ED 135 365 IR 004 456

AUTHOR	Brown, John S.; And Others
TITLE	Steps Toward a Theoretical Foundation for Complex, Knowledge-based CAI.
INSTITUTION	Bolt, Beranek and Newman, Inc., Cambridge, Mass.
SPONS AGENCY	Army Research Inst. for the Behavioral and Social Sciences, Arlington, Va.; Navy Personnel Research and Development Center, San Diego, Calif.
REPORT NO	BBN R-3135; ICAI R-2
PUB DATE	Aug 75
CONTRACT	DAHC-19-74-C-0060
NOTE	148p.
EDRS PRICE	MF-$0.83 HC-$7.35 Plus Postage.
DESCRIPTORS	Cognitive Processes; *Computer Assisted Instruction; *Computer Science; Display Systems; Educational Games; Input Output Devices; Programed Instruction; *Programed Tutoring; Programing Languages; Research Projects; Tutoring

ABSTRACT

This report describes research directed at designing and evaluating computer assisted instructional (CAI) systems capable of inferring structural models of a student's reasoning strategies and identifying his underlying misconceptions. Several prototype systems using representative domains of knowledge were constructed. From these an information processing framework comprising models of expert reasoners, adaptive tutors, and students, have evolved. Section 1 describes two paradigmatic instructional systems involving a decision making and a gaming environment. Section 2 explores issues of building intelligent instructional systems over more complex domains of knowledge. Section 3 describes research related to the design of robust intelligent systems. (Author/WBC)

BBN Report No. 3135
ICAI Report No. 2

STEPS TOWARD A THEORETICAL FOUNDATION FOR COMPLEX, KNOWLEDGE-BASED CAI

By

John Seely Brown
Richard Burton
Mark Miller
Johan deKleer
Stephen Purcell
Catherine Hausmann
Robert Bobrow

August 1975*

This research was supported, in part, by the Army Research Institute, and by the Navy Personnel Research and Development Center under Contract No. DAHC-19-74-C-0060.

*A limited printing of this report was done in 1975 and is now being reprinted as a report in our Intelligent Computer Assisted Instruction (ICAI) series.

ABSTRACT

This report describes research directed at designing and evaluating instructional systems which are able to use their knowledge to mimic some of the capabilities of a good tutor such as being able to construct/infer structural models of a student's reasoning strategies and to identify his underlying misconceptions. Our basic research methodology has been to construct several highly modifiable, prototype systems, each of which emphasizes some aspect of our intelligent CAI paradigm. By carefully choosing restricted (but representative) domains of knowledge for these prototype systems, an information processing framework comprising models of expert reasoners, adaptive tutors, and students has evolved.

This report is broken into three sections. The first section describes two paradigmatic instructional systems built around a decision making and a gaming environment. Each of these systems illustrates some of the needs and techniques for automatically inducing and using a structural model of the student's reasoning strategies. The second section moves away from these elegant but simple domains of knowledge and explores some of the issues of building intelligent instructional systems over more complex domains of knowledge, such as remedial mathematics. The last describes some research related to the general issue of how to design robust intelligent systems.

ACKNOWLEDGEMENTS

We are indebted to Dr. Leon Nawrocki for his numerous, and insightful comments on the rough draft of this document and for his suggestions during the preliminary stages of this research. We have also profited from discussion with Drs. Beatrice Farr, Bruce Knerr and Dexter Fletcher. Dr. Farr was the first person to convince us of the need for structural models of the learner (as opposed to models of learning). Her editorial suggestions were also very helpful in debugging the final version of the report. Dr. Fletcher was instrumental in increasing our awareness of the latent complexities of constructing meaningful structural models of the learner. Finally we would like to thank Dr. Guy Groen for his contribution to the notion of paradigmatic instructional systems.

TABLE OF CONTENTS

Page

INTRODUCTION

We are at the threshold of a dramatic advancement in computer technology which should change the way computers are employed in instruction. This technological advance will decrease the cost of computer hardware to the extent that each student will have available computational resources which are currently restricted to a few elite users. The greatest challenge facing educational technologists is to productively harness the increased availability of computational power to provide an equally dramatic improvement in the quality and cost-effectiveness of instructional systems. Traditional computer-assisted instruction (CAI) paradigms were developed under the assumption that computational power is a scarce resource and these paradigms are, for the most part, incapable of exploiting the latest technological advances. To effectively use the increased availability of computational power requires a re-evaluation of the role of the computer in instructional paradigms.

This report describes research directed at understanding and designing instructional systems which take fuller advantage of the computational power afforded by new technology. This new kind of instructional system will be able to do more than spew forth its knowledge as factual information. It will be able to use its knowledge to form structural models of a student's reasoning strategies, to determine his strengths and weaknesses, and to identify his misconceptions. Once it forms such a model of the learner, it will then use this knowledge to determine when and how to provide remediation, heuristic recommendations ("hints"), or further instruction. To expect an instructional system to make such decisions solely on the basis of its own knowledge represents a substantial shift away from the basic notion of fixed instructional (linear or branching) sequences toward a more autonomous, truly adaptive, individualized instruction.

Our basic methodology for understanding the design and operation of such instructional systems is to construct several highly modifiable, prototype systems, each of which emphasizes some aspect of the overall adaptive approach. By carefully choosing restricted (but representative) domains of knowledge for the prototype system, we have begun to uncover a viable information processing framework of "intelligent" instructional systems, comprising models of expert reasoners, adaptive tutors, and students. This general framework will guide us in coping with less restricted domains of knowledge and in increasing the tutorial capabilities over the limited domains.

Prior to the research described in this report, we have constructed CAI systems that use their built-in intelligence to evaluate a student's answers and construct counter-examples to his hypotheses <Brown et al 74>. From these systems, arose the need for tutorial initiative on the part of the system to interrupt the student and comment on some failing in the student's behavior. For any but the most trivial domain, it is clear that such interruption had to be based on an accurate "model" of the particular student in order to avoid inappropriate or irrelevant comments. One of the aims of the research performed under this contract was the development of such models. From this research we discovered the difficulty of constructing a system that can automatically formulate a model of what the student is doing which succinctly represents his reasoning and knowledge to the extent necessary for isolating his fundamental misconceptions. To complicate the development task further, the "intelligent" adaptive tutor must then be able to take advantage of the synthesized information (e.g. structural models of the student's reasoning and knowledge) in order to generate pedagogically sound criticisms or hints.

This report is broken into three sections. The first section (chapters one and two) begins with an explanation of the need for a fundamental change in point of view with regard to the design of complex CAI systems. The remainder of chapter one and chapter two describe two paradigmatic instructional systems which were constructed to investigate (and establish the foundation for) a new viewpoint for CAI. In the proposed p[illegible]digm the instructional system itself contains an "intelligent" tutoring module capable of automatically inducing and using a structural model of the student's reasoning strategies. Chapter one describes a system to monitor a student's decision making activities over the domain of "attribute blocks". Chapter two describes a system to enhance the educational effectiveness of a Plato drill-and-practice game. In addition to the tutoring modules, both systems contain expert problem-solving modules which assist the tutoring module in constructing the model of the student and assist the student when help is requested. The subject domains of these systems were necessarily restricted to being structurally "ideal" or simple in order to expedite the investigation.

The second section (chapters three and four) moves away from these elegant but simple domains of knowledge and explores the problem of building intelligent adaptive instructional systems over more complex domains of knowledge, in particular the domains of electronics and remedial mathematics. Chapter three (on electronics) focuses primarily on a logical theory of troubleshooting and offers a precise information processing model of how an expert troubleshooter reasons. This model will play a fundamental role in the design of any complete "intelligent" instructional system for teaching electronics via the troubleshooting methodology. Chapter four addresses the problem of teaching procedural knowledge (as opposed to the axiomatic knowledge underlying basic electronics). For these initial investigations we have chosen the large and mostly unspecified procedural knowledge that underlies the ability to solve high school level algebra problems. By making explicit all the tacit or implicit knowledge comprising the procedural skills of algebra simplification, we will achieve the first step in creating an instructional system that can isolate and remediate "bugs" or mistakes contained in a person's internal representation of these procedures (skills).

The last section (chapters five and six) describes some research related to the general issue of how to design and use intelligent instructional systems. Chapter five focuses on the use of the "intelligence" or knowledge base embedded in the instructional system to achieve a new dimension in man-machine communications. Chapter six sets forth some general guidelines for the design of intelligent computer assisted instructional systems. As such, this last chapter establishes a theoretical cornerstone for a wide range of instructional systems which can fully exploit the computational powers of tomorrow's computers in order to achieve higher quality adaptive instruction.

A word of caution to the reader. Each of these chapters contains many technical details of prototype systems which carry out their designated tasks. As Weizenbaum demonstrated with his ELIZA program <Weizenbaum 66>, the external appearance of programs can be very deceptive and the merits of programs attempting something as complex as tutoring cannot be understood by its behavior. Much of what is novel and interesting in this report is immersed in details of how tasks are performed. We therefore encourage the reader to pay more than passing attention to technical aspects of each chapter. Only those portions of the systems which illustrate either useful techniques or points of view on system organization are described. It is all too easy to talk about structural models of a student and it is all too hard to understand what is involved in actually constructing them -- even over simple knowledge domains.

SECTION I - Paradigmatic Intelligent Instructional Systems

CHAPTER 1
ADAPTIVE INSTRUCTION AND COMPLEX KNOWLEDGE-BASED CAI*

In recent years, the cost of computer hardware (central processing units, memories and the like) has been declining very rapidly. The most obvious manifestation of this is the dramatic drop in cost of the pocket calculator. It is now possible to buy a fairly versatile microcomputer (the Hewlett Packard HP 65) in department stores for a price comparable to an expensive stereo receiver. There is every indication that this trend will continue, and that we will be seeing computers of much greater power available in this price range. Even large scale computers suc as the PDP-10 will be available at a price well within the budget of the average school system. As a result, we can expect CAI systems to become more and more complex. In particular, it seems reasonable to expect a greater use of techniques developed in the field of artificial intelligence, whose applicability has been limited by their use of features available, until recently, in only the largest and most sophisticated computer installations.

On the whole, a CAI system is justifiable only if it is doing something that cannot be done equally well without the use of a computer. The most unique capability of the computer is its ability to make very rapid decisions regarding what to do next. Because of this, it is possible, in principle, to design a system with an adaptive capability that responds rapidly to the needs of an individual student. In the early days of CAI, there was great optimism regarding this kind of capability. In fact, it was claimed that a CAI system would be able to provide the equivalent of a personal tutor for every student. In much current CAI, however, this capability has somehow fallen by the wayside! This is partly because humans are good adaptive devices, in their own fashion. In fact, there are systems such as TICCIT in which the adaptive component is left entirely up to the user. However, it seems reasonable to suppose that there are situations where the computer is superior, and it seems reasonable to ask why such uses have not been developed extensively. A major component of the answer lies in the presently accepted view of adaptive instruction. The problem of adaptive instruction has classically been formulated as the "branching problem" of specifying when, in an instructional sequence, a branch should occur. We will argue that this is the wrong view of the problem and that there is something wrong with the whole notion of an "instructional sequence".

(*)This chapter is a preliminary draft of a paper by Groen, Brown and Burton which was presented by Groen at the AERA conference.

One version of this classical formulation is illustrated in Figure 1.1 <Groen & Atkinson 66>. The basic idea was that the instructional process could be broken down into a sequence of stages. Each stage consisted of a set of displays to which the student responded. The system had three basic components: a history, consisting of a representation of the responses made by the student; a model of the student's learning process; and a decision procedure which enabled the system to decide, on the basis of the history and the model, which stimuli were to be presented during the next stage. Inherent in the approach was the notion that, given a precise enough model, and a precise enough criterion, an optimal decision procedure could be found.

As Atkinson has demonstrated in his more recent work, this is a reasonable framework for looking at vocabulary learning and possibly certain kinds of drill and practice. However, it does not extend to more complex situations! There are at least three reasons for this:

(1) The simplistic notions of "stimulus" and "response" lack the expressive power needed to describe large, complex knowledge structures and the processes which manipulate them.

(2) Developing adequate mathematical models for complex information processing tasks is extremely difficult and may not be possible.

(3) The flow diagram notation, with its strict flow of control is not a realistic representation of the adaptive systems employing more complex structures.

An example of such a complex system is shown in Figure 1.2, which is taken from a description of a system for teaching the programming language BASIC. Clearly, it is impossible to separate out the dynamics implied by a diagram like this into a strict sequence of stimulus and response components. What we need are techniques for making such a system coherent, so that we can isolate aspects of it, construct models for it, set up desirable criteria and devise satisfactory procedures for meeting these criteria. In the case of the paired associate learning tasks (for which the simple flowcharts of Figure 1.1 were appropriate), researchers were able to operate within the framework of a simple paradigmatic scenario. The problem in creating adaptive <u>intelligent</u> CAI at the present time is to formulate such a paradigmatic scenario. The question which must be answered positively if there is to be any rationale for the design of adaptive CAI systems is whether such paradigmatic scenarios exist. The remainder of this chapter will

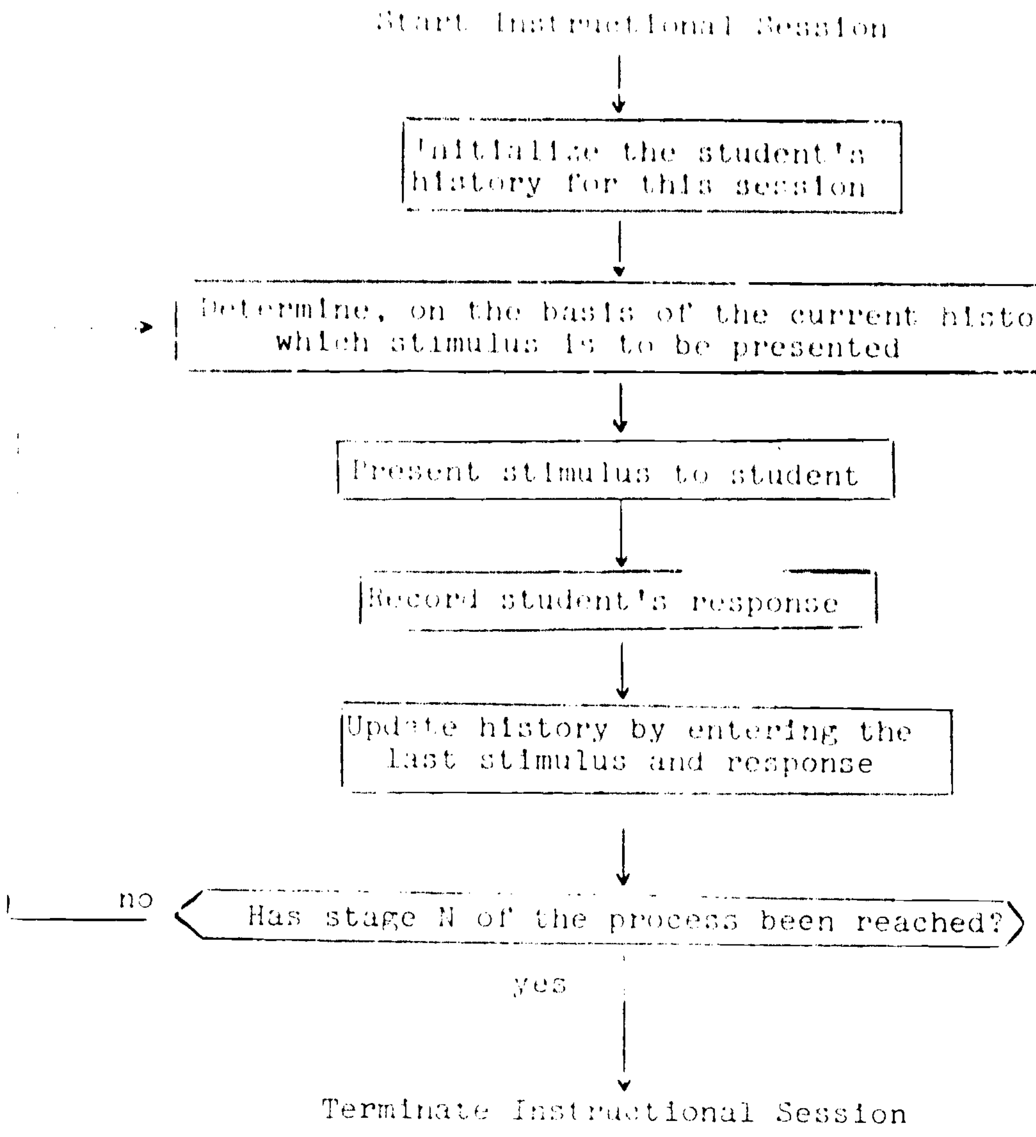

Start Instructional Session
Initialize the student's history for this session
Determine, on the basis of the current history which stimulus is to be presented
Present stimulus to student
Record student's response
Update history by entering the last stimulus and response
no
Has stage N of the process been reached?
yes
Terminate Instructional Session

