AD-A034 844 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MASS F/G 5/9
'INTELLIGENT' COMPUTER ASSISTED INSTRUCTION (CAI) APPLICATIONS.(U)
OCT 76 J S BROWN, R BURTON, J DEKLEER F41609-75-C-0032
AFHRL-TR-76-67 NL

UNCLASSIFIED

1 OF 1
AD-A
034844

END
DATE
FILMED
3-7-77
NTIS

U.S. DEPARTMENT OF COMMERCE
National Technical Information Service

AD-A034 844

'INTELLIGENT' COMPUTER ASSISTED INSTRUCTION (CAI) APPLICATIONS

BOLT BERANEK AND NEWMAN, INCORPORATED
CAMBRIDGE, MASSACHUSETTS

OCTOBER 1976

028095

AFHRL-TR-76-67

AIR FORCE

HUMAN RESOURCES

LABORATORY

AD A034844

"INTELLIGENT" COMPUTER ASSISTED INSTRUCTION (CAI) APPLICATIONS

By

John Seely Brown
Richard Burton
Johan DeKleer
Ned Benhaim

Bolt Beranek and Newman, Incorporated
50 Moulton Street
Cambridge, Massachusetts 02138

TECHNICAL TRAINING DIVISION
Lowry Air Force Base, Colorado 80230

October 1976
Interim Report for Period February 1975 – September 1975

Approved for public release; distribution unlimited.

DDC
JAN 26 1977
C

REPRODUCED BY
NATIONAL TECHNICAL
INFORMATION SERVICE
U.S. DEPARTMENT OF COMMERCE
SPRINGFIELD, VA. 22161

AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235

NOTICE

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This interim report was submitted by Bolt Beranek and Newman, Incorporated, 50 Moulton Street, Cambridge, Massachusetts 02138, under contract F41609-75-C-0032, project 1121, with Technical Training Division, Air Force Human Resources Laboratory (AFSC), Lowry Air Force Base, Colorado 80230. Mr. Ed Gardner, Computer Based Systems Branch, was the contract monitor.

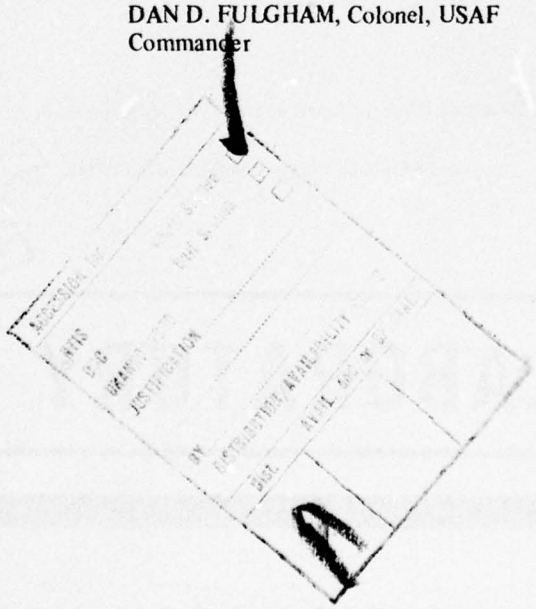
This report has been reviewed and cleared for open publication and/or public release by the appropriate Office of Information (OI) in accordance with AFR 190-17 and DoDD 5230.9. There is no objection to unlimited distribution of this report to the public at large, or by DDC to the National Technical Information Service (NTIS).

This technical report has been reviewed and is approved.

MARTY R. ROCKWAY, Technical Director
Technical Training Division

Approved for publication.

DAN D. FULGHAM, Colonel, USAF
Commander



Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFHRL-TR-76-67	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* "INTELLIGENT" COMPUTER ASSISTED INSTRUCTION (CAI) APPLICATIONS		5. TYPE OF REPORT & PERIOD COVERED Interim February 1975 – September 1975 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) John Seely Brown, Johan DeKleer Richard Burton, Ned Benhaim		8. CONTRACT OR GRANT NUMBER(s) F41609-75-C-0032
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman, Incorporated 50 Moulton Street Cambridge, Massachusetts 02138		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62703F 11210215
11. CONTROLLING OFFICE NAME AND ADDRESS HQ Air Force Human Resources Laboratory (AFSC) Brooks Air Force Base, Texas 78235		12. REPORT DATE October 1976 13. NUMBER OF PAGES<br~~80~~ 83
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Technical Training Division Air Force Human Resources Laboratory Lowry Air Force Base, Colorado 80230		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

This research was partially funded by Advanced Research Projects Agency, 1400 Wilson Boulevard, Arlington, Virginia 22209.

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

computer assisted instruction
computer programming
electronic equipment
electronic troubleshooting
electronics training
SOPHIE
SOPHisticated Instructional Environment
technical training

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Interim work is documented describing efforts to modify computer techniques used to recognize and process English language requests to an instructional simulator. The conversion from a hand-coded to a table driven technique are described in detail. Other modifications to a simulation based computer assisted instruction program to allow a gaming situation to be used in the classroom are also documented, and a student protocol taken with this program is presented. Textual support materials used in conjunction with the on-line computer instruction are also presented.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

SUMMARY

PROBLEM

The training of troubleshooting principles has always been difficult in the military and civilian community. Techniques which have proven to be successful have also been expensive and labor intensive. Within electronics training this has proven to be especially true, probably due to a large variety of devices and technologies which are used in military equipment. This continuing investigation has attempted to develop methods for applying computer based simulation techniques to assist in the training of electronic techniques to assist in the training of electronic technicians more thoroughly and less intensive labor than must be done currently. The goal has been to develop in the student an understanding of the cause-and-effect relationships between circuit elements, by enabling the computer to interact with the student in English as he is led through an investigation of the operation and repair of a malfunctioning electronic device. The development of programming and pedagogical techniques to achieve this end has been the focus of this research.

APPROACH

The work documented in this technical report consists of modification of internal aspects of the programming techniques used in earlier versions of this software to improve the speed and generality of the techniques, incorporation of these techniques into new instructional situations, and the description of supportive textual materials used when the materials are presented to students. The new instructional situation allows two students to engage in a gaming scenario in which each inserts faults into the other's circuits and them must predict the operation of the circuit as the opponent troubleshoots it. Scores are kept by the computer, based upon the ability of the fault inserter to predict the effects of his inserted fault upon elements of the circuit. The internal changes improve the ability of the program to model the student's understanding of the circuit.

RESULTS

The gaming situation and textual support materials developed will be used in an experiment in the second phase of this contract to determine how well this technique works with actual students. The success of this effort will be documented in the final technical report.

CONCLUSIONS

The new technique developed for processing English language requests entered by the student, appears to be both more general and faster than the one previously used. The impact upon the overall instructional situation will be known after the second phase.

TABLE OF CONTENTS

	Page
Chapter 1: Introduction.	1
Overview.	1
Future Plans.	3
Chapter 2: Lesson Material	4
Lesson Text	4
Fundamentals of a Regulated Power Supply.	5
Introduction.	5
Regulators.	5
Circuits.	7
The Series Regulating Element	9
Comparison and Reference.	9
Controller.	11
Regulating Element.	12
Current Source.	14
Voltage Reference	18
Current Limiting.	18
Controls.	19
Chapter 3: Gaming Scenarios.	20
Technical Details	20
Protocol of Two Students Engaged in a Gaming Scenario	22
Chapter 4: Developments in the Natural Language Front-End	29
Augmented Transition Networks	29
Reasons for a Change.	32
Fuzziness	35
Other Object Languages.	36
Chapter 5: Deductive-Based Circuit Understander.	37
An Overall Perspective of Troubleshooting	37
Towards a Structural Theory of Troubleshooting. . .	38
Simple Local Analysis	40
A Simple Theory of Troubleshooting.	45
Unexpected Complexities of the Simple Theory. . . .	46
The Necessity and Utility of Other Knowledge. . . .	59
Extensions.	61
Top Level User Flags and Functions.	61
Protocol with Trace of What "New" Information is Being Deduced.	63
References	68
Appendix A: Contrasting Formalisms	69
The NODE Rule from SOPHIE's LISP/BNF Semantic Grammar	
BNF Version	69
LISP Version.	70
The NODE Network from SOPHIE's Semantic ATN	71
Sample of NODE Network Parsing a Phrase	73
ATN Object Code of NODE Network	75

"Intelligent" Computer Assisted Instruction
(CAI) Applications

CHAPTER 1
INTRODUCTION

In this report we will describe the technical details of the first five months work on our contract "Application of Intelligent Computer Assisted Instruction (CAI)." As an interim report we shall make every effort to describe details in sufficient depth to enable the interested reader to understand fully the theory behind the programs we have been developing. Before delving into these details we provide a brief overview of our work.

OVERVIEW

We have given the highest priority to laying the ground work for performing a series of experiments at Bolt Beranek and Newman (BBN) and Lowry AFB, CO, concerning student reactions to various versions of SOPHIE (i.e. SOPHisticated Instructional Environment). Since SOPHIE is not, in itself, a teaching system and relies exclusively on student initiative, we realized that before any meaningful experiment could be conducted some kind of tutoring system had to be constructed "on top of" SOPHIE. Toward this end, two separate projects were initiated. The first consisted of writing an extensive lesson sequence which would gradually ease the student a) into using the facilities offered by SOPHIE, and b) into having a conceptual understanding of the complex power supply currently being modeled in SOPHIE. It seemed questionable to teach the student how to troubleshoot such an instrument if he didn't really understand what the instrument was meant to do and how it actually worked. In fact one of the issues we want to experimentally investigate concerns how the student can use the troubleshooting scenario to gain a better understanding of the underlying teleology of the instrument. We believe that a better understanding of electronics arises from troubleshooting, and better troubleshooting behavior can arise out of increased awareness of the structure and function of a circuit.

Simultaneously to designing this lesson plan or agenda, we have been extending SOPHIE in the direction of having it support a variety of gaming scenarios. These scenarios focus on having two persons or teams "compete" in troubleshooting a circuit wherein, alternately, one team inserts a fault into the circuit and the other team attempts

to isolate it by making measurements. Since each measurement has an assigned cost, a total score can be computed by summing up the cost of all the measurements.

Within this gaming environment, there are a host of learning strategies which can be explored, many of which were impossible, or at least impractical, to realize before SOPHIE. These scenarios also have the advantage of minimizing the need for a tutoring system to have complex models of the student since the student makes his own decisions much in the spirit of "learner control" as found in TICCIT.*

We have thus far extended SOPHIE to support many of the desired gaming scenarios and are therefore in a position to experiment with these scenarios along with the more traditional use of SOPHIE as directed by our lesson plan.

The third avenue of research we have been pursuing is directed more toward generative CAI issues. The first of these concerns a major attempt to formalize and generalize the novel semantic parser underlying SOPHIE so that it can be more easily used in other CAI systems and in particular so that it can be used on a CAMIL** machine directly! The techniques of semantic parsing have proved to be a significant advance over what was previously thought possible. In particular, the leverage that semantic grammars provide in handling context and dialogue problems have enabled friendly and useful front-end natural language processors to be constructed where the more traditional syntax-based schemes have run into serious obstacles. However, on the negative side, semantic grammars have appeared to be excessively ad hoc, lacking any formal structure.

Our response to the ad hoc nature of semantic grammars was to take a high level formal language which was designed for writing syntax grammars and adapt it for use in writing our semantic grammars. The beauty of this approach is that not only is this formal language designed to expedite the writing and understanding of complex grammars, but that we have written, under an in-house project, a compiler for programs (i.e. grammars) written in this high level language. This compiler generates a program in the target language (LISP)*** which is the natural language processor. That is, the resulting program constitutes a parser plus its grammar. What makes this exceptionally exciting is that the compiler (running under LISP) can generate a CAMIL or

---

*An Overview of the TICCIT Program, The Mitre Corporation, January 1974.

**Computer Assisted/Managed Instruction Language, the language being used in the Advanced Instructional System project at Lowry AFHRL/TT.

***INTERLISP Reference Manual, Warren Teitelman et. al., Xerox Palo Alto Research Center, December 1975.

PASCAL* program comprising a complete natural language parser plus grammar! Although SOPHIE's entire grammar has now been recoded in this high level language and has been passed through the compiler thereby generating an "object" program, the compiler has not been made to generate CAMIL code, nor has the run time environment for CAMIL been written. We plan to wait on this until CAMIL is operational. However, we have included enough discussion of this language and compiler in later sections to indicate that this task should be straight-forward.

Finally, we have started to extend a more axiom-like circuit understander (developed last year under our Army Research Institute (ARI) contract**) so that it can engage in teleological reasoning about a circuit. Unlike SOPHIE this system is surprisingly small and has very little circuit dependent knowledge. It can provide causal explanations for its "findings" and can generate, on the fly, all information that can be deduced from a given measurement. Later in this report we provide a description of this system, a brief discussion of our current extensions and a simple example of its operation.

This system can either reside on top of SOPHIE thereby augmenting SOPHIE's explanatory capabilities or it can be coupled to our semantic net to augment its capabilities. In either case the techniques being developed in it will play a substantial role in constructing a generative system that can "understand" novel circuits. However, the full development of this system is a long range project.

FUTURE PLANS

We are currently waiting for a PLATO*** terminal interface box. When this arrives we shall modify SOPHIE so that it can use the graphics facilities provided by this terminal. If this arrives soon, we shall consider using it in our experiments. Otherwise we shall have to settle for the less ideal non-graphics terminals.

We shall also be modifying SOPHIE to enable it to handle the range of circuits that occurs in our lesson agenda. We also will be bringing up a separate semantic net generator and retrieval component for storing and retrieving causal knowledge about a circuit. This system will be coupled with lesson agendas to enable us to explore simpler, quasi-intelligent CAI systems.

---

*The Programming Language PASCAL, N. Wirth, Acta Informatica, 1, 35-63, 1971.

**ARI contract number DAHC19-74-C-0060.

***The TUTOR Language, Bruce A. Sherwood, Computer-based Education Research Laboratory, University of Illinois, June 1974; PLATO Curricular Materials, Elizabeth R. Lyman, Computer-based Education Research Laboratory, University of Illinois, December 1975.

CHAPTER 2
LESSON MATERIAL

This chapter contains a linearized version of our tentative lesson text. In writing it as a self-contained linear text, we have made numerous assumptions about the level of prior knowledge of the user. The text also contains transitional paragraphs (as opposed to disjoint frames) which open up the possibility of using it as a booklet which a student could peruse before using the system. However, our intention is to break this material up into frames with branching capabilities dependent on the level of performance of the given student. We are also considering creating a semantic/conceptual network of the frames so that a student could traverse the material in various ways and so that the system could automatically extract relevant nodes as determined by the answer evaluator.

Once the various sub-circuits have been encoded in SOPHIE's simulators and semantic nets, we shall also explore opening up parts of the lesson to more student initiative experimentation. After the lesson material becomes more finalized we hope to run instructors through it in order to get their reaction. We recognized that this lesson may be already too long and too complex to conduct extensive experimentation with it. This raises the possibility that we might want to run subjects over a longer period of time. However, such considerations are probably premature until we actually run some of our own test subjects through it.

LESSON TEXT

The power supply with which you are experimenting is the Heathkit IP-28, a good example of a modern, electronically regulated, solid-state laboratory power supply. This section of the on-line electronic workshop and tutorial will discuss the principles and theory underlying the operation of the IP-28.

Our approach in stating "how it works" (for troubleshooting) will be to first show you what must be accomplished in a general way by <u>any</u> circuit fulfilling the function of a regulated power supply. This will be a general or "top-level" description without specific form; that is to say, at this stage we won't consider any specific circuit but will instead describe building blocks functionally -- by the job they have to do. Later, we shall introduce and explain circuits which are simplified versions

of the real circuits embodied in the IP-28. SOPHIE will model those so you can experiment with them too. Finally, we shall elaborate these simple circuits into the more complex ones of the real IP-28.

FUNDAMENTALS OF A REGULATED POWER SUPPLY

INTRODUCTION

The primary function of a power supply is to change the alternating current of power utilities to direct current. Thus, a power supply needs to have a rectifier and filter to do this.

A rectifier and filter will provide dc power but such a combination alone has an unstable output voltage; that is, its output voltage changes as the load current changes. Since modern electronic equipment requires very steady voltages, a sophisticated power supply like the IP-28 has additional circuits to stabilize the voltage provided by a simple rectifier and filter, and to provide the facility to vary it. Most of the power supply's complexity lies in these additional circuits which are related to the rectifier and filter as shown in Figure 1.

REGULATORS

The class of regulators we shall be concerned with are called <u>active</u> regulators. This is because they maintain constant output voltage by actively <u>doing</u> something. If, for any reason, the output voltage departs from its nominal or required value, these types of regulators <u>react</u> in a manner which returns the output voltage to where it should be.

There are three important functions which must be performed by a regulator to accomplish this task. These are fundamental and easily stated:

1. <u>Control of output voltage</u>
   The regulator must be able to vary the output voltage over a wide range. It does this by means of a <u>REGULATING ELEMENT</u>.

2. <u>Sensing and Comparison</u>
   In order to correct any deviation of the output voltage from its required value, the regulator must be able to sense that there has been a deviation. It does this by comparing the output voltage to a REFERENCE VOLTAGE and generating a control signal proportional to the degree of discrepancy.

