, op/sh/op, op/op/sh, sh/op/op sh/sh/sh, beta-low or beta-high.

Table 4: Table of common fault modes.

This model has a number of shortcomings. It could be more accurate. For example, in the active (on) region, although the voltage-current relationship is roughly linear, the model only propagates the region's extremes.

3.3 Comparison to other propagation schemes

LOCAL is similar to Stallman and Sussman's EL which is also based on propagation of constraints [8]. LOCAL differs from EL in three fundamental ways. First, since the task is troubleshooting we need to make assumptions about the faultedness of components, not just about operating regions of components. Hence our nogood sets (which represent inconsistent sets of assumptions) include assumptions about faultedness. Second, since values are known only approximately, the quantities propagated are ranges instead of numbers. Third, since the quantities are ranges, two quantities can split or overlap; one of them can be "better" or "worse" than another; and like EL they can be equal and unequal. One of the consequences of incomparable (e.g., overlapping) ranges and the possibility of differing underlying assumptions for the same circuit value is that LOCAL must often propagate multiple values for the same circuit quantity which leads to substantial complexity. For instance, the kind of assumption made about the faultedness of a component depends on the ranges of the quantities being propagated through the component. We discuss this in more detail in the following sections. These generalizations precluded LOCAL from propagating variables (known as anonymous objects in EL).

Mode	**Explanation**
ok/op/op	collector is open, but BE junction is functional.
op/op/ok	emitter is open, but BC junction is functional.
op/ok/op	base is open, but EC junction is functional.
op/op/op	all terminals open.
op/sh/op	base is open, and the emitter is shorted to the collector.
sh/op/op	collector is open, and the base is shorted to the emitter.
op/op/sh	emitter is open, and the collector is shorted to the base.
sh/sh/sh	all terminals are shorted with each other.

Table 5: Transistor fault modes.

4 Fault modes

So far we have placed almost no restrictions on how components can fail. Nevertheless, the framework of coincidences, conflicts and corroborations provides a very general framework for eliminating components from suspicion. However, physical structure dictates that each component can only fail in a small number of characteristic ways. Knowing these fault modes greatly increases LOCAL's diagnostic power. For example, if none of the possible fault modes of a component are consistent with the observations, then that component cannot be faulted. This principle is at the heart of this section.

The fault modes for components are enumerated in Table 4. The transistor has so many possible fault modes that it is impossible to choose succinct labels for each. The modes are listed in the form EB/EC/BC where each pair refers to a transistor "junction" which is one of three modes: (1) OK, the junction is working correctly, (2) SH, the junction is shorted, or (3) OP, the junction is open. The possibilities are explained in Table 5.

Each fault mode can be described by a distinct model. We could introduce an assumption for each possible fault mode and generalize our theory of conflict and corroborations extract fault mode information. However, this is computationally infeasible. Instead, LOCAL uses explicit fault models, but exploits a set of new inferential mechanisms to exploit them. Thus, the assumption sets underlying propagations remain exactly as they were before. We presume that if a component is faulted, then it remains in the same fault mode throughout the entire troubleshooting scenario.

LOCAL associates a list of possible fault modes with each component. (In order to be able to later explain its deductions, LOCAL also records the reasons for each inference it makes about fault modes.) If all of the possible modes of a component are eliminated, then that component is considered verified (i.e., a singleton good). On the other hand, if a component is definitely determined to be in a particular fault mode, then that component is the fault (i.e., a singleton nogood). Knowledge of fault modes usually does not improve the goods or nogoods produced by any individual coincidence. Rather, information gleaned from combinations of coincidences now combine non-linearly. For example, one conflict may indicate that if the resistor is faulted, then it must be high or open while another indicates that if the resistor is faulted, then it must be low or shorted. The combination of these two conflicts indicate the resistor is unfaulted.

LOCAL incorporates three fault mode detection strategies: teleology, mode consistency, and qualitative propagation. We describe each in turn.

Teleology. If a particular component is faulted and the overall circuit is manifesting a symptom, then the faulty component must be manifesting a symptom. It follows that, if a component is not displaying a symptom of a known fault mode, then it isn't in that fault mode. For example, if a diode is open and causing a circuit symptom, then the voltage across it should be either very high or very low (far below zero) — if the voltage isn't, then the diode can't be faulted. Notice that this depends critically on knowing the purpose of the circuit; after all, any open diode with no voltage across it would be verified by this rule. Thus, teleological inferences are enabled only if the flag (see Section 2.2) which indicates the circuit is malfunctioning is set.

Mode consistency. Fault modes whose behavior is inconsistent with any propagations is eliminated. For instance, if there is any voltage across a diode, then it cannot be shorted (since if it were shorted the voltage would be zero). The fault mode is eliminated regardless of whether the propagation has assumptions. If the propagation is incorrect, then one of its underlying components must be faulted in which case the component in question isn't faulted. Therefore it is valid to eliminate the fault mode independent of the correctness of the propagation (unless of course the component itself is an assumption of the propagation in which case nothing can be deduced about the fault modes).

Qualitative propagation. This strategy examines the conflicting propagations to identify what fault modes could have caused the particular conflict. Suppose that through a long propagation chain, we determine the current through a resistor and use Ohm's Law to predict the voltage across it. A subsequent measurement indicates this predicted voltage is too low. If the resistor is truly faulted, then its resistance must be too high or must be open. On the other hand, if it is not faulted, then the current supplied to it must have been too low; the process then recurses examining the component that was used to deduce that high current. (In either case, the resistor cannot be in the fault modes shorted or low.)

The justification structure produced by LOCAL records how each component contributes to the propagation. This information is used to determine the fault mode of the component, or which erroneous input propagations (if any) could cause the observed symptoms. Consider the example of a high voltage deduced by applying Ohm's Law. The resistor expert records the justification of the voltage with an annotation (RESISTORI) that it used a current to deduce a voltage by Ohm's Law. The qualitative propagation rule is: "All current-to-voltage propagations, produced by a resistor, that are too high indicate either that the resistor is shorted or low, or that the resistor's input current is too high."

The qualitative propagator, instead of propagating ranges, through the circuit topology, propagates the tokens “high” and “low” in reverse direction through the justification of the problematic propagation. The tokens refer to whether the propagation is strictly higher or lower than was actually measured. The potential difficulty is that, if a component occurs more than once, it is not easy to tell which contribution dominates. To account for this, LOCAL performs the entire qualitative propagation as one unit, identifying which fault modes in which components could have caused the observed symptoms and localizing the fault to the union of these. Thus, if the same resistor contributes twice to the same conflict, one explained by high or open and the other explained by low or shorted, no deduction about that resistor is made other than it is under suspicion.

The qualitative propagator is invoked for every conflict, whether it is between two propagations or a measurement and a propagation. For example, if the range of propagation A lies entirely below the range of propagation B, then it propagates “high” down B and “low” down A. This will collect all fault modes which could remove the conflict by moving A up or B down. In the usual case, one of A or B is a measurement, so it is only necessary to propagate “high” or “low” down one propagation.

4.1 Diode fault modes

For the sake of brevity, we only analyze the fault modes of a diode in detail. The following teleological rule for the diode is based on the presupposition that the circuit contains only one fault and that it is currently manifesting a symptom. If the voltage across the diode is less than some maximum, then the diode cannot be open since it cannot be causing a symptom. If the current through the diode is less than enough to warrant it being considered on, then it cannot be shorted since if it were shorted and causing a symptom, then it would be conducting a significant amount of current:

If $V_H \leq V_{MAX}$, then the diode cannot be open. If $I_H \leq I_{ON}$, then the diode cannot be shorted.