FIGURE 1.1
Flow Diagram for an Instructional System

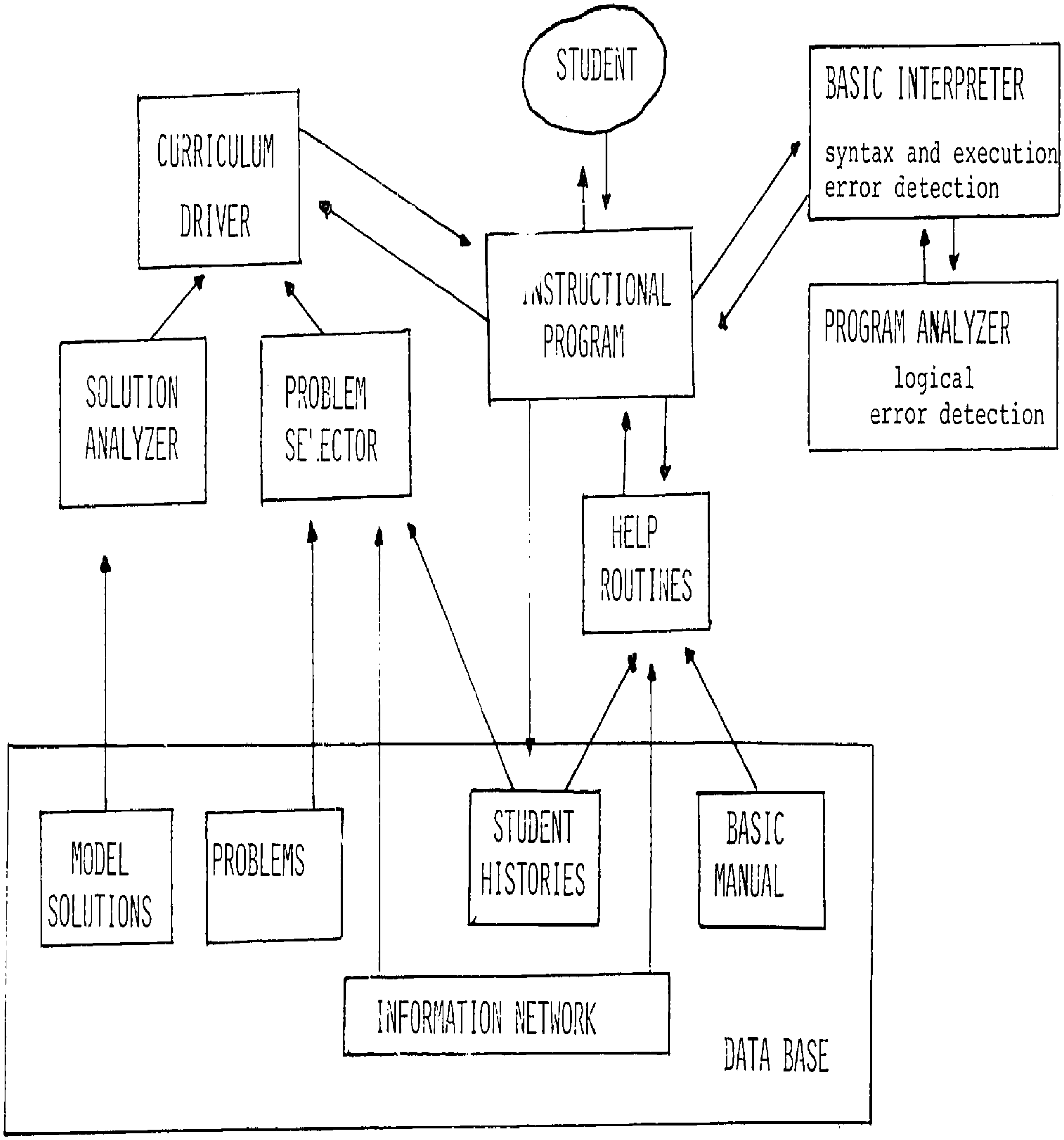


Figure 1.2

BIP's Information Flow Diagram

be devoted to exploring a specific instance that provides an example of what we mean by a paradigmatic system.

A prerequisite for a paradigmatic system is a subject domain which has a simple and elegant structure. The domain must have a logical formulation which is both well-defined and easily specifiable. Its logical structure must also support natural mappings (analogies) into the kinds of complex and real world domains hat instructional systems are intended to handle.

A domain which appears to be ideal for this purpose derives from that part of the world of manipulatory mathematics known as Attribute Blocks. Although Attribute Blocks can be used to explore a rich variety of interesting common sense reasoning principles, we will focus on just one application, a game which combines the notions of logic, decision making and hypothesis formation into an interesting exercise on how to ask optimal questions and how to draw inferences from the answers.

Description of the Game

This game is played with the 32 attribute blocks, a deck of attribute cards and 2 looped strings. Each block has three attributes:

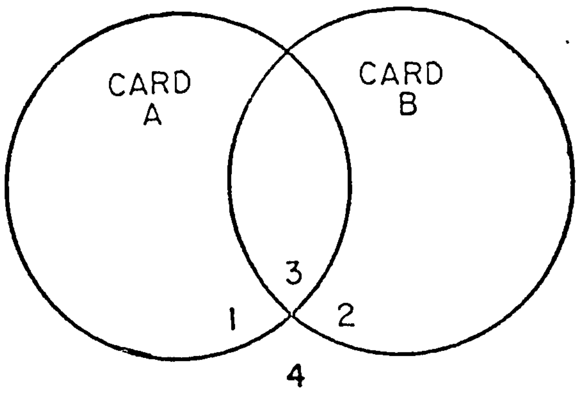
SIZE: small or large
COLOR: red or yellow or green or blue
SHAPE: triangle or square or circle or diamond

There is one block in the set of 32 for each possible combination of the values of the three attributes.

The deck is made up of 18 cards. Written on each card is an attribute value or the negation of a value.

1. LARGE	7. TRIANGLE	13. NOT YELLOW
2. SMALL	8. CIRCLE	14. NOT GREEN
3. BLUE	9. SQUARE	15. NOT TRIANGLE
4. RED	10. DIAMOND	16. NOT CIRCLE
5. YELLOW	11. NOT BLUE	17. NOT SQUARE
6. GREEN	12. NOT RED	18. NOT DIAMOND

The student takes the two looped strings and overlaps them like:



This arrangement of the looped strings has created four areas. Area 1 is inside the string on the left (loop A) and outside the string on the right (loop B). Area 2 is inside loop B and outside loop A. Area 3 is inside both loops. Area 4 is outside both loops.

The [illegible]res labelled Card A and Card B in Areas 1 and 2 represent [illegible]vo cards which the teacher chooses from the deck of cards. The student is NOT told which cards have been chosen. The object of the game is for the student to guess the attribute value written on these two cards. To do this the student chooses blocks one at a time and asks the teacher where the block goes (according to the rule that a block is placed inside of loop A only if it satisfies the value on Card A, and inside of loop B only if it satisfies the value on Card B, e.g. if Card A=SQUARE and Card B=NOT BLUE, then the Large Yellow Square goes in Area 3). The student continues choosing blocks, asking where they go, and placing them there until he believes he has placed enough blocks to uniquely identify (i.e. deduce) what both of the cards are.

What can a computer do for this environment?

Manipulatory math tools represent, in our opinion, one of the best uses of simple, inexpensive technology that we have encountered. It was therefore with some trepidation that we considered contaminating this otherwise simple domain with "high technology". Of course, if it served our criteria for paradigmatic systems, that might be enough justification, but after watching numerous people use attribute blocks we felt there were many important tasks that could be better accomplished (and in some cases, only accomplished) by having a knowledge based CAI system.

Protocols

Instead of providing a theoretical description of the kind of facilities our system provides for this highly structured environment, we will provide three annotated protocols. Each protocol builds on the previous one and illustrates additional tutoring features which are realized by having the instructional system contain additional information processing capabilities. However before we can meaningfully describe these protocols we must refer to the basic architecture of this system. Figure 1.3 illustrates the functional decomposition of the system into the modules referred to below.

The first protocol stems from a relatively simple version of Figure 1.3 in which there are only three monitors, and a tutor which performs no mediation of the output of these monitors. The first monitor (who heavily utilizes the "expert") evaluates the student's conjecture about what a card is and decides 1) if the conjecture is necessarily correct (i.e. do the blocks placements entail that card and only that card), 2) if it is consistent with the known information (i.e. block placements but that there are still other possibilities for the card), or 3) if the conjecture is inconsistent and if so selects a counter-example. The second monitor determines if the placement of the chosen block could have been deduced from the existing blocks, i.e. is it a logically redundant block. From a strict information theoretic point of view, such choices reflect informationless questions. Detecting such questions is an easy task for the monitor to perform but a difficult one for a human teacher. The third monitor determines when there is sufficient information (blocks already placed) to be able to deduce what one of the cards is.

These monitors are constantly watching the environment (i.e. student moves) and "wake up" the tutor whenever they detect something of interest, informing the alerted tutor of what it was that they found worth waking him up for. Note that even in this elementary version of the system the exact flow of control is not easy to specify since it depends on when and why a monitor interrupts the normal course of action of the system.

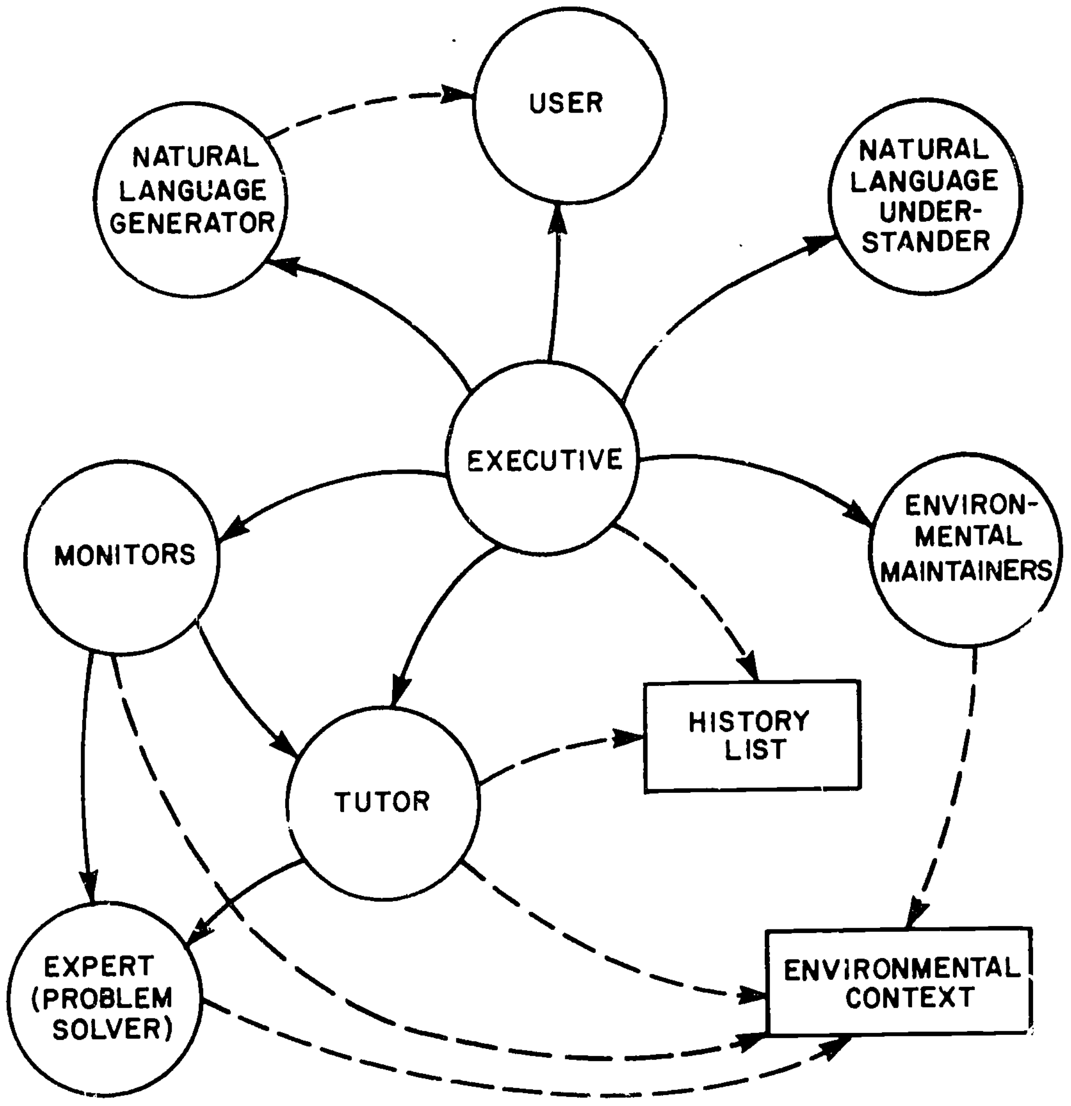

USER
NATURAL LANGUAGE GENERATOR
NATURAL LANGUAGE UNDER-STANDER
EXECUTIVE
MONITORS
ENVIRON-MENTAL MAINTAINERS
HISTORY LIST
TUTOR
EXPERT (PROBLEM SOLVER)
ENVIRONMENTAL CONTEXT

Figure 1.3

Diagram of Blocks System

PROTOCOL 1
Attribute Block Game

This is an example of a student using a simple version of the Attribute Blocks system. The lines which begin with "* " were typed by the student.

WELCOME TO THE ATTRIBUTE BLOCKS WORLD

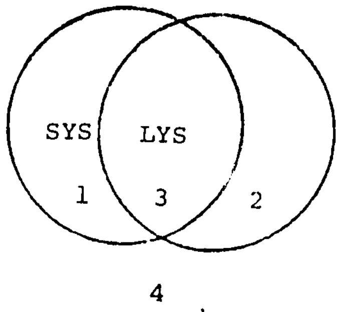
TWO CARDS HAVE BEEN CHOSEN.
System selects two cards.

----CHOOSE A BLOCK----
* LARGE Y SQ

The student tells the system which block he would like placed. The system will recognize any reasonable abbreviation of a block description.

PUT THE LARGE YELLOW SQUARE IN AREA 3

System tells him where to place his block in accordance with what the cards are.



----CHOOSE A BLOCK----
* SMALL YELLOW SQUARE
PUT THE SMALL YELLOW SQUARE IN AREA 1
YOU SHOULD BE ABLE TO DEDUCE CARD B NOW.

The monitor watching the possibilities space wakes up the tutor to tell the student a potentially surprising fact: There is only one possible theory left for Card B even though there isn't a single block in AREA 2 and only two blocks placed! This is an example of the type of monitoring humans are not very good at.

----CHOOSE A BLOCK----
* IS CARD B LARGE?

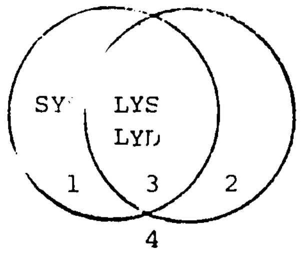
Student forms a conjecture. In the non-computerized version of the game, this question had only two answers; yes and no. That is, the conjecture is the same as the card or it is not. The tutor allows this to be answered in three ways: Yes, No, or Maybe. In this case LARGE is the only <u>possible</u> value Card B could have; so the answer is yes.

YES, THAT IS CORRECT.

----CHOOSE A BLOCK----
* IS CARD A SMALL?

Student forms another conjecture. In this case, the tutor finds a counter-example to his conjecture so the answer is "no and here's why".

NO. IF CARD A WERE SMALL, THEN THE LARGE YELLOW SQUARE WHICH IS IN AREA 3 WOULD BE OUTSIDE OF LOOP A.



----CHOOSE A BLOCK----
* LARGE Y DI
PUT THE LARGE YELLOW DIAMOND IN AREA 3

----CHOOSE A BLOCK----
* IS CARD A YELLOW?

The student forms another conjecture. In this case the tutor informs him that his "guess" is consistent with the known information but that it is just a guess because there are still other possibilities. This is an example of "maybe." Note that a human teacher would have a hard time distinguishing this fact and would simply answer yes or no depending on whether Card A was Yellow!

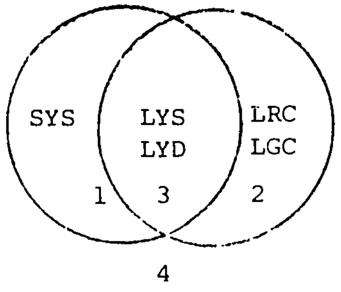
IT COULD BE, BUT IT COULD ALSO BE NOT RED, NOT GREEN, NOT BLUE, NOT CIRCLE, OR NOT TRIANGLE.

----CHOOSE A BLOCK----
* L R CI
PUT THE LARGE RED CIRCLE IN AREA 2

This particular block performs a perfect split of the hypothesis space, i.e. it rules out NOT GREEN, NOT BLUE, and NOT TRIANGLE.

----CHOOSE A BLOCK----
* L G CI
PUT THE LARGE GREEN CIRCLE IN AREA 2

Rules out NOT RED.