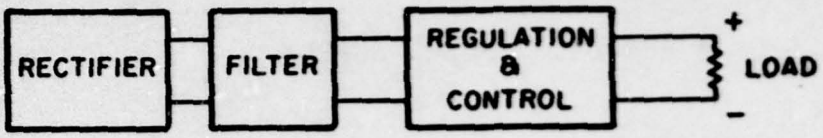


Figure 1

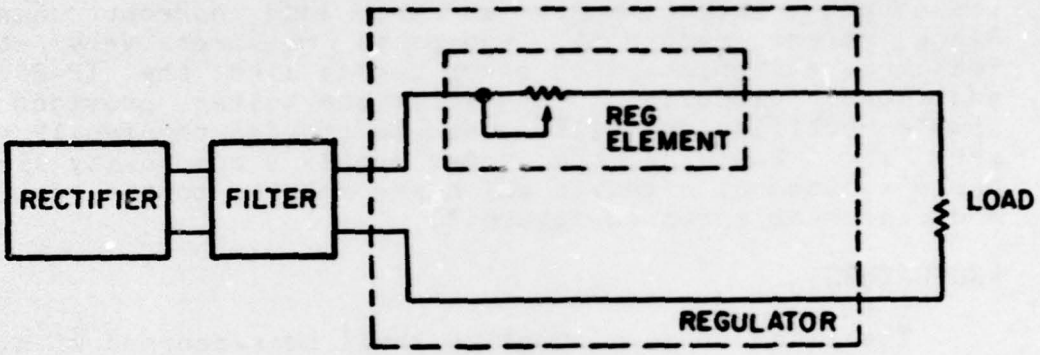


Figure 2

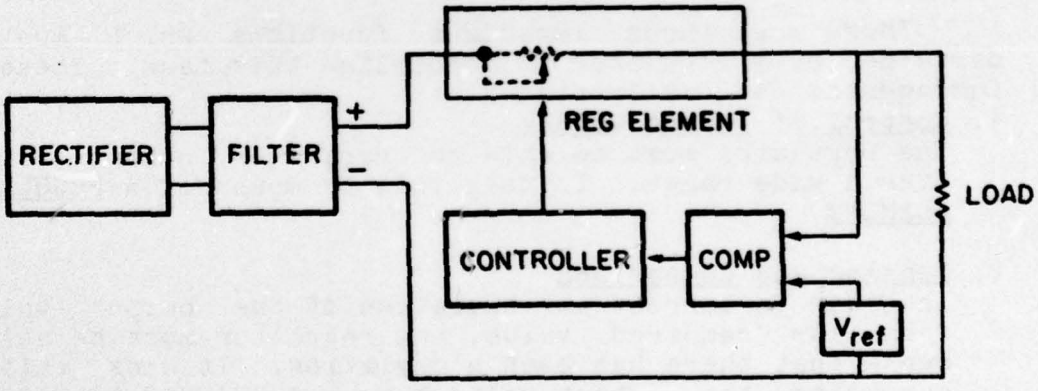


Figure 3

3. CONTROL OF THE REGULATING ELEMENT
   Finally, the signal generated by the sensing and comparison function must be translated into a command which can adjust the regulating element. We shall call the unit which performs this function a CONTROLLER. It is the job of the controller to accept a discrepancy or error signal from the comparison function and to translate this into a command which can adjust the REGULATING ELEMENT.

These three functions are common to a great many power supplies, and in fact, to a whole class of systems known as negative feedback systems. There are many ways of implementing these functions. In particular, the nature of the regulating element determines the classification of the regulator, i.e. whether it is a "shunt regulator", "series-regulator," or other type of regulator.

The IP-28's regulator is of a type called a SERIES-REGULATOR or SERIES-PASS regulator. It is called this because the regulating-element is effectively placed in series with the load and controls the output voltage by performing essentially like a variable resistor as shown in Figure 2.

Let's review briefly what we have discussed until now. The rectifier and filter change ac voltage into a dc voltage which is applied to the load through a series REGULATING ELEMENT whose effective resistance can be varied thereby changing the voltage across the load. A comparison function compares the output voltage to a fixed REFERENCE voltage source (Vref) and if there exists a difference between the two, actuates the controller. The latter generates a command to adjust the REGULATING ELEMENT so that the discrepancy between output voltage and reference voltage disappears. This feedback action is continuous and dynamic whenever the power supply is in operation.

Having understood the functional operation of our power supply, we are ready now to fill in the blocks with electronic components.

CIRCUITS

Figure 4 is a first realization of a power supply built along the lines suggested by the functional block diagram of Figure 3. The circuits shown are simplified versions of the real ones found in the IP-28 but they work nevertheless and SOPHIE has modeled them so you can experiment before tackling their more complicated versions which do not differ

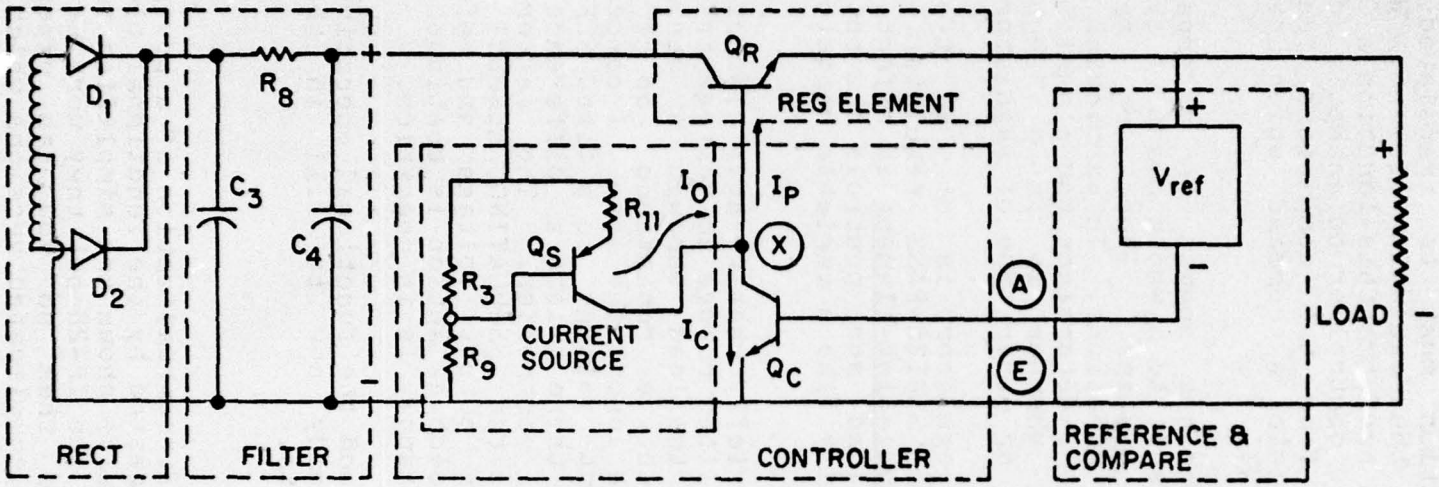

D1
D2
R8
C3
C4
RECT
FILTER
QR
REG ELEMENT
R11
IO
IP
QS
R3
R9
CURRENT SOURCE
IC
QC
X
A
E
CONTROLLER
Vref
+
-
LOAD
REFERENCE & COMPARE

Figure 4

in principle. The latter are more complicated only for reasons of increased performance, overcoming limitations of individual components, adding protection, etc.

THE SERIES REGULATING ELEMENT

The series regulating element of our simplified power supply is a single transistor (Qr). Its collector is connected to the output of the filter and its emitter toward the load. The collector-emitter resistance of Qr is thus placed in series with the load. Since it is a property of a transistor operating in its active region (that its collector-emitter resistance varies as the base current is changed) Qr behaves effectively as a variable resistor whose resistance is controlled by a current. As the base current (IP) of Qr varies, the voltage drop from collector to emitter changes and consequently, the voltage applied to the load changes according to a simple voltage divider action. We have then, our means of controlling the output voltage of the power supply. It is important to realize that while Qr does indeed behave like a current controlled variable resistance, it nevertheless must still conform to the requirements of any transistor operating in its active region as an amplifier. Thus, in a non-faulted functioning circuit, Qr has its base-emitter junction forward-biased, its collector-base junction reverse-biased, and its collector current related to the base current by:

$$\frac{Ic}{Ib} = \beta$$

where β= dc current gain. (This symbol is called the beta of a transistor.) Remember this when you begin to troubleshoot the IP-28.

COMPARISON AND REFERENCE

Before we discuss the CONTROLLER which varies the collector-emitter resistance of Qr, let us see, with reference to Figure 5, how an error, or discrepancy signal is generated in the IP-28. This error signal, you will recall, is what tells the controller to readjust the regulating element Qr in a way which brings the output voltage back to its required value. The functional block diagram of Figure 5, shows the error signal being generated by a mysterious comparison block which accepts the output voltage and the reference voltage as inputs, and which outputs an error signal, presumably when the output voltages of the power supply does not match the reference voltage.

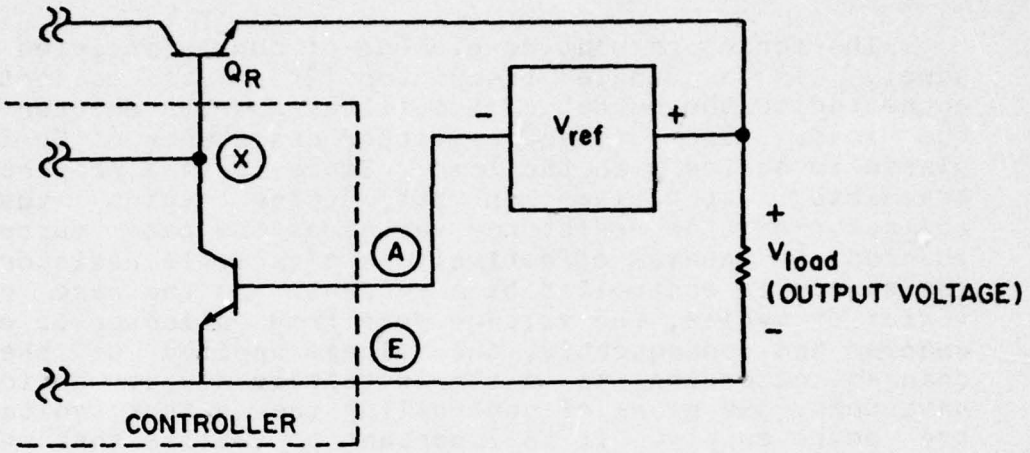


Figure 5

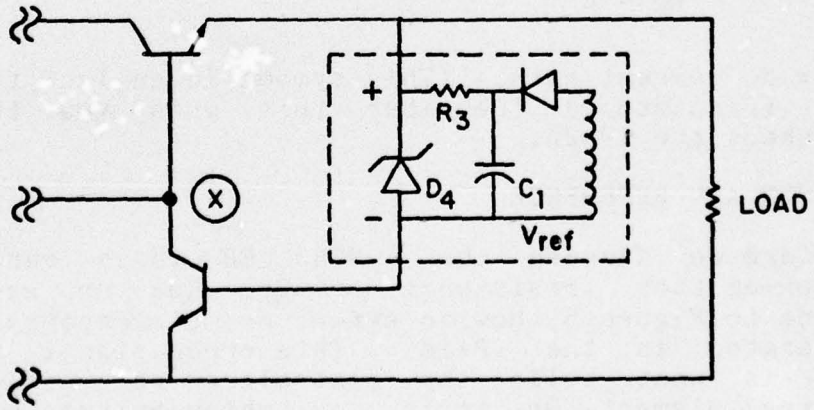


Figure 6

In the IP-28, the comparison between output and reference voltages is performed in a deceptively simple fashion: output voltage and reference voltage are subtracted from one another by simply "bucking" them, that is, by placing them in series with one another. It may be easier to see this if we redraw the pertinent portion of Figure 4 as shown in Figure 5. Now it is clear from Figure 5 that Vae, the voltage between nodes A and E which is applied to the controller, is just the difference between reference voltage Vref and output voltage, Vload. Therefore, when the output voltage rises above Vref, Vae becomes positive and drives the controller. The latter in turn adjusts Qr as we shall see shortly. First, however, let's discuss Vref briefly. The primary requirement for Vref in this application is that it be isolated or "floating." A battery having the required voltage would be suitable and could in fact be wired into a circuit. Batteries do run down, however, so, in the IP-28, Vref is a complete power supply all by itself. It has a transformer, a rectifier, a filter, and of course, a regulator! For our simplified power supply it is enough to have Vref consist of the crudest type of half-wave rectifier, capacitor, and zener diode as a regulating element (Figure 6).

CONTROLLER

As we have seen, Vae is the result of the comparison function. It is the job of the controller to accept Vae as input and to translate this into a suitable, compensating drive for Qr, the regulating element. Reference to Figure 4 shows the controller to be made of a constant current source consisting of Qs, R3, R9, and R11 and of a control transistor Qc. The purpose of the constant current source is to maintain an unvarying current Io flowing out of the collector of Qs. For the moment, we shall postpone a discussion of how the constant current works, but it is important at this state to realize that Io never changes in a properly functioning circuit. Notice that Io flows into junction X and divides into two currents (Ip) and (Ic). If transistor Qc is off, all of Io flows in the upper branch. This condition presents the maximum drive to Qr, causing the latter's effective resistance to be the lowest possible. It is also the condition under which the power supply delivers its maximum current to the load. When Qc begins to conduct as a result of positive drive voltage Vae being applied to its base, a portion of Io is diverted away from the base of Qr resulting in less drive to that transistor and a consequent increase in its collector emitter resistance. The controller thus functions by using the input voltage Vae to vary the conduction through Qc which in turn shunts

current away from the base of Qr.

Having discussed the circuits making up all the functional blocks (with the exception of the rectifier and filter since you are probably quite familiar with these) we can now summarize the operation of the IP-28.

Unregulated dc voltage is applied to the load through the collector-emitter path of Qr, the series-regulating transistor. When the supply is first turned on, the output voltage is zero and therefore less than the reference voltage Vref. Vae is negative and Qc completely cut off. All of Io thus flows into the base of Qr and the output voltage tries to rise to its maximum possible value which, by design, is greater than Vref. At some point, the output voltage becomes greater than Vref causing Vae to become positive. Qc begins to conduct and shunts current away from the base of Qr. The voltage drop across the collector and emitter of Qr increases as a result of the decrease in base drive Ip and continues to decrease until the output voltage stabilizes at some value very close to Vref. Henceforth, the output voltage remains equal to Vref and the regulating action just described acts continually to maintain it at that value.

By now you should be thoroughly comfortable with the concepts involved in the simplified IP-28, so it is time to consider the "real," more complex IP-28. Figure 7 is the schematic for it as published by Heathkit. Although it looks somewhat more complicated, it is identical in function to the simplified IP-28. In the following section, we shall consider the differences between the two.

The "real" IP-28 has:

..... a more complex regulating element
..... a more complex current source
..... a more complex voltage reference
..... Protective circuits to limit maximum current out of the supply
.....a few controls to vary output voltage and current limits

Let's look at the reasons for the added complexity.

REGULATING ELEMENT

In the simplified IP-28 the regulating element consisted of a single transistor Qr. If the supply had to deliver 1 amp to the load and the output of the current source were .6 ma, what do you think the β of Qr should be?

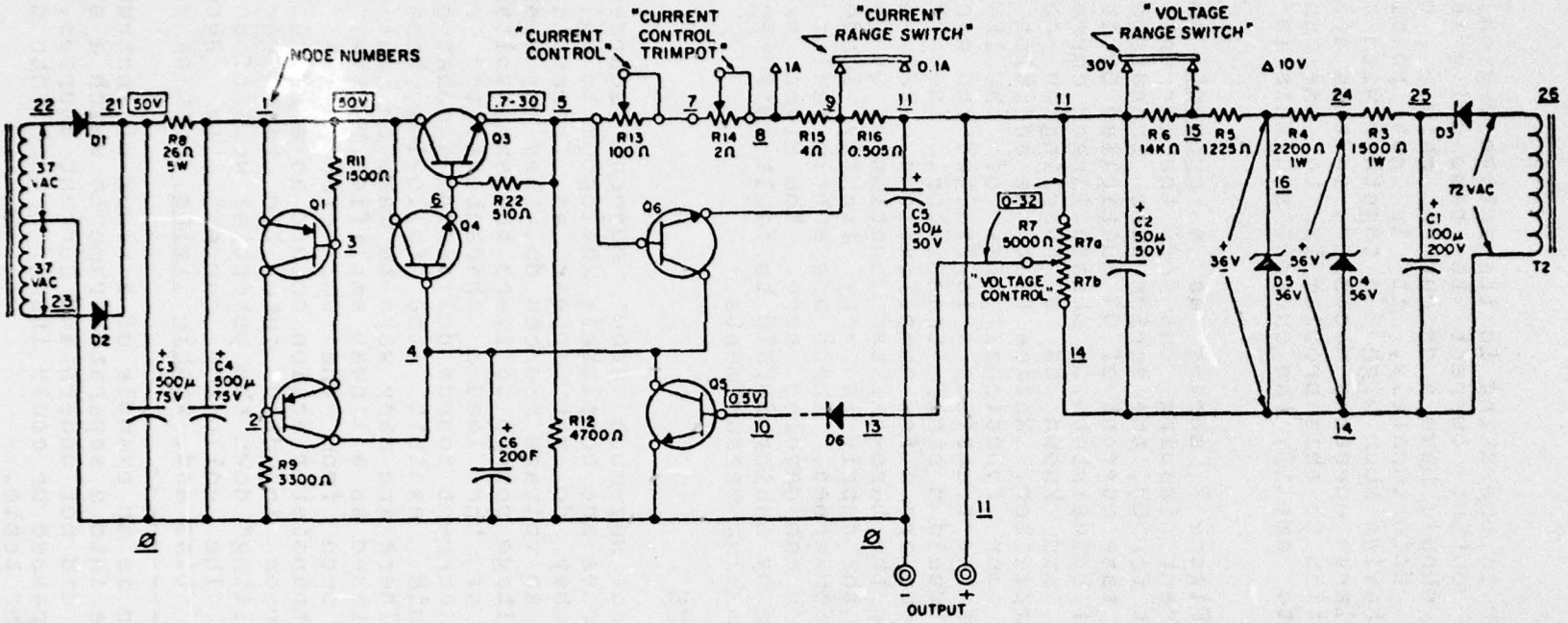

NODE NUMBERS
"CURRENT CONTROL"
"CURRENT CONTROL TRIMPOT"
"CURRENT RANGE SWITCH"
"VOLTAGE RANGE SWITCH"
1A
0.1A
30V
10V
50V
.7-30
0-32
0.5V
37 VAC
37 VAC
72 VAC
D1
D2
D3
D4
D5
D6
R8 26Ω 5W
R9 3300Ω
R11 1500Ω
R12 4700Ω
R13 100Ω
R14 2Ω
R15 4Ω
R16 0.505Ω
R22 510Ω
R7 5000Ω
R7a
R7b
R6 14KΩ
R5 1225Ω
R4 2200Ω 1W
R3 1500Ω 1W
C1 100μ 200V
C2 50μ 50V
C3 500μ 75V
C4 500μ 75V
C5 50μ 50V
C6 200F
Q1
Q3
Q4
Q5
Q6
36V
56V
56V
"VOLTAGE CONTROL"
T2
OUTPUT

Figure 7

Since Qr is operating in its active region, the ratio of required output current to base drive is about 1700. Therefore Qr should have a dc current gain β of at least that much. Unfortunately, it is not possible to find transistors having such high B's together with the capacity to pass the large currents necessary in this application. A typical solution to this problem is to use an additional transistor to amplify the current and this is what is done here.