The mode consistency rule for a diode is as follows. If the voltage across a diode is greater than zero, then it cannot be shorted since the definition of shorted is that the voltage across the diode is zero. Similarly if the current through a diode is greater than zero, then it cannot be open:

If $V_L \geq .1$, then the diode cannot be shorted. If $I_L \geq I_{OFF}$, then the diode cannot be open.

When the value propagated by the diode leads to a later conflict, the fact that the propagation was high or low can be used to determine the fault modes in which the diode could be. The diode model can make six different propagations: (1) from a new voltage deduce that the diode is on and propagate

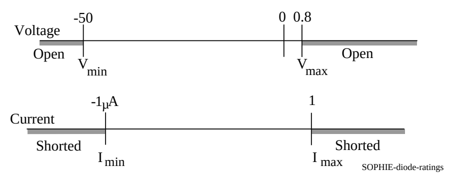


Figure 8: Diode Ratings

the corresponding current, (2) from a new voltage, deduce that the diode is off and propagate the corresponding current, (3) from a new current, deduce that the diode is off and propagate the corresponding voltage, (4) from a new current, deduce that the diode is on and propagate the corresponding voltage, (5) add extreme maximum and minimum voltages to the propagation database before any measurements are performed, (6) add extreme initial maximum and minimum currents.

For brevity we discuss only the rules for cases (1), (5) and (6). (These qualitative propagation rules will presume all the preceding diode rules.) If the current propagated by the diode model is high (i.e., the predicted is higher than the measured), then the diode must be shorted since the current flowing through it must be less than I_{MIN}. No erroneous value of the voltage propagated into the diode can cause this symptom since there is no valid region of operation in which the diode current is deduced to be less than I_{MIN} (I_{MIN} is negative, and represents the maximum reverse current flow).

On the other hand, if the propagated current flowing through the diode is low (i.e., the predicted is lower than the measured), the diode must be shorted, or the voltage across the diode is high (indicating that the diode is off rather than it being on).

For the propagation $V \rightarrow I$ for diode off: If $I =$ high, then the diode must be shorted, and the symptom cannot be caused by V. If $I =$ low, then the diode must be shorted, or $V =$ high.

The diode being open never arises since a voltage low enough to indicate the diode is off will also trigger the behavioral deduction that the diode cannot be open.

The notion of state is crucial in understanding the behavior of the diode. For example, the propagation rule "If $V_H \leq V_{OFF}$, then propagate the range $I = [-\infty, I_{ON}]$" is better thought of as "If $V_H \leq V_{OFF}$, then the diode must be off, and if the diode is off, then the current flowing through it must be $I = [-\infty, I_{ON}]$." Therefore the state of the diode is explicitly recorded (with its underlying assumptions, of course) and the record is referred to in explanations.

Cases (5) and (6) are much simpler.

If $I = I_{MIN}$ =high, then the diode must be shorted.
If $I = I_{MAX}$ =low, then the diode must be shorted.
If $V = V_{MAX}$ =low, then the diode must be open. If
$V = V_{MIN}$ =high, then the diode must be open.

4.2 Active troubleshooting extended

The simple scoring procedure for proposed measurements of Section 2.4 needs to be considerably extended to accommodate the additional inferential complexities introduced into LOCAL. The actual scoring procedure used expands the simple approach to account for the following effects:

- There may be multiple propagations for each circuit quantity.
- Propagations can have secondary assumptions. These assumptions contribute to a conflict but not to a corroboration.
- Coincidences can no longer be simply divided into conflicts and corroborations (see Figure 5). Some hypothetical coincidences turn out to be neither corroborations nor conflicts, thereby making the scores of hypothetical measurements higher than they should be. The wider the range of the propagated value, the less probable an interesting coincidence will occur.
- A measurement may eliminate some but not all of the fault modes of a component.
- Due to incompleteness, important coincidences may occur only among propagations caused by hypothetical measurements. LOCAL cannot calculate these effects, making the score of certain measurements lower than they should be.

The actual scoring procedure used by LOCAL attempts to accommodate all these affects. However, the procedure is ad hoc and is not described here.

5 Circuit-specific knowledge

The first measurement, one even a neophyte debugger would perform, is to measure the output. This is always the single most informative measurement, but LOCAL can draw no diagnostic conclusions from it. LOCAL can rarely draw any diagnostic inferences from the second measurement as well as it is usually too far from the output to produce any coincidences. Only after the student has made a number of measurements have enough values been propagated to produce some useful coincidences.

One approach to improving LOCAL's diagnostic powers would be to use more precise component models coupled with more powerful constraint propagation techniques which could solve simultaneous equations over ranges. We decided not to adopt this approach for a number of reasons:

- The required research task is very difficult. Even under the simplification that circuit values are single valued and component models are linear, troubleshooting is still difficult. Under this simplification a system like EL could do the propagation, but initially every coincidence would depend on every component since every component will in some way (usually small) influence every value, thus making most coincidences (and certainly all conflicts) uninteresting.
- The resulting algorithms would more than likely be computationally intractable.
- LOCAL's reasoning would diverge so far from a human's that it would no longer be able to produce useful explanations for its deductions.

So how do technicians troubleshoot successfully? Technicians and engineers seem to draw extensively on causal, qualitative and teleological reasoning when troubleshooting circuits. They exploit an intuitive "understanding how the circuit works" to troubleshoot it. These forms of reasoning are exactly what are missing from LOCAL and contribute to its poor performance in the early phases of troubleshooting. LOCAL employs a myopic view of the circuit and thus fails to utilize the global mechanism through which the circuit functions. Its myopic view is appropriate when the fault has been isolated to some module or group of components; it is entirely inappropriate for making initial measurements when not enough propagations have been made.

Unfortunately, in 1977, the state of knowledge about causal and teleological reasoning was not deep enough to permit their inclusion in SOPHIE III (even in 1992 the field has not made enough advances to do without circuit-specific knowledge). Our approach was, instead, to augment LOCAL by a simple rule-based system which draws the missing conclusions about component faults. This rule-based system, unlike the propagator, is circuit specific and must be changed for every new circuit SOPHIE III encounters.

The ensuing loss of generality is unfortunate. In order to mitigate this loss of generality we impose two principles when we encode this circuit-specific knowledge. First, we impose a strict discipline upon its form, making addition, modification and explanation of circuit-specific knowledge more stylized and less error-prone. Second, the propagator and its underlying mechanisms for handling assumptions and fault modes are used as a way of simplifying the deductions that the rule-based system needs to make. Consider a simple example, suppose we had the rule "if output voltage is low, then the regulator is shorted." Suppose we measure the load current, and LOCAL propagates it through the load to determine that the output voltage is low. This triggers the regulator shorted rule under the assumption that the load is correct. Therefore we have learned that either regulator is shorted or the load is faulted. Utilizing LOCAL we need only one rule to handle both of these cases. Because of LOCAL's propagating power we need only a few of the thousands of rules we would otherwise need to have.