20

```
----CHOOSE A BLOCK----
*  HELP
CARD A CAN NOW BE
YELLOW, NOT CIRCLE.

CARD B CAN NOW BE
LARGE.
```

HELP calls the expert to generate the possibilities left for each card. Now the student must distinguish between the two.

```
----CHOOSE A BLOCK----
*  S Y DI
PUT THE SMALL YELLOW DIAMOND IN AREA 1
YOU DIDN'T GAIN ANY INFORMATION BY THAT.
```

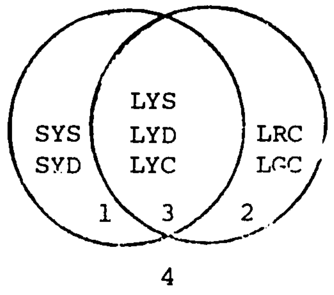
Another monitor decides that the answer to that question or block placement was deducible from the known information. Since the block the student chose contained both possible values for CARD A (it is both YELLOW and NOT CIRCLE), it necessarily had to fall in CARD A's area (AREAS 1 or 3), and therefore couldn't give him any new information about the value of Card A.

```
----CHOOSE A BLOCK----
*  L Y CI
PUT THE LARGE YELLOW CIRCLE IN AREA 3
YOU SHOULD BE ABLE TO DEDUCE CARD A NOW.
```

This block contains the negation of <u>one</u> of the hypotheses for CARD A (NEGATION(NOT CIRCLE))=CIRCLE; and holds the other one constant (YELLOW). This is the correct strategy for breaking down a hypothesis space. The tutor tells him that he did just that and now should be able to deduce the card.

```
----CHOOSE A BLOCK----
*  IS CARD A YELLOW
YES, THAT IS CORRECT.
YOU HAVE NOW GUESSED BOTH CARDS.
```

--which he does!



21

This second protocol reflects a slightly more versatile tutor. In this version, the tutor performs some services besides acting as a mouthpiece for a monitor. For example, when the tutor is alerted that a card can now be deduced, it tries to get the student to see why this is the case. Also, whenever the tutor is alerted by the monitor, which is watching for a student to choose a redundant block, the tutor, in consultation with the expert, invokes one of several strategies for trying to get the student to realize i) the concept or meaning of redundancy, and ii) why his particular choice was redundant.

PROTOCOL 2

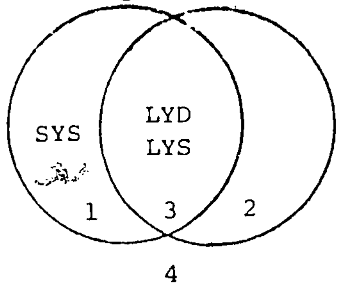
Examples of Tutoring

These are examples of the tutoring which would have transpired in protocol 1 using the more sophisticated tutor. The lines which begin with "**" and "> " were typed by the student, where "**" represents a student's statement and ">" represents a student's response to a tutorial query.

Example when student doesn't deduce a card:

```
----CHOOSE A BLOCK----
**LARGE YELLOW SQUARE
Put the LARGE YELLOW SQUARE in AREA 3.

**S Y SQ
Put the SMALL YELLOW SQUARE in AREA 1
```

The student should be able to figure out what Card B is. But instead of guessing it, he places another block.



```
**L Y DI
Can you tell me some possibilities for  rd B?
> LARGE, SQUARE AND YELLOW
```

The tutor asks the student about Card B. It is possible that the student has just not considered it because the blocks both fall in loop A. When the student gives three possibilities the tutor evaluates each giving counter-examples to the ones which are incorrect.

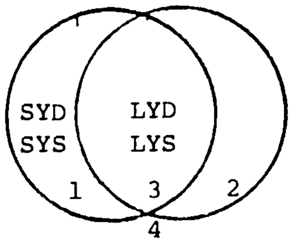
Card B could be LARGE.
But if Card B were SQUARE then the SMALL YELLOW SQUARE which is in Area 1 would be INSIDE of Loop B.
And if Card B were YELLOW then the SMALL YELLOW SQUARE which is in Area 1 would be INSIDE of Loop B.

By taking the initiative, the tutor has gotten the student to observe that his reasoning is incorrect.

Put the LARGE YELLOW DIAMOND in AREA 3

Example of tutoring after placement of a worthless block:

The possible theories are:
Card A can now be YELLOW or NOT CIRCLE.
Card B can now be LARGE.



**S Y DI

The monitor has noticed that the student is placing a worthless block so the tutor is called. The tutor tries to get the student to realize that he could predict where that block will fall.

Where do you think the SMALL YELLOW DIAMOND will go?
> AREA 1
It's true. It could go there.
Is there anyplace else the SMALL YELLOW DIAMOND could go?
> AREA 4

If the student thinks there is someplace else it could go, he must have an incorrect idea of what Card A could be (since he already knows what Card B is.)

What value could Card A have which would cause the SMALL YELLOW DIAMOND to go in Area 4?
> NOT CIRCLE
If Card A were NOT CIRCLE, the SMALL YELLOW DIAMOND would go INSIDE of Loop A and NOT in Area 4.

What value could Card A have which would cause
the SMALL YELLOW DIAMOND to go in Area 4?
> NOT DIAMOND
If Card A were NOT DIAMOND then the LARGE YELLOW DIAMOND
which is in Area 3 would be OUTSIDE of Loop A

What value could Card A have which would cause
the SMALL YELLOW DIAMOND to go in Area 4?
> NONE
Is there anyplace besides Area 1 that the SMALL
YELLOW DIAMOND could go?
> NO
When there is only one place a block can go, it's
not helping you guess what the cards are.

Put the SMALL YELLOW DIAMOND in AREA 1

Example of an alternative tutoring strategy after a
worthless block:

The possible theories are:
Card A can now be YELLOW or NOT CIRCLE.
Card B can now be LARGE.

**S Y DIA

An alternative strategy of the tutor when called by the monitor is to make the student aware that his block is not splitting his hypothesis space. The first step is to get him to verbalize the possibilities. If he were wrong on any of them, counter-examples would be given. If there were more, the tutor would work with the subset the student gives.

Can you tell me some possibilities for Card A?
> YELLOW AND NOT CIRCLE
Both of those are possibilities.

Next the tutor asks if his block splits his hypothesis space. Since he believes it would, the tutor walks him through the deductions to show him differently.

Will the SMALL YELLOW DIAMOND eliminate either of those

```
theories?
> YES
Would the SMALL YELLOW DIAMOND go INSIDE or OUTSIDE
of a loop which had a card value YELLOW?
> INSIDE
Right
Would the SMALL YELLOW DIAMOND go INSIDE or OUTSIDE
of a loop which had a card value NOT CIRCLE?
> I
Right
So you know that the SMALL YELLOW DIAMOND goes INSIDE of
Loop A no matter which of those values Card A has.

Put the SMALL YELLOW DIAMOND in AREA 1
That was an information-less block!
```

The third protocol reflects a substantially expanded set of monitors. In this version, there are monitors which watch for instances when higher order information can be logically deduced (besides just the identity of a card). Such an example might be that, given the current placement of blocks, one can logically conclude that a card could not be any SHAPE. These more abstract characterizations of the remaining possibilities for a card stem from knowing general rules or principles about the structure of this environment. For example, one rule about shapes is if there are two blocks of the same shape with one of them inside a loop and the other outside of a loop, then the card for that loop can't be either a positive or negative shape! This is a non-trivial abstraction and helping students to discover such abstractions is a worthwhile goal. This third version has monitors that understand all these abstractions and whenever one is encountered, it alerts the tutor who can then question the student much as was done in the previous protocol.

In addition to these monitors, the expert's capabilities have been expanded to compute the actual and expected information gain from any particular question (i.e. block choice). This information enables the student to experiment with the effectiveness of different decision strategies and enables the tutor to provide substantive hints as to what is the next best question to ask (i.e. block to choose). This facility in conjunction with the HELP command, which invokes the expert to compute all consistent possibilities for each card, provides an entirely new dimension to this environment -- a dimension that would be nearly impossible to realize without a computer.

PROTOCOL 3

Information and Event Monitoring

This protocol is the same game that was played in Protocol 1, i.e. the same values for the cards and the same sequence of blocks. For this protocol however the information monitor and the event monitors have been turned on. The protocol also points out the use of the HINT command. The st [illegible] nt typed the lines which begin with "**" which is followed by either a block choice or a request for "HELP" or a hint.

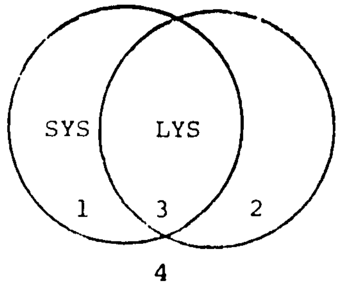
```
Cards A and B have been chosen

--- CHOOSE A BLOCK ---
**LARGE Y SQ
Put the LARGE YELLOW SQUARE in AREA 3
The expected information content of that block was 1.0
The actual information content was 1.0
```

> The unit of information can be thought of a. "the number of blocks required to uniquely o nine both cards". The best block generally has expected information of about 1.

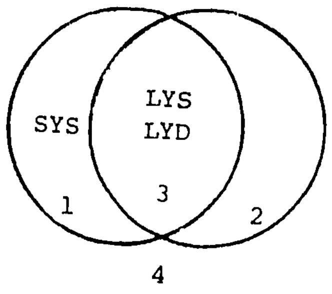
```
**SMALL YELLOW SQUARE
Put the SMALL YELLOW SQUARE in AREA 1
The expected information content of that block was .50
The actual information content was 1.67
```

> Notice that in this case the student gambled with a "bad" block and won. The information monitor provides the student a new tool with which to compare strategies and choose between alternative ways of making decisions.
>
> Below are the event monitors which were fired by the placement of this block. In future systems, these monitors would invoke the tutor similar to the way the placement of a worthless block does now. Notice the number of event monitors which fired -- this confirms the fact that the actual information gained by this block was quite high.

```
Card B must be size
Card B must be positive
Card B can't be shape
```



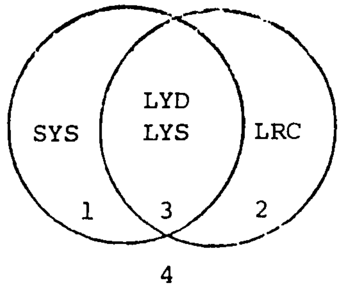
```
Card B can't be color
Card B can't be negative
Card B can't be positive shape
Card B can't be positive color
Card A can't be size
Card B can't be negative color
Card B can't be negative shape
```



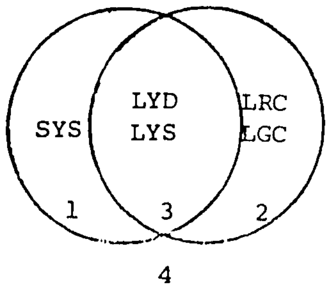
```
**L Y DI
Put the LARGE YELLOW DIAMOND in AREA 3
The expected information content of that block was .41
The actual information content was .21

Card A can't be positive shape

**HINT
I think placing a NOT YELLOW CIRCLE OR TRIANGLE
would be a good idea.
```

The HINT command uses the information measure to determine the set of Blocks with the highest expected gain. An intensional description of this set is then calculated and given to the student. The intensional description allows the student to see which attributes are critical to the splitting of the hypothesis space.

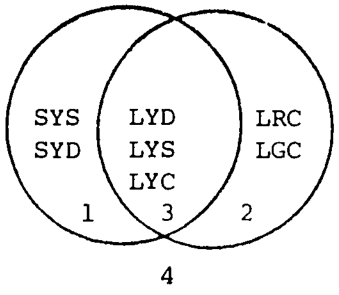
```
**L R CI
Put the LARGE RED CIRCLE in AREA 2
The expected information content of that block was .5
The actual information content was .5
```



```
**HELP
Card A can now be YELLOW, NOT RED or NOT CIRCLE.
Card B can now be LARGE.

**HINT
I think placing a NOT YELLOW thing
which is not CIRCLE, a BLUE OR GREEN
thing or a NOT RED CIRCLE would be a good idea.

**L G CIR
Put the LARGE GREEN CIRCLE in AREA 2
The expected information content of that block was .46
The actual information content was .29

Card A can't be negative color

**HELP
Card A can now be YELLOW or NOT CIRCLE.
Card B can now be LARGE.
```



```
**S Y DIA
Put the SMALL YELLOW DIAMOND in AREA 1
The expected information content of that block was 0.0
The actual information content was 0.0

**HINT
I think placing a NOT YELLOW thing which is not CIRCLE
or a YELLOW CIRCLE would be a good idea.

**L Y CIR
Put the LARGE YELLOW CIRCLE in AREA 3
The expected information content of that block was .5
The actual information content was .5

Card A must be positive color
Card A must be color
Card A must be positive
Card A can't be shape
Card A can't be negative
Card A can't be negative shape

**STOP
```



It must now be clear that the logical information being computed for the tutor is approaching information overload and if the tutor were simply to pass it on to the student or, for example, query him each time a monitor deduced some fact, the student would collapse into a hysterical twitching heap! The tutor is in need of some guiding principles for determining which of this information is important to a particular student at a given moment. But how can the tutor make any rational decisions along these lines without having at his disposal a model of what might currently be important to the student? Here we have a definite need for a structural model of the student. In particular, the model should make explicit which rules or generalizations the student already knows, which ones he clearly does not know and which ones he has used sometimes incorrectly.

There is a beautifully simple paradigm for using and constructing such a model. Associated with each monitor is a set of rules he could use to derive or achieve its particular goal (fact). Any time a monitor achieves its goals, it need inform the tutor not only of his success (i.e. Card A can't be any SHAPE) but must also inform the tutor of the way it deduced this fact.* (This might necessitate additional calls on the expert.) Then, the tutor

----

(*)For the moment let us assume this derivation is unique.

can either decide that the student has already shown mastery of these rules, or if in doubt, he can decide to query the student about the conclusion. If the student answers the question correctly then the tutor has evidence that he can successfully use these rules to derive the appropriate information. Whether the student answers the question correctly or not, the tutor has gained new information about which rules he knows and how he can combine them. (Of course, there may be higher-level considerations that dissuade the tutor from asking any question at the moment that a monitor discovers something.)

Production Rules:

The rules comprising the structure of this environment can be viewed as <u>production rule schemes</u> in the sense of Newell and Simon. Similar formulations are currently being exploited by many cognitive psychologists. As such, the blocks world provides a nearly open ended range of possibilities for examining how to induce production rule based, structural models of a student.

System Description

The Attribute Blocks system has been structured to allow experimentation with various tutorial and assistance modules. In this section we will describe the overall structure of the Blocks system, and the workings and responsibilities of the individual modules. Figure 1.2 shows the basic organization of processes and data within the Blocks system.

Executive

The executive has responsibility for the control flow within the system. The typical control path to process a student's move is as follows: The executive reads the input statement from the student and passes it to the natural language understander. The natural language understander identifies the intent of the statement and returns its semantic structure. The semantic structure contains the pertinent information for one of the environment maintainers (Unless, of course, the statement was directed to the tutor or one of the monitors.) The executive calls the proper maintenance routine which carries out the proper change to the environment. Then, before the student is told the result of his statement (e.g. where a particular block goes), the "pre-answer" tutor is called. At this point the tutor knows the student's move and what the result of that move will be. The tutor can, for example, query the student about what he had in mind by placing that block (while the student still has it in mind) or he can point out some things that the student should have known but didn't (because if he had, his present statement would have been different). After the tutor has finished, the executive calls the natural language generator to tell the student the result of his statement. Then the tutor is called again, this time to further explain the results of the statement or possibly the ramifications of it. This is also when the various monitors are invoked.

The executive also maintains the history list of student interactions. For each interaction, the student's statement, the result generated by the system, any advice or tutoring he was given and the context in which the statement occurred are saved on the history list. The history list is used by the tutor to find old possibilities lists and also to check on the types of tutoring it has given the student in the past.

Natural Language Understander

The classes of sentences required by the Attribute Blocks system to date have been fairly straightforward and are handled easily by a small semantic grammar based processor <Brown & Burton 75>. The Blocks grammar has about 15 semantic categories and involves very little complexity. However the flexibility allowed by the goal directed nature of the semantic grammar was particularly useful in the rules for recognizing descriptions of blocks. Without it, the understander would have been much harder to write.

In writing the parser for Blocks, the semantic grammar framework was extended by the addition of commands. Commands are one word directives to the system such as HELP, HINT, STOP, SHUT/UP, EXPERIMENTING, etc. A facility was added to the parsing mechanism to allow words to be defined as commands and then recognized in a bottom-up manner which short-circuits the regular top-down parsing scheme. This facility has allowed new commands to be added quickly and easily.

Environment maintenance

The Blocks system is built around an environment of a student playing with attribute blocks. This environment consists of the values of the cards, the locations of blocks which have been placed, and the possible theories for the cards which are consistent with the placed blocks.* The tasks involved in the Blocks laboratory are performed by procedural specialists called environment maintainers.

(*)The list of possibilities can be recalculated from the blocks but was deemed important enough to make it part of the environment.

These tasks include placing and removing blocks (including determining where blocks should go) printing the present board configuration, setting up the cards to begin a game and stopping the session. The effect of each of these maintenance actions is to change some portion of the data base which is examined by the other modules.