Q4 in Figure 7 serves as a current amplifier for control current issuing out of the current source. The drive current for Q3, the series-pass transistor, is thus roughly the base current of Q4 multiplied by its B. Such a connection of transistors to obtain larger current gain is a classic one and known as a "Darlington connection" or Darlington amplifier. Notice that the addition of Q4 has not changed our functional view of the IP-28; in fact, unless one of the transistors is faulted you can treat the two like you would a single transistor, (see Figure 8). Of course there are a few complications. R22 and R12 are added to keep the base-emitter junction of Q4 forward biased or "on" when the supply is very lightly loaded. Simple? Yes, but be forwarned. There are subtle faults with strange symptoms which can develop here. You might explore these possibilities by asking SOPHIE to fault this region (perhaps R22) and by making measurements.

CURRENT SOURCE

We did not say much about current sources earlier.* Current sources are completely analogous to voltage sources but since they do not appear as frequently in our environment as voltage sources do, they may seem strange at first. A voltage source delivers the same voltage to a load regardless of the load's characteristics as shown in Figure 9. A current source delivering a constant current of 1 amp would exhibit the following behavior (see Figure 10). There are many ways to make current sources. A transistor biased as a linear amplifier behaves as a current source when seen from the collector (see Figure 11). Recall that transistor action occurs as a result of carriers (which have been injected into the base region by the emitter) "falling" down the potential well of collector-base back bias. The collector current is determined by emitter-base bias and, within limits, nothing you can do to

---

*This section is an example of a block of instruction which could be made into a separate frame to which a student could branch if he did not understand current sources. The frame could be bypassed or could itself branch into more esoteric aspects of the topic.

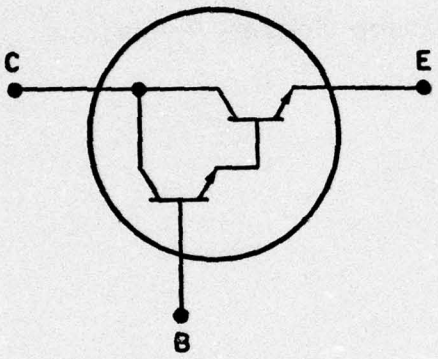
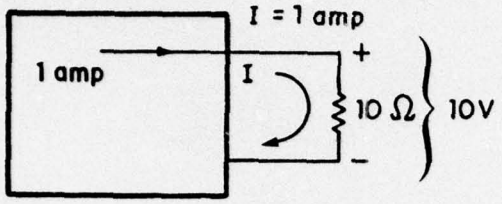
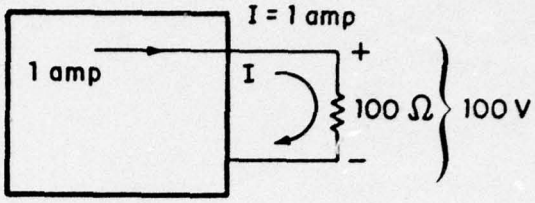


Figure 8

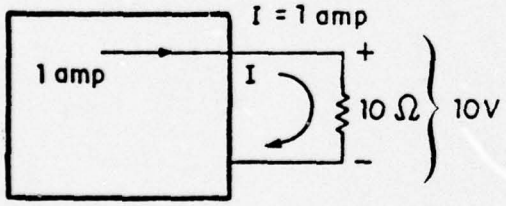


10 volt voltage source delivers 1 amp to 10Ω load.

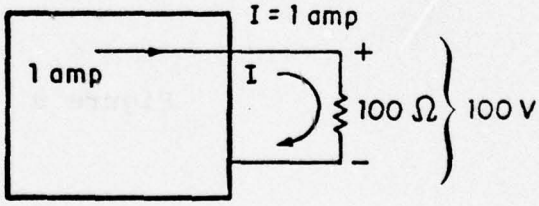


When the load is changed to 100Ω, the voltage delivers 0.1 amp. Voltage remains at 10 volts.

Figure 9 Voltage Source Behaviour



1 amp current source delivers 1 amp through 10Ω load.



When the load is changed to 100Ω, the current source continues to deliver 1 amp. The voltage across the load now rises to 100 volts!

Figure 10 Current Source Behaviour

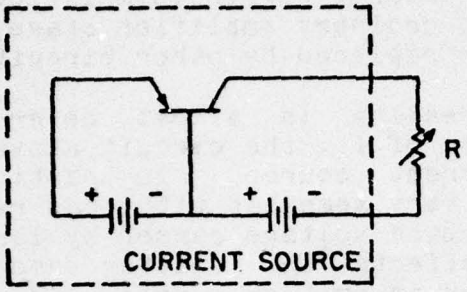


Figure 11

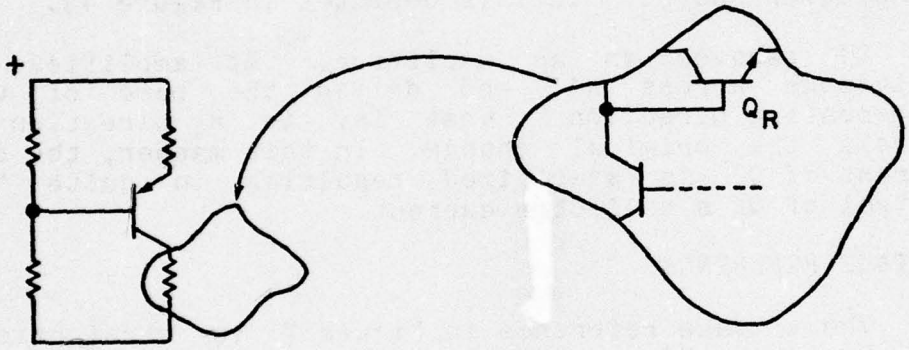


Figure 12

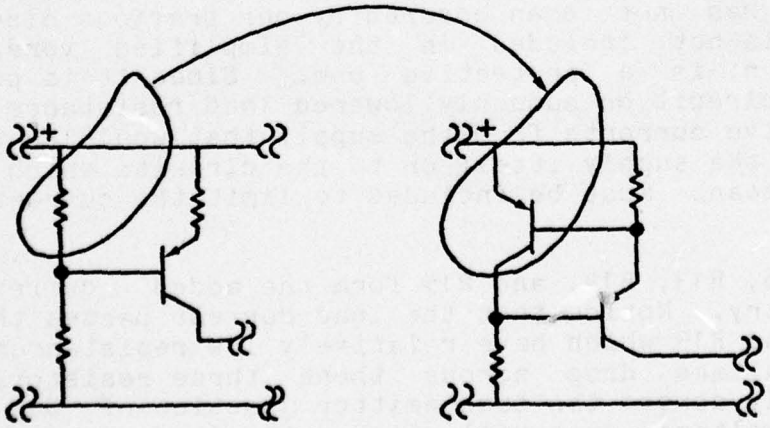


Figure 13

the collector circuit will change the collector current. The current source of the simplified power supply is thus seen to be an ordinary amplifier stage, with the collector load resistor replaced by other circuits (see Figure 12).

Since reality is almost never as neat as our idealizations of it, the circuit above falls short of being a perfect current source. In particular, the collector current does vary somewhat with load resistance. The change in collector-base voltage caused by large variations of load resistance affects the emitter base circuit which in turn causes changes in emitter current and hence, in collector current.

The more complex current source of the "real" IP-28 incorporates an additional transistor Q1 which replaces R of the simpler source. This is depicted in Figure 13.

Q1 behaves as an amplifier. It amplifies small variations across R11 and drives the base of Q2 in a compensating direction that is, in a direction which cancels the original change. In this manner, the emitter current of Q2 is stabilized resulting in quite "tight" control of Q2's collector current.

VOLTAGE REFERENCE

The voltage reference in Figure 7 is straightforward. T2, D3, C1, R3 and D4 form a half-wave rectifier, filter, and zener regulator which you'll recognize to be identical to the reference supply of the simplified IP-28.

CURRENT LIMITING

There is an additional function added in the real IP-28 which has not been covered by our previous discussion and which is not included in the simplified version. That function is a protective one. Since it is possible (by short-circuit or suddenly lowered load resistance) to draw excessive currents from the supply that would be damaging to either the supply itself or to the circuits which it powers, some means must be included to limit the current to a safe value.

Q6, R13, R14, and R15 form the added current-limiting circuitry. Notice that the load current passes through R13, R14, and R15 which have relatively low resistances and that the voltage drop across these three resistors is placed directly across the base emitter junction of Q6. Clearly, this voltage drop will rise as the load current rises. As

long as this drop does not reach the value of base-emitter voltage required to "turn on" Q6, the latter remains effectively out of the circuit. However, when the output current of the power supply reaches a certain value, the voltage drop becomes high enough to drive Q6 into an active state and collector current begins to flow from the current source into the load. Since the current source delivers a constant current, there is less available to drive the regulating element and the power supply shuts down, i.e. refuses to deliver any more current. We should mention that when this happens, the power supply undergoes a transition to a different mode of operation. The main feedback element is now the chain of resistors across the base-emitter of Q6, and it senses output current instead of output voltage. In effect, the power supply has shifted mode and instead of operating as a voltage source, it now operates as a current source (though not a very accurate one). It is well worth your time to study this transition using the experimental facilities of SOPHIE.

CONTROLS

The real IP-28 does have controls to adjust the output voltage, the point at which the supply limits, and to change ranges of operation. We leave it as an exercise for the student to discover how this is done.

CHAPTER 3
GAMING SCENARIOS

SOPHIE can be viewed as a collection of tools which can be used to quickly create new scenarios for training. As an example of one such use, we built the following gaming scenario. The game is played by two trainees, Student A and Student B who are each using his own terminal. The terminals communicate in such a way that each student can work at his own pace and only has to wait for his opponent when absolutely necessary. Student A introduces a fault into the circuit. Student B must then find the fault by performing a series of measurements. Each measurement has a cost and a total cost is computed for the series of measurements he makes in discovering the fault. While Student B is troubleshooting, Student A is predicting the results of the measurements made by Student B. The cost of Student B's measurements and the results of how well Student A predicted the consequences of his fault are combined to give a score for Student A. After the fault is isolated, the roles are reversed and the game is played again.

TECHNICAL DETAILS

The communication between two students in a gaming scenario is done via files. Each student has his own version of SOPHIE (running in its own fork) so that he works at his own pace and only has to wait for his opponent when absolutely necessary. The situation is shown in Figure 3.

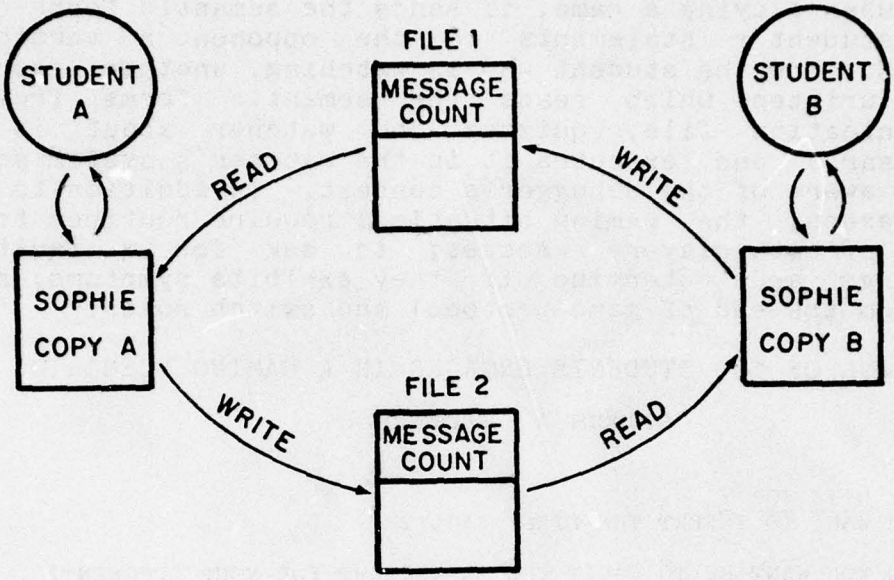


FIGURE 3

Each copy of SOPHIE has a file that it can write on to send messages to the other, and a file that it reads from to receive messages. The communication channels are initially set up by having one job open the files and the other job waiting for them to be opened. This is done using the new SOPHIE commands "Play with job number X" and "Wait for an opponent". The active job must have the passive job's number to allow more than one game at once. The number is incorporated into the file names to insure their uniqueness.

The following protocol is used on each communication file to insure that messages get passed correctly. The first word of each file contains a count of the number of messages which have been written on the file. This is set to zero when the file is first opened. To send a message, the writing job prints (using regular LISP routines) the message on the file at the end. When the writing has finished it increments the message count. (This prevents the reader from getting incomplete messages.) To receive a message the reading job must keep count of the number of messages it has read from the file. There is a message waiting to be read whenever the count of the number of messages on the file (the first word) is greater than the number of messages the receiving job has read. When this

occurs, the receiving job reads (using LISP READ) a message from the file beginning where its last read message ended.

During a game, one student is debugging the instrument and the other student is being quizzed about the debugger's measurements. To implement this within SOPHIE required the following changes: the normal SOPHIE exec was rewritten so that when playing a game, it sends the semantic forms of all the student's statements to the opponent's version of SOPHIE. For the student who is watching, another executive was written which reads the semantic forms from the communication file, quizzes the watcher about it (if necessary) and executes it in the watcher's system so that he is aware of the debugger's context. In addition to these two execs, the gaming situations require routines to keep track of both players' scores; to ask for a fault and settings and determine if they exhibits symptoms; and to perform the end of game protocol and switch roles.

PROTOCOL OF TWO STUDENTS ENGAGED IN A GAMING SCENARIO

STUDENT A's PROTOCOL

```
>>PLAY WITH JOB 7
DO YOU WANT TO INSERT THE FIRST FAULT?
> YES
HOW DO YOU WANT ME TO FAULT THE INSTRUMENT FOR YOUR OPPONENT?
>> LET R22 HAVE THE WRONG VALUE.
IS IT OKAY IF I USE 25 FOR THE VALUE OF R22 ?
> NO
WHAT VALUE SHOULD I USE?
> 8
NOW THE VALUE OF R22 IS 8
```

Student A is allowed to fault any part in the instrument and can increase his score by choosing a fault which is difficult to find. However, he is expected to be able to predict the consequences of his fault in two ways. First he must set up the instrument controls so that the instrument is malfunctioning. It is to his advantage to choose settings which make his fault look like lots of other faults. This encourages him to do careful analysis of the effects of the controls. Second, each time Student B makes a measurement, Student A will be asked to predict the results of that measurement. This forces him to do analysis on how his fault interacts with all of the functional blocks in the circuit.

```
PLEASE GIVE ME SOME CONTROL SETTINGS WHICH WOULD CAUSE YOUR FAULT
TO EXHIBIT SOME EXTERNAL SYMPTOM.
WHAT SHOULD THE CURRENT CONTROL BE?
> 1.0
WHAT SHOULD THE CURRENT RANGE SWITCH BE?
> HIGH
WHAT SHOULD THE LOAD BE?
> 2000
WHAT SHOULD THE VOLTAGE CONTROL BE?
> .8
WHAT SHOULD THE VOLTAGE RANGE SWITCH BE?
> HIGH
THOSE DON'T SHOW A SYMPTOM.
THE OUTPUT VOLTAGE IS 24.3 VOLTS WHICH IS CORRECT
BECAUSE IN A WORKING CIRCUIT THE OUTPUT VOLTAGE IS 24.3 VOLTS.
```

```
WOULD YOU LIKE TO TRY A DIFFERENT FAULT?
> NO
```

The first settings Student A picked didn't show a symptom. He is penalized because Student B knows this fact before he even begins to troubleshooting.

```
PLEASE GIVE ME SOME CONTROL SETTINGS WHICH WOULD CAUSE YOUR FAULT
TO EXHIBIT SOME EXTERNAL SYMPTOM.
WHAT SHOULD THE CURRENT CONTROL BE?
> .94
WHAT SHOULD THE CURRENT RANGE SWITCH BE?
> HIGH
WHAT SHOULD THE LOAD BE?
> 30
WHAT SHOULD THE VOLTAGE CONTROL BE?
> 1.0
WHAT SHOULD THE VOLTAGE RANGE SWITCH BE?
> HIGH
THOSE DO SHOW A SYMPTOM.
YOUR OPPONENT IS NOW FINDING THE FAULT.
```

For an interaction which Student B makes, Student A may be asked to predict the result. This querying of Student A is shown between the long line of dashes and the short one. Then Student A is shown what transpired at Student B's terminal.

```
-----------------
DO YOU THINK THE OUTPUT VOLTAGE IS HIGH, LOW OR ABOUT RIGHT?
> LOW
THAT'S RIGHT
YOUR SCORE SO FAR IS 1 OUT OF 1
-----
>> WHAT IS THE OUTPUT
THE OUTPUT VOLTAGE IS 1.86 VOLTS.
ACCUMULATED COST = 1


-----------------
-----
>> IS THAT CORRECT?
THAT VALUE IS INCORRECT BECAUSE IN A WORKING CIRCUIT
THE OUTPUT VOLTAGE IS 2.93 VOLTS.
ACCUMULATED COST = 5


-----------------
THE CURRENT CONTROL IS NOW 1.0.
DO YOU THINK THE OUTPUT VOLTAGE IS HIGH, LOW OR ABOUT RIGHT?
> LOW
THAT'S RIGHT
YOUR SCORE SO FAR IS 2 OUT OF 2
-----
>> WHAT IS THE OUTPUT WHEN THE CC IS 1
THE OUTPUT VOLTAGE IS 1.83 VOLTS.
ACCUMULATED COST = 6




-----------------
DO YOU THINK THE BASE EMITTER VOLTAGE OF Q5 IS HIGH, LOW OR ABOUT
 RIGHT?
> RIGHT
```

```
NO
THE BASE EMITTER VOLTAGE OF Q5 IS -.28 MILLIVOLTS.
WHICH IS INCORRECT BECAUSE IN A WORKING CIRCUIT
THE BASE EMITTER VOLTAGE OF Q5 IS .532 VOLTS.
YOUR SCORE SO FAR IS 2 OUT OF 3
-----
>> WHAT IS THE BASE EMITTER VOLTAGE OF Q5
THE BASE EMITTER VOLTAGE OF Q5 IS -.28 MILLIVOLTS.
ACCUMULATED COST = 8


-----------------
YOUR OPPONENT THOUGHT THERE WAS SOMETHING WRONG WITH Q5
WHICH TURNED OUT TO BE AN UNREASONABLE HYPOTHESIS
-----
>> COULD Q5 BE SHORTED?
NO
```