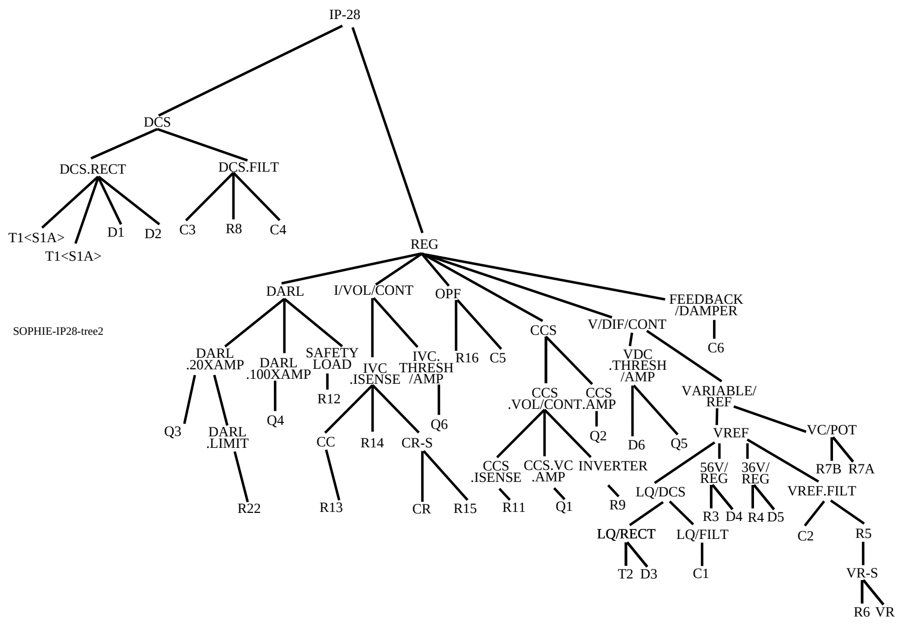


Figure 9: Hierarchical decomposition of IP-28 modules. See Appendix for a description of the modules.

In addition the fact that circuit-specific rules lose us generality, we also lose some capacity to make intelligent explanations since the rules are produced by a person and not by LOCAL. Fortunately, there are four effects that mitigate this loss. First, the rules are very simple, and thus any particular troubleshooting deduction uses a sequence of them, thus admitting a kind of deductive argument but with prestored explanation for each step in the sequence. Second, the rigid discipline on the form of the rules makes it much easier to impose a coherent structure on them. Third, LOCAL's propagating power reduces the required rule set to an essential core. Fourth, the contribution LOCAL makes to each deduction is easily explained through its existing explanation facilities.

Although a circuit is constructed from components, it is more easily understood as consisting of a small number of interacting modules which in turn, consist of other modules. The hierarchy of modules of a circuit are represented by a tree, the root of which is the circuit itself, the nodes of which are its modules and submodules, and the leaves of which are its components. Figure 9 illustrates the modules of the IP-28.

5.1 The qualitative-quantitative interface

LOCAL continuously checks for any propagation which affects a key set of voltages and currents (called *watchpoints*). When a value is found for one of these watchpoints, it maps the quantitative range to a qualitative value and asserts the qualitative value in a distinct database. This assertion is recorded along with the assumptions underlying the triggering propagation. This correspondence is specified by a translation table. Consider the watch-point at the output of the IP-28 voltage reference (the voltage between nodes N11 and N14). The translation table is (`high` $\geq 31.0 \geq$ `eq` $\geq 27.0 \geq$ `low`). The numbers in the table delineate the different qualitative ranges. If the voltage is above 31.0 volts, then `(V (N11 N14) high)` is added to the qualitative data base. If the voltage lies between 31 and 27, then `(V (N11 N14) eq)` is asserted. The voltage [30 , 32] overlaps `high` and `eq`, so this voltage results in the assertion `(V (N11 N14) (OR high eq))`. The 12 watchpoints and their translation tables for the IP-28 are given by Table 6.

It is important to distinguish `high` and `low` used in this teleological sense, to the "high" and "low" used in the qualitative propagator. In the teleological sense `high` means the value is higher than it should be compared to a working (i.e., unfaulted) circuit; while "high" in the propagator means the value was higher than predicted, based on previously made measurements in the same (i.e., faulted) circuit which may already be higher than it should be in a working circuit.

Semiconductor components can have many operating regions or states, each modeled by a distinctly different characteristic behavior. Whenever LOCAL determines an operating region of a component, it adds an assertion of the form `(STATE <component> <state>)` to the qualitative database. The possible states were summarized in Table 1.

Watchpoint	**Translation table**
B/Q4	(normal $\geq$ -.00055 $\geq$ on)
C/Q2	(high $\geq$ .0008 $\geq$ normal $\geq$ .00056 $\geq$ low)
L/R8	(high $\geq$ 1.3 $\geq$ heavy $\geq$ .8 $\geq$ light)
(N1 GROUND)	(high $\geq$ 52.0 $\geq$ ok $\geq$ 45.0 $\geq$ low $\geq$ 30.0 $\geq$ verylow)
(N11 GROUND)	Function of front panel settings
(N11 N14)	Function of front panel settings
(N15 N16)	(high $\geq$ 10.0 $\geq$ normal $\geq$ 1.0 $\geq$ low)
(N16 N14)	(high $\geq$ 37.0 $\geq$ eq $\geq$ 35.0 $\geq$ low)
(N21 N1)	(high $\geq$ 2.6 $\geq$ normal)
(N23 N14)	(high $\geq$ 85.0 $\geq$ normal $\geq$ 60.0 $\geq$ low $\geq$ 1.0 $\geq$ zero $\geq$ 0.0)
(N24 N14)	(high $\geq$ 58.0 $\geq$ eq $\geq$ 52.0 $\geq$ low)
(N4 GROUND)	(high $\geq$ 45.0 $\geq$ normal)

Table 6: Watchpoints and translations for IP-28.

5.2 Module modes

Just like components, modules can have behavioral modes. The module mode describes both the physical condition of the module (faulted or valid) and its behavior. A module may have multiple valid and faulted modes. In writing down the rules one needs to be very clear what the modes actually mean, otherwise the rule set may become inconsistent and SOPHIE III will encounter an irreconcilable contradiction (i.e., manifested as an empty nogood). Modules typically have many terminals, therefore we adopt the convention that module modes are associated with the quantities at one designated module port. For example, the mode of the constant current source is completely determined by its output voltage and current. The modes used in SOPHIE III are chosen by the knowledge engineer on the basis of simplicity, and the majority of the modules have only one or two (e.g., `high` and `low`) faulted modes.

A module can be unfaulted yet be exhibiting the behavior of a faulted mode because the behavior of some external component makes the module's behavior appear to be faulty. Therefore we adopt the convention that a module is operating in a faulted mode if it is exhibiting faulty behavior *and* contains a fault causing that symptom. A module which contains no faults is by definition operating in a valid mode. For example, the regulator section of the IP-28 has four faulty modes: `over-v-lim` — regulator output voltage is higher than the voltage control setting, `over-i-lim` — regulator output current is higher than the current control setting, `low` — regulator output is lower than either the voltage or current control setting, and `very-low` — regulator output voltage is less than 1.2 volts and the output is lower than the control settings. The regulator can exhibit the faulty behavior `low` either because it contains the fault or because it is being supplied with a faulty signal (e.g., the DC source is `low`). The possible fault modes need not be disjoint; the modes `low` and `very-low` overlap. The same

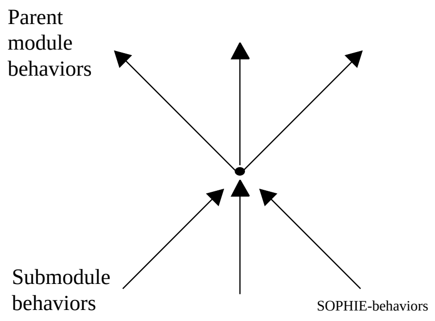


Figure 10: A Node in the Behavior Tree

component fault can cause two different modes depending on the precise shift in parameter value (e.g., R8 `high` can mean R8 has a resistance anywhere between 3 and ∞ ohms) and the front panel control settings. As module behaviors need not be disjoint, it is not only necessary to know all the possible behaviors of a module, but also which behaviors are inconsistent with each other. For example, the regulator behavior `low` is inconsistent with behavior `high`, but not with behavior `very-low`.