Monitors

To study the effects of various types of services which the Attribute Blocks laboratory could provide, several different types of monitors were designed and implemented. Protocol 3 presents the same example that was presented in Protocol 1 but with different monitors in effect.

Remaining possibilities monitor

In order to allow the student to see the effect that placing certain blocks had on the theory sets for each of the cards, a monitor was written which calculates all of the possible values for each of the cards from a configuration of blocks. Using this monitor, together with the ability to remove blocks, the student can discover how certain blocks (Questions) will split a set of possible theories. This monitor can also wake up the tutoring routines when worthless blocks are placed or when a set of possible theories is reduced to one element.

Information Gain monitor

The Attribute blocks world is an excellent domain to study problems of making decisions such as what makes a good question. The expected information gain of a block and the actual information gained provide a valuable metric for evaluating alternative questions. The expected information gain of a block is the sum over the four areas of the amount of information gained by that block falling in that area, times the probability of it falling there. The amount of information gained from a block falling in an area is the logarithm of the percentage of possible theories that block eliminated (in the cross product of theories for card A and theories for card B). The logarithm is taken base 4 since each question has four possible answers (each block could go in one of four possible areas). Since the beginning theory space has 324 members (18x18), the expected number of questions required to isolate one individual element is LOG 324 (base 4) or about 4.2. When the total actual information gain of the student's blocks totals 4.2, he can deduce both cards. By seeing the expected and actual information gain, the student can begin to develop intuitions about "good" questions.

Event monitors

Event monitors provide a means of monitoring the remaining list of possibilities for the occurrence of a particular event. An event occurs when a generalized class of values must be the case or can't be the case. For protocol 3 there were classes monitored for size, shape, color, positive shape, positive color, negative shape, negative color, positives and negatives. A class is defined as a list of theory values. For example the positive shape class is (TRIANGLE CIRCLE DIAMOND SQUARE). Each time the student places a block, the new possibilities list is checked for a change of status with respect to each of the classes. There are two changes of status which are considered worthy of note. One is when the possibilities list no longer intersects class. In this case, that class has just become impossible. The other important change of status occurs when the possibilities list becomes a subset of one of the classes. In this case, that class has just become required. In protocol 3 the monitors merely evoke printing functions which herald the event to the student. In a more fully developed system, these monitors would invoke the tutoring component as is currently done when a worthless block is placed. The tutor would then have the option of exploring the reasons for the event with the student.

Pre-answer tutor

The "pre-answer tutor" springs into action when either (1) the student has placed a worthless block or (2) the student has placed a block when he should have been able to deduce a card but didn't. The tutor attempts through a series of pre-stored questions to direct the student's attention to aspects of the situation that he may have missed. Protocol 2 shows several examples of this tutor intervening in a student's session. At present the tutor has three strategies, (1) If the student fails to deduce a card, try to get him to say what he thinks it could be and show him by counter example where he is wrong. (2) If the student places a worthless block, try to get him to predict where it will go and convince him that is the only place it could go. (3) If the student places a worthless block, get him to verbalize some remaining theories and choose a block which would distinguish between them.

From experiments, we have found this tutoring to be valuable although at present much too oppressive! When to tutor and when not to is a very difficult problem whose solution will require a structural model of the student. The following chapter is devoted to precisely this issue.

CHAPTER 2
AN INTELLIGENT TUTORING AND STUDENT MODELLING SYSTEM

In this chapter we describe a paradigmatic CAI system that was built to investigate the problems of 1) developing a representation for a logically adequate model of a learner, 2) constructing actual models of learners in learning situations, and 3) constructing a tutor which could use the constructed models to provide (on its own) well-timed, insightful comments. The interactions between these three problems dictate that they be attacked simultaneously. The logical adequacy of a student model cannot be investigated without specifying <u>how</u> the model is to be used by the tutoring system. Likewise, there is no point in inventing a representation for a model which is structurally more complicated than one which the system can automatically induce.

In classical CAI, the tutoring behavior is locally controlled by a predetermined instructional branching sequence which, at best, references a coarse model of the student. This differs substantially from our view, in which the tutoring module has complete freedom to interrupt the student at any time and must use its knowledge of the domain together with the synthesized model to decide <u>what</u> to say and <u>when</u> to say it. The viability of this approach depends critically on how well the model represents the student's reasoning strategies and current state of knowledge. If the tutor is to deviate from a predetermined instructional sequence, its new course of action must be based on its reasoning capabilities and the minute details of a student's strengths and weaknesses!

In order to gain some leverage on building a system that could actually construct and use a model of the student, we chose a domain in which we could easily construct an expert program that the tutor could call on for evaluating the student's behavior. The domain of knowledge chosen was the PLATO game "How the West Was Won." A provocative doctoral thesis by Cecily Resnick <Resnick 75> describes some preliminary experiments which question the effectiveness of this game as a learning environment. Taking her thesis as our starting point, we attempted to transform this arithmetic game into a highly productive learning environment by adding a student modeller and an intelligent, sensitive tutoring program. The tutor's comments were to be sufficiently insightful and well-timed that students continue to view the game as exciting and fun but at the same time actually learn something! The problem in building this system was to make the tutor neither too vocal -- so that it would neither be constantly babbling at the mouth, destroying the appeal of the game, nor so reticent that little would actually be learned. We felt that this could be achieved by bringing powerful information processing techniques to bear, thereby opening up an exciting new domain of CAI.

Description of "How the West Was Won"*

This game is played with two opponents (the computer usually being one of the opponents), on a game board like that in Figure 2.1. The object of the game is to get to the last town on the map (position 70). On each turn a player gets 3 spinners (random numbers). He can combine the values of the spinners using any 2 (different) arithmetic operators (+, -, * or /). The value of the arithmetic expression he makes is the number of spaces he gets to move. (He must also specify the answer.) If he makes a negative number, he moves backwards.

Along the way there are shortcuts and towns. If a player lands on a shortcut, he advances to the other end (e.g. from 5 to 13 in Figure 2.1). If he lands on a town, he goes on to the next town. When a player lands on the same place as his opponent, unless he is in a town, his opponent goes back two towns. To win, a player must land exactly on the last town. Both players get the same number of turns, so ties are possible.

Why Tutor at All?

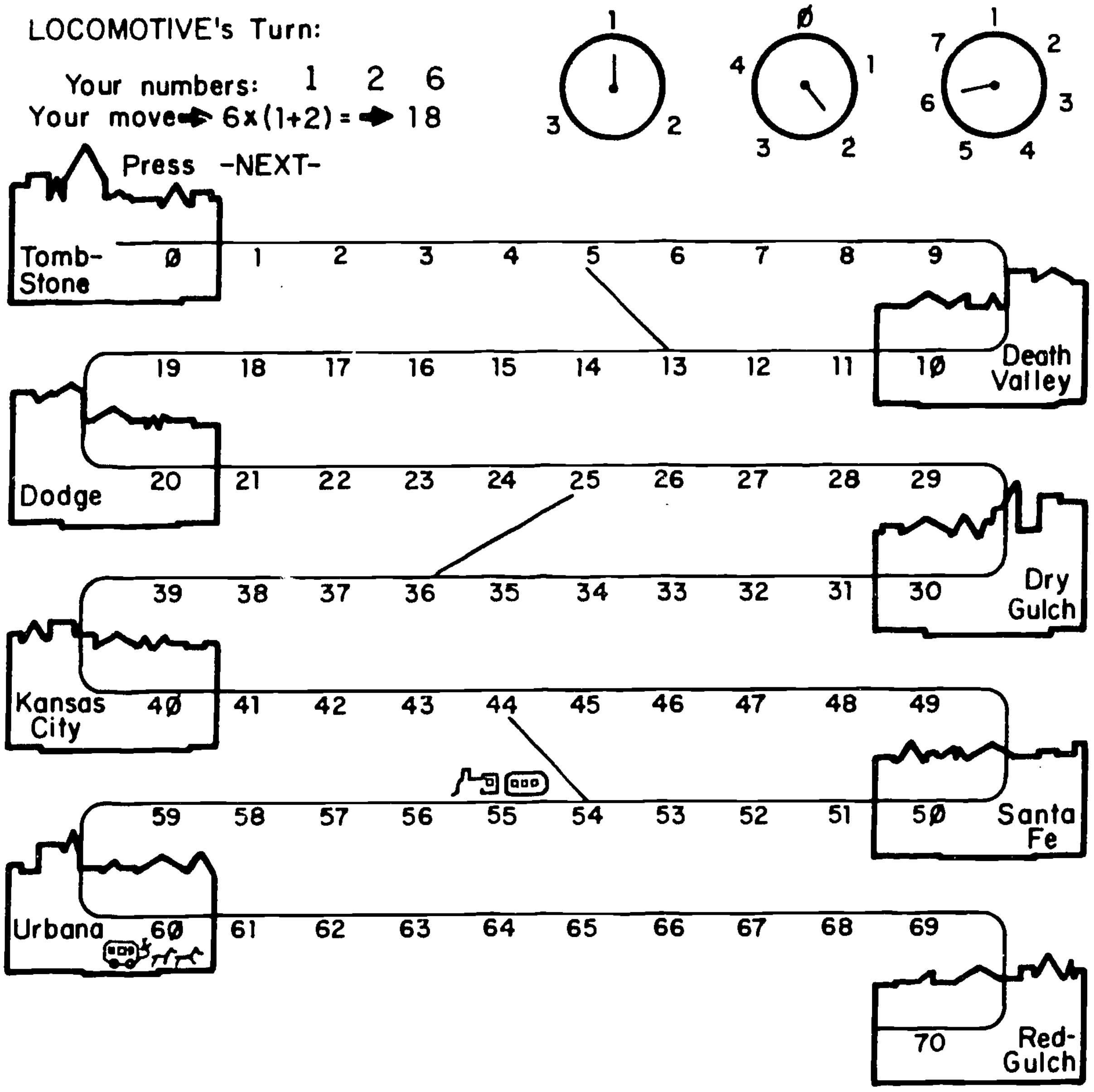
A central premise of complex, knowledge-based CAI is that good tutoring can point out structure in an environment which might have otherwise been missed, and by so doing allows the student to enrich his understanding of (and skills in) the environment. In West, an untutored (unwatched) student may tend to become fixed on a subset of the available moves and hence miss the potential richness of the game. For example, a student may adopt the strategy of adding the first two spinners and multiplying the result by the third spinner, (A+B)*C. Since the third spinner tends to be the largest, this strategy is close to the strategy of multiplying the largest number by the sum of the other two numbers (which produces the largest possible number). If this strategy is augmented by a rule that prevents moving off the board (i.e. a simple end game strategy) it generates a respectable game. Notice, however, that much is missed. The student is unaware of the special moves such as bumps and therefore of such questions as, "Is it better to send my opponent back 14 or get 9 ahead of him?" In fact,

---

(*)This game was written by Bonnie Anderson for the PLATO Elementary Mathematics Project.

FIGURE 2.1

Game Board for WEST (from PLATO terminal)


LOCOMOTIVE's Turn:

Your numbers: 1 2 6

Your move ➡ 6x(1+2) = ➡ 18

Press -NEXT-

35

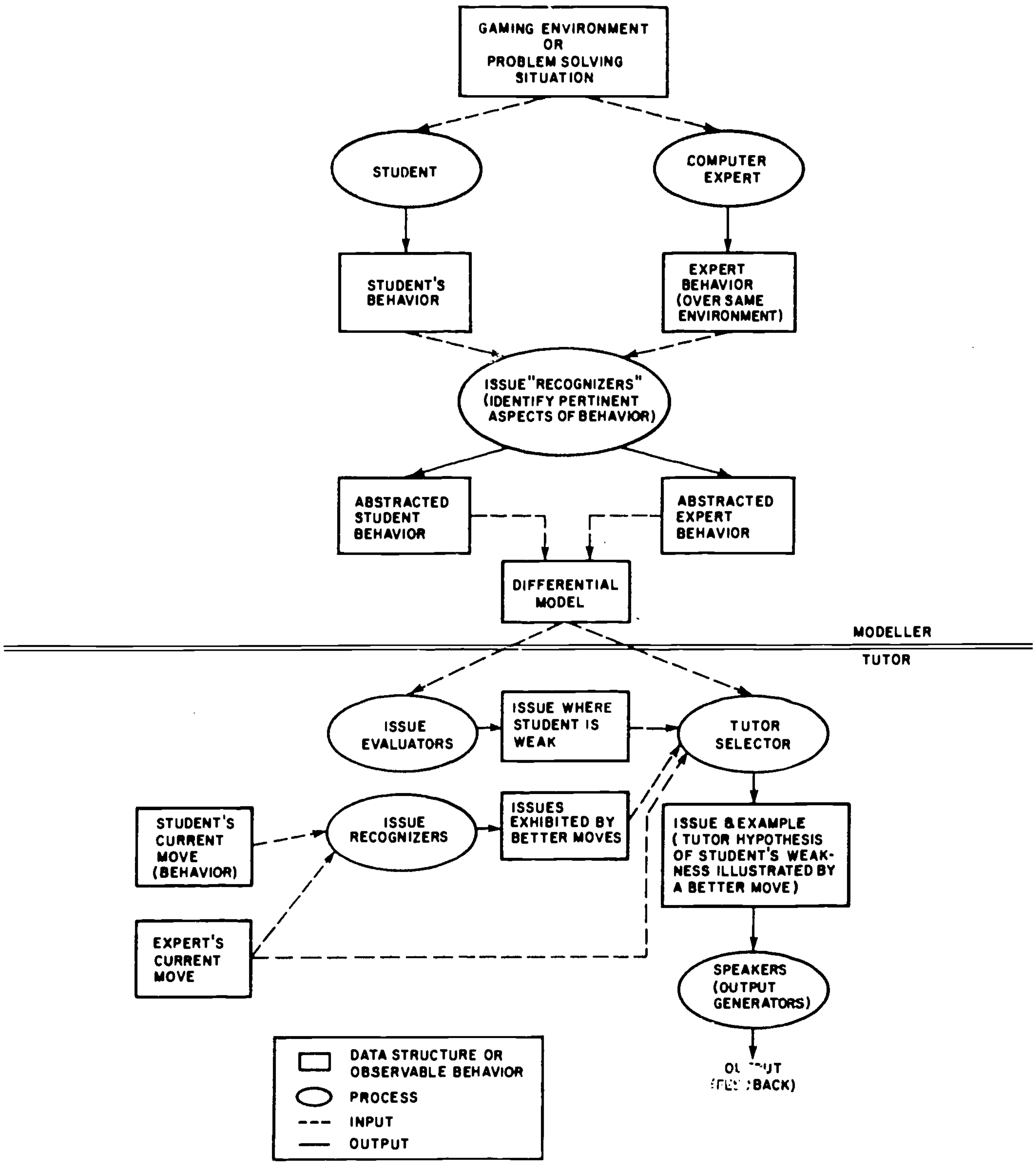
since the student generates only one move, he misses the whole notion of strategies for deciding between alternative moves. From an arithmetic point of view, he is performing one calculation per move instead of the dozens which he would have to perform to answer questions such as, "What numbers can I form with these spinners?" or "Can I make an 8 with these spinners?" By interjecting comments and suggesting better moves to the student from time to time (not too often), the tutor tries to widen the student's view of the game.

Tutoring by Issue and Example -- a General Paradigm

The paradigm of "issues and examples" was developed to provide the tutoring system with the means to focus on relevant portions of the student's behavior. The important aspects (skills) of the domain (i.e. what the student is expected to know or learn) are identified as a collection of "issues". The issues determine what parts of the student's behavior are being monitored by the tutor. The issues are implemented as procedural specialists which watch the student's behavior for evidence that the student uses or does not use their particular concept or skill. As the student plays, a model of how he is performing, with respect to each issue, is constructed. When he makes a "bad" move a tutorial component uses the model to decide why the student did not make a better move, that is, which issue he missed. Once an issue has been determined, the tutor might decide to present an explanation of that issue together with a better move which illustrates the issue. In this way, the student can see the usefulness of the "issue" at a time when he will be most receptive to the idea presented -- immediately after he has thought about the problem.

Figure 2.2 presents a diagram of the modelling/tutorial process underlying the paradigm. The first major component of the process is the construction of a model of the student's behavior. The model is constructed from an environment in which the student is solving a problem (in this case, a move in a game). Within this environment the student exhibits a certain behavior (such as making a move). The important aspects of this behavior (the issues) are abstracted into the model by the issue "recognizers". This abstracting is also done with respect to the behavior of a computer-based "Expert" in the same environment. The two abstractions are compared to provide a <u>differential</u> <u>model</u> of the student's behavior, which indicates those issues on which the student is weak. Notice that without the Expert it is not possible to determine whether the student is weak in some area or whether the need for that skill has arisen infrequently in the student's experience.