Here Student A may be queried as to whether or not Student B's hypothesis was correct.

```
-----------------
DO YOU THINK THE VOLTAGE ACROSS R7A IS HIGH, LOW OR ABOUT RIGHT?
> RIGHT
THAT'S RIGHT
YOUR SCORE SO FAR IS 3 OUT OF 4
-----
>> WHAT IS THE VOLTAGE ACROSS THE VOLTAGE CONTROL?
THE VOLTAGE ACROSS R7A IS 28.9 VOLTS.
ACCUMULATED COST = 10

-----------------
DO YOU THINK THE COLLECTOR CURRENT OF Q2 IS HIGH, LOW OR ABOUT RIGHT?
> RIGHT
THAT'S RIGHT
YOUR SCORE SO FAR IS 4 OUT OF 5
-----
>> WHAT IS THE COLLECTOR CURRENT OF Q2
THE COLLECTOR CURRENT OF Q2 IS -.59 MILLIAMPS.
ACCUMULATED COST = 14

-----------------
DO YOU THINK THE BASE EMITTER VOLTAGE OF Q3 IS HIGH, LOW OR ABOUT
 RIGHT?
> RIGHT
THAT'S RIGHT
YOUR SCORE SO FAR IS 5 OUT OF 6
-----
>> WHAT IS THE VBE OF Q3
THE BASE EMITTER VOLTAGE OF Q3 IS .48 VOLTS.
ACCUMULATED COST = 16
```

```
-----------------
YOUR OPPONENT TRIED TO REPLACE Q3
-----
>> REPLACE Q3
Q3 NOT REPLACED.
ACCUMULATED COST = 19
```

Here again Student A maybe queried as to whether or not it was reasonable to replace Q3.

```
-----------------
DO YOU THINK THE CURRENT THRU R22 IS HIGH, LOW OR ABOUT RIGHT?
> HIGH
THAT'S RIGHT
YOUR SCORE SO FAR IS 6 OUT OF 7
-----
>> IS THE CURRENT THRU R22 CORRECT
THE CURRENT THRU R22 IS 60 MILLIAMPS.
WHICH IS INCORRECT BECAUSE IN A WORKING CIRCUIT
THE CURRENT THRU R22 IS 1.8 MILLIAMPS.
ACCUMULATED COST = 27

-----------------
>> REPLACE R22
R22 REPLACED
ACCUMULATED COST = 30
```

Student B found the fault in a total cost of 30. Student A is given a percentage of this score depending on how well he predicted the consequences of his fault.

```
YOUR OPPONENT FOUND THE FAULT. HIS ACCUMULATED SCORE WAS 30
YOU KNEW 6 OF 7 THE MEASUREMENTS YOUR OPPONENT MADE SO YOU GET
25.7 POINTS.

NOW IT'S YOUR OPPONENT'S TURN TO CHOOSE THE FAULT.
 . . .
```

STUDENT B's PROTOCOL

```
>>WAIT FOR AN OPPONENT
YOUR OPPONENT IS JOB 32
```

Student B waits for his opponent to fault the instrument and set its controls so that it is exhibiting a symptom.

```
YOUR OPPONENT IS 'FIXING' THE INSTRUMENT.
HE HAS CHOSEN A FAULT AND IS SETTING UP THE INSTRUMENT CONTROLS.
HE HAS SET UP THE INSTRUMENT AS FOLLOWS:
CC=1.0, CR=HIGH, LOAD=2000, VC=.8, VR=HIGH
BUT THOSE SETTINGS DON'T EXHIBIT A SYMPTOM SO HE IS TRYING
TO PICK A DIFFERENT SET.
```

The settings which student A picked didn't show a symptom so he has to find some others that do. Student B now has the advantage of knowing some settings which don't show a symptom without being charged for it.

```
HE HAS SET UP THE INSTRUMENT AS FOLLOWS:
CC=.94, CR=HIGH, LOAD=30, VC=1.0, VR=HIGH
YOU MAY BEGIN TROUBLE SHOOTING.

>>WHAT IS THE OUTPUT
THE OUTPUT VOLTAGE IS 1.86 VOLTS.
ACCUMULATED COST = 1
```

Student B now troubleshoots the instrument with a cost being assigned to each of his measurements. External measurements are cheap.

```
>>IS THAT CORRECT
THAT VALUE IS INCORRECT BECAUSE IN A WORKING CIRCUIT
THE OUTPUT VOLTAGE IS 2.93 VOLTS.
ACCUMULATED COST = 5
```

Student B can use a working circuit to make reference measurements but he is charged for it.

```
>>WHAT IS THE OV WHEN THE CC IS 1
WITH THE CURRENT CONTROL NOW SET TO 1
THE OUTPUT VOLTAGE IS 1.83 VOLTS.
ACCUMULATED COST = 6

>>WHAT IS THE BASE EMITTER VOLTAGE OF Q5
THE BASE EMITTER VOLTAGE OF Q5 IS -.28 MILLIVOLTS.
ACCUMULATED COST = 8
```

Internal measurements are more expensive than external ones.

```
>>COULD Q5 BE SHORTED
SHOULD Q5 HAVE FAULT MODE EB/EC/BC = SH/SH/SH?
> NO
WHICH TERMINAL SHOULD BE MODELLED OPEN?
> COLLECTOR
OK, I'LL USE EB/EC/BC = SH/OP/OP
WHEN THE SETTINGS WERE:
        CC=1, CR=HIGH, LOAD=30, VC=1.0, VR=HIGH,
THAT FAULT DOES NOT EXPLAIN THE FOLLOWING SYMPTOM.
YOU OBSERVED THAT THE OUTPUT VOLTAGE WAS 1.83 VOLTS.
IF Q5 HAD FAULT MODE EB/EC/BC = SH/OP/OP IT WOULD BE 30 VOLTS.
IN A WORKING CIRCUIT IT IS 29.8 VOLTS.
SHALL I CONTINUE COMMENTING ON YOUR HYPOTHESIS?
> NO
```

At present hypothesis checking is free to encourage the student to present his ideas to SOPHIE for evaluation.

```
>>WHAT IS THE VOLTAGE ACROSS THE VOLTAGE CONTROL
THE VOLTAGE ACROSS R7A IS 28.9 VOLTS.
ACCUMULATED COST = 10

>>WHAT IS THE COLLECTOR CURRENT OF Q2
THE COLLECTOR CURRENT OF Q2 IS -.61 MILLIAMPS.
ACCUMULATED COST = 14
```

Internal current measurements are expensive.

```
>>WHAT IS THE VBE OF Q3
THE BASE EMITTER VOLTAGE OF Q3 IS .49 VOLTS.
ACCUMULATED COST = 16

>>REPLACE Q3
I AM GOING TO ASK YOU SOME QUESTIONS ABOUT HOW Q3 IS FAULTED.

ARE ANY JUNCTIONS SHORTED?
> NO
ARE ANY TERMINALS OPEN?
> YES
WHICH TERMINALS ARE OPEN?
> EMITTER
ANY OTHERS?
> NO
THAT'S NOT ACTUALLY WHAT'S WRONG. LET'S SEE IF IT WAS REASONABLE.

UNDER THE SETTINGS:
CC=1, CR=HIGH, LOAD=30, VC=1.0,VR=HIGH
THE FAULT WHICH YOU PROPOSED WILL PROPAGATE.
LIKELY FAULTS WHICH MIGHT OCCUR ARE:
R22 OPEN

WOULD YOU LIKE ME TO CONTINUE TESTING YOUR HYPOTHESIS IN A CIRCUIT
IN WHICH R22 IS OPEN ALSO?
```

```
> NO

WOULD YOU LIKE ME TO COMMENT ON YOUR HYPOTHESIS AS IF NO OTHER PARTS
HAD BLOWN?
> YES

THAT FAULT AGREES WITH ALL OF THE MEASUREMENTS THAT YOU HAVE TAKEN
UNDER THE PRESENT SETTINGS.
SHALL I CONTINUE COMMENTING ON YOUR HYPOTHESIS?
> NO
Q3 NOT REPLACED.
ACCUMULATED COST = 19

>>IS THE CURRENT THRU R22 CORRECT
THE CURRENT THRU R22 IS 61 MILLIAMPS WHICH IS INCORRECT
BECAUSE IN A WORKING CIRCUIT THE CURRENT THRU R22 IS 1.8 MILLIAMPS.
ACCUMULATED COST = 27
```

For this question the student is charged for both the internal current measurement and the reference measurement.

```
>>REPLACE R22
IS R22 OPEN OR SHORTED OR DOES IT HAVE THE WRONG VALUE?
> ITS VALUE IS WRONG
IS THE VALUE TOO HIGH OR TOO LOW?
> LOW
R22 REPLACED.
ACCUMULATED COST = 30
```

Student B finds the fault, is told Student A's score and is given a chance to insert a fault of his own.

```
YOUR TOTAL COST WAS 30
YOUR OPPONENT KNEW WHAT 6 OUT OF 7 OF YOUR MEASUREMENTS WOULD BE
SO HE GETS 25.7 POINTS.
NOW IT'S YOUR TURN TO INSERT A FAULT.
HOW DO YOU WANT ME TO FAULT THE INSTRUMENT FOR YOUR OPPONENT?
 . . .
```

CHAPTER 4

DEVELOPMENTS IN THE NATURAL LANGUAGE FRONT-END

During the past two years SOPHIE's natural language processor has developed, via ad hoc methods, into a system which is fast and surprisingly robust. The system uses a grammar based on semantic categories written in the programming language LISP. While expressing the grammar as programs has benefits in the area of efficiency and allows complete freedom to explore new extensions, the technique is lacking in perspicuity. The lack of perspicuity has three main problems which concern us here: (1) One is the difficulty encountered when trying to modify or extend the grammar. (2) The second is the problem of trying to communicate the extent of the grammar to either a user or a colleague. (3) The third is the problem of trying to re-implement the grammar on a machine which does not support LISP. These difficulties have been partially overcome by using a second, parallel representation of the grammar in a BNF-like specification language. This, however, requires supporting two different representations of the same information and does not really solve problems (1) or (3). To overcome these difficulties, SOPHIE's natural language front is being re-written in an Augmented Transition Network.

The Augmented Transition Network (ATN) formalism was developed as a conceptually and computationally efficient representation for natural language grammars, and has been used successfully in several natural language processing systems <Woods, Kaplan and Nash-Webber, 1972>[1], <Simmons, 1975>[2] and <Bates, 1975>[3]. Its advantages over phrase-structure formalisms are (1) perspicuity, (2) generative power, (3) efficiency of representation, (4) flexibility for experimentation, and (5) efficiency of operation.

AUGMENTED TRANSITION NETWORKS

Some years ago, Chomsky <Chomsky, 1957>[4] introduced the notion that the process of language generation or language recognition could be viewed in terms of a machine. One of the simplest of such models is the finite state machine. This machine starts off in its initial state looking at the first symbol (or word) of its input sentence and then moves from state to state as it gobbles up the remaining input symbols. The sentence is accepted if the machine stops in one of its final states after having processed the entire input string and is rejected otherwise. A convenient way of

representing finite state machines is as a transition graph, in which the states correspond to the nodes of the graph and the transitions between states correspond to its arcs. Each arc is labelled with a symbol whose appearance in the input can cause the given transition.

In an augmented transition network the notion of a transition graph has been modified in three ways. One is by the addition of a recursion mechanism which allows the labels on the arcs to be non-terminal symbols which themselves correspond to networks. The second is by the addition of arbitrary conditions on the arcs which must be satisfied in order for an arc to be followed. The third is by the inclusion of a set of structure building actions on the arcs, together with a set of registers for holding partially built structures*. Figure 4 is a specification of a language for representing augmented transition networks. The specification is given in the form of an extended context free grammar in which alternative ways of forming a constituent are represented on separate lines and the symbol "+" is used to indicate arbitrary repeatable constituents.

```
<transition network> := (<arc set> <arc set>+)
<arc set> := (<state> <arc>+)
<arc> := (CAT <category name> <test> <action>+ <term act>)
         (WRD <word> <test> <action>+ <term act>)
         (PUSH <state> <test> <action>+ <term act>)
         (TST <arbitrary label> <test> <action>+ <term act>)
         (POP <form> <test>)
         (VIR <constituent name> <test> <action>+ <term act>)
         (JUMP <state> <test> <action>+)
<action> := (SETR <register> <form>)
            (SENDR <register> <form>)
            (LIFTR <register> <form>)
            (HOLD <constituent name> <form>)
<term act> := (TO <state>)
<form> := (GETR <register>)
          LEX
          *
          (GETWORDF <feature>)
          (BUILDQ <fragment> <register>+)
          (LIST <form>+)
          (APPEND <form> <form>)
          (QUOTE <arbitrary structure>)
```

FIGURE 4

A Language for Representing ATNs

---

*The following discussion is a condensation from <Woods, 1970>[5] to which the reader is referred.

The non-terminal symbols are lowercase English descriptions enclosed in angle brackets. All other symbols except "+" are terminals. Non-terminals not given in Figure 4 have names which should be self-explanatory.

The first element of each arc is a word indicating the type of arc. For CAT, WRD and PUSH arcs, the arc type together with the second element corresponds to the label on a finite state transition graph. The third element is the additional test. A CAT arc can be followed if the current input symbol is a member of the lexical category named on the arc (and if the test is satisfied). A PUSH arc causes a recursive invocation of the network beginning at the state indicated (if the test is satisfied). The WRD arc can be followed if the current input symbol is the word named on the arc (and if the test is satisfied). The TST arc can be followed if the test is satisfied (the label is ignored). The VIR arc can be followed if there is a constituent of the named type on the HOLD list (and it satisfies the test). In all of these arcs, the actions are structure building actions and the terminal action specifies the state to which control is passed as a result of the transition. After CAT, WRD and TST arcs, the input is advanced. After VIR and PUSH arcs it is not. The JUMP arc can be followed whenever the test is satisfied. Control is passed to the state specified in the second element of the arc without advancing the input. The POP arc indicates under what conditions the state is to be considered a final state and the form of the constituent to be returned.

The actions, forms and tests on an arc may be arbitrary functions of the register contents. Figure 4 presents a useful set which illustrates major features of the ATN. The first three actions specified in Figure 4 cause the contents of the indicated register to be set to the value of the indicated form. SETR causes this to be done at the current level of computation, SENDR at the next lower level of embedding (so that information can be sent down during a push) and LIFTR at the next higher level of computation (so that additional information can be returned to higher levels). The HOLD action places a form on the HOLD list to be used at a later place in the computation by a VIR arc.

GETR is a function whose value is the contents of the named register. LEX is a form whose value is the current input symbol. * is a form whose value is (1) in the actions of a CAT arc the root form of the current input symbol; (2) in the actions of a PUSH arc the value of the lower computation; or (3) in the actions following a VIR arc the constituent removed from the HOLD list. GETWORDF is a

function which determines the value of a specified feature of the input word. BUILDQ is a general structure building form which places the values of the given registers into a specified template (fragment). The remaining three forms make a list out of the specified arguments (LIST), append two lists together to make a single list (APPEND) and produce as a value the (unevaluated) argument form. A sample augmented transition network is given in Appendix A.

REASONS FOR A CHANGE

Our own approach to writing grammars has been influenced by the ATN which was seriously considered at the beginning of the SOPHIE project, but rejected as being too slow. Recent development at BBN of an ATN compiler has made the ATN approach more attractive. A complete description of the compiling process is given in Appendix A. Briefly what the compiler does is to input an ATN and output a program which parses sentences in the way dictated by the ATN. Two aspects of the compiler make it especially attractive to our application. One is that the program is optimized to the complexity of the ATN so that SOPHIE's natural language processor incurs no overhead for the features it does not use. The other is that the production of parsing programs in different languages (e.g. CAMIL) can be done by changing the object language output by the compiler without changing the representation of the grammar.

The advantages of using an ATN over the present technique fall into three general areas: (1) conciseness, (2) conceptual effectiveness and (3) available facilities. By conciseness we mean that writing a grammar as an ATN takes less characters than writing one in LISP. It gets this conciseness by not requiring the specification of details of the parsing process at the same level required in LISP. Most of these differences stem from the fact that the ATN assumes it has a machine whose operations are designed for parsing, while LISP assumes it has a lambda calculus machine. The most straightforward example is that LISP assumes a function has one value. A function call to look for a non-terminal while parsing (PUSH), must return at least two values: the structure of the constituent found and the place in the input where the parsing stopped. A good deal of complexity is added to the LISP rules to maintain the variable which has to be introduced to return the structure of the constituent. Other examples include the binding of variables and the specification of control structure as ANDs, ORs and CONDs.

The conciseness of the ATN results in a grammar which is easier to change, easier to write and debug, and easier to understand (and hence to communicate). We realize that conciseness does not necessarily lead to these results (APL being a prime example in computer languages, mathematics in general being another); however, this is not a problem. The correspondence between the grammar rules in LISP and ATN is very close. The concepts which were expressed as LISP code can be expressed in nearly the same way as an ATN but in fewer symbols.