The knowledge engineer must construct a knowledge structure which indicates how each module's behavior contributes to the behaviors of its parent. We call this structure the behavior tree (although it is really a lattice). Every node in the behavior tree corresponds to a behavioral mode of a module. The incoming edges (see Figure 10) to a node correspond to the behaviors of its submodules that could cause the node's own behavior; the outgoing edges from a node correspond to the parent behaviors that the node can cause. As a module behavior can produce different behaviors in its parent, a node may have multiple outgoing edges. The terminals of the behavior tree are the fault modes of the components.

A node in the behavior tree is specified by the module name and the mode it is in, abbreviated `(<module> <mode>)`. Figure 11 describes a small part of the IP-28 behavior tree. (The actual tree takes about two pages to print so we have not included it in this paper.)

In order to be used successfully the behavior tree must obey the following principles:

- Every possible mode (faulty or valid) must be included[3].
- Every node (except for the overall module) must have a parent.
- Include all possible ways in which a module can affect a parent.

[3]Some faults immediately cause others, so these are left out of the tree by the single-fault presupposition.

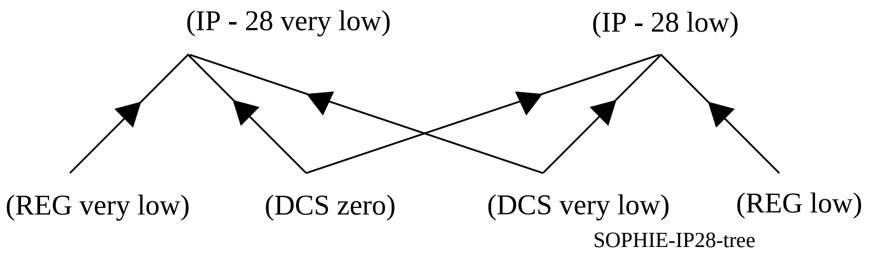


Figure 11: Part of the IP-28 behavior tree. The behavior (IP-28 very-low) can be caused by (REG very-low), (DCS zero), or (DCS very-low). The behavior (IP-28 low) can be caused by (DCS zero), (DCS very-low), or (REG low). Note that (DCS zero) and (DCS very-low) can cause both (IP-28 very-low) or (IP-28 low).

5.3 Mode detection rules

The assertions in the qualitative data base trigger rules which determine module modes. Determining a module's fault mode tells us a lot about the circuit as the only remaining fault modes are those which can cause it (i.e., are below the observed node in the behavior tree). This section and the next fleshes out this intuition.

The mode detection rules need not pinpoint the exact behavioral mode of the module. They can determine that the module is in one of a set of possible modes (a *localization*), or determine that the module cannot be in some modes (an *elimination*). The rules for VARIABLE/REF (the variable voltage reference of the IP-28) are:

```
(V (N11 N14) high) -> (LOCALIZE VARIABLE/REF high ok)
(V (N11 N14) eq) ->  (LOCALIZE VARIABLE/REF ok)
(V (N11 N14) low) -> (LOCALIZE VARIABLE/REF low)
```

`(LOCALIZE <module> <mode1> <mode2> ...)` indicates that the module must be in one of the given modes. The first rule states that if the voltage between nodes N11 and N14 is high, then the module `VARIABLE/REF`, if faulted, must be in mode `high`. `ok` is added in the first case because the IP-28 voltage-reference is designed to actively correct a low voltage on its output terminal, but not a high voltage. Thus if a low voltage appears, then the voltage reference must be faulted, but if a high voltage appears, then some external module could also be supplying a voltage to its port. This latter possibility never arises in normal operation, but can in faulty operation.

The rules for a constant current source are a good example of the expressive power of using multiple valid modes. A peculiarity of a non-ideal current source is that it cannot supply a constant current into an arbitrarily high-impedance load. Its behavior when presented with a high-impedance load is deceiving. The high (higher than it should be teleologically) output voltage suggests that the current source is high, while the low output current suggests it is too low.

The constant current source (`CCS`) is modeled by four modes, two of which are faulted and two of which are valid: `high`(faulted), `low`(faulted), `saturated`(ok), and `normal`(ok):

```
(V (N4 GROUND) high) -> (LOCALIZE CCS saturated high)
(V (N4 GROUND) normal) -> (LOCALIZE CCS high low normal)
(I C/Q2 high) -> (LOCALIZE CCS high)
(I C/Q2 normal) -> (LOCALIZE CCS normal)
(I C/Q2 low) -> (LOCALIZE CCS saturated low)
```

In the case where the fault lies outside the constant current source resulting in a high impedance load, the high voltage measurement indicates that the source is either in modes `high` or `saturated`, and the low current measurement indicates that the source is either in modes `low` or `saturated`. Hence a high voltage combined with a low current indicates that it is in the `saturated` mode and therefore not faulted.

When a propagation triggers a mode-detection rule, its assumptions, if any, modify the effect of the rule (see next section). In addition, in limited circumstances, the qualitative propagator is invoked. For example, if the rule,

```
(V (N11 N14) low) -> (LOCALIZE VARIABLE/REF low),
```

is triggered, then the fault either lies in VARIABLE/REF or the triggering propagation is low. The qualitative propagator can then be invoked on the trigger restricting the fault modes of its underlying assumptions which are not in VARIABLE/REF. The situation is quite different for the rule:

```
(V (N11 N14) high) -> (LOCALIZE VARIABLE/REF high ok)
```

In this case, there is little point to invoking the qualitative propagator as the fault might well lie outside VARIABLE/REF or the assumptions underlying the triggering propagation (because of the `ok` in the second rule consequence). The general principle is that the qualitative propagator can only restrict the fault modes of those components that would otherwise be verified by the consequent. These conditions are clearly overly restrictive, and further research would be required to determine exactly under which conditions the qualitative propagator should be invoked.

Although the behavior tree contains a large number of modules and their modes, it is only necessary to include a handful of rules to detect module modes — the remaining module modes are either entailed by LOCAL's propagations or only affected indirectly. This can be seen from the simple fact that 12 watch-points are completely adequate to diagnose all faults in the IP-28.

5.4 Reasoning with the behavior tree

Once the mode detection rules identify a module's mode(s), reasoning over the behavior tree identifies which faults can cause the observations. The behavior tree is primarily a data structure to make it convenient to write and explain circuit-specific rules. The presence of the behavior tree does not directly reduce the number of rules needed (except in so far as it makes rule writing a much simpler task). All of the consequents of the mode detection rules could be replaced by a set of component mode eliminations or localizations. For example, `(LOCALIZE VARIABLE/REF low)` could be replaced by a list of all the component fault modes which cause the `VARIABLE/REF` to be `low` and there would be no change in diagnostic precision. However, with this crude representation one is far more prone to omit component modes.