Figure 2.2


GAMING ENVIRONMENT OR PROBLEM SOLVING SITUATION
STUDENT
COMPUTER EXPERT
STUDENT'S BEHAVIOR
EXPERT BEHAVIOR (OVER SAME ENVIRONMENT)
ISSUE "RECOGNIZERS" (IDENTIFY PERTINENT ASPECTS OF BEHAVIOR)
ABSTRACTED STUDENT BEHAVIOR
ABSTRACTED EXPERT BEHAVIOR
DIFFERENTIAL MODEL
MODELLER
TUTOR
ISSUE EVALUATORS
ISSUE WHERE STUDENT IS WEAK
TUTOR SELECTOR
STUDENT'S CURRENT MOVE (BEHAVIOR)
ISSUE RECOGNIZERS
ISSUES EXHIBITED BY BETTER MOVES
ISSUE & EXAMPLE (TUTOR HYPOTHESIS OF STUDENT'S WEAKNESS ILLUSTRATED BY A BETTER MOVE)
EXPERT'S CURRENT MOVE
SPEAKERS (OUTPUT GENERATORS)
OUTPUT (FEEDBACK)
DATA STRUCTURE OR OBSERVABLE BEHAVIOR
PROCESS
INPUT
OUTPUT

The second major component of Figure 2.2 is the "Tutor". When the student makes a less than optimal move (as determined by comparing his move with those of the Expert) the Tutor uses the issue "evaluators" which scan the student model to create a list of issues on which the student is weak. From the Expert's list of better moves, the tutor uses the "issue" recognizers to determine which issues are illustrated by better moves. From these two lists (the "weak" issues and the better move issues), the tutor selects an issue and a good move which illustrates it. The selected issue and example are then passed to the output generators which produce the feedback to the student.

We would like to stress two points in the above process. One is the necessity of the Expert, and the other is the importance of identifying the critical issues. The Expert provides a measure for evaluating the student's behavior in unpredicted situations. The issues define those structured or conceptual components of the environment which the student is expected to learn and they provide a handle to structure and direct the exploration of the environment.

Protocol

Before discussing the modelling/tutoring process in greater detail, we present in Figure 2.3 a protocol of a student playing WEST. The tutoring component has been accelerated to generate more feedback than normal. In particular, the normal paced tutor seldom hassles the student two moves in a row. In the protocol, all lines typed by the student begin with "=> ". The lines which are indented 5 spaces are information determined by the model or the tutor and are not seen by the student. Annotations are indented.

FIGURE 2.3

```
Welcome to "How the West Was Won".
What is your name?
=> Debbie
Do you want to go first?
=> Yes

It's YOUR turn.. Spinners are: 1 2 2
What arithmetic expression did YOU form?
=> 1*2+2
What number does that give you?
=> 4
```

38

```
        The numbers that could have been made were:
        -2 2 -1 -3 5 3 0 4 6 1
        "best" move is (2*2)+1.
        delta-ways: 13-1, 10-8, 6-1, 4-5, 3-2, 2-3, 1-3
        Move ranks 11 which is FAIR
```

The indented information above is generated by the EXPERT module on each turn. This information is not seen by the student but is used by the TUTOR, MODEL EVALUATOR and STUDENT MODELLER modules. It is included here to indicate the complexity of information that is used by the system to follow the student's behavior. The first line gives the range of numbers that could legally be made using the spinners. The second line gives the "best" move according to the metric of maximizing the difference between you and your opponent. (These two pieces of information are available to the student through the "hint" key if she wants it.) The third line (beginning "delta-ways") represents the legal move space. The first number in each pair gives the number of spaces ahead the student would be (i.e. delta). The second number is the number of different arithmetic patterns which could have been used to achieve that lead. In this case there is one move which would have put Debbie 13 ahead (making 5 and taking the shortcut), eight moves which would have put her 10 ahead (making 0 and getting advanced to the next town), one that would put her 6 ahead, etc. Her move (4 ahead) ranked 11 because there were 10 better ones. 1+(2*2) is considered the same as (2*2)+1 when determining the rankings. For the rest of the protocol only the Rank of the student's move will be shown.

```
Debbie is at 4 -- the COMPUTER is at 0

It's MY turn..  Spinners are: 1 2 3
My expression is:  (1*2)+3
Which gives ME 5
I took a SHORTCUT
the COMPUTER is at 13 -- Debbie is at 4

It's YOUR turn..  Spinners are: 2 1 2
What arithmetic expression did YOU form?
=> 2*1+2
What number does that give you?
=> 4
```

Debbie makes another less-than-optimal move but two moves is too early to start tutoring.

```
    Move ranks 9 which is FAIR
```

```
Debbie is at 8 -- the COMPUTER is at 13

It's MY turn.. Spinners are: 1 2 1
My expression is:  (1+1)*2
Which gives ME 4
the COMPUTER is at 17 -- Debbie is at 8

It's YOUR turn..  Spinners are: 1 1 5
What arithmetic expression did YOU form?
=> 1*1+5
What number does that give you?
=> 6
     Move ranks 3 which is GOOD
     The ISSUES are (PATTERN PARENS STRATEGY)
```

> These are the tutorable issues which have been determined by considering the student model together with the moves which are better. Debbie's first three moves have all had the same pattern, a*b+c which does not require parentheses; whereas the best move uses a different pattern which does require parentheses so the issues of PATTERN and PARENS are noticed. The issue of PATTERN was chosen because it is more specific than PARENS. The STRATEGY issue indicates that there is no coherent pattern arising for the student's moves, as will be discussed later.

```
It's beginning to seem like you stick to the same old
standard pattern in your move.  The really expert players
try a whole bunch of possibilities.  An example of a good
expression would be:  (1+1)*5, which is a pretty good move.
So you could have been at 18 with ME finishing up the turn
at 17.
```

> After the presentation of a better move, the student may be allowed to retake her turn. This usually makes the student more receptive to the advice. In this case Debbie decides not to put herself in front of her opponent.

```
Would you like to take your turn over?
=> No
Debbie is at 14 -- the COMPUTER is at 17

It's MY turn..  Spinners are: 3 3 2
My expression is:  (3+2)*3
Which gives ME 15
the COMPUTER is at 32 -- Debbie is at 14

It's YOUR turn..  Spinners are: 3 4 5
What arithmetic expression did YOU form?
=> 3+4*5
```

```
What number does that give you?
=> 23

      Move ranks 7 which is FAIR
      The ISSUES are (PARENS ORDER STRATEGY)
```

In this case the tutor picks on the issue of parentheses use. ORDER (using the spinner in a particular order) was not a issue last turn because none of the better moves required changing the order. (In a less condensed session, the tutor would not comment on two moves in a row.)

```
I notice that you don't use parentheses very often.  I think
you could win more often if you tried using parentheses.
For example, you could have made the expression:  (3+4)*5,
which is alright for an amateur!  So you could have been at
49 and I would only have been at 32.

Would you like to take your turn over?
=> Yes
```

In this case Debbie retakes her turn. The players are put in their original position and the spinners are the same. Retaking a turn allows the student to go thru the motions of the suggested move for themselves and makes it easier to remember.

```
Debbie is at 14 -- the COMPUTER is at 32
It's YOUR turn..  Spinners are: 3 4 5
What arithmetic expression did YOU form?
=> (3+4)*5
What number does that give you?
=> 35
Debbie is at 49 -- the COMPUTER is at 32

It's MY turn..  Spinners are: 2 2 1
My expression is:  2*(1-2)
Which gives ME -2
I landed in a town.
the COMPUTER is at 40 -- Debbie is at 49

It's YOUR turn..  Spinners are: 1 2 3
What arithmetic expression did YOU form?
=> 2+1*3
What number does that give you?
=> 5
     Move ranks 13 which is POOR
     The ISSUES are (DIVIDE MINUS STRATEGY)
```

Up to this point the only operations Debbie has used are addition and multiplication, so division and subtraction become issues because there are better moves which use them.

```
You shouldn't forget that it's alright to divide, so long as
the answer comes out evenly.  Especially when there is a
good move not too far away, you should consider division as
a way to make the right number.  This expression is a
typical example:  (1+2)/3, which would have resulted in a
TOWN!  Then you would have been at 60 while I would have
ended up at 40.

Debbie is at 54 -- the COMPUTER is at 40

It's MY turn..  Spinners are: 2 4 2
My expression is:  2+(4/2)
Which gives ME 4
I took a SHORTCUT
I bumped YOU
the COMPUTER is at 54 -- Debbie is at 40

It's YOUR turn..  Spinners are: 1 2 2
What arithmetic expression did YOU form?
=> 1+2*2
What number does that give you?
=> 5
     Move ranks 11 which is POOR
     The ISSUES are (BUMP SHORTCUT ORDER STRATEGY)

You don't seem to be bumping very much.  Bumps are hard to
get but are usually a good idea.  One good example would be
the expression:  (1*2)+2, which would give you a SHORTCUT
and a BUMP!!  So you could have been at 54 while I would
have ended up at 40.

Debbie is at 45 -- the COMPUTER is at 54

It's MY turn..  Spinners are: 1 2 1
My expression is:  (1+1)*2
Which gives ME 4
the COMPUTER is at 58 -- Debbie is at 45

It's YOUR turn..  Spinners are: 2 0 7
What arithmetic expression did YOU form?
```

The hint command (it is actually a single key) provides the student with three levels of help. First it tells the student what numbers are possible thereby allowing him to pick one for himself. If he wants more help, the system will pick one for him. Finally the system will tell him what expression would make that number. Some students use the first level of hint as a check that they have though of all

```
        the numbers.

 => Hint
 The numbers you should be able to make are -14 -9 -2 2
 5 9 -7 7 14 0 -5

 What arithmetic expression did YOU form?
 => Hint
 I think moving 5 would be a good idea.

 What arithmetic expression did YOU form?
 => Hint
 You can make 5 by the expression (0+7)-2

 What arithmetic expression did YOU form?
 => 2*0+7
 What number does that give you?
 => 14
```

The WEST system also contains a simple arithmetic expression diagnostician which looks for mixed up precedence.

```
 MULTIPLICATION is done before ADDITION so 2*0+7 is equal
 to (2*0)+7 not 2*(0+7).
 Would you like to change your expression?
 => Yes

 Spinners are: 2 0 7
 What arithmetic expression did YOU form?
 => 0+2*7
 What number does that give you?
 => 14
      Move ranks 3 which is GOOD
 Debbie is at 59 -- the COMPUTER is at 58

 It's MY turn.. Spinners are: 3 1 3
 My expression is: (1+3)*3
 Which gives ME 12
 the COMPUTER is at 70 -- Debbie is at 59
 I win.

 The COMPUTER has won 382, lost 26 and tied 98
 Debbie has won 0, lost 1 and tied 0

 Hope we can play again sometime.
```

43

Issues

As stated earlier, the issues define those aspects of the environment which are abstracted into the model. The issues currently modelled over the West domain are:

order of spinners	the spinners don't have to be used in any particular order.
parentheses:	the use of parentheses is allowed and frequently valuable.
backwards:	if the result of a move is negative the player moves backwards which can sometimes lead to a special move.
special moves:	trying for towns, bumps, shortcuts is part of a good strategy.
subtraction:	subtraction is legal and sometimes useful.
division:	division is legal and sometimes useful.
pattern:	the operations can be used in any order, i.e. more than a small number of move patterns should be used.
strategy:	a strategy for looking for moves should be used, and alternative moves should be considered.

Each issue has procedural information associated with it. For each issue there is a function (called the recognizer) which determines whether a move exhibits that issue. The recognizers are used by the modeller to update the model on each turn and by the tutor to determine 1) if any of the most recent turns exhibited that issue and 2) whether any of the good examples exhibit the issue. Each issue also has an evaluation specialist associated with it (the evaluator) which can look at a student model and determine whether the student is weak in that issue. In addition, each issue also has a output generation function (called the speaker) which explains the issue to the student i.e. "I notice that you seldom move backwards".

FIGURE 2.4

```
(RANKS ((1 . 3)
        (2 . 1)
        (4 . 1)
        (7 . 1
        (10 . 1)
        (11 . [illegible])
        (16 . 1))

QUALITIES   (OPTIMAL 3 GOOD 2 FAIR 4 POOR 0 ERROR 0)

PATTERNS
 (WFF1 (OPTIMAL 0 GOOD 0 FAIR 1 POOR 0 ERROR 0 MISSED/BEST/MOVE 2)
 (WFF2 (OPTIMAL 0 GOOD 0 FAIR 0 POOR 0 ERROR 0 MISSED/BEST/MOVE 4)
 (WFF3 (OPTIMAL 1 GOOD 0 FAIR 0 POOR 0 ERROR 0 MISSED/BEST/MOVE 4)
 (WFF4 (OPTIMAL 1 GOOD 2 FAIR 3 POOR 0 ERROR 0 MISSED/BEST/MOVE 2)
 (WFF5 (OPTIMAL 0 GOOD 0 FAIR 0 POOR 0 ERROR 0 MISSED/BEST/MOVE 0)
 (WFF6 (OPTIMAL 0 GOOD 0 FAIR 0 POOR 0 ERROR 0 MISSED/BEST/MOVE 0)
 (WFF7 (OPTIMAL 0 GOOD 0 FAIR 0 POOR 0 ERROR 0 MISSED/BEST/MOVE 0)
 (WFF8 (OPTIMAL 0 GOOD 0 FAIR 0 POOR 0 ERROR 0 MISSED/BEST/MOVE 3)
 (WFF9 (OPTIMAL 0 GOOD 0 FAIR 0 POOR 0 ERROR 0 MISSED/BEST/MOVE 1)
 (WFF10 (OPTIMAL 0 GOOD 0 FAIR 0 POOR 0 ERROR 0 MISSED/BEST/MOVE 0)
 (WFF11 (OPTIMAL 0 GOOD 0 FAIR 0 POOR 0 ERROR 0 MISSED/BEST/MOVE 0)
 (WFF12 (OPTIMAL 1 GOOD 0 FAIR 0 POOR 0 ERROR 0 MISSED/BEST/MOVE 2)
 (WFF13 (OPTIMAL 0 GOOD 0 FAIR 0 POOR 0 ERROR 0 MISSED/BEST/MOVE 0)
 (WFF14 (OPTIMAL 0 GOOD 0 FAIR 0 POOR 0 ERROR 0 MISSED/BEST/MOVE 0)
 (WFF15 (OPTIMAL 0 GOOD 0 FAIR 0 POOR 0 ERROR 0 MISSED/BEST/MOVE 2)
 (WFF16 (OPTIMAL 0 GOOD 0 FAIR 0 POOR 0 ERROR 0 MISSED/BEST/MOVE 0))

ORDERS  (ORIG (GOOD 4 POOR 4)
        REV (GOOD 0 POOR 0)
        LMS (GOOD 1 POOR 0)
        SML (GOOD 0 POOR 0)
        OTHER (GOOD 0 POOR 0))

PARENS   (NECESSARY 2 OTHER 0 NONE 7 ERRORS 0)

DIRECTIONS   (FORWARD (GOOD 5 POOR 4 WAS/BEST/MOVE 9)
              BACKWARD (GOOD 0 POOR 0 WAS/BEST/MOVE 0))

SPECIALS   (TOWN (TOOK 3 WAS/BEST/MOVE 5)
            BUMP (TOOK 0 WAS/BEST/MOVE 1)
            SHORTCUT (TOOK 0 WAS/BEST/MOVE 2))

STRATEGIES  (SPECIAL 3 MAXDELTA 3 MAXVAL 3 ENDGAME 1 MAXNUMB 1 OTHER 6)

ARITHMETIC/ERRORS 0

TOTAL/MOVES 9

GAMES/PLAYED (WON 0 LOST 1 TIED 1))
```

Model

The model maintained by the West system is a record of how the student has performed with respect to a particular set of issues. It is built incrementally after each move by the issue recognizers. Figure 2.4 shows Debbie's model at the end of the protocol. In this section we shall discuss Debbie's model in detail and explain how it is constructed.

The model is broken down into sub-parts which are maintained independently by the recognizers. The major parts of the model deal with the general quality of the move, the form of the move expression (pattern), the order of the spinners, the use of parentheses, the possible strategies, and the use of special moves.

Rank and Quality

Each move that the student makes is judged and given a ranking and a quality class. The general criterion for judging a move is how it compares to what the (mathematically optimal) expert would do in the same situation. This expert works by generating all the possible moves. Each of the moves is then simulated to find the ending positions of both the player and his opponent. For the legal moves, the difference between the player's final position and his opponent's final position (called the delta or difference) is calculated. For example if a player starts at 5 and his opponent at 25, the delta for a move of 5 is -5 since the player would finish his turn at 20 (after getting a town) while his opponent would remain at 25. The legal moves are ordered from largest delta to smallest delta. The rank of the student's move is its position on this list (1 being optimal).* The quality of a move is a further classification of the RANK as OPTIMAL, (Rank=1), GOOD (Rank=2-6), FAIR (Rank=7-20) or POOR (Rank>20).** Both the rank and quality of each move are saved in the model. This information is used by the tutor to determine the general "strength" of a player so that "weak" players are not criticized for making GOOD (as opposed to OPTIMAL) moves.

(*)If the same number could have been calculated several different ways, all of the possibilities excluding commutativity of addition and multiplication, are included on the list. This has the effect of ranking the moves according to the number of ways of getting a better move, e.g. even though 8 was the only number the student could have made that was better than the 5 he made, if there are six ways of making an 8, the student's move ranks 7th.