The second area of improvement deals with conceptual effectiveness. One example of conceptual efficiency can be seen by considering the implementation of case structured rules. In a typical case structure, the verb expresses the function or relation name and the subject, object and prepositional phrases express the arguments of the function. Let us assume for the purpose of this discussion that we are looking at three different cases (location, means, and time, e.g. John went to the store by car at 10 o'clock). In the BNF abstraction of the semantic grammar, one would be encouraged to write:

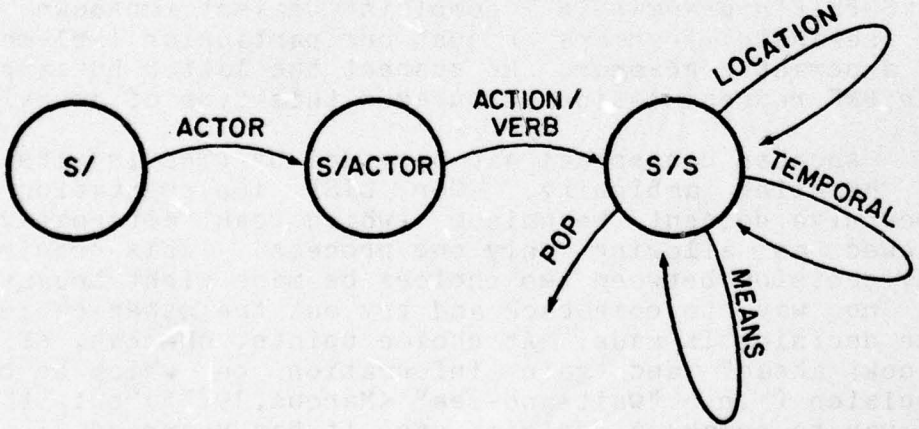
<statement>:= <actor><action/verb><location><means><temporal>

Since the last three cases can appear in any order, one must also write 5 other rules:

<statement>:= <actor><action/verb><location><temporal><means>

.
.
.

In an ATN one is inclined towards:



which expresses more clearly the case rule. There is no

reason why in the LISP version of the grammar one couldn't write loops which are exactly analogous to the ATN (the ATN compiler after all produces such code!) but the BNF does not encourage one to think this way. An alternative BNF method would be:

<action>:= <actor><action/verb><action1>
<action1>:= <action1><temporal>
<action1>:= <action1><location>
<action1>:= <action1><means>

which is easier to write but has the disadvantage of being left-recursive. To implement it, one is forced to write the LISP equivalent of the ATN which creates a difference between the BNF representation and the actual implementation.

The ATN framework also lends itself to postponing decisions about a sentence thereby allowing different paths to stay together. In the present SOPHIE grammar there are top level rules for <set>, a command to change one of the control settings and <modify>, a command to fault the instrument in some way. Sentence (1) is a <set> and (2) is a <modify>.

(1) Suppose the current control is high.
(2) Suppose the current control is shorted.

The two parse paths for these sentences should be the same for the first five words, but they are separated immediately by the rules <set> and <modify>. An ATN encourages structuring the grammar so that the decision between <set> and <modify> is postponed so that the paths remain together. It could be argued that the fact that this example occurred in SOPHIE's grammar is a complaint against top-down parsing or semantic grammars or just our particular implementation of a semantic grammar. We suspect the latter but argue that our BNF representation encourages this type of behavior.

Another conceptual aid provided by ATNs is its method of handling ambiguity. Our LISP implementation uses a recursive descent technique (which can alternatively be viewed as allowing only one process). This requires that any decision between two choices be made right because there is no way to come back and try out the other choice <u>after</u> the decision is made. At choice points, one can, of course, "look ahead" and gain information on which to base the decision (i.e. "wait-and-see" <Marcus, 1975>)[6] but there is no way to remake a decision once it has returned.

The result of this can be most easily seen by considering the lexical aspects of the parsing. A prepass collapses compound words, expands abbreviations, etc. This allows the grammar to be much simpler because it can look for units like "Voltage/Control" instead of having to decode the noun phrase "voltage control." Unfortunately without the ability to handle ambiguity, this rewriting can only be done on words which have no other possible meaning. So for example, when the grammar is extended to handle:

Does the voltage control the current limiting sections?

the compound voltage/control would have to be removed from the prepass rules. This reduces the amount of bottom-up processing which can be done and results in a slower processing time. This also makes compound rules difficult to write because all possible uses of the individual words must be considered to avoid errors. Another example is the use of "C" as an abbreviation. Depending on context, it could possibly mean either current or collector so it could not be conveniently allowed as an abbreviation for either but would have to be recognized in the grammar.

The third general area which provides impetus for the change is the facilities which have been developed around ATNs. These range from mechanisms for handling general linguistic phenomena such as coordination and quantification to particular routines for working with large dictionaries. We will list some of these facilities and briefly discuss each.

Conjunction: Woods <Woods,1973>[7] describes a general machine for dealing with coordination in English which is based on the ATN framework. While it is not implemented yet in the compiler, it should greatly ease the addition of conjunction to the grammar in a systematic way.

Debugging Techniques: The ATN compiler provides a very complete set of debugging tools which have been designed to debug grammars. These have been found to be very helpful in the task of debugging grammars.

Lexical Routines: The ATN has a complete facility for handling lexical problems in parsing. These include morphological routines, compound words, and facilities to handle an off-line dictionary.

FUZZINESS

An exciting use of the non-determinism provided by ATNs involves fuzziness. In a one process (recursive descent) implementation, the rule which checks for a word must decide (with information passed down from higher rules) whether to skip a word or not. The critical information which is not available when this decision has to be made is whether or not there is another parse which would use that word. In the ATN, it is possible to suspend a parse and come back to it after all other paths have been tried. Fuzziness could be implemented so that rather than skip a word and continue, it can skip a word and suspend, waiting for the other parses to fail or suspend. The end effect may well be that sentences are allowed to get fuzzier.

OTHER OBJECT LANGUAGES

The ATN provides a complete description of the grammar which is independent of any particular programming language (unless the ATN formalism is called a programming language). This makes the problem of converting the grammar to other programming languages much easier. To change implementation languages one has only to change the code generation parts of the grammar compiler to output the new object language and implement the run-time environment used by the ATN.* Changing the object language to most algebraic languages which have some string capabilities should be fairly straightforward.

---

*The runtime environment consists of the lexical and dictionary routines, the arc actions and the accessing functions.

CHAPTER 5
DEDUCTIVE-BASED CIRCUIT UNDERSTANDER

In this chapter we describe i) our more general-purpose deductive circuit understander*, ii) recent modifications made to it under this contract, and iii) an example of it running on top of SOPHIE. We are including a prior write-up of the system since it has never been described to AFHRL technical personnel. Understanding its underlying theory is mandatory before one can understand our recent extensions, how it might be employed in a small generative CAI environment, and how it could augment SOPHIE.

AN OVERALL PERSPECTIVE OF TROUBLESHOOTING

Electrical Engineering provides a vast amount of information about mathematical relations between quantities in electronic circuits. In fact for the kind of circuits studied in this paper one can calculate the voltage and current at any point in the circuit using sufficiently complicated mathematics. The use of such complex mathematics is never seen in actual situations! Most often the only mathematics one uses in circuit troubleshooting and understanding is of a very simple type such as in the application of Kirchoff's laws. For more complex situations it becomes more useful to model only those aspects which are interesting, ignoring other aspects. This will of course simplify the problem, but on the other hand we must discover just what these interesting qualities are and be aware of the fact that they ignore certain details (so in certain contexts they can behave incorrectly). This type of analysis is most useful for studying the behavior of collections of (connected) components. We will call such an interesting collection a device. A device is a set of components or other devices interconnected in a particular way to achieve a certain effect. Electronics already has a language for describing the behavior of devices and the handling of exceptions.

There are two approaches to understanding circuits, the quantitative (Kirchoff's laws) and the qualitative (e.g. amplifiers). Each provides different information and is used in different circumstances. As we shall see later in the paper these two approaches require radically different troubleshooting strategies.

---

*The initial design, implementation and write-up of this system was done under a prior contract with ARI. Modifications made to this system under this contract have been fairly minimal (two man-months). We are currently expanding and tailoring it to handle some generative tasks.

TOWARDS A STRUCTURAL THEORY OF TROUBLESHOOTING

The way to obtain new information about a circuit is to make a measurement. In troubleshooting, new information is provided by coincidences. In the most general sense a coincidence occurs when a value at one particular point in the circuit can be deduced in a number of different ways. Such a coincidence provides information about the assumptions made in the deductions. A coincidence can occur in many different ways; it can be the difference between an expected value and a measured value (e.g. expected output voltage of the power supply and the actual measured value); it can be the difference between a value predicted by Ohm's law and a measured value; or it can be the difference between an expected value and the value predicted by the circuit designer. There are numerous other possibilities.

In general, a troubleshooting investigation into a particular circuit proceeds primarily in two phases. The first involves discovering more values such as currents and voltages occurring at various points in the circuit, and the second involves finding coincidences. The usefulness of coincidences is based on the fact that nothing can be discovered about the correctness of the circuit with a measurement unless something is known about the value at that point of the circuit in the first place. If nothing is known about that point, a measurement will say nothing about the correctness of the components. One actual measurement implies many other values in the circuit. The first phase of the investigation involves discovering many such values in the circuit and the second involves making measurements at those points for which we know the implied values so that we can see whether the circuit is acting like it should or something is wrong.

We will call such an implication a propagation and the discovery of a value for a point we already know a propagated value for a coincidence. When these two values are equal we will call such a coincidence a corroboration and when they are different, we will call it a conflict.

Information about the faultiness of components in the circuit can only be gained through coincidences. Propagations involve making certain assumptions about the circuit and then predicting values at other points from these. These assumptions can be of many kinds. Some of them involve just assuming the component itself is working correctly, for example, deriving the current through a resistor from the voltage across it. Others require knowing something about how the circuit should work, thus predicting

what values should be. For example, knowing the transistor is acting as a class A amplifier, we can assume it is always forward-biased. Coincidences between propagated values and new measurements provide information about the assumptions made in the propagation.

Coincidences between propagated values and values derived from knowing how the circuit should work require a teleological description of the circuit. As indicated earlier, this paper does not investigate these latter kinds of assumptions. Instead, this paper investigates propagations employing only assumptions about the components themselves. Although, at first sight, the teleological analysis of troubleshooting is the more interesting, it cannot effectively function without being able to propagate measurements in the circuit! Also, human troubleshooters use less and less teleological information as they narrow down to a particular fault, and even in the narrowing down process there is a constant switching from using teleological values and propagating them in nonteleological ways. So every theory of troubleshooting must include knowledge about local and nonteleological deductions.

It may appear that this local kind of circuit reasoning is essentially trivial and thus should not be investigated. This paper will show that the issues of local nonteleological reasoning are, in fact, very difficult. Some of the problems are specific only to electronics. Others have a very broad range of application to the structure of knowledge. However, if we want to understand troubleshooting all these issues have to be attacked, not just the more interesting teleological ones.

Some of the problems arise partially because the nonteleological knowledge should interact with the teleological knowledge. A particularly difficult problem which will arise again and again is the question of how far to propagate values. Often the propagations will be absurd, and only a small amount of teleological knowledge would have recognized these situations. Part of the effort of this paper is used for determining what other kinds of knowledge and interaction are required aside from the nonteleological in order to troubleshoot circuits effectively.

The sections that follow present an evolution of the knowledge required. The first sections will present a simple theory about local reasoning and troubleshooting. Then the problems of the approach will be investigated, and some of them answered by a more sophisticated theory. Then

the deficiencies of the theory and how it must interact with more teleological knowledge will be discussed.

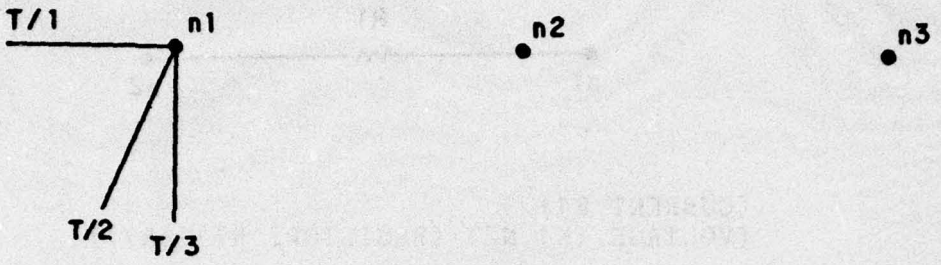
SIMPLE LOCAL ANALYSIS

The domain of electronics under consideration will be restricted to DC circuits. These are circuits consisting of resistors, diodes, zener diodes, capacitors, transistors, switches, potentiometers and DC voltage sources. All AC effects will be ignored although an analogous type of analysis would work for AC circuits. It will be assumed that the topology of the circuit does not change so that faults such as wiring errors or accidental shorts will not be considered as possible faults.

In this section we will present a simple theory of propagation. Initially, only numeric values will be propagated. Interacting local experts produce the local analysis. Each kind of component has a special expert which, from given input conditions on its terminals, computes voltages and currents on other terminals. For example, the expert for a transistor might, when it sees a base emitter voltage of less than .55 volts, infer a zero current through the collector.

In order to give explanations for deductions a record is kept as to which expert made the particular deduction. Most propagations make assumptions about the components involved in making it and these are stored on a list along with the propagated value. Propagations are represented as:
(<type> <location> (<local-expert> <component> <arg>) <assumption-list>)
where:
<type> is VOLTAGE or CURRENT.
<location> is a pair of nodes for a voltage and a terminal for a current.

The simplest kinds of propagations require no assumptions at all, these are the Kirchoff voltage and current laws.



The circuit consists of components such as resistors and capacitors etc., terminals of these components are connected to nodes at which two or more terminals are joined. In the above diagram T/1, T/2 and T/3 are terminals and N1, N2 and N3 are nodes. Currents are normally associated with terminals, and voltages with nodes.

Kirchoff's current law states that if all but one of the terminal currents of a component or node are known, the last terminal current can be deduced.

```
(CURRENT T/1)
(CURRENT T/2)
(CURRENT T/3 (KIRCHOFFI N1) NIL)
```

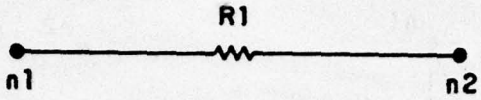
Since faults in circuit topology are not considered, KIRCHOFFI makes no new assumptions about the circuit.

Kirchoff's voltage law states that if two voltages are known relative to a common point, the voltage between the two other nodes can be computed:

```
(VOLTAGE (N1 N2))
(VOLTAGE (N2 N3))
(VOLTAGE (N1 N3) (KIRCHOFFV N1 N2 N3) NIL)
```

As with KIRCHOFFI, KIRCHOFFV makes no new assumptions about the circuit.

One of the most basic types of the circuit elements is the resistor. Assuming the resistance of the resistor to be correct, the voltage and current can be deduced from each other using Ohm's law:



(CURRENT R1)
(VOLTAGE (N1 N2) (RESISTORI R1) (R1))

(VOLTAGE (N1 N2))
(CURRENT R1 (RESISTORV R1) (R1))

(In all the example propagations presented so far it was assumed that the prerequisite values had no assumptions, otherwise they would have been included in the final assumption list.)

These three kinds of propagations suggest a simple propagation theory. First, Kirchoff's voltage law can be applied to every new voltage discovered in the circuit. Then for every node and component in the circuit Kirchoff's current law can be applied. Finally, for every component which has a newly discovered current into it or voltage across it, its VIC (voltage current curve) is studied to determine further propagations. If this produces any new voltages or currents, the procedure is repeated.

This procedure can be easily implemented as a program. Strategies need to be developed to avoid making duplicate propagations, the basic way to do this is to consider only newly discovered values for making new deductions. For each component type an expert can be constructed, we have already seen the resistor and Kirchoff's laws experts. A uniform interaction between the general propagator and the experts can easily be developed.

The current through a capacitor is always zero, so the current contribution of a capacitor terminal to a node can always be determined.

(CURRENT C (CAPACITOR C) (C))

Similarly, the voltage across a closed switch is zero.

(VOLTAGE (N1 N2) (SWITCH VR) (VR))

The remaining kinds of components are semiconductor devices, these devices are very different from the previously discussed kinds of components. Transistors, diodes and zener diodes have discontinuous regions of operation. Semiconductor devices have different regions of operation and each region has a different VIC so that a region of operation must be determined before any VIC can be used. The transistor has the added complication that it is a three terminal device.