The inference mechanism operates on a database which keeps a record of the remaining possible modes for each module. All localizations are converted to their equivalent eliminations so we need only consider inferences resulting from an elimination. A distinct such data base is kept for each control panel setting. However, one common database is kept for the component fault modes as we presume the physical fault cannot change. This is sufficient to flow all the deductions from one set of control settings to another. (For example, if a module is actually faulted under one setting it must be in some faulted mode under all settings.)

Consider the elimination of a faulty mode. If a faulty node is eliminated, the nodes below it can no longer be the cause; each of these lower nodes is examined to see whether it still has a parent (i.e., can still cause any other behavior), and if not, it too is eliminated. Analogously, if a mode is eliminated it can no longer cause any of its parents, so each parent behavior it can cause is examined to see whether it still can be caused by another behavior, and if not it too is eliminated. One elimination can thus recursively generate many more. If all possible faulty behaviors of a module are eliminated, then the module cannot be faulted. It is unecessary to explicitly implement the intuitive principle that if a module is unfaulted, then all of its submodules are unfaulted. This is a simple consequence of the fault mode elimination mechanism just outlined. If all of a module's faulty behaviors have been eliminated, then its submodules have no possible faulty behaviors either.

Elimination of valid modes is simpler. While a faulty behavior can be caused by a faulty behavior in any one of the submodules, a valid behavior only results if all of the submodules are behaving validly. Thus the elimination of a valid node has no direct consequences on the modes below (unless there is only one) it since the elimination of any one of these alone would be sufficient to eliminate the mode itself. On the other hand, when a valid mode is eliminated its parent cannot happen unless it could also be caused by an alternative valid mode of the module. If all possible valid behaviors are eliminated, then the module must be faulted. Under the single-fault assumption and the fact that modules do not overlap, any sibling modules are automatically unfaulted. If a module is determined to contain a fault, then all of its parent modules are, by definition, also faulted. Since the particular faulty mode need not be known, this is achieved by eliminating all of the valid modes of the parent modules.

The propagations which cause the original elimination may have underlying assumptions which modify the consequent eliminations and localizations. These assumptions must be carried along during the recursive eliminations, continuously enforcing the restriction that an elimination of a faulty mode cannot depend on an assumption about the module in question. Eliminations of valid modes can, of course, depend on the module itself. An attempt to eliminate a mode of a module upon which the triggering propagation depends is ignored, however, the recursive eliminations are still attempted. Usually the underlying assumptions refer to only some of the components of the module, in which case the elimination still leads to deductions about that part of the module which does not overlap with the assumptions underlying the triggering propagation.

By far the majority of SOPHIE III's circuit-specific rules detect module modes. The behavior tree, unfortunately, can only represent hierarchical module interactions, and cannot represent causal interactions among same-level modules. SOPHIE III employs three additional rule types to capture these latter type of knowledge. We discuss each individually in the following subsections.

5.5 Good neighbor rules

Sometimes when a module is discovered to be unfaulted and behaving normally (as distinguished from being unfaulted but behaving symptomatically) its "suppliers" and "consumers" can also be determined to be behaving normally, and hence to be unfaulted. Consider an example from the IP-28. The constant-voltage reference of this power-supply (see Figure 4) consists of a raw DC voltage which is first regulated to 56 volts and then regulated down to 36 volts. The only supplier of power to the 36-volt regulator is the 56-volt regulator, and the only consumer of the 56 volt output is the 36-volt regulator. If voltage of the second reference is 36 volts, then it is operating normally and unfaulted, and the 56-volt regulator must be unfaulted and operating normally. To facilitate good neighbor rules SOPHIE III maintains an additional mode for every module, `operating-ok`, which is consistent with every other good mode, and inconsistent with every other faulty mode. The `operating-ok` mode indicates a module is unfaulted *and* behaving normally. Thus, localizing a module's mode to `operating-ok` is stronger than just localizing its mode to just `ok`.

The module-detection rule,

```
(V (N11 N16) eq) -> (LOCALIZE 36V/REG operating-ok)
```

also triggers the good neighbor rule,

```
(36V/REG operating-ok) -> (LOCALIZE 56V/REG operating-ok),
```

which, in turn, triggers the good neighbor rule,

```
(56V/REG operating-ok) -> (LOCALIZE LQ/DCS operating-ok).
```

5.6 Bad neighbor rules

Faulty behavior can be caused by neighboring *faulty* behaviors. This fact can be stated by bad neighbor rules of the form:

```
(BEHAVIOR <module1> <mode1>) -> (BEHAVIOR <module2> <mode2>)
```

Such a rule states that if `<module1>` is observed behaving in faulty mode `<mode1>` that this can be explained by faulty module `<module2>` behaving in `<mode2>`. Consider the voltage-reference example again. If the output of the 36-volt regulator is low, then it is either faulted, being supplied with a bad signal (the 56-volt regulator is low), or being presented a bad load (the output filter of the voltage reference is shorted). One example of a bad neighbor rule is:

```
(BEHAVIOR 36V/REG low) -> (BEHAVIOR 56V/REG low)
```

Bad neighbor rules are invoked by mode detection rules or other bad neighbor rules. For example, the mode detection rule:

```
(V (N11 N16) low) -> (LOCALIZE 36V/REG low ok),
```

is rewritten as:

```
(V (N11 N16) low) -> (BEHAVIOR 36V/REG low).
```

This rule now has the additional effect that `(BEHAVIOR 36V/REG low)` needs to be explained. As a `BEHAVIOR` assertion states that the behavior is in a particular mode which may or may not originate within the module, the `ok` mode is always implicit as a possibility. Another bad neighbor rule is:

```
(BEHAVIOR 56V/REG low) -> (BEHAVIOR LQ/DCS low).
```

All bad neighbor rules are triggered and run until quiescent and the fault is localized to one of the faults mentioned in one of the bad neighbor rules. Thus, the result is a disjunctive localize, e.g., the above example becomes a `(OR (LOCALIZE 56V/REG low) (LOCALIZE 36V/REG low) (LOCALIZE LQ/DCS zero) ...)`.

5.7 Exception rules

The third and last type of rule does not fit any particular conceptual pattern. They are all of the form:

```
<measurement> -> (ELIMINATE <module> <mode>).
```

These are like mode detection rules, but the watchpoint is not at the module's boundary. For example, one of the three such rules for the IP-28 is:

```
(V (N11 N14) EQ) -> (ELIMINATE Q5 sh/op/op).
```

There are two components between Q5 (see Figure 4) and the nodes N11 and N14. Unfortunately, this rather inelegant rule is needed. This particular fault mode for Q5 makes the voltage regulator inoperative, and also shorts the output of the voltage reference to the output of the overall power-supply. The voltage reference is very weak, so the main power-supply will dominate. Thus, if the output of the voltage-reference is normal, Q5 must not be shorted in this way. (If it were, then it would not be voltage regulating, causing the voltage across the reference to be high.) This is a very subtle inference which would not be generally possible without a rather sophisticated causal and teleological model of the IP-28.

5.8 Handling disjunction

As ranges can overlap multiple qualitative values, the database can contain disjunctive assertions such as `(V (N11 N14) (OR high eq))`. Such assertions are handled by processing each of the individual disjuncts individually and then intersecting any resulting module eliminations. In the example, the disjunction is processed by first identifying all the mode eliminations of `(V (N11 N14) high)` and then identifying the mode eliminations of `(V (N11 N14) eq)`, and only permanently recording those consequences which are common to both. This paradigm is used for all types of circuit-specific rules.