(**)The total number of legal moves can vary greatly but it is typically on the order of 35.

Patterns

The pattern recognizer deals with the form of a student's move expression. A move is classified into one of 16 possible patterns (WFFn) according to the operations used in the expression and the order in which they are performed. See Figure 2.5. For example, WFF1 corresponds to taking the difference of the sum of two numbers and the third number. The mapping between the 16 well-formed expression numbers and the specific operations and order of evaluation is given in Figure 2.5. Once the WFF number of a move has been ascertained, the appropriate pattern counter is incremented. The "goodness" of the move determines which quality class subfield to increment for that pattern. This provides a profile of the student's use of each pattern. Debbie's model (Figure 2.4) indicates that she favors WFF4 (X+Y*Z).

Figure 2.5

WFF Number	Form	Needs Parentheses
WFF1	(A+B)-C	No
WFF2	(A*B)/C	No
WFF3	(A+B)*C	Yes
WFF4	(A*B)+C	No
WFF5	(A+B)/C	Yes
WFF6	(A*B)/C	No
WFF7	A-(B+C)	Yes
WFF8	A/(B*C)	Yes
WFF9	A/(B+C)	Yes
WFF10	A-(B*C)	No
WFF11	A+(B/C)	No
WFF12	A*(B-C)	Yes
WFF13	A-(B/C)	No
WFF14	(A-B)/C	Yes
WFF15	A/(B-C)	Yes
WFF16	(A/B)-C	No

In addition, for those moves in which the student's move was not optimal, the MISSED/BEST/MOVE field for all of the patterns which would have given an optimal move are incremented. This information points out areas where a student may be weak and can also be used to avoid criticizing the student about issues which were never to his advantage to use.

The pattern section of the model is an example of several different issue evaluators using the same recognizer. For example the subtraction evaluator knows which of the patterns use subtraction and can thus find a profile for the student's use of subtraction. Similarly the division evaluator knows which patterns use division. The parenthesis evaluator can also use the MISSED/BEST/MOVE field of the pattern section to determine if the lack of parentheses has affected the student's performance.

Orders

The ORDERS field of the model keeps information about the order in which the spinners appear in the student's move. The order of a move is classified as one of the following: (1) ORIG the spinner in the expression occur in the same order as they were given; (2) REV, they occur in the reverse of the order given; (3) LMS, they occur in decreasing order; (4) SML, they occur in increasing order of (5) OTHER, they occur in some other order. Within each class, the moves are kept in two subfields, GOOD and POOR. This information is used to make sure the student realizes that he may change the order of the spinners. For example, Debbie's model shows that she used the original ordering in 8 out of 9 moves.

Parentheses

The PARENS part of the model records the student's use of parentheses. Each move is classified as having no parentheses (NONE), as having parentheses which were NECESSARY as in (1+3)*4 or as having unnecessary parentheses (OTHER). In addition, a count is kept of the number of parenthesis errors the student makes while trying to form her expression. From this information, the evaluator can tell if the student understands the purpose of, and feels comfortable using, parentheses.

Directions

The DIRECTIONS part of the model records directional (forwards or backwards) information about the student's moves. Each move is classified as forward or backward and the appropriate sub-field (GOOD or POOR) is incremented (depending on the quality of the move.) The optimal move is then classified as forward or backward and the WAS/BEST/MOVE subfield of the appropriate field is also incremented.* From this the evaluator can determine if moving backwards has ever been the best move to make. Notice that in the two games Debbie played, moving backwards was never the best move. If a situation which exhibits an issue refuses to come up, it would be feasible within the tutoring/modeling

----

(*)As is the case in several parts of the model, there may be several different "best" moves which would classify differently. In most cases we have opted for picking one of the best ones and using that under the combined assumptions that (1) it will average out and (2) that field in the model is not used for anything critical enough that the difference will be significant.

paradigm to have the tutor try to set up "interesting" situations. For example, the spinner values could be (temporarily) dynamically biased to increase their likelihood, or hypothetical cases proposed and discussed with the student ("What would you have done if...").

Specials

The SPECIALS section of the model monitors the student's performance with regard to special moves, i.e. towns, bumps and shortcuts. There is a field in the model for each type of special move which records both the number of times the student made that type of move, and the number of times this was in agreement with the optimal strategy. It's primary use is to ensure that the student is aware of each type of special move.

Strategies

The recognizer for the strategy issue keeps track of possible strategies that the student might be using. It does this by recording for each of the student's moves, the strategies under which that move is optimal. At present there are five strategies that are recognized; 1) SPECIAL, always try to make a town, bump or shortcut 2) MAXDELTA, always try to maximize the value of your position minus your opponent's, 3) MAXVAL, always try to get the farthest along; 4) ENDGAME, get to 70; 5) MAXNUMB, always try to make the largest number possible. These strategies are not exclusive and any particular move may be optimal under several different strategies. However, if the student is consistently using one of the strategies, it should begin to show up. Any move which is not optimal under any of the strategies is stored under the field OTHER. As a rough approximation, any time OTHER becomes greater than any of the strategies, the student is not playing any particular strategy.* Debbie made six non-strategic moves, but she made three which were optimal under all three of the strategies MAXDELTA, SPECIAL and MAXVAL. The fact that these were her last three moves would not show up in the model and is one reason why the model must be augmented with a history list.

(*)The picture is complicated by the other issues. As an extreme case, if the student doesn't use subtraction division or know about towns, bumps and shortcuts and doesn't know how to use parentheses, her strategy would probably show up as OTHER. This does not mean that the student does not have a coherent strategy just that her set of possible moves is very limited. For this reason strategy should be one of the last issues to be pressed (which emphasizes the need for ordering the issues.)

Other

In addition to the major sections previously discussed, other information is saved in the model. Such information includes the total number of moves made by the student, his won-loss record, and the number of arithmetic errors he has made.

History List

The history list in West contains a complete temporal record of what has occurred in the session. This includes for each move, the spinners, the expression entered by the student (both parsed and linear forms), the results of the move (bumps, towns, etc.) and the final position. In addition, a record is maintained of all the errors made by the student and the advice given by the tutor. At present the history list is used to check the recent moves made by the student. This prevents the tutor from "hassling" the student about an issue with respect to which he has performed satisfactorily in the last (say) three moves.

Another possible use for the history list deals with the problem of "changing the point of view" of the modeller. The modeller evaluates a move based on its comparison to an expert's move in the same situation. This expert must use some strategy to decide which move is best. (For example, is it better to get one farther or to be on a town?). Whatever strategy the expert uses, (it currently uses the maximum delta strategy), it may not be the same strategy employed by the student. When this is the case, the student's moves won't be evaluated correctly using the expert's strategy. Since the reason for tutoring the student is not necessarily to teach him our notion of strategy, but instead to see that he is aware of the range of issues, it might be beneficial to criticize the student within his own strategy. If we discover that the student is playing a coherent but different strategy (either by asking him or by noticing patterns in his model*) the modeller can try to re-model the student using the history list and an expert who plays under the student's strategy. If we are correct about the student's strategy, this new model should indicate a better player. (At this point, if we verbalize this strategy to the student, we can make him aware of it and hence willing to consider alternatives. This gives him a purpose to the arithmetic practice; i.e. a tool in studying strategy.)

(*)The types of patterns in the model might be a large number of moves which are optimal in a strategy, together with general strengths in other areas, i.e. when the student is making less than optimal moves which can't be explained by the issues.

The Tutor

In the previous sections we discussed the structure of the student model that is constructed. In this section we shall see how this model is used by the evaluators to determine when to tutor the student.

Patterns

The PATTERN evaluator checks the student model to see if the student is varying the form of his move. As mentioned earlier, the pattern recognizer classifies each move as one of 16 patterns depending on the operations and their order of operation. Thus if the student is always forming A+B*C, the WFF4 field of the Pattern section will have a large portion of the moves. Notice, however, that constant use of a single pattern does not necessarily indicate that the student is stuck. It may be the case that in these particular situations, the student made the best move. For this reason, the evaluator looks at the non-OPTIMAL subfields of each WFF to determine how often the student used a form when it was not the optimal thing to do. The actual algorithm it uses (which is subject to change) to determine if the student is stuck in a pattern is "Has the student used this pattern non-optimally more than 75% of the times that he has not used it optimally."

Orders

The ORDER evaluator checks to see if the student is trying to use the spinners in alternative orders. Again, the important factor here is how the student's behavior compares with the expert's, i.e. how many times has the student used the same order when he could have done better with a different one. Currently, the student is judged weak if he has used the original ordering on more FAIR or POOR moves than if he has used any other ordering. This also has the effect of quickly informing the student in case he is unaware that he is allowed to change the order of the spinners.

Parentheses

The PARENTHESES evaluators check to see if the student uses parentheses. For this issue, the student is judged weak if he has used parentheses less times than he has not used them. This is one area in which the evaluator should be extended. The pertinent question here is not "Does the student use parentheses", but "Does the student use parentheses when they are required". A more complicated evaluator could determine this from the MISSED/BEST/MOVE subfield of each WFF form in the PATTERN section of the model. The sum of those subfields for the WFF forms which require parentheses is the number of times the student has missed an optimal move which required parentheses.

Strategies

The Strategy evaluator checks to make sure the student is playing some type of strategy. The present system will only criticize the student when he appears to have no coherent strategy. That is, it will not criticize a student's strategy, only his lack of one. A student is judged to lack a strategy if he has made more moves which are not optimal under any one of the recognized strategies than moves which were optimal. A more extensive version of tne strategy evaluator should be able to deduce precisely what strategy the student is using. In addition to the first information about the five strategies that the recognizer has been explicitly monitoring, the Strategy evaluator can use the information in the PATTERN section to confirm more complicated strategies. Once the student's strategy is discovered, the comments by the tutor can contrast the new example with the student's present strategy. This would lead to much more pertinent and correctly individualized comments. In addition, if the system knew the student's strategy, this would feedback into the model building process by conditioning the recognizers' view of a move.

Directions

The direction evaluator checks to make sure the student is aware that moving backwards is both legal and sometimes beneficial. Since it is possible that a student who has played two or three games of West has never been in a situation which called for a backward move, knowing the expert's behavior is essential to avoid unfair criticism. Using the expert's behavior (the WAS/BEST/MOVE subfield), the student is judged weak only if he has moved backwards less than half the number of times it was optimal to do so.

Specials

One of the things which we discovered by conducting experiments with early versions of West was that students often overlooked towns, shortcuts and bumps when they played. Prior to this discovery, these moves were grouped together under the class SPECIALS. The process we went through to increase the complexity of the modelling/tutoring system in this area provides some idea of the ease with which new issues can be added. The recognizer, for what had previously been specials, was extended to keep records by the type of special move (for both the student's move and the expert's.) Similarly, the evaluator had to be changed to consider towns, bumps and shortcuts separately. But the same function which performs the decision (e.g. "Has the student done x at least half of the times it was optimal to do so?") could be used in all cases. Also the predicates which were used by the recognizers to determine if the student's move used a town (for example) had to be made available to the tutor to filter the list of possible examples (better moves). Finally three speakers had to be written to print appropriate comments to the student (e.g. "You shouldn't be afraid to bump me. I don't get mad," etc.) In all the whole conversion took about two hours (<u>within the already established framework</u>).

The Speakers

In the WEST system the speakers are very simple. Each has three or four possible phrases for each of three or four parts of an explanatory paragraph. This implementation has the advantages of being easy to build and providing a reasonable variety of comments. However, there are problems using speakers which are too simple. The main limitation is that a speaker is not aware of the context in which it must "talk" (i.e. player positions, moves, etc.) and must therefore be overly general or risk making inappropriate comments.

<u>Methodology and Experimental Data</u>

When we began designing this system we faced uncertainty about what should go into a student model and how to guide the tutor into making insightful comments at relevant, and only relevant, times in the game. Because of the total lack of any comprehensive theory for how to develop and use these models, the system was designed so that it could be easily and drastically modified. That way, we could run subjects on the system, observe the systems's behavior and the student's reactions, and eventually compare the system's behavior to that of human tutors (ourselves).

Several hundred hours of subjects have now been run using the system, culminating in an experiment with 18 Boston University summer school students (from the School of Education). On a day by day basis, over the two months of our initial experiments we constantly changed and expanded our system, taking into account the flaws that were manifested by the results of the prior day's experiments. We also did some fine tuning of the strategies of the tutor to make it coincide more with our own intuitions. These initial experiments included a substantial variety of subjects ranging from ten-year-olds (mostly from the Montessori School), professionals and subjects closely resembling DOD trainees.

Initially, the system would only print out the student model at the end of each experimental run. However, it quickly became clear that incremental changes in the model were nearly as important as the end product. Consequently the system now "dumps" a complete model of the student every four moves. In addition, the system dumps most of the information computed by the "expert" along with whatever tutorial comments were made. This expert-produced information provides a substantial tool for helping us evaluate the tutor.

During the first week in July, after we decided that our system had reached a fairly competent stage of tutoring, we ran the 18 student teachers (from Boston University). After they finished using the system we asked them to fill out a questionnaire. In addition we held a one hour discussion in which the students discussed their reactions to our system.

Questionnaire

Of the 18 people involved in the experiment, 12 filled out and returned the questionnaire. The following comments apply to this sample.

All but one subject received advice from the tutor. The general feeling about the Tutor was quite favorable. Nine subjects stated that the Tutor's comments were appropriate to what they were doing. Of the two who disagreed, one said that the Tutor was offering a strategy which he didn't feel he should follow because it would leave him "vulnerable to attack". Eight out of 10 students found the comments helpful in learning a better way to play the game and nine out of 10 felt that the Tutor manifested a good understanding of their weaknesses! One subject commented "I misunderstood a rule; the computer picked it up in the 2nd game."

We are quite encouraged by these results. Not only did the subjects sense the "intelligence" of the Tutor in knowing when to offer appropriate suggestions, but from the discussion that followed they seemed to enjoy the Tutor's support.

Two of the questions asked them to verbalize rules that they were using. The first asked if they could state the rule that, given any 3 numbers (other than 0), gives them the mathematical expression which evaluates to the largest number possible. Seven subjects stated the rule fairly precisely (sum of the two smallest numbers times the largest). The others gave incomplete descriptions like "Add, then multiply". and "Sum of 2 numbers times the 3rd number." One subject could only express the rule using an example: (2+2)*4. From the discussion that followed, it appeared most of the subjects could perform this biggest number operation, but had difficulty stating exactly what they were doing. This further confirms our belief than an intelligent monitor who can <u>deduce</u> the rules under which a student is operating is superior to one which requires the <u>student</u> to verbalize what he is doing.

In the second question we asked what strategy they used when they played WEST. Amazingly, only three subjects were able to give completely coherent descriptions. Most seemed to be suggesting the following: "I would usually maximize my move, but would always try for a town first. I guess I had a hierarchy of moves. Distance - If I was behind, or far enough ahead of the computer. Towns - if I could get to it and jump a town. Shortcuts - if applicable. It really depended on my position at the time. Safe towns looked mighty good. I always would bump back the computer if possible." One subject said she used the strategy "Tried to get to 10's to go from town to town," although her actual play did not mimic this verbalized strategy.

No one had played WEST on PLATO before. In fact no one had played WEST at all before, so comparisons with other systems couldn't be made. 5 out of 12 felt that the computer sometimes cheats, a result Resnick found in her studies of children playing WEST. This is especially apparent when the student is losing. The student feels that the computer is not choosing its spins randomly, but is dealing itself bigger numbers. In fact, our version of WEST is set up so that the Expert biases its moves when the student is falling far behind. It limits itself to, at most, 3 on the last spinner when it's ahead.

Because most of the subjects were elementary school teachers, we asked them on the questionnaire and in the discussion afterwards, for comments or suggestions about the tutoring aspects of this game for their own students. The response was enthusiastic. The group felt that their students would love playing WEST: "Even high school students would love the game." It is interesting to note that this was in response to our version of WEST that includes tutoring (which all but one received). We had wondered whether the tutoring aspects would make WEST lose its game appeal. Our fears seemed to be unfounded.

One subject felt that "some students would lose sight of the math and get caught up in one response set." He thought "the Tutor can eliminate this fault if it can detect (and it apparently does) when the same mode is repeated."

An interesting aside about adult subjects who are unfamiliar with computers concerns how free they feel to explore the terminal: We labelled one of the keys HINT. We did not tell them that they could press it or what would happen if they did. Only 2 people pressed it even once. It's clear to us that making use of all the facilities of a system, at least with adults, does not come without explicitly instruction.

Conclusions

The overall sense we had from building and experimenting with this system is that it is so easy to talk about student models and yet it is so fantastically difficult to actually construct a system that can grow an insightful model of the student and then use this model in a sensitive way to tutor the student. The pedagogical value of drawing tutorial examples from the student's work seems beyond reproach, yet the intelligence the system must have to successfully act on its own is considerable. Constructing a tutor who is constantly criticizing is relatively straightforward. The point is to make one that only interrupts when a skilled human tutor would and then generates a succinct remedial comment.