The diode is the simplest kind of semiconductor device. Basically, the only thing we can say about it in our simple propagation theory is that if it is back biased, the current through it must be zero.

```
(CURRENT D (DIODEV) (D))
```

For the zener diode we can propagate more values. If the current through a zener diode is greater than some threshold, the voltage across it must be at its breakdown voltage.

```
(VOLTAGE Z (ZENERI) (Z))
```

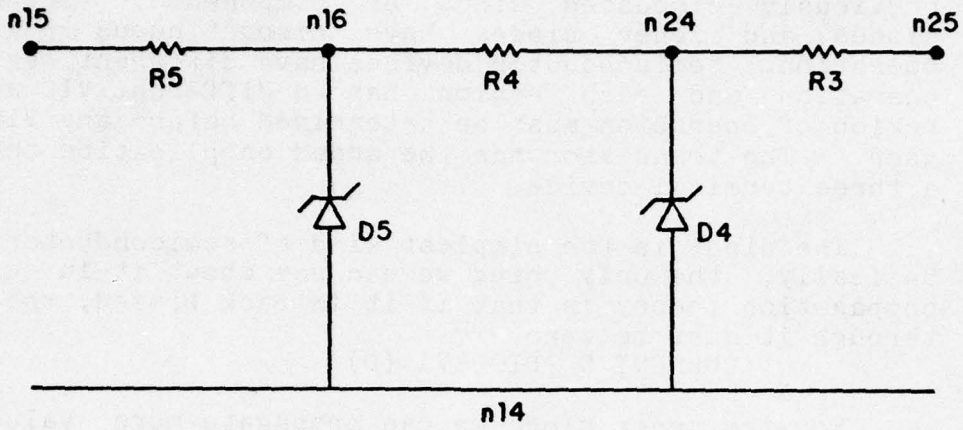
If the voltage across a zener diode is less than its breakdown voltage the current through it must be zero.

```
(CURRENT Z (ZENERV) (Z))
```

The transistor is the most difficult of all devices to deal with. This is both because it has discontinuous characteristics of a semiconductor device and because it is a three terminal device. If the current through any of the transistor's terminals is known, the current through the other terminals can be determined using the beta characteristics of the device. Furthermore, if the voltage across the base-emitter junction is less than some threshold (.55 volts for silicon transistors), the current flowing through any of its terminals should be zero also.

```
(CURRENT C/Q1 (BETA Q1 B/Q1) (Q1))
(CURRENT C/Q1 (TRANOFF Q1) (Q1))
```

Having experts for each component type as has been just described makes it possible to propagate measurements throughout the circuits. As an example, consider the following circuit fragment:



Assume that the fault in this circuit is that D4 has a breakdown voltage too low and the measurements of output voltage and voltage across D5 have just been made. The propagations that can be made are:

```
(VOLTAGE (N15 N14))
(VOLTAGE (N16 N15) (KIRCHOFFV N16 N14 N15) NIL)
(CURRENT R5        (RESISTORV R5) (R5))
(CURRENT D5        (ZENERV D5) (D5))
```

the voltage across the zener D5 is less than its breakdown

```
(CURRENT R4        (KIRCHOFFI N16) (R5 D5))
(VOLTAGE (N24 N16) (RESISTORI R4) (R4 R5 D5))
(VOLTAGE (N24 N14) (KIRCHOFFV N24 N16 N14) (R4 R5 D5))
(VOLTAGE (N24 N15) (KIRCHOFFV N24 N16 N15) (R4 R5 D5))
(CURRENT D4        (ZENERV D4) (D4 R4 R5 D5))
```

the voltage across the zener D9 is less than its breakdown.

```
(CURRENT R3        (KIRCHOFFI N24) (D4 R4 R5 D5))
(VOLTAGE (N24 N25) (RESISTORI R3) (R3 D4 R4 R5 D5))
(VOLTAGE (N25 N14) (KIRCHOFFV N25 N24 N14) (R3 D4 R4 R5 D5))
(VOLTAGE (N25 N16) (KIRCHOFFV N25 N24 N16) (R3 D4 R4 R5 D5))
(VOLTAGE (N25 N15) (KIRCHOFFV N25 N24 N15) (R3 D4 R4 R5 D5))
```

The propagation proceeds one deduction at a time; never is it necessary to make two simultaneous assumptions at the same time in order to get to the next step in the propagation chain. The propagation can always go through some intermediate step.

A SIMPLE THEORY OF TROUBLESHOOTING

This section examines how the propagation strategy of the previous section can be used to troubleshoot the circuit. The ideas of conflicts and corroborations between propagation will be used to show how the propagator can be used to help in troubleshooting the circuit. In this simple theory we will assume that coincidences only occur between propagated values and actual measurements.

The meaning of the coincidences depends critically on the kinds of assumptions that the propagator makes. For the coincidences to be of interest every assumption made in the derivation must be mentioned, and a violation of any assumption about a component must mean that component is faulted. Then, when a conflict occurs, one of the components of the derivation must be faulted. Furthermore, if the coincidence was a corroboration, all the components about which assumptions were made are probably unfaulted.

The usefulness of the coincidence depends critically on how many faults the circuit contains. The usual case is that there is only one fault in the circuit. Even the case where there is more than one fault in the circuit, the approach of initially assuming only a single fault in the circuit is probably a good one.

If there is only one fault in the circuit, all the components not mentioned in the derivation of the conflict, must be unfaulted. If a coincidence occurs, all the components used in the derivation can be assumed to be unfaulted. In a multiple fault situation these would be invalid deductions: in a conflict only one of the faulted components need be involved and in a corroboration two faults could cancel out each other to produce a correct final value.

If, in the propagation example of the previous section, the voltage between N25 and N14 was discovered to conflict with the propagated value, one of R3, D4, R4, R5 and D5 must be faulted. But, if the values were in corroboration, all the components would have been determined to be unfaulted.

Now that the fault has been reduced to one of R3, D4, R4, R5 and D5, the propagations can be used to determine what measurement should be taken next. The best sequence of measurements to undertake is, of course, the one which will find the faulted component in the fewest number of new measurements. Assuming that the relative probability of which component is faulted is not known, the best strategy

is a binary search. This is done by examining all propagations in the circuit, eliminating from their assumption lists components already determined to be correct. and picking a measurement to coincide with that propagation whose number of assumptions is nearest to half the number of possibly faulted components.

In the example there are five possibly faulted components; hence the best propagations to choose, are those with two or three assumptions. This means either measuring the current through R4, the voltage across D4, the voltage across R4 or the voltage between N24 and N15. All the other measurements, in the worst case, can eliminate only one of the possibly faulted components from consideration.

The current through R4 is measured. This coincidence is a corroboration; so R5 and D5 are verified to be correct. Therefore one of R3, D4 and R4 must be faulted. At this point there are too few possible faults to make a binary search necessary. Any measurement which would coincide with any propagation having R3, D4 or R4 as assumptions, but not all three at once, is a good one. One such measurement is the current through D4. This conflict would indicate that D4 is faulted.

This kind of circuit analysis can be used for simple kinds of troubleshooting. Of course, the troubleshooting as indicated cannot really begin effectively until the first conflict has been found. However, in a more teleological framework, teleological assumptions can also be used in the propagations. (This transistor is a class A amplifier so its base-emitter voltage must be about .6 volts.) When teleological assumptions have to be made, the derivations will of course no longer be complete. That is, a conflict or corroboration will not necessarily say anything about the components if some teleological assumption was made in the propagation. But, as with assumptions about components, conflicts and corroborations will still comment on the validity of the teleological assumptions in an analogous way. The information provided by a conflict or corroboration with a teleological assumption needs a special kind of knowledge to make use of it.

UNEXPECTED COMPLEXITIES OF THE SIMPLE THEORY

The discussion of the previous section presents an interesting and, on the surface, a very simple scheme for troubleshooting. Unfortunately, the entire approach is fraught with difficult problems! This section deals with some of these problems and attempts to provide a solution to

them within the original framework. Such an investigation will clarify the deficiencies of using only local circuit knowledge for troubleshooting.

Basically, three kinds of problems arise. Firstly, the handling of corroborations and conflicts leads to faulty assertions in certain situations and thus they should be examined much more closely. Secondly, it will be shown that the propagation scheme, the knowledge contained in the experts and the troubleshooting strategy are all incomplete. All of them cannot make certain kinds of deductions which one might expect them to in the framework that has been outlined. Finally, accuracy is a problem; all components and measurements have an error associated with them (if only a truncation or roundoff error), and errors cause many kinds of difficulties in the entire strategy. (In the remainder of this paper it will be assumed that the circuit under consideration contains only a single fault.)

The nature of corroborations requires closer scrutiny. It has already been shown that every component which a derivation depends on is in the assumption list of that derivation, and so a conflict localizes the faulted component to one of those in the mentioned assumption list. For corroborations the simple troubleshooting scheme used the principle that a coincidence indicated that all of the components in the assumption list were cleared from suspicion. This principle must be studied with much greater scrutiny as there are a number of cases for which this principle doesn't hold.

In order to do this we must examine the precise nature of the propagations, and, more importantly, examine the relation between a single value used in a propagation with the final propagated value. Consider a propagated value derived from studying the component D; let the resulting current or voltage value be f(D). The propagator is entirely linear; so the propagated value at any point can be written as a linear expression of sums of products involving measured and propagated values. For every component, current and voltage vary directly with each other and not inversely. Hence, in the expression for the final propagated value, f(D) can never appear in the denominator. So the final value can be written as:

$$\text{value} = (f(D) \cdot b) + c$$

where b and c are arbitrary expressions not involving D. The relation between f(D) and the final propagated value is characterized by b. By studying the nature of component experts, the structure of b can be determined. Every expert either multiplies the incoming value (we will denote this

value which is used by the component expert to derive f(D), v(D)) by a parameter, or applies a simple less than or greater than test to the incoming value v(D) to obtain a propagated value. As many components of this type can be involved in a single propagation, each propagation of this kind has a predicate associated with it indicating what conditions must be true for the propagation to hold. With both kinds of propagations there is a problem if b is zero. In that case, f(D) has no influence on the final value and so a coincidence indicates nothing about the validity of f(D).

In the case where a predicate must hold for the propagation to be made, a corroboration only indicates that the incoming value v(D) is within a certain range, thus saying little about the assumptions which were used to derive v(D). Note that, however, in a conflict situation the predicate is violated, and thus v(D) being incorrect is a possibility that cannot be ignored. Any single propagation makes many assumptions, some of them may involve predicates, others may not. In a corroboratory coincidence the only assumptions which cannot be substantiated are those which were made to determine the v(D) which the component expert for D used only in a test, all the remaining assumptions can be handled with the usual corroboration scheme. We shall call such assumptions, which corroborations do not remove from suspicion, the <u>secondary</u> assumptions of the propagation, and the remaining, the <u>primary</u> assumptions.

The situation for which b is zero can be partially characterized. Using the same assumption more than once in a propagation is relatively rare. In such a single assumption propagation b must be a single term, consisting of a product of parameters (resistances, betas, etc.) or their inverses, and since no circuit parameter is zero, b cannot be zero.

Every occurrence of an assumption about D in a propagation introduces another term to b. Each of these terms must still be a product of parameters. Unfortunately, at this point in our research we cannot give a proof why b=0 is impossible, but only an appeal to a somewhat heuristic argument. Consider the case where b is zero. It has already been shown that b is a product of circuit parameters, and so is independent of any measurements. That means whatever value f(D) has that value, no matter how extreme, has absolutely no influence on the final propagated value. That seems absurd, so b must never be zero.

What makes this discussion only an argument and not a proof is that:
(1) Any manipulation on the circuit to alter the actual value f(D) must also shift c and value. (just changing the specifications of D results in nothing - one interpretation of the argument is: no matter what specification D has, in this particular propagated value it has no influence).
(2) The idea of b=0 being absurd is extremely difficult to formalize and it is intimately dependent on the exact nature of the component experts.
In conclusion, it should be noted that we have not been able to discover any propagation (in a coherent circuit) for which b was zero, and so it seems a workable hypothesis that b cannot be zero. Of course, if b is very small, accuracy issues become critical, but this will be discussed later.

The propagation scheme cannot make all the propagations that one might reasonably expect. Incompleteness of this type manifests itself in two ways, yet in both certain obvious propagations are not made. One is just a problem of circuit representation, and the other is an inherent problem of the propagator.

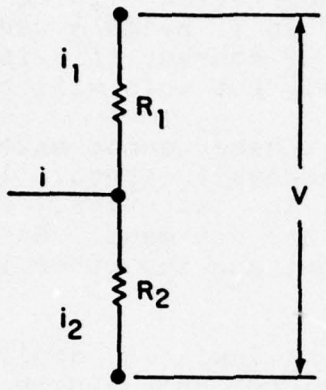
Kirchoff's current law can apply to collections of components and nodes, not just single components and nodes. Recognizing relevant functional blocks in the topology of the circuit is a tedious (yet performable) task. Circuit diagrams usually present a visual organization so that such functional blocks (and teleological organization) become clear.

The process of propagation as outlined consists of using a newly discovered value to call an expert which can use that value to make new discoveries. The called expert then looks at the environment and from this deduces new values for the component it is an expert about. The communication with the environment always involves numeric values. Experts cannot communicate with each other, neither can they handle abstract quantities. Furthermore, propagation stops when a coincidence occurs and iteration toward an accurate solution is never attempted. This can become a severe limitation in certain feedback situations.

This entire scheme is motivated by what we see in human troubleshooters. The strategy has some very surprising limitations. The fact that only one expert is invoked at any one time means that only one assumption can be made at any step in the propagation process. This means that propagations which require two simultaneous assumptions cannot be made. Most propagations which require more than

one assumption do not require simultaneous assumptions as they can be derived using some intermediate propagation (e.g. all the previously discussed examples).

One such case requiring simultaneous assumptions is the voltage divider.



Suppose V and i are known, the current through R1 (and hence through R2) can be propagated by simultaneously assuming the correctness of both R1 and R2.

$$V = i1\ R1 + i2\ R2$$
$$i = i1 - i2$$
$$i1 = (V - i\ R2)/(R1+R2)$$

Admittedly, the voltage divider is an important enough entity that it should be handled as a special case pattern, however, problems of this kind of incompleteness will arise in other situations, and it will not be possible to design a special case pattern for each of them.

If multiple faults are allowed, simultaneous assumptions must be handled with even greater caution. For example, a propagation involving a simultaneous assumption can propagate a correct value even though both components which the assumptions were about were faulted. In the case of a voltage divider, the resistance of both R1 and R2 could shift without affecting the voltage at the tap, yet the voltage divider would present an erroneous load to the

voltage source it was connected to.

In order to illustrate some other difficulties, propagations requiring simultaneous assumptions can be characterized differently. If a measurement is made for which a propagation can be made to coincide with the original measurement, a previously incomplete propagation has completed. The coincidence indicates that if the propagator could have made an abstract hypothetical measurement and used a relaxation or algebraic method, the actual value for that point could then have been determined without making the measurement in the first place. However, since the current propagation scheme cannot make such hypothetical measurements, a later measurement might play the role of generating the hypothetical measurement. Unfortunately, the coincidence rarely occurs at the exact point of the measurement; all propagations proceed in a breadth first direction from the original measurement point, and even if this was modified, it would not alleviate the difficulty because the new measurement might only cause a later propagation (some distance away from the original measurement point) which plays the role of a hypothetical measurement. The problem is then, that coincidences need not be between propagated values and measured values, but can also be between two propagated values.

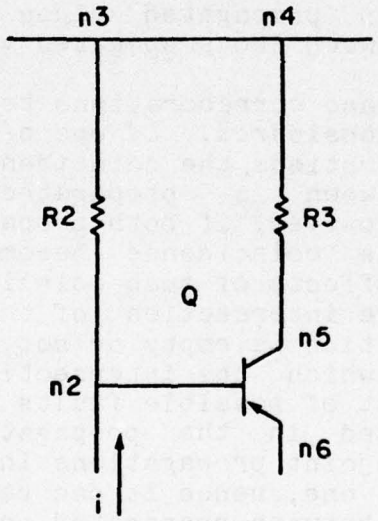
Conflicts and corroborations between propagated values must then be considered. If one of the propagations has no unverified assumptions, the coincidence can be handled as if it were between a propagated value and an actual measurement. However, if both propagations have unverified assumptions, the coincidence becomes far more difficult to analyze. The effects of such coincidences depend critically on whether the intersection of the unverified assumptions in each propagation is empty or not. First we will examine the case in which the intersection is empty. A conflict reduces the list of possible faults to the union of the assumptions used in the propagations. The corroboration between two disjoint propagations indicates that this value is the correct one, hence it can be treated as two separate corroborations between propagated and measured values.

The case of a nonempty intersection is the most difficult. If the coincidence was a corroboration, a fault in the intersection could have caused both propagations to be incorrect yet corroborating. Yet, what can be said about the nonintersecting assumptions in the propagations? If there was a fault in one of the nonintersecting primary assumptions, it must have caused a conflict, so all the nonintersecting primary assumptions can be verified to be

correct. If the coincidence was a conflict, the list of possibly faulty components can be reduced to the union of the assumptions. In this case it is very tempting to remove from suspicion all those components mentioned in the intersection, this would capture the notion that correct propagations from a single (albeit incorrect) value must always corroborate each other or, equivalently, that each point in the circuit has only two values associated with it: a correct value and a faulted value.

In the counterexample, an emitter current is propagated through a transistor to obtain propagated values for the base and collector currents. The base emitter junction of this transistor has shorted and consequently both these propagated values will be in conflict with the actual values in the circuit. These two values will also conflict with each other.

Consider the circuit fragment:



The circuit is faulted with the base emitter junction of Q shorted, with the collector terminal open. Thus far the voltage at N2 and N5, and a current into N2 have been measured. Next the emitter current of Q is measured from which the base and collector currents are propagated. The

initial measurement did not coincide with any propagated value, yet a conflict will occur within the propagations caused by this measurement. Furthermore, the values of the conflicting propagations conflict with the actually measured value. The exact point at which this conflict will occur depends on internal details of the propagator. Two obvious points at which the conflict can occur is the voltage at N3-N4 and the current through the base of the transistor.

```
(VOLTAGE (N2 N0))
(VOLTAGE (N5 N0))
(CURRENT E/Q)


(CURRENT B/Q (BETA Q E/Q) (Q))
(CURRENT C/Q (BETA Q E/Q) (Q))
(CURRENT R2 (KIRCHOFF N2) (Q))
(CURRENT R3 (KIRCHOFF N5) (Q))
(VOLTAGE (N4 N5) (RESISTORI R3) (R3 Q))
(VOLTAGE (N3 N2) (RESISTORI R2) (R2 Q))
(VOLTAGE (N3 N0) (LOOP N3 N2 N0) (R2 Q))
(VOLTAGE (N4 N0) (LOOP N4 N5 N0) (R3 Q))
```

This results in two conflicting voltages at N3-N4, one is higher than the actual value in the circuit, and the other is lower.

All measurements in the circuit and all circuit parameters have errors associated with them. Even if we assumed perfect measurements, truncation and roundoff errors would cause problems. One way to view the problem is to study the size of b relative to the error in c. If b is smaller than the error in b, a large error in some f(D) could be undetected. Again we see the greatest problem lies with corroborations. In a corroborating coincidence we must make absolutely sure that an error in any of the verified assumptions could have been detected in the <u>value</u> (i.e., b is not too small).

The solution is quite simple; instead of propagating numeric values through the circuit, we propagate values and their tolerances, or just ranges of values. Each measurement and circuit parameter could have a tolerance associated with it, and the arithmetic operations could be modified to handle ranges instead of numeric values. Instead of computing b and its tolerance, the propagator could note whenever an error in some incoming value could be obscured in larger errors in other values (remember, errors in parameters and measurements are usually percentages, and thus adding a large value and a small value will often

obscure an error in the small value). Since such problems occur only with addition and subtraction of ranges, KIRCHOFFV and KIRCHOFFI are the only experts which need to be directly concerned with the accuracy issue.