5.9 Example of reasoning with the behavior tree

The top-level of the behavior tree for the IP-28 is illustrated in Figure 12. Valid modes are omitted. Suppose the output of the IP-28 is measured to be too low. A mode detection rule monitoring the output applies and does a localization of `(IP-28 low)`. This is transformed into the equivalent elimination of all those modes inconsistent with low: `(IP-28 high)` and `(IP-28 ok)`. Consider the elimination of `(IP-28 high)`. Looking at the tree we see that `(IP-28 high)` can be caused by `(REG not-limiting)`, `(REG over-v-lim)`, and `(REG over-i-lim)`. These modes have only one parent so they can be eliminated. The eliminations recurse until the component level is reached. For example:

```
    (ELIMINATE REG over-v-lim)
 -> (ELIMINATE V/DIF/CONT always-off)
 -> (ELIMINATE VDC.THRESH/AMP always-off)
 -> (ELIMINATE Q5 op/op/op)
```

Suppose another measurement localizes `VARIABLE/REF` be `low`. `(VARIABLE/REF ok)` is inconsistent with it and is eliminated. There is no way to tell which submodule of `VARIABLE/REF` is faulted, so the eliminations do not recurse. Reasoning up the behavior tree we obtain:

```
   (LOCALIZE VARIABLE/REF low)
-> (LOCALIZE V/DIF/CONT low)
-> (LOCALIZE REG low)
-> (LOCALIZE IP-28 limit-too-low)
```

The `(LOCALIZE V/DIF/CONT low)` is a new piece of information we did not know before. Presuming that the circuit had only a single fault, all components not explicitly listed as faulted or contained within `VARIABLE/REF` must be `ok`. Figure 13 presents the faulty behaviors that are still possible along with those connections in the behavior tree that are relevant to them.

5.10 Troubleshooting with circuit-specific knowledge

The same techniques for using the general knowledge to score measurements apply to the circuit-specific knowledge. Recall that the best strategy is to identify a measurement with the maximum expected information value. SOPHIE III computes the expected diagnostic scores for every possible coincidence and

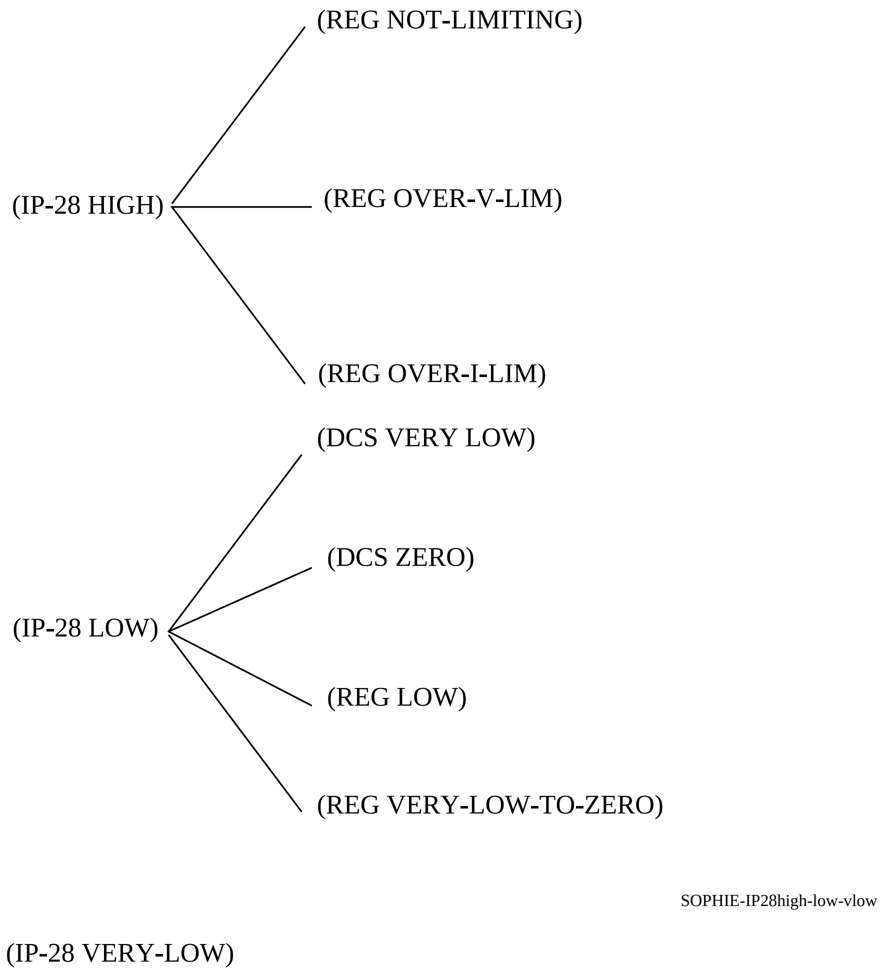


Figure 12: Top of IP-28's behavior tree

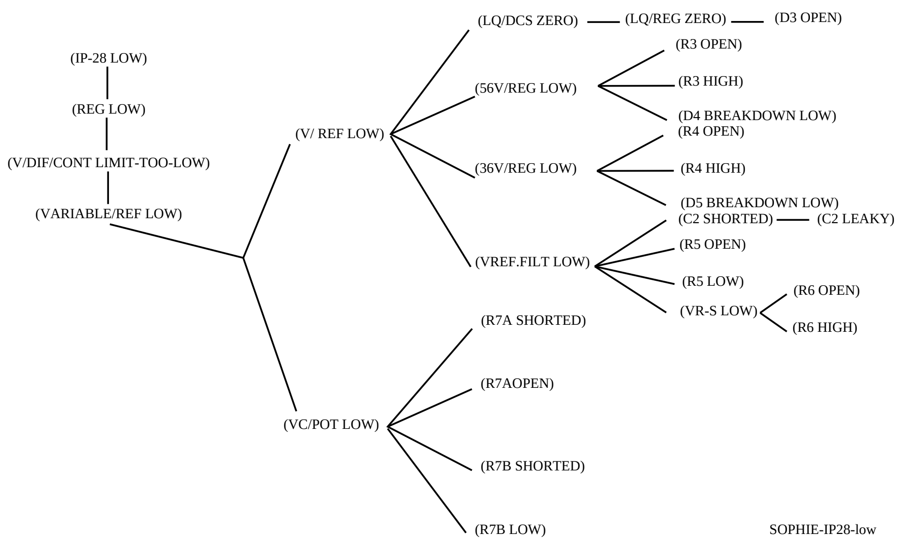


Figure 13: Final IP-28 behavior tree which shows the only remaining possible component faults are the 16 leaves (in 10 components).

at every watch-point. The score is computed on the basis of the combined diagnostic results from both LOCAL and the circuit specific rules. Obviously, early in the troubleshooting the measurements at the watch-points have higher scores; later in the troubleshooting the scores for hypothetical coincidences with propagations will be preferred.