We feel that our WEST system provides the beginning of a theory of how this can be accomplished. It also provides a glimpse of technical issues which must be confronted in actually constructing an operational system that can grow and use student models in a provocative way.

SECTION II - Component for Complex Intelligent CAI System

CHAPTER 3
STRUCTURAL KNOWLEDGE IN TROUBLESHOOTING ELECTRONIC CIRCUITS*

Introduction

This chapter is an investigation into the structural knowledge required to troubleshoot electronic circuits. Of particular interest will be the ability for a system to utilize this knowledge to discuss troubleshooting as it takes place. This ability would include debugging a circuit and giving a reason why each measurement was taken or just giving comments on other suggested measurements. Such a system could function as the basis for modelling the knowledge of an expert troubleshooter, for construc'ing a structural model of a student's reasoning process and state of knowledge in electronics, (i.e. by restricting the techniques of the expert until his "simulated" behavior coincides with the student's), and for having a tutor converse with a student

Natural vs. Unnatural Inference Schemes

There are many approaches to building a computer based troubleshooting system varying from troubleshooting by synthesis to formal theorem proving. Some of these methods are convenient for enabling the computer to generate explanations of its decisions, but with others it is almost impossible to generate explanations. An example of a successful troubleshooting strategy which cannot generate very good explanations is troubleshooting by synthesis. A circuit simulator is used by faulting components to see what the measurements would be like, and then comparing these values to the observations made in the circuit with the unknown fault. This generates a list of possible faults implied by the measurements taken so far and suggests future measurements. The problem with this strategy, as with many others, is that conclusions for troubleshooting are reached using unnatural inferencing rules, (i.e. such as in SOPHIE). Unless the inferences made by the troubleshooting system are reasonably understandable by humans it is not useful for purposes of explanation or for modelling a student! Unfortunately this constraint makes the troubleshooter-modelling (program) very complicated.

The position this chapter takes is to study <u>how people reason</u> in troubleshooting and then to formalize our findings by constructing a system to produce troubleshooting

(*)This chapter is based on a paper presented by Johan Dekleer at the Canadian Society for Computational Studies of Intelligence Workshop.

explanations, models of experts and models of particular students. This research also provides a formal representation of the knowledge and strategies actually used to qualitatively understand electronics and to perform troubleshooting. As such, it might also serve as a formal methodology for determining the kinds of strategies and knowledge that should be taught in intermediate level training courses in electronics.

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 chapter, 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 chapter these two approaches require radically different troubleshooting strategies.

Towards a Structural Theory of Troubleshooting

The way to obtain new information about the 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 as it should, or if 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 faultedness 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 provides information about the assumptions made in the propagation.

Coincidences between propagated values and values derived from knowing how the circuit should work requires a teleolcgical description of the circuit. As indicated earlier, this chapter does not investigate these latter kinds of assumptions. Instead, this chapter 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 non teleological ways. So every theory of troubleshooting must include knowledge about local and nonteleological deductions.

It may appear that in fact this local kind of circuit reasoning is essentially trivial and thus should not be investigated. This chapter 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 particular 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 chapter is directed into determining what other kinds of knowledge and interaction is 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.

The only constraint we will impose on the proposed theory is that the explanations for propagations and deductions it makes about the faultedness of components be easily understandable. To achieve this there are two options. The first is to ignore the explanation problem, attack the the troubleshooting problem in any possible way and then approach the problem of explanation separately. The second approach is to design the inferencing schemes the troubleshooter uses to be very close to that which humans use or could understand, thus eliminating the explanation problem. The former approach is tempting because there exist complete troubleshooting strategies which cannot generate explanation. An example of such a troubleshooting scheme is troubleshooting by synthesis: a circuit simulator is used to search for all possible faults that explain the current measurements, then a similar search is made to identify all useful measurements. Unfortunately, the process is extremely time consuming and it is inherently not able to produce explanations of any kind. For these reasons, as well as interest in the study of reasoning processes, this investigation will take the latter approach.

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.

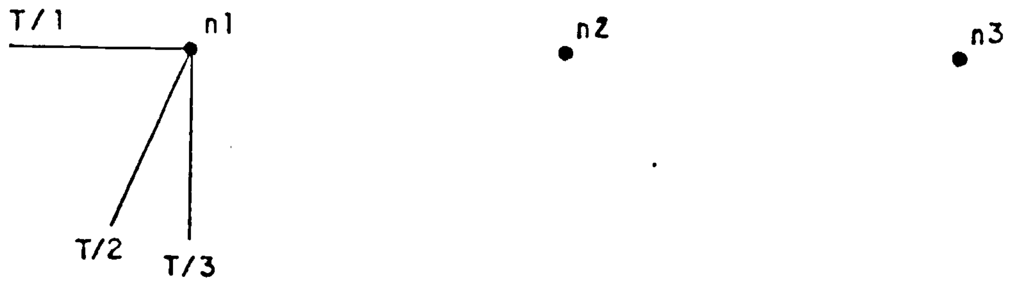
In order to give explanations for deductions, a record is kept as to which expert made the particular deduction. Most propagations make assumptions about the components involved in making it, and these are stored on a list along with the propagated value. Propagations are represented as:
(<type> <location> (<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.

62



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 is known, the last terminal current can be deduced.

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

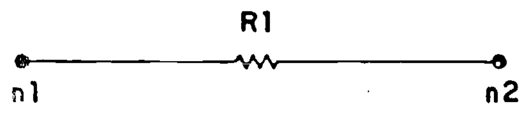
Since faults in circuit topology are not considered KIRCHOFFI makes no new assumptions about the circuit.

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 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 only consider 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 curren' 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 d' cussed 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 an added complication in 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))
```

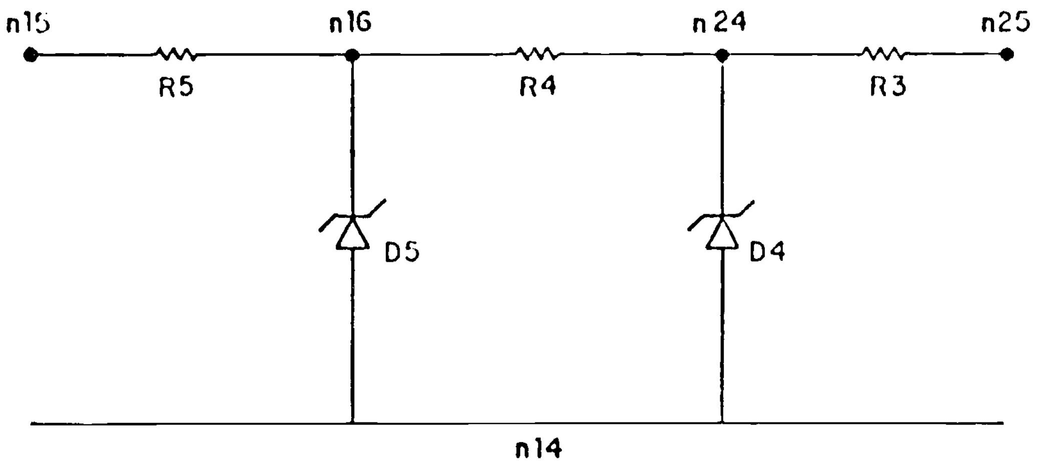
If the voltage across a zener diode is less than its breakdown voltage, the current through it must be zero.

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

Transistors are 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 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 whic 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 5 possibly faulted components, hence the best propagations to choose, are those with 2 or 3 assumptions. That means either measuring the current through R4, 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.

Proceeding in this scenario 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.

68

Basically, three kinds of problems arise. First, the handling of corroborations and conflicts leads to faulty assertions in certain situations and thus they should be examined much more closely. Second, 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 these cause many kinds of difficulties in the entire strategy. (In the remainder of this chapter 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 c mponent which a derivation depends on is in the assumption list of that derivation, and so a conflict thus 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:

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

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. Since 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, however, that in a conflict situation the predicate is violated, and thus the possibility of V(D) being incorrect 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 only used 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 <u>value</u>. (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 as 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.

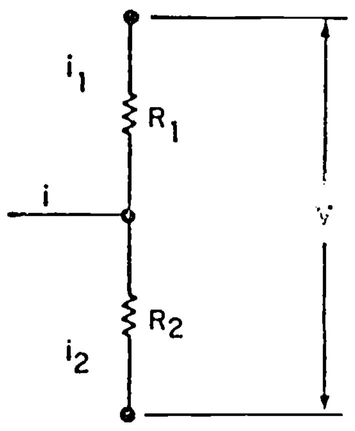
Kirchoff's current law can apply to collections of components and nodes, not just single components and nodes. Recognizing relevant functional blocks in the topology of the circuit is a tedious (yet performable) task. Circuit diagrams usually present a visual organization so that such functional blocks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

71



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.

```
 = 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 to which it was connected.

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 been 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 then, is that coincidences need not be between propagated values and measured values, but can also be between two propagated values.