Assuming that errors are roughly proportional to the magnitude of the propagated values, those propagations involved in a sum whose magnitude is less than the error in the final result should not be verified in a corroboration of the final value. (As this assumption is not always true, some assumptions may not be verified in a corroboration when they should be.) KIRCHOFFV and KIRCHOFFI can easily check for such propagations. Fortunately, a category for assumptions which should not be verified in a corroboration has already been defined; these are the secondary assumptions. So, primary assumptions of the incoming values into a KIRCHOFF may become secondary assumptions of the final result.

As usual, this theory of handling accuracy has subtle problems. If the only possible effect of a particular f(D) was described in a propagation, then no matter how insignificant its contribution was to the final value, a coincidence should verify D since it wouldn't matter in such a case if D were faulted or not. Furthermore, the propagation through certain components is so discontinuous that no matter how insignificant its propagatory contribution is, a fault in the final value would so greatly affect the propagation that the assumption in question should really be treated as a major assumption. An example of the former is a switch in series with a resistor, and an example of the latter is a zener diode contributing zero current to a node.

Consider the case of a resistor in series with a switch. The only contribution of that switch to the circuit is in the voltage across the switch and the resistor. A voltage across a closed switch is zero, so unless the resistance of the resistor is zero the switch becomes a secondary assumption of the final voltage. Unfortunately, a corroboration with that voltage should indicate the switch was acting correctly.

Similarly, a zener diode contributing zero current to a node will always become a secondary assumption of the KIRCHOFFI propagation. But, a corroboration should indicate that the zener was functioning correctly. This is because this propagation would not even have been possible if the voltage across the zener were near its breakdown. A heuristic solution to this problem is not to secondarize

propagations with zero value which were just propagated from discontinuous devices. This, of course, makes the teleological assumption that the discontinuous component makes a significant contribution whenever it is contributing a non-zero value, as is almost always the case with the switch, diode, zener diode and transistor.

Accuracy brings along other problems, testing for equality between ranges becomes a rather useless concept. A simple workable strategy is to use a rough approximation measure such as accepting two ranges as equal if the corresponding endpoints of the two ranges are within a certain percentage of each other. More satisfactorily, the actual width of the range should also enter into consideration so that if one end of the range is extremely small relative to the other, a much more liberal percentage is used to compare the smaller endpoints. One certainly would want the range [0 , 1] to be roughly equal to 10E-6 , 1. Using the percentage of the endpoint largest in magnitude as a fixed range to compare the smaller endpoints appears to be the best strategy. A coincidence can be of three kinds: the ranges can be approximately equal (or just significantly overlapping) which is a corroboration, the ranges can be disjoint which is a conflict and the ranges can overlap but not significantly which provides no information at all.

Having the the propagator propagating ranges brings up the idea of allowing components to individually propagate higher and lower limits in the circuit. Every diode could propagate a non-negative current through itself. A voltage could be propagated at every part of the circuit whose upper limit was the magnitude of the sum of all the voltage sources in the circuit. More interestingly, it could handle the problem of having a range propagated over a discontinuous device: a [-1 , +1] current range into a diode should have its lower limit modified to 0 (i.e. [0 , +1]). Interesting as such new kinds of propagations may be, they require separate derivations for the upper and lower limits for each range, and thus introduce incredible difficulties for handling coincidences.

When a significant propagation occurs which overlaps a test point of a discontinuous component, the best strategy is to interpret that measurement to have too wide an error associated with it and stop the propagation there. In general, when error tolerances in propagated values become absurd (a significant fraction or multiple of the central value) the propagation should be artificially stopped.

There remain certain characteristics of the devices that are not captured in the propagation scheme. These are usually the maximum ratings of the components. The base emitter voltage of a transistor cannot exceed a certain value, the voltage across a capacitor cannot exceed its breakdown voltage, the voltage across a zener diode cannot exceed its breakdown voltage, the power dissipation in a resistor cannot exceed its wattage rating, etc. These, in fact, can be quite easily captured by simple modifications of the component experts. Each expert could check whenever it was invoked whether any ratings about the component were exceeded. Such situations of excess can be of two kinds; the final value calculated to compare to a rating may or may not involve the component itself as an assumption. If the component itself is used as an assumption, the situation can be treated as a conflict with the calculated rating. Otherwise, if the component itself is not mentioned in the assumptions, the situation must again be handled as a conflict, except that the component in question must not be removed from suspicion.

Often a component which is considered possibly faulted because of a conflict can really be eliminated from suspicion by examining exactly what kind of fault the conflict implies the component might have. (For example, all the currents in the transistor must shift so that Kirchoff's law is violated, or, a more trivial case a capacitor for which the fault of too low a current is entertained.

In order to determine the kind of faults a particular conflict implies it must be known whether the value is high or low. This can only be determined for conflicts with measured values. For conflicts between propagated values, there is no convenient way of determining the possible faults except by hypothesizing all the possible high/low combinations and using the intersection of all the results.

We must tackle the problem about how to scan back through the propagation to determine what faults in the components could have caused the final conflict. Of course, a straightforward way to do this would be to compute b for every component f(D) involved in the propagation. For every two terminal component the possible fault can be immediately determined from b (unless of course we have the inaccurate case where the range for a spans zero). The only three terminal device, the transistor, requires a more careful examination as it has many possible fault modes, and a single consideration of a propagation from it may not uniquely determine the fault mode.

Continuing in the spirit of the original propagation scheme, a different method than computing b should be used. A simple scheme can be derived which only has difficulties in certain kinds of multiple assumption propagations. The conflict indicated that the propagation was in error by a certain shift in value in a certain direction. This shift can be propagated backwards through all the experts except KIRCHOFFI and KIRCHOFFV. The Kirchoff's laws experts involve addition, so each of the original contributors to the sum must be examined. For those contributors whose (unverified) assumption list does not intersect with any of the other assumption lists, the shift can be propagated back, after adding the appropriate shift caused by the remaining contributors. For those contributors with intersecting contributions it must be determined for each of the intersecting components whether all contributions of all the possible faults do not act against each other (e.g. will a shift in the resistance of the component both increase a current contribution to a node and decrease it through another path?). For such canceling intersections, nothing can be said about the intersecting component. In actuality, all this does is capture qualitatively whether the signs of the terms of b are different and thus canceling. It should be noted, that if it really turns out to be the case that a b can be zero, such a scheme could be used to at least eliminate faulty verifications from taking place, again at the cost of sometimes not verifying probably unfaulted components.

Incompleteness in the propagation scheme introduces incompleteness in the troubleshooting scheme. Even if the propagation scheme were complete the troubleshooting scheme would be incomplete. The earlier answer to what is the next best measurement is inaccurate. The measurement which reduces the list of possible faults by the greatest number is not necessarily the best measurement. Future measurements must also be taken into consideration, a poor first measurement may set the stage for an exceptionally good second measurement.

The choice of best measurement depends of course on what is currently known about the circuit. The most general approach would be to try every possible sequence of hypothetical measurements and choose the first measurement of the best sequence as the next measurement. Again, that would be an incredible, and unnatural computation task. The current troubleshooting scheme does not try to generate all possible sequences and only considers making those measurements about which it already knows something (so to produce a coincidence).

Since only measurements at points about which something is known are considered the information provided by coincidences between propagated values (the result of incompleteness in the propagator) cannot enter into consideration. The basic simple paradigm of the troubleshooter was to make no hypothetical measurements and look only at those propagations with unverified assumptions as those to try to coincide with. Unexpected information, such as that provided by coincidences between propagated values, cannot be considered in that paradigm (making hypothetical measurements of course would handle this problem).

Issues of accuracy are sufficiently captured by primary and secondary assumptions. The binary search for the best measurement must of course be reorganized. Since a corroboration may eliminate less components from suspicion than a conflict could, the search is not purely binary. A workable solution is to just take the average of the number of components which would be verified in each case as the measurements rating. Then that measurement whose rating was nearest to half the number of faulted components could be chosen as the next measurement.

There remains the issue of generating an explanation for this choice. Although the above argument for deriving a future choice of measurement could be made understandable to humans it does not always generate a very good explanation. A large part of the explanation for a future choice of measurement involves indicating why a certain component cannot be faulted (incomplete understanding by the student). Once a component is eliminated from suspicion for any reason it is never considered again. However, a later measurement might give a considerably better explanation for its unfaultedness. The problem of generating good explanations, of course, also must take into account a model of the student and what he knows about the electronics and the particular circuit in question. This is a topic of current investigation.

On the topic of selecting the most comprehensible choice from a number of otherwise equally good measurements something can be said. The above scheme for selecting measurements does not take into account how "close" the measurement is to the actual components in question. For example a voltage measurement across two unverified resistors is just as good as a measurement many nodes away which also has only those two resistors as unverified assumptions. Fortunately these can be easily detected: just remove from the list of possible measurements all those

which are propagated from other elements on the list.

THE NECESSITY AND UTILITY OF OTHER KNOWLEDGE

In this section we will attempt to characterize where and why local and nonteleological reasoning fails. Indeed many of these failures have been already demonstrated. Our method of attack will be from two directions. First, inherent problems in the earlier propagation scheme can be alleviated with other knowledge about the circuit. Second, many of the kinds of troubleshooting strategies we see in humans cannot be captured by even a generalization of the proposed scheme. One of the basic issues is that of teleology. The more teleological information one has about the circuit, the more different the troubleshooting process becomes. Currently, most of the ideas presented so far in this paper so far have been implemented in a program so that many of the discussions derive their observations from actual interactions with the program.

The most arresting observation is that the propagator cannot propagate values very far, and at other times it propagates values beyond the point of absurdity. Examining those propagations which go too far, the most dominant characteristic is that either the value itself has too high an error associated with it, or that the propagation itself is not relevant to the issues in question. The former problem can be more easily answered by more stringent controls on the errors in propagations. The latter requires an idea of localization of interaction. This idea of a theater of interactions would limit senseless propagation; however, it requires a more hierarchical description of the circuit (will be discussed later).

The idea that every measurement must have a purpose points out the basic problem: our troubleshooter cannot make intelligent measurements until it has, by accident, limited the number of possible faults to a small subset of all the components in the circuit. After this discovery has been made, it can make fairly intelligent suggestions. However, as such a discovery is usually made when the set of possible faults is reduced to about five components, it can only intelligently troubleshoot in the last few (two or three) measurements that are made in the circuit.

Clearly, many more measurements are made before this discovery and the troubleshooter cannot do anything intelligent during this period. It will be shown, however, that the propagation scheme and the ideas of corroborations and conflicts can be effectively used even

during this period.

The only way intelligent measurements can be made during this period is by knowing something about how the circuit should be behaving, or just how it behaves. This requires teleological information about the circuit. For example, just to know that the circuit is faulted and requires troubleshooting requires teleology. In the situations where the propagator did not propagate very far, the problem usually was that some simple teleological assumption could have been made. The voltages and currents at many points in the circuit remain relatively constant for all instantiations of the circuit, and furthermore many of them can be easily deduced (e.g. knowing certain voltage and current sources such as the power supply, knowing contributions by certain components to be small, etc.). Propagation can then proceed much further. Of course, the handling of coincidences requires modifications, and a new kind of strategy to deal with teleological propagations needs to be developed.

If sufficient teleology about the circuit is known so that the transfer functions of certain groups of components are known, assumptions of the form "assuming x is in the correct state" or "assuming x is working correctly" can be made. Issues of structuring such a hierarchical and teleological description are being investigated <Brown & Sussman, 1974>[8].

The propagation scheme of the previous sections can be used to understand the implications of these assumptions by propagating them in the circuit, and to determine all the isomorphisms of a particular set of measurements so that the appropriate values for the teleological description mechanisms can be discovered no matter what measurements are made.

However, as indicated earlier, a new procedure has to be made to handle coincidences. At a low level coincidences can be used quite simply. When it is discovered that a certain voltage is lower than it should be, a search can be made in the topology of the circuit as to faults in which components might have caused such a shift. This would work most of the time, except in cases where complex feedback paths were present. Coincidences and corroborations involving assumptions concerning collections of components need to be handled differently. If an entire collection of components is working correctly, all the components inside of it can be assumed to be working correctly. But, if a collection of components is possibly working incorrectly a

measurement must be chosen within the actual collection or module which can best determine what could be wrong. While the previous deduction required extrinsic knowledge about the module, the search for such a fault within a module requires an intrinsic description of it.

In searching for reasons why a certain value is not teleologically what it should be, it is important to note that in examining the behavior of a particular component or module that the reason for its apparent faulty behavior can lie either with itself, or what it is delivering values to or what is supplying values to it.

EXTENSIONS

Two basic modifications have been made to this original system in addition to exploring how these ideas may be exploited in the kind of generative system being designed for AFHRL. The first modification has been the addition of ranges to the propagations specialists. This addition has lead to several unexpected problems, some of which have now been resolved. The second addition has been to add some teleological knowledge to each component specialist so that the system can now push a propagation further than before. The simplest example (and one that does not illustrate the real power of teleological reasoning) is that when the propagator discovers that Q5 is conducting, then by assuming Q5 is OK, it can postulate the approximate value of the Vbe and then propagate that value. Then, by knowing what the output voltage is, it can push the Vbe voltage back so as to determine the output voltage component of the voltage reference source. Higher level specialists for groups of components, as opposed to single components, are also being written. With the addition of these specialists, such as a constant current source specialist, we can easily add higher level teleological knowledge to handle such situations as predicting when Q5 is off then Q6 is on, assuming the other "devices" are working.

TOP LEVEL USER FLAGS AND FUNCTIONS

The following list contains the special flags, options, and top level functions of the inference package:

INTERFLAG

Set to T will print out two lines of information every time something new is discovered about the circuit.

INTERRESET[]

This will cause INTER to forget everything it knows about this particular circuit. You have to do this every time you change the circuit (new fault, settings).

FAULTS[]

If INTER has seen a conflict, this prints out a complete list of the possibly faulted components and (for a few) how they could be faulted.

MEASVALUE[TYPE POINT]

(Again after a conflict). This will make a comment about the usefulness of the next measurement before!! you make it. TYPE is either VOLTAGE or CURRENT. POINT must be a SEMNET terminal. The voltage across R4 MEASVALUE[VOLTAGE (N16 N24)], the current through R22: MEASVALUE[CURRENT L/R22].

SUGGEST[]

(Again after a conflict). This will print out a list of the best measurements open to you.

WHYMEASURE[TYPE POINT]

(Again after a conflict). This will give an explanation as to why that hypothetical measurement is any good. (Usual use is to understand the suggestions SUGGEST returns. It will, however, work on less optimal measurements.]

EXPLAIN[TYPE POINT] - being debugged.

Gives one explanation about why a value was propagated to that point.

INFORM[TYPE VALUE POINT]

This function acts as an input point enabling the inference module to be interfaced to SOPHIE, a simulation package or student's assertions, etc. If

TYPE=VOLTAGE, positive means from (CAR POINT) to (CADDR POINT)>. If TYPE=CURRENT, positive means out of some node. In this way time can be saved by completely ignoring the circuit simulator and the natural language processor.

This inference system can currently make some comments on its own about the students' measurements. Before a conflict has occurred, it can distinguish whether the measurement you made was useless - i.e., it was logically entailed by the previous measurements. A measurement for which the first conflict occurs is given a special comment so that the user can tell that MEASVALUE, SUGGEST and WHYMEASURE can now give interesting responses. After that it can make a much more critical comment about useful and useless measurements. These comments will all be after the measurement has taken place, and currently the code has not been included to make a comment about how good the measurement was (using MEASVALUE). Later it might make comments like "Although you had little reason to make the measurements, you were lucky and it reduced the possible fault space by a large amount." That requires more thought.

It is also possible to trace interesting functions and get comprehensible results. The two interesting functions are CONFLICT which handles the conflicts, and SETFAULTSTATUS which is called every time the fault status of some component changes. With INTERFLAG set to T it is possible to determine where and why the conflicts occurred.