Although SOPHIE III is now a nearly optimal troubleshooter of the IP-28, its scoring of early student measurements remains poor. Consider the case where the student makes a measurement which is neither at a watch-point, nor coincides with an existing propagation. Our scheme does not compute any expected score for such measurements. Also, it does not consider the influence of propagations into multiple watchpoints, or into a watchpoint but with underlying assumptions. We implemented a version of SOPHIE III which computes better scores by actually simulating the propagation and behavior tree effects of various outcomes for each measurement point. Unfortunately, this version was too slow to be of practical use. Therefore, as a consequence we did not allow SOPHIE III to critique a student's measurement on the basis of easily computed but approximate scores.

We used a alternative, cruder, technique to critique the student's early measurements. We relied on the observation that SOPHIE III's score for a measurement is always too low (as it doesn't simulate the propagation, it cannot foresee all the results of measurements). On the other hand, once a student has made a particular measurement SOPHIE III can precisely determine the actual information gain produced by the measurement. Given the topology of the IP-28 every early measurement, no matter what its outcome, should provide substantial diagnostic information. Therefore, when a student's actual measurement produces substantially less information than the expected score for some other measurement, we can be certain the measurement is substantially suboptimal. At this point the coach can critique the student for his poor measurement.

6 Conclusions

In the introduction we laid out a set of five desiderata for the inferential machinery. SOPHIE III's electronic expert meets all of the criteria fairly well. However, we do not yet have a general diagnostician for arbitrary circuits, but we have made a major step. The necessity of circuit-specific rules is unfortunate, so let us re-examine those. The following table enumerates the types and number of circuit-specific rules used in SOPHIE III:

Mode detection rules	33
Good neighbor rules	4
Bad neighbor rules	4
Exception rules	9
Total	50

The fact that SOPHIE III uses only 50 circuit-specific rules deserves further discussion as this number seems both too high and too low. If we added mode detection rules for every possible module, as well as all possible good and bad neighbor rules, then we would probably require on the order of 1000 rules. Almost all of these rules are unnecessary because LOCAL propagation mechanism makes it necessary to represent only the core few.

Although SOPHIE III requires only a few rules to adequately troubleshoot the IP-28, does not mean that these rules are easy to obtain. Even after extensive experience with SOPHIE III we cannot be totally confident that it can handle every possible fault. It would be far more useful if we could build a diagnostician which did not depend on any circuit-specific rules at all. This suggests two directions for future research. First, LOCAL's inferential capabilities could be expanded in some way to be more sophisticated. Second, there might be some way to construct the necessary rules in another more computationally intensive process which examined the circuit much more carefully.

6.1 A re-examination of presuppositions

In order to draw diagnostic inferences, SOPHIE III makes a number of presuppositions. These presuppositions impose a restriction on its capabilities because all of them are violated in some situations. The consequences of violating them range from encountering irreconcilable contradictions (i.e., empty nogoods) to very inefficient troubleshooting. In this section, we briefly re-examine some of those presuppositions.

The presuppositions listed in the introduction fall into two classes. The first class consists of good troubleshooting heuristics. These should be assumed to hold only until evidence is discovered to the contrary. This class includes:

- The circuit contains a single fault.
- Faults only occur in components; not in circuit topology.
- All component fault modes are known.
- The circuit is non-intermittent.
- The faulty component is currently manifesting a symptom.

In a more general theory of troubleshooting, they should be treated as *defeasible assumptions* subject to explicit reasoning, much like the component assumptions of LOCAL. They are, in essence, a deeper level of assumptions, examined only when all the component assumptions have been eliminated. Every deduction made by LOCAL which depends on one of these presupposition should explicitly include this fact as an underlying assumption of that deduction. Unlike the component assumptions, these *presuppositional assumptions* influence the inference mechanism making the deductions. Within this extended framework, LOCAL, would become just a special case of a more powerful reasoner that was capable of "deliberating" over its own actions and theories.

The single-fault presupposition is the most pervasive. Many of SOPHIE III's deductions depend on it. The ideas of coincidence, conflict and corroboration are generalizable to multiple faults, but many of the the specific inference strategies used by LOCAL do not apply directly. For example, without the single-fault presupposition a corroboration can not verify the underlying components nor a conflict verify the components that aren't mentioned by its underlying assumptions. Also, much of the reasoning mechanism of the behavior tree is no longer valid. The combination of two faults might cause an unusual behavior which is not captured by any of the module's fault modes.

For a very nonobvious reason, multiple independent faults occur more frequently than one otherwise would expect within our framework. Designers typically design circuits so that they can handle wide variations in component parameters. This allows designers to use cheaper components with larger tolerances as well as to cope with changes in parameter values with temperature and age. Manufacturers typically produce components in only a few fixed tolerance levels and therefore the tolerance specified by the designer is always tighter than that of the component actually used. Thus a component may be faulted (exceed its tolerance), but the circuit still function correctly. The component definitely is faulted: The same component might not work in another circuit where the designer's tolerance is closer to the manufacturer's. The difficulty ensues when another component fault does cause the circuit to manifest a global symptom. The circuit now has multiple independent faults, and SOPHIE III may well encounter an irreconcilable contradiction troubleshooting it. This suggests that

LOCAL should use the designer's, not the manufacturer's tolerances, for component parameters.

We have been discussing one alternative to the single-fault case, that of multiple independent faults. A more common case is that of multiple consequential faults. For example, an output transistor may short causing another transistor to open. The single-fault heuristic often works well for such multiple faults. Frequently, the component that originally faulted will no longer be manifesting a symptom since the consequential fault has effectively made the originally faulted component appear unfaulted by isolating it. In the example, the voltages and currents around the output transistor would be negligible if the transistor it caused to open was responsible for delivering it its power. Since the output transistor now has no power, it will appear to be fault-free, thereby temporarily reducing the situation to a single-fault case.

The second presupposition, that SOPHIE III presumes all possible faults are known, is again good heuristic, but again does not always hold. Possible topological faults include a short between two nodes which do not share a common component (e.g., two geometrically adjacent, but topologically distant and components which have been physically bent so that their leads touch) or an open node which disconnects some components from others.

A more subtle violation occurs with component fault modes, since it is hard to hypothesize all the ways in which a component can fault. For example, if the diode faulted so as to become a resistor, with the voltage across it greater than V_{MAX} and the current through it greater than I_{MIN}, then the propagator would eventually encounter a conflict with no underlying assumptions. An example of such a conflict, a measurement of the voltage across the diode would indicate it was open, while a measurement of the current through it would indicate that it couldn't be. Fortunately, this is an extremely rare fault mode for diodes. Diodes usually fault by becoming either nearly shorted or completely open — and these are the modes currently modeled.

The non-intermittency presupposition is equivalent to assuming that nothing changes between the times measurements are taken and that the fault does not change when the control panel or load is changed.

The presupposition that the circuit's present symptom is a direct consequence of some component presently behaving symptomatically, is only true for circuits which do not have some kind of "memory." Suppose the circuit had a circuit breaker on its input which blew every time the power supply was plugged in. At the time we observe the power supply output it is manifesting a symptom (no output), but every component is behaving correctly. As a consequence LOCAL would encounter an irreconcilable contradiction. The problem is, of course, that the circuit breaker "remembers" that some component was shorting out when the power supply was first plugged in, even though the component might not be doing so at present. This is not a isolated case, most good power supplies, for example, employ a kind of electronic circuit breaker in addition to the usual regulation as an extra measure of protection for the circuitry connected to the power supply. This type of fault is notoriously hard to find since the troubleshooter does not get the opportunity to see the faulted component manifest its symptom.

The remaining presuppositions are limitations which need to be addressed by some future theory:

- Only DC behavior is important.
- All faults are equally likely.
- All measurements are equally easy to make.
- All fault modes are equally likely.

In many circuits AC behavior is important, so LOCAL needs to be extended to deal with time and AC effects. As transistors are far more likely to fail than resistors, the simple measurement scoring function LOCAL does not zero in on more probable faults earlier. In the simulated electronics laboratory all measurements cost the same, but in real circuits some measurements are far more easy to make than others. For example, DC currents are much harder to measure than voltages because they usually require physically removing one of the component's terminals. Resistors are far more likely to open than to short, therefore if we have eliminated opening as a fault mode for a resistor which should treat shorting as a very low probability fault.

6.2 Final observations

There are many promising directions along which the work described in this paper could have continued. However, two very different problems, one pragmatic and one intellectual, stood in our way. The pragmatic issue was the need for large address space and personal Lisp machines; the intellectual issue was our increased awareness that we did not really know what it meant to "understand" how a complex piece of equipment works. In particular we did not know what mental models the experts had of a given system's functioning, nor did we know how these models were learned, for they certainly weren't explicitly taught.

The computational issue that we faced with SOPHIE III was that it barely fits into the (256K words) address space of a PDP-10. SPICE, SOPHIE I and SOPHIE II each already occupied its own separate address space so nothing could be gained by running SOPHIE III stand-alone. This made it extremely difficult to extend SOPHIE III, especially to expand its coaching capabilities along the lines developing out of our coaching research on mathematical games. Address space limitations were not our only concern: To do "formative" evaluations, discovering the *reasons* for the pedagogical success and failures of our system, we needed the speed of efficient, dedicated Lisp machines. Without this speed, student reactions to the long and unpredictable delays often swamped our experimental probes.

The issue concerning the need for a theory of human understanding of complex systems, in particular circuits, was clearly the more challenging one. Indeed, much of our recent research has been directed at attacking this problem.

It quickly became clear to us that the work that went into SOPHIE II and III on explanation put the cart before the horse. We had no adequate theory of what it meant to understand a circuit and hence no well defined “target” model of what we wanted the student to learn. As a consequence no real theory of explanation was forthcoming.

In our experiments with SOPHIE I and II we substantiated that the beginner and expert alike prefer to reason about the circuit in qualitative and causal terms. The students preferred qualitative explanations and were only comfortable about their understanding if it was in terms of a qualitative causal mechanism, but SOPHIE III’s only reference to causality is in the precompiled explanations for its circuit-specific rules. It could not generate new ones nor could it expand old ones. A significant step toward the necessary robust theory of causality was achieved the following year by de Kleer, in his doctoral dissertation presenting a theory of qualitative causal reasoning about electronics [3]. His theory has turned out to provide a basis for constructing human-oriented explanations.

7 Acknowledgments

The SOPHIE project has been influenced and helped by numerous people. We are especially indebted to Alan Bell and Richard Burton for the substantial amount of work he did on SOPHIE I. Richard Rubenstein and Ned Benhaim helped a great deal on SOPHIE II. We are also grateful to Ed Gardner for making possible the original SOPHIE project and to Harry O’Neil and Dexter Fletcher for their constant encouragement and willingness to shelter the project from various bureaucratic constraints. The project was initially funded by the Air Force Human Resource Laboratory at Lowry Air Force Base and later by DARPA/CTO and the Tri-Service training laboratories. Numerous people have read preliminary drafts and have made invaluable suggestions. Daniel G. Bobrow, Bruce Buchanan, Bill Clancey, Jaime Carbonell, Ken Forbus, Olivier Raimon and Rachel Rutherford deserve special thanks for their patience and insights.

References

[1] Brown, J.S., Burton, R. R. and de Kleer, J., Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III, in: D. Sleeman and J.S. Brown (Eds.), *Intelligent Tutoring Systems*, (Academic Press, New York, 1982) 227–282.

[2] de Kleer, J., Local methods of localizing faults in electronic circuits, Artificial Intelligence Laboratory, AIM-394, Cambridge: M.I.T., 1976.

[3] de Kleer, J., How circuits work, *Artificial Intelligence* **24** (1984) 205–280; also in: D.G. Bobrow (Ed.), *Reasoning About Physical Systems* (MIT Press and North Holland, 1985) 205–280. 11-April 24, 1992-May 14, 2019

[4] de Kleer, J. and Brown, J.S., Mental models of physical mechanisms and their acquisition, in: J.R. Anderson (Ed.), *Cognitive Skills and their Acquisition*, (Erlbaum, Hillsdale, NJ, 1981) 285–310.

[5] Doyle, J., A truth maintenance system, *Artificial Intelligence* **12** (1979) 231–272.

[6] Nagel, L.W. and D.O. Pederson, Simulation program with integrated circuit emphasis, *Proceedings of the Sixteenth Midwest Symposium on Circuit Theory*, Waterloo, Canada, 1973.

[7] Rieger, C. and Grinberg, M., The declarative representation and procedural simulation of causality in physical mechanisms, in: *Proceedings IJCAI-77*, Cambridge, MA, (August, 1977), 250–256.

[8] Stallman, R. and Sussman, G.J., Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis, *Artificial Intelligence* **9** (1977) 135–196.

Appendix: IP-28 Modules

`DCS` The main DC source for the power supply.

`DCS.RECT` The rectified DC power.

`DCS.FILT` Filters the raw DC provided by the rectifier section.

`REG` Regulates the output based on the information fed back by the output sensors.

`DARL` The darlington pair which is the central controlling element in the power supply.

`DARL.20XAMP` The second stage of the darlington which amplifies approximately 20 times.

`DARL.LIMIT` A resistance which limits the gain of the second darlington transistor.

`DARL.100XAMP` The first stage of the darlington which amplifies approximately 100 times.

`SAFETY/LOAD` The minimal load on the darlington.

`I/VOL/CNT` This controls the amount of current provided to the load.

`IVC.SENSE` Senses whether the amount of current provided to the load is over the limit.

`CC` A variable control which sets the maximum amount of current provided to the load.

`CR-S` A switch which sets the range of the current control.

`IVC.THRESH/AMP` Amplifies the signal provided by the current sensor.

`OPF` Filters the output after regulation.

`CCS` A source of constant current used to establish a reference.

`CCS.VOL/CONT` A constant current source used to control another current source.

`CCS.ISENSE` A resistance which is used to sense the output current of the CCS.VOL/CONT

`CCS.VC.AMP` Amplifies the signal provided by the current sensor.

`INVERTER` Because of the arrangement of components, the current source can be inverted.

`CCS.AMP` Amplifies the output of the simple current source to provide a strong reference.

`V/DIF/CONT` This computes the difference between the reference and the output.

`VARIABLE/REF` Provides a variable reference voltage for comparison.

`VREF` Provides a fixed reference voltage.

`LQ/DCS` A separate source of unregulated DC used to establish a reference.

`LQ/RECT` Provides the raw DC for the reference.

`LQ/FILT` Filters the raw DC provided by the rectifier section.

`56V/REG` A 56 volt reference source.

`36V/REG` A 36 volt reference source.

`VREF.FILT` An output filter for the fixed voltage reference.

`VR-S` Sets the range of the voltage control.

`VC/POT` Controls the reference voltage and hence the maximum output voltage.

`VDC.THRESH/AMP` Computes the difference between the output and the reference and amplifies it.

`FEEDBACK/DAMPER` Damps out any potential oscillation caused by the feedback.