PROTOCOL WITH TRACE OF WHAT "NEW" INFORMATION IS BEING DEDUCED

```
SOPHIE2 is active

>>INSERT A FAULT

THE INSTRUMENT HAS A FAULT AND ITS PRESENT CONTROL SETTINGS ARE:
CC          1.0
CR          HIGH
LOAD        20
VC          1.0
VR          HIGH

>>WHAT  IS THE OUTPUT VOLTAGE?
THE OUTPUT VOLTAGE IS 11.7 VOLTS.
```

```
(NEWINFO (VOLTAGE (12.30294 . 11.13124) (N11 GROUND) (FLAT ((MEAS
M0001)))))
(NEWINFO (VOLTAGE (-11.13124 . -12.30294) (GROUND N15) (GLAT ((LOOP
GROUND N11 N15)) (VR))))
(NEWINFO (CURRENT (.6151472 . .556518) N/OP (FLAT (VCVT LOAD))
 (LOAD))))
(NEWINFO (CURRENT (-.5565618 . -.6151472) P/OP (FLAT ((VCVT LOAD))
(LOAD))))
"SYMPTOM FOUND"

>>WHAT IS THE VBE OF Q5
THE BASE EMITTER VOLTAGE OF Q5 IS -.037 MILLIVOLTS.

(NEWINFO (VOLTAGE (-3.547876E-5 . -3.921335E-5) (N10 GROUND) (FLAT
((MEAS M0002)))))
(NEWINFO (VOLTAGE (-11.13127 . -12.30298) (N10 N15) (FLAT ((LOOP N10
GROUND N15)) (VR))))
(NEWINFO (VOLTAGE (-11.13127 . -12.30298) (N10 N11) (FLAT ((LOOP N10
GROUND N11)) NIL)))
(NEWINFO (CURRENT 0.0 E/Q5 (FLAT ((TRANOFF Q5)) (Q5))))

>>WHAT IS THE IC OF Q5
THE COLLECTOR CURRENT OF Q5 IS 0.0 AMPS.

(NEWINFO (CURRENT (0.0 . 0.0) C/Q5 (FLAT ((MEAS M0003)))))
(NEWINFO (CURRENT (0.0 . 0.0) B/Q5 (FLAT ((BETA Q5 C/Q5)) (Q5))))

COINCIDENCE:
OLDREASON = (FLAT ((TRANOFF Q5)) (Q5))
NEWREASON = (FLAT ((BETA Q5 C/Q5)) (Q5))
OLDVALUE = 0.0
NEWVALUE = (0.0 . 0.0)
POINT = E/Q5

   CORROBORATE:
   REASON1 = (FLAT ((TRANOFF Q5)) (Q5))
   REASON2 = (FLAT ((BETA Q5 C/Q5)) (Q5))
   POINT = E/Q5
(NEWINFO (CURRENT (0.0 . 0.0) A/D6 (FLAT ((KIRCHOFF N10)) (Q5))))

>>WHAT IS THE VBE OF Q6
THE BASE EMITTER VOLTAGE OF Q6 IS .349 VOLTS.

(NEWINFO (VOLTAGE (.36644 . .3315409) (N5 N9) (FLAT ((MEAS M0004)))))
(NEWINFO (VOLTAGE (.36644 . .3315409) (N5 N8) (FLAT ((LOOP N5 N9 N8))
(CR))))

>>WHAT IS THE OUTPUT CURRENT
THE OUTPUT CURRENT IS .586 AMPS.
```

```
COINCIDENCE:
OLDREASON = (FLAT ((VCVT LOAD)) (LOAD))
NEWREASON = (FLAT ((MEAS M0005)))
OLDVALUE = (.6151472 . .5565618)
NEWVALUE = (.6148398 . .5562836)
POINT = N/OP

   CORROBORATE:
   REASON1 = (FLAT ((VCVT LOAD)) (LOAD))
   REASON2 = (FLAT ((MEAS M0005)))
   POINT = N/OP
SHORTERPROOF

     SETFAULTSTATUS:
     DEVICES = (LOAD)
     REASONS = (OK)

>>WHAT IS THE IC OF Q2
THE COLLECTOR CURRENT OF Q2 IS -.59 MILLIAMPS.

(NEWINFO (CURRENT (.000624971 . .0005651116) C/Q2 (FLAT ((MEA
 M0006)))))
(NEWINFO (CURRENT (8.922815E-6 . 4.347013E-6) B/Q2 (FLAT ((BETA Q2
C/Q2)) (Q2))))
(NEWINFO (CURRENT (-.0005694587 . -.0006335199) E/Q2 (FLAT ((BETA
Q2 C/Q2)) (Q2))))

>>WHAT IS THE V BETWEEN NODES 4 AND 5
THE VOLTAGE BETWEEN N4 AND N5 IS 1.64 VOLTS.

(NEWINFO (VOLTAGE (1.723952 . 1.559766) (N4 N5) (FLAT ((MEAS
 M0007)))))
(NEWINFO (VOLTAGE (2.090392 . 1.81307) (N4 N8) (FLAT ((LOOP N4 N5
N8)) (CR))))
(NEWINFO (VOLTAGE (2.090392 . 1.891307) (N4 N9) (FLAT ((LOOP N4 N5
N9)) NIL)))

>>WHAT IS THE VOLTAGE ACROSS R22
THE VOLTAGE ACROSS R22 IS .875 VOLTS.

(NEWINFO (VOLTAGE (.9184789 . .8310047) (N6 N5) (FLAT ((MEAS
 M0008)))))
(NEWINFO (VOLTAGE (-.6412872 . -.8929473) (N6 N4) (FLAT ((LOOP N 6
N5 N4)) NIL)))
(NEWINFO (VOLTAGE 1.284919 . 1.162546) (N6 N8) (FLAT ((LOOP N6 N5
N8)) (CR))))
(NEWINFO (VOLTAGE 1. 284919 . 1.162546) (N6 N9) (FLAT ((LOOP N6 N5
N9)) NIL)))
(NEWINFO (CURRENT (.001800939 . .001629421) R/R22 (FLAT ((VCVT R22))
(R22))))
```

```
(NEWINFO (CURRENT (-.001629421 . -.001800939) L/R22 (FLAT ((VCVT R22))
(R22))))

>>WHAT IS THE I THRU R22
THE CURRENT THRU R22 IS 35 MILLIAMPS.

COINCIDENCE:
OLDREASON = (FLAT ((VCVT R22)) (R22))
NEWREASON = (FLAT ((MEAS M0009)))
OLD VALUE = (.001800939 . .001629421)
NEWVALUE = (.03673916 . .03324019)
POINT = R/R22

   CONFLICT:
   REASON1 = (FLAT ((VCVT R22)) (R22))
   REASON2 = (FLAT ((MEAS M0009)))
   POINT = R/R22
SHORTERPROOF

     SETFAULTSTATUS:
     DEVICES = (R22)
     REASONS = (UNKNOWN)
YOU KNOW ENOUGH TO BE ABLE TO DEDUCE WHICH COMPONENT IS FAULTED.

>>WHAT IS THE I THRU C5
THE CURRENT THRU C5 IS 0.0 AMPS.

COINCIDENCE:
OLDREASON = (FLAT ((CAPACITOR C5)) (C5))
NEWREASON = (FLAT ((MEAS M0010)))
OLDVALUE = 0.0
NEWVALUE = (1.230295E-9 . 1.113124E-9)
POINT = N/C5

   CORROBORATE:
   REASON1 = (FLAT ((CAPACITOR C5)) (C5))
   REASON2 = (FLAT ((MEAS M0010)))
   POINT = N/C5

THAT MEASUREMENT WAS USELESS,
IT NEITHER TOLD YOU MORE ABOUT THE CIRCUIT,
NOR DID IT ELIMINATE ANY OF THE POSSIBLE FAULTS.

>>REPLACE R22
IS R22 OPEN OR SHORTED OR DOES IT HAVE THE WRONG VALUE?
> WRONG VALUE
IS THE VALUE TOO HIGH OR TOO LOW?
> LOW
R22
REPLACED.
```

Now you should take measurements to see if there is anything else wrong with the circuit.

References

[1] Woods, W.A., R.M. Kaplan, and B. Nash-Webber, "The Lunar Sciences Natural Language Information System: Final Report." BBN Report 2378, Bolt Beranek and Newman Inc., Cambridge, Massachusetts, 1972.

[2] Simmons, R.F. "The clown's world." In R. Schank and B.L. Nash-Webber (Eds.), Proceedings of a Workshop on Theoretical Issues in Natural Language Processing, Cambridge, Massachusetts, 1975.

[3] Bates, M. "Syntactic Analysis in a Speech Understanding System." BBN Report 3116, Bolt Beranek and Newman Inc., Cambridge, Massachusetts, 1975.

[4] Chomsky, N. Syntactic Structures. The Hague: Mouton and Co., 1957.

[5] Woods, W.A. "Transition Network Grammars for Natural Language Analysis." Communications of the ACM. 13(1970), 591-606.

[6] Marcus, M. "Diagnosis as a Notion of Grammar." In R. Schank and B.L. Nash-Webber (Eds.), Proceedings of a Workshop on Theoretical Issues in Natural Language Processing, Cambridge, Massachusetts, 1975.

[7] Woods, W.A. "An Experimental Parsing System for Transition Network Grammars." In Randall Rustin (Ed.), Natural Language Processing, New York: Algorithmics Press, 1973.

[8] Brown, A.L. and G.J. Sussman, "Localization of Failures in Radio Circuits A Study in Causal and Teleological Reasoning." Memo No. 319, M.I.T., Artificial Intelligence Laboratory, Cambridge, Massachusetts, 1974.

APPENDIX A
CONTRASTING FORMALISMS

This appendix gives examples of a semantic grammar rule expressed both as LISP functions and an ATN. It also shows a trace of the ATN version parsing a phrase.

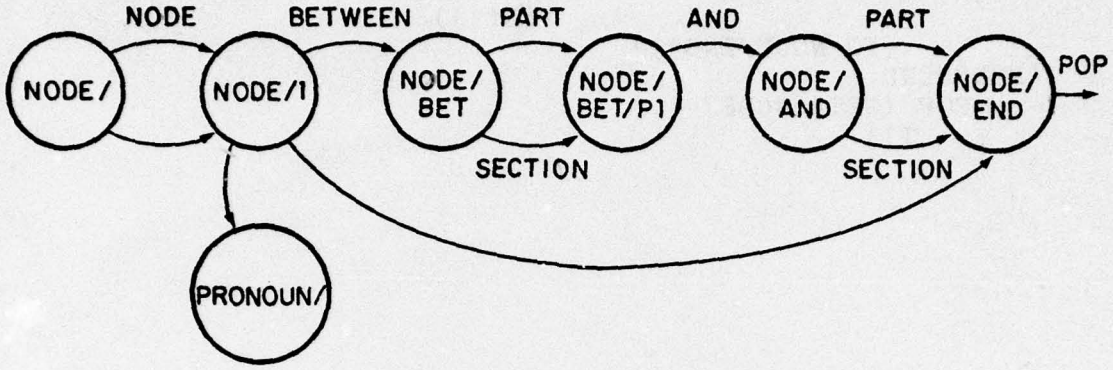
THE <NODE> RULE FROM SOPHIE'S LISP/BNF SEMANTIC GRAMMAR
BNF VERSION

```
<node> := junction of <part/spec> and <part/spec>
          node between <section> and <section>
          [point] between <part/spec> and <part/spec>
          <node/name> ! [node] <node/number>
          <pronoun>
```

LISP VERSION

```
(DEFINEQ
(<NODE>
  (LAMBDA (STR N)
    (PROG (TS1 R1)
          (RETURN (COND
                    ((AND (SETQ TS1
                            (OR (GOBBLE (CHECKLST STR
                                                  (QUOTE (JUNCTION
                                                          POINT)))
                                        (QUOTE (OF BETWEEN)))
                                (CHECKWRD STR (QUOTE BETWEEN))))
                          (SETQ TS1 (OR (<PART/SPEC> (CDR TS1))
                                        (<SECTION> (CDR TS1))))
                          (SETQ R1 RESULT)
                          (SETQ TS1 (CDR (CHECKWRD (CDR TS1)
                                                   (QUOTE AND))))
                          (SETQ STR (OR (<PART/SPEC> TS1)
                                        (<SECTION> TS1)))
                          (SETQ RESULT (LIST (QUOTE NODE/BETWEEN)
                                             R1 RESULT))
                          STR))
                    ((<NODE/NAME> STR))
                    ((SETQ TS1
                        (<NODE/NUMBER>
                          (OR (CDR (CHECKLST STR (QUOTE (NODE))
                                             N))
                              STR)
                          N))
                      (SETQ RESULT (PACK (LIST (QUOTE N)
                                               RESULT)))
                      TS1)
                    ((<PRONOUN> STR N (QUOTE (NODE)))))))))
)
```

THE NODE NETWORK FROM SOPHIE'S SEMANTIC ATN




```
(NODE/
      (WRD (NODE N) T
           (TO NODE/1))
      (JUMP NODE/1 T))
(NODE/1
      (WRD (BETWEEN JUNCTION) T
           (TO NODE/BET))
      (CAT NODE T
           (SETR NODE *)
           (TO NODE/END))
      (CAT INTEGER (AND (IGREATERP * -1)
                        (ILESSP * 27))
           (SETR NODE (PACK (LIST (QUOTE N)
                                  *)))
           (TO NODE/END))
      (JUMP PRONOUN/ (NULLR NOPRO)
           (SETRQ TYPES (NODE))))
(NODE/BET
   (PUSH PART/ T
        (SETR PART1 *)
        (TO NODE/BET/P1))
   (CAT SECTION T
        (SETR PART1 *)
        (TO NODE/BET/P1)))
(NODE/BET/P1
```

```
   (WRD AND T
        (TO NODE/BET/AND)))
(NODE/BET/AND
   (PUSH PART/ T
        (SETR NODE (BUILDQ (NODE/BETWEEN + *)
                           PART1))
        (TO NODE/END))
   (CAT SECTION T
        (SETR NODE (BUILDQ (NODE/BETWEEN + *)
                           PART1))
        (TO NODE/END)))
(NODE/END
   (POP (GETR NODE)
      T))
)
```

SAMPLE OF NODE NETWORK PARSING A PHRASE

```
PARSE((NODE BETWEEN Q5 AND D6]
Parsing:   (NODE BETWEEN Q5 AND D6)
Starting alternative 0
At arc NODE/
Node = (((NODE KNOWNWORD (&)) ((BETWEEN KNOWNWORD &) (&
 &))))
Taking WRD (NODE N) arc NODE/-1-1
Entering state NODE/1
Node = (((BETWEEN KNOWNWORD (&)) ((Q5 PART & TRANSISTOR &)
 (& &))))
Taking WRD (BETWEEN JUNCTION) arc NODE/1-1-1
Entering state NODE/BET
Node = (((Q5 PART (&) TRANSISTOR (&)) ((AND KNOWNWORD &) (&
 NIL))))
      Storing alt 1 for arc NODE/BET-2
Taking PUSH arc NODE/BET-1
PUSHing for PART/
 Taking CAT PART arc PART/-1-1
    Setting PART to Q5

Entering state PART/END
 Node = (((AND KNOWNWORD (&)) ((D6 PART & DIODE &) NIL)))
 Taking POP arc PART/END-1
 Trying to POP
(Continuing arc NODE/BET-1-PUSH)
   Setting PART1 to Q5
Entering state NODE/BET/P1
Node = (((AND KNOWNWORD (&)) ((D6 PART & DIODE &) NIL)))
Taking WRD AND arc NODE/BET/P1-1
Entering state NODE/BET/AND
Node = (((D6 PART (&) DIODE (&)) NIL))
      Storing alt 3 for arc NODE/BET/AND-2
Taking PUSH arc NODE/BET/AND-1
PUSHing for PART/
 Taking CAT PART arc PART/-1-1
    Setting PART to D6

Entering state PART/END
 Node = (NIL)
 Taking POP arc PART/END-1
 Trying to POP
(Continuing arc NODE/BET/AND-1-PUSH)
   Setting NODE to (NODE/BETWEEN Q5 D6)
Entering state NODE/END
Node = (NIL)
Taking POP arc NODE/END-1
Trying to POP
Trying to SUCCEED
```

(NODE/BETWEEN Q5 D6)

ATN OBJECT CODE OF NODE NETWORK

```
NODE/
    (COND
      ((ARCMEM (NODE N))
        (NEXTLEXALT NODE/)
        (TRACEARC WRD (NODE N)
                  NODE/-1-1)
        (DOTO NODE/1)
        (GO NODE/1)))
NODE/-1-2
    (NEXTLEXALT NODE/)
    (TRACEARC JUMP NIL NODE/-1-2)
    (DOJUMP NODE/1)
    (GO NODE/1)
NODE/1
    (COND
      ((ARCMEM (BETWEEN JUNCTION))
        (NEXTLEXALT NODE/1)
        (TRACEARC WRD (BETWEEN JUNCTION)
                  NODE/1-1-1)
        (DOTO NODE/BET)
        (GO NODE/BET)))
NODE/1-1-2
    (COND
      ((ARCCAT NODE)
        (NEXTLEXALT NODE/1)
        (TRACEARC CAT NODE NODE/1-1-2)
        (SETR (QUOTE NODE)*)
        (DOTO NODE/END)
        (GO NODE/END)))
NODE/1-1-3
    (COND
      ((AND (ARCCAT INTEGER)
            (IGREATERP * -1)
            (ILESSP * 27))
        (NEXTLEXALT NODE/1)
        (TRACEARC CAT INTEGER NODE/1-1-3)
        (SETR (QUOTE NODE)
              (PACK (LIST (QUOTE N)*)))
        (DOTO NODE/END)
        (GO NODE/END)))
NODE/1-1-4
    (COND
      ((NULLR NOPRO)
        (NEXTLEXALT NODE/1)
        (TRACEARC JUMP NIL NODE/1-1-4)
        (SETRQ TYPES (NODE))
        (DOJUMP PRONOUN/)
        (GO PRONOUN/)))
```

```
    (CHANGESTATEQ NODE/1)
    (GO NEXTLEX)
NODE/BET
    (COND
      ((STRINGLEFTP)
        (ALTARC NODE/BET-2)
        (TRACEARC PUSH NIL NODE/BET-1)
        (DOPUSH PART/ NODE/BET-1-PUSH)
        (GO PART/)))
NODE/BET-2
    (COND
      ((ARCCAT SECTION)
        (NEXTLEXALT NODE/BET)
        (TRACEARC CAT SECTION NODE/BET-2)
        (SETR (QUOTE PART1)*)
        (DOTO NODE/BET/P1)
        (GO NODE/BET/P1)))
    (CHANGESTATEQ NODE/BET)
    (GO NEXTLEX)
NODE/BET-1-PUSH
    (SETR (QUOTE PART1)*)
    (DOPTO NODE/BET/P1)
    (GO NODE/BET/P1)
NODE/BET/AND
    (COND
      ((STRINGLEFTP)
        (ALTARC NODE/BET/AND-2)
        (TRACEARC PUSH NIL NODE/BET/AND-1)
        (DOPUSH PART/ NODE/BET/AND-1-PUSH)
        (GO PART/)))
NODE/BET/AND-2
    (COND
      ((ARCCAT SECTION)
        (NEXTLEXALT NODE/BET/AND)
        (TRACEARC CAT SECTION NODE/BET/AND-2)
        (SETR (QUOTE NODE)
              (BUILDQ (NODE/BETWEEN + *)
                      PART1))
        (DOTO NODE/END)
        (GO NODE/END)))
    (CHANGESTATEQ NODE/BET/AND)
    (GO NEXTLEX)
NODE/BET/AND-1-PUSH
    (SETR (QUOTE NODE)
          (BUILDQ (NODE/BETWEEN + *)
                  PART1))
    (DOPTO NODE/END)
    (GO NODE/END)
NODE/BET/P1
    (COND
      ((ARCWRD AND)
```

```
            (NEXTLEXALT NODE/BET/P1)
            (TRACEARC WRD AND NODE/BET/P1-1)
            (DOTO NODE/BET/AND)
            (GO NODE/BET/AND)))
        (CHANGESTATEQ NODE/BET/P1)
        (GO NEXTLEX)
NODE/END
        (NEXTLEXALT NODE/END)
        (TRACEARC POP NIL NODE/END-1)
        (DOPOP (GETR NODE)
               NIL)
        (GO EVALARC)
```


☆U.S. GOVERNMENT PRINTING OFFICE: 1977-771-057/20