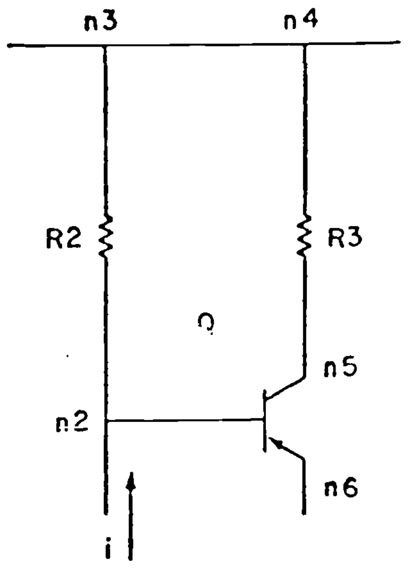
Conflicts and corroborations between propagated values must then be considered. If one of the propagations has no unverified assumptions, the coincidence can be handled as if it were between a propagated value and an actual measurement. However, if both propagations have unverified assumptions, the coincidence 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 has 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 at the voltage at N3-N4 and the current through the base of the transistor.

```
(VOLTAGE (N2 NO))
(VOLTAGE (N5 NO))
(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 NO) (LOOP N3 N2 NO) (R2 Q))
(VOLTAGE (N4 NO) (LOOP N4 N5 NO) (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 c, a large error in some f(D) could go detected. 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 in values are roughly proportional to their magnitude, 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 zener was functioning correctly. That is because the propagation would not even have been possible if the voltage across the zener was 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 was used as an assumption, the situation can be treated as a conflict with the calculated rating.

Otherwise, if the component itself was 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. (i.e. all the currents in the transistor must shift so that Kirchoff's law is violated, or, a more trivial case (which should be handled differently but nevertheless serves as a good example) 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 value 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 method different from computing b should be used. A simple scheme can be derived, which has difficulties only 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 b can be zero, such a scheme could be used at least to 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 explicitly known are considered, the information provided by coincidences between solely propagated values cannot enter into consideration. Thus the basic simple paradigm of the troubleshooter is to make no hypothetical measurements and only those propagations with unverified assumptions. Unexpected information, such as that provided by coincidences between propagated values, is not considered.

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 already been 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 in this paper so far have been implemented in a program so that much of the discussions derive their observations from actual interactions with the program.

Observing the propagations the propagator makes, the most arresting observation is that it 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 of 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. More on that later.

The idea that every measurement must have a purpose points out the basic problem: our troubleshooter cannot make intelligen .easurements 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 only can intelligently troubleshoot in the last few (two or three) measurements that are made in .he circuit.

Clearly, many more measurements are [illegible] before this discovery and the troubleshooter [illegible] 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 dealing with structuring such a hierarchical and teleological description are being investigated <Brown & Sussman 74>.

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 in order to locate faulty components which 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 made within the 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.

When searching for reasons why a certain value is not teleologically what it should be, it is important to note that when 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.

Future Research

Although the discussion of the previous section sketches out the necessity for teleology, more basic research has to be done into the nature of teleology and more work has to be done on the local analysis to even use the little we do know about teleology.

Once a propagation has been made, it is currently impossible to remove it. This raises many problems. For example, if teleology told you that a voltage at a particular point was 10 volts and you measured it and it turned out to be 20 volts the strategy would be stuck. There is no ability to forget the original 10 volts and its propagations. In our simple single fault theory this is not fatal; a clear conflict occurred, and the single fault has been found -- teleology! Similar problems occur in purely local analysis. Suppose a zener-diode is off, then we could propagate a voltage less than its breakdown voltage across it. Unfortunately, if at some later time we would measure the actual voltage we would only be able to tell if the diodes breakdown voltage was perhaps too high. The original propagation could not be forgotten.

The necessity is for another assumption type which, when conflicted with, will merely cause another assumption to be chosen instead of concluding faultedness. This would help both the problem of interacting with teleology and of repropagating better values. Sussman & Stallman are using this idea for circuit analysis. Arbitrary assumptions are made about the the states of the devices in the circuit and when conflicts occur different states are chosen. This proceeds until a consistent assignment of states has been found. The current troubleshooting strategy could be extended this way. It is probably a poor idea to choose states arbitrarily, since that would be extremely time consuming. A state should be chosen only if some reason has been discovered for it. This strategy is workable for troubleshooting because many more values are known about the circuit than in circuit analysis where the point is to discover these values. In troubleshooting, extra faulted states should be modeled so that when a contradiction occurs components should be forced into different faulted states. This would be done by forgetting the VIC of the old state and assuming the VIC for the faulted state.

Forgetting a propagation is in general nontrivial. If a propagation is forgotten, all the propagations that ensued from it have to be forgotten also. Unfortunately, forgetting this propagation is simple compared to the other forgetting that has to be done. Namely, all inferences about the circuit, not just those directly about measurements, have to reconsidered. For example, coincidences and their implications on the faultedness of components have to be modified. Most difficult of all, deleted propagations may have blocked the further propagation of other values (for example in coincidences). These blocked propagations must now be allowed to proceed.

Forgetting will have to be implemented by storing the assumptions of every deduction in a data base so that whenever an assumption is deleted all deductions depending on that assumption be deleted also. The simple solution of a CONNIVER context mechanism is not useful in this applic ion because the context mechanism can only be used effectivel if one knows, a priori, what things might need to be forgotten and that a total ordering for these assumpt is known. Both of these conditions are impossible to meet.

With such an ability to forget, more versatile local propagations can be made, multiple faults can be handled and teleology h must eventually be added can begin to be handled.

CHAPTER 4
TOWARDS TEACHING MATHEMATICAL PROBLEM SOLVING

Introduction

There are two intertwined long term goals that motivated the research reported in this chapter. The first goal is to establish the theoretical foundations and a procedural knowledge base for an "intelligent" generative tutorial system for mathematical literacy*. The second is to gain insight into how to build better information processing models of tutors for this and other domains of knowledge.

Before proceeding with the technical discussion, we include below a hypothetical protocol of a student using the kind of tutorial system we are working toward. The protocol will provide a glimpse of the kinds of tutoring we expect our final system to be able to provide by using its own built-in reasoning strategies. At the outset we stress that constructing a machine to solve algebra problems is relatively straightforward, as it is to construct a system to check the validity of each step of a proposed solution. The challenge, however, is to exploit these techniques to enable an instructional system to make sense out of what a student is doing. That is, the system must be able to analyze his solution paths sufficiently deeply that detected errors can be succinctly and meaningful described, appropriate hints can be given, and automatic remediation and problem generation can be performed. These requirements reflect the subtleties of the kind of tutoring which is possible when the tutor can induce a model of how the student is reasoning, and what his underlying "bugs" or points of confusion are.

This and the next chapter will describe our efforts, thus far, in understanding, designing and implementing such a system.

Basic Assumptions Underlying this Research

One of our basic assumptions is that a major breakthrough in the effectiveness of CAI will be achieved by constructing a system that can follow all of the intermediate steps that a student is performing while solving a given exercise. By having sufficient built-in generative "intelligence", the system can construct an

---

(*) Mathematical literacy concerns the procedural skills of mathematics as opposed to its axiomatic theory, (e.g. the kind of knowledge needed to enable students to comfortably manipulate and understand the mathematical formulas encountered in basic courses in mechanics, navigation, flight dynamics, electronics.)

PROTOCOL

WELCOME TO THE COMPUTER BASED ALGEBRA WORKSHOP!
WHAT IS YOUR NAME?

>> max

AS I RECALL, WE WERE WORKING ON REDUCTIONS INVOLVING COMPLEX FRACTIONS. YOU WERE HAVING A BIT OF DIFFICULTY DECIDING WHEN TO MULTIPLY OUT A PRODUCT, OR FACTOR IT FURTHER. ARE YOU READY TO TRY A NEW PROBLEM?

>>yes

OK, THIS ONE IS A LITTLE HARDER THAN THE LAST ONE.

The system needs to employ a metric for the problem complexity which reflects, not only some absolute notions, but also a knowledge of which rules and heuristics the student has mastered.

PLEASE REDUCE:

$$\{A \div [(2A-3B)(2A \div 3B)]\} + 1/2.$$

YOU CAN WRITE ON THE TABLET NOW.

>> $A \times \frac{1}{[(2A-3B)\left(\frac{2A}{3B}\right)]}$

THAT'S A GOOD START. YOU HAVE CHANGED ONE OF THE DIVISIONS TO A MULTIPLICATION, AND THE OTHER TO FRACTION FORMAT. YOU ALSO GOT RID OF THE BRACES, WHICH WEREN'T REALLY NECESSARY. PLEASE CONTINUE (BUT DON'T FORGET THE 1/2)!

Occasionally the system may comment even on correct work, ensuring that the sequence of steps which were performed is made explicit. Notice that a system which constantly told the student, "You must also copy the '1/2'" would be very frustrating [illegible]

>> $\frac{A}{(2A-3b)\left(\frac{2a}{3b}\right)} + \frac{1}{2}$

Note that the system must be aware of common notational variations and ambiguities, such as lower/upper case for literals.

>> $\frac{(2a-3b)\times 2a}{3b} = \frac{A}{\left(\frac{4a^2-6ab}{3b}\right)} + \frac{1}{2}$

Recognising that only the "attended" parts of an expression have been written, while unchanging parts are not copied until later, can require considerable problem solving.

>> $\frac{A}{\frac{2a(a-3b)}{3b}} + \frac{1}{2}$

I AM AFRAID THAT'S NOT QUITE RIGHT, MAX. DO YOU SEE WHERE YOU HAVE MADE A SLIGHT MISTAKE?

Understanding an error means isolating its precise location in the expression (and when it occurred in the rule-path), categorizing it as one of a class of common error types, and relating it to the student model.

>> no

PERHAPS IF YOU LOOK AT THE LINE ABOVE. ON THAT LINE, YOU WERE MULTIPLYING OUT IN THE DENOMINATOR. NOW IN THIS LINE, YOU HAVE TRIED TO FACTOR IT AGAIN, RIGHT?

Remediation interacts with hint-generation when the students are helped discover their errors for themselves.

>> yes

Multiplying out [illegible] then re-factoring an expression is a rational form violation. [illegible] in the problem-solving strategies, rather than, or [illegible] addition to, particular operational errors.

BUT INSTEAD OF HAVING THE "2A" ON THE RIGHT, THIS TIME YOU BROUGHT IT OUT TO THE LEFT. DOES IT MATTER WHAT ORDER YOU DO THE MULTIPLYING?

This is the underlying principle which justifies the claim that the multiply-factor sequence was non-productive.

>> of course not !

BUT THEN?

>> i guess i left out a 2 in (2a-3b).

GOOD! LET'S CONTINUE WITH YOUR NEW PROBLEM, AS IF WE HADN'T NOTICED THE MISTAKE, AND SEE WHAT DEVELOPS.

The student must learn to recognize mistakes, by developing higher-order rules to help him check his work, and also to sense when something seems wrong. Developing these skills requires letting him sometimes proceed, noting that his error has simply redefined the problem. Since our system has some basic "understanding", it can simply solve this new problem to see if there is anything of potential interest in letting him proceed with it. By referring to a model of his current skills, the system can also tell if the student is apt to be able to handle the new problem in the first place.

WHAT WOULD YOU HAVE WANTED TO DO NEXT?

>> $$\frac{\overset{1}{\cancel{A}}}{\dfrac{2\cancel{A}(a-3b)}{3b}} + \frac{1}{2}$$

Here, he simply [illegible] the A's. The system must understand cancellation.

GO AHEAD FROM THERE.

>> $$\frac{3b}{2(a-3b)} + \frac{1}{2}$$

The system should [illegible] ease with which the complex fraction [illegible] modifications to the student model.

>> $$\frac{3b+1}{2(a-3b)}$$

While the system [illegible], it has led to an opportunity to [illegible], so nothing is said at first.

>> $$\frac{\cancel{3}b+1}{2(a-3b)} = \boxed{\frac{1}{a}}$$

The error has been manifested.

WELL, YOU CERTAINLY FINISHED THAT ONE OFF FAST! UNFORTUNATELY, YOUR WORK IS NOT QUITE RIGHT. THERE WAS A LITTLE ERROR UP ABOVE, BUT THIS LINE HAS A BIG MISTAKE, WHICH LOTS OF PEOPLE GET CONFUSED ABOUT. DO YOU SEE IT?

>> well, i wasn't sure if the denominator should be (a-0)=a, or (a-1).

Students commonly generate "pseudo-rules" to handle unusual (or forgotten) situations, which arise because of other errors.

YOU'RE ON THE RIGHT TRACK. WHAT IF THE PROBLEM HAD BEEN:

$\frac{3b}{3b} = ?$

'LD THE ANSWER HAVE BEEN $\frac{1}{1}=1$, OR $\frac{0}{0}$ = UNDEFINED?

>> it would be 1.

GOOD NOW SUPPOSE YOU WERE TO EVALUATE THE FIRST EXPRESSION IN THAT LAST LINE, BEFORE CANCELLING 3b, WHEN a=1 AND b=2. WHAT WOULD YOU GET?

The intrinsic justification for correctness of algebraic rules is part of their semantics. The literals in algebraic formulas represent numbers. The rules are expected to preserve equivalence over all assignments of numerical values to literals.

>> $\frac{3(2)+1}{2(1-3(2))} = \frac{7}{2(-5)} = \frac{-7}{10}$

WHAT ABOUT THE SECOND, $\frac{1}{a}$, WHEN a=1?

The system must exercise caution in choosing its numerical examples, to avoid confusing "accidental equalities."

>> $\frac{1}{1} = 1$

SO THE FIRST TERM IS NOT THE SAME AS THE SECOND, RIGHT? WOULD

$\frac{1}{a-1} \cdot \frac{1}{1-1} = \boxed{\frac{1}{0}}$ BE ANY BETTER?

Assignment of credit, blame for an error to an incorrect [illegible] can be avoided by pointing out the correct role of the [illegible] question.

>> no

SO WHAT DOES THAT IMPLY?

The student is encouraged to take an active role in finding the underlying difficulty.

>> it can't be either of them. i am beginning to think that something else
>> is wrong.

But the student can't be expected to do everything - if he could, he wouldn't be here.

THERE IS. YOU CAN'T CANCEL OUT THE "[illegible]"'s UNLESS THEY ARE COMMON FACTORS OF EVERY TERM IN THE NUMERATOR AND DENOMINATOR. SINCE THAT IS NOT THE CASE HERE, YOU HAVE CHANGED THE VALUE OF THE EXPRESSION.

Explaining the prerequisites for the rule. The student's "bug" is in the conditions part of the rule.

>> i see.

GOOD. NOW LET'S BACK UP A BIT. HERE IS WHAT YOU WROTE BEFORE:

$$\frac{3b+1}{2(a-3b)} = \boxed{\frac{1}{a}}$$

SINCE IT WAS ILLEGAL TO CANCEL THE "3b"'s, WAS THERE ANYTHING ELSE TO BE DONE?

Illustrating the relationship of the error to the particular context in which it occured.

>> i don't think so.

RIGHT. SO THE ANSWER WOULD BE $\frac{3b+1}{2(a-3b)}$. PERHAPS YOU FELT THAT IT WASN'T <u>SIMPLE</u> ENOUGH TO BE THE FINAL ANSWER. THIS MAY HAVE ENCOURAGED YOU TO RESORT TO "SHADY TACTICS"! BUT WHEN THE ANSWER ISN'T SIMPLE, IT MAY BE BECAUSE OF AN ERROR YOU MADE <u>BEFORE</u> -- OR IT MAY JUST BE THAT SOME PROBLEMS HAVE "MESSY" ANSWERS. WHICH DO YOU THINK IS THE CASE HERE?

And drawing a general lesson about problem solving heuristics.

>> well, since we changed the problem way back in the beginning, it <u>could</u>
>> have a messy answer.

THAT'S TRUE. WE REDEFINED THE PROBLEM. BUT THERE <u>IS</u> ANOTHER ERROR -- GOING FROM:

$$\frac{3b}{2(a-3b)} + \frac{1}{2}$$

TO: $\frac{3b+1}{2(a-3b)}$

SINCE YOU USUALLY DON'T MAKE THIS SORT OF ERROR, I'LL JUST TELL YOU WHAT IT IS: YOU FORGOT TO MULTIPLY THE NUMERATOR OF THE 1/2 BY THE CONFACTOR, (a-3b), WHEN YOU COMBINED THE TWO FRACTIONS. DO YOU SEE WHAT I AM SAYING:

"Telling" can be a valid teaching strategy too! Especially when the difficulty is not fundamental. But the description must be in terms that the student will understand!

>> yes

SO WHAT SHOULD COME AFTER $\frac{3b}{2(a-3b)} + \frac{1}{2}$?

>> $\frac{3b}{2(a-3b)} + \frac{1(a-3b)}{2(a-3b)}$

OK, GO ON...

$$\frac{3b+(a-3b)}{2(a-3b)}$$

$$\frac{3b+a-3b}{2(a-3b)} = \boxed{\frac{a}{2(a-3b)}}$$

THAT'S FINE.

accurate model of steps which the student performed (including implicit operations), and then identify and critique the <u>fundamental</u> errors that were committed, as opposed to simply evaluating whether or not the <u>answer</u> is correct.

It is also assumed that the effectiveness of a tutor can be greatly improved by constructing a <u>detailed</u> <u>model</u> of the current state of <u>each</u> student's entire set of problem solving procedures for the domain, and by taking that model into consideration when formulating tutorial comments appropriate to the student's problem solving behavior.

Another assumption is that the effectiveness of particular remedial strategies can be significantly increased by constructing and using a detailed explanation of the nature of the error which occurred. This means that each particular error should be seen by the tutor as an instance of a category of related <u>error</u> <u>types</u>, and the occurrence of each class of errors would be seen in turn as indicative of specific weaknesses in the student's understanding. Tutorial assistance is to be keyed to these specific weaknesses.

A final assumption, implicit in our goals, is that it is possible and sometimes desirable to instruct students in a procedural, "how to do it" type of knowledge of a domain, without belaboring the formal theory underlying that domain. More simply, we require that our system understand and explain algebra from the point of view, not of the abstract <u>logical</u> structure of the knowledge, but of a <u>reorganized</u>, <u>learner</u> <u>oriented</u> structuring of how he is to <u>use</u> the knowledge for solving algebra problems.

Implications of these assumptions

An immediate implication of our approach -- if it is to be useful as a practical educational technology -- is that the creation of a detailed representation of the path of intermediate steps which the student must have traced out, (given his previous state, his current input, and prior knowledge of the student and algebra in general), will need to be done efficiently. Furthermore, it must be computationally feasible to construct a thorough, <u>simulatable</u> model of the student's <u>overall</u> knowledge of algebraic procedures. This can be achieved by abstracting from <u>individual</u> problem solving protocols, in conjunction with prior expectations based upon the system's own procedures and an understanding of common error types.

Moreover, in order for the tutoring system to be widely applicable to students with different backgrounds, the <u>taxonomy of errors</u> must reflect <u>universal</u> kinds of misunderstandings, rather than artifacts of particular teaching styles or techniques.

The expert system for algebraic manipulation must represent and process algebraic entities in a natural, human-like way. At each step, the system must be capable of generating <u>explanations</u> for its actions, in terms of a <u>network of procedures</u>, which <u>embody</u> the pedagogically reorganized knowledge.

Synthetic Approach

Our approach to constructing such a system has taken two forms: synthetic and analytic. The synthetic approach concerns the design and partial implementation of a collection of modules (programmed in LISP) to analyze and critique a student's solution path. We have run a limited set of experiments on the operational portions of these modules. The purpose of these experiments is to predict the computational behavior of the modules with respect to different formulations of production rules <Newell 72> which encode chunks of algebraic <u>know-how</u>. In particular we have been concerned with understanding the combinatorics of the search space involved in filling in the missing steps in a student's proposed solution, and in characterizing exactly what is wrong with a student's answer. Such an understanding is crucial in determining the extent to which brute-force search strategies over the set of production rules must be supplanted by more sophisticated heuristic techniques.

As the student becomes more proficient, the combinatorial nature of the search space seems to warrant more powerful heuristics, since students perform an increasing number of algebraic manipulations "in their heads". We have already given a good deal of thought to surmounting these problems, and a few of our ideas seem promising. A distinction between strategies (higher order rules which determine the <u>order</u> of (basic) rule applications), and the set of (primitive and macro) operations which the student is known to be using, greatly constrains the search space. Without it, any sequence of operations would be a possibility. Furthermore, rules are hierarchically structured - one need not expand sub-rules until the top level solution <u>plan</u> has been discovered. This is similar to the use of MACROPS in the ABSTRIPS system <Sacerdoti 73>. Heuristic equivalence checking, based upon a hash-coding algorithm described by Martin <Martin 71>, further constrains the search by eliminating consideration of error operations.

Use of Production Rules

In retrospect, some of the design decisions which were made in our preliminary efforts seem somewhat unsatisfactory. For example, the organization of the system around relatively simple rewrite or production rules may have been unwise. A more general form of production rule, or "schema" now seems preferable. Such a schema might take a form such as:

[pattern + conditions => action => new pattern]

in which the right hand side can represent an entire class of expressions -- such as might be generated by one of the common error categories. Moreover, there should be a provision for higher order, or "meta-rules" for recognizing transformations. On the basis of introspection, such rules are frequently employed by skilled tutors. Examples might be something like the following*:

```
(a) IF MATCH <initial-expression> WITH
      ((!#1@SIGNP #2@NUMBERP #3@SIGNP #4@NUMBERP)
    AND MATCH <final-expression> WITH
      (!#5@SIGNP #6@NUMBERP),
    THEN conclude that RULE was
      "Addition-Of-Two-Signed-Numbers"
    with the following
    CAVEAT:  IF #6 is not in the appropriate relation  to
    #2 and #4, then propose that student error occured.

(b) IF MATCH <initial-expression> WITH
      (!#1@AXPRP '= !#2AXPRP)
    AND MATCH <final-expression> WITH
      (#3@LITATOMP '= !#$@AXPRP)
    THEN conclude that RULE was
      "Solve-For-Litatom" with the following
    CAVEAT:  IF the value of #1, when all occurrences  of
    #3   are   REPLACE-d  by  #4,  is  NOT  EQUAL  to  the
    corresponding  value  of  #2,  then  a  student  error
    occurred.  Suspect the sub-rules:
        (1) "transpose all terms involving #1 to left";
        (2) "transpose all terms not involving #1 to the
         right";
        (3) "add like terms on each side".
```

Routines should also be constructed which provide a structural description of the differences between two expressions. These might be based on such features as: different numbers of parentheses; different numbers of

---

(*)With all due apologies for the cryptic pattern-matcher syntax!

92

terms; different numbers of Lambda (bound) variables; presence of "=" signs; etc.

Analytic Approach:

The other approach (i.e. analytic) that we have been pursuing concerns a careful analysis of the knowledge to be embedded in the tutoring system. It also includes a specification of the kinds of logical tasks the system will have to perform in analyzing and critiquing a student's solution. These investigations have been empirically driven in the sense that we have run several experiments and performed (by hand) detailed protocol analyses (akin to the methodology of Newell & Simon) of the solution paths and errors generated by students.

In order to provide an intuitive grasp of the kinds of knowledge and reasoning involved in constructing a structural model of a student's behavior in solving an algebra problem, we have annotated an actual subject's solution path to one of our experimental problems. This indicates the kind of reasoning required to construct such models. We stress, again, that it is relatively straightforward to have a computer system generate a reasonable solution to this problem. The challenge is to adequately formalize the reasoning strategies and knowledge that tutors use in understanding what a student is (probably) doing, so that similar strategies can be employed by a computer program to enable it to figure out the student's reasoning processes and points of confusion.

The algebraic problem this subject has been asked to solve is to reduce an algebraic expression (Line L0 below) to a minimal form (i.e. to a single reduced fraction). Let us proceed to generate a hypothetical explanation of what a good tutor might be thinking while grading or understanding this subject's solutions.

[L0] $\{A \div [(2A - 3B)(2A + 3B)]\} + \frac{1}{2}$

[L1] [illegible] $+ \frac{1}{2}$

[L2] [illegible] $+ \frac{1}{2}$

[L3] [illegible] $+ \frac{1}{2}$

The analysis of L0 through L2 is straightforward, but the transition to L3 is puzzling. The inversion operation, which brings the "3B" into the numerator, suggests a moderate degree of sophistication. But the (very common) illegal cancellation indicates considerable confusion. There are some clues which help to refine the description, narrowing the range of contexts which would be likely to elicit this error. Note the subject did <u>not</u> perform, (2A-3B)(2A/3B), which is a very similar mistake. The difference appears to focus on whether the sum of terms occurs in the numerator or denominator.

[L4] $$\frac{1}{4A^2-2} + \frac{1}{2}$$

[L5] $$\frac{1}{2(2A-1)} + \frac{1}{2}$$

In going from L4 to L5, the exponent of "A" is lost. This could be attributed to either a copy error, followed by a legal factoring, or a factoring error. A flag should be set to watch for further occurrences of the latter, but there is not enough information <u>here</u> to be certain.

[L6] $$\frac{1}{2A-1} + 1 = 2A-1+(2A-1) = \boxed{\frac{4A-2}{2(2A+1)}}$$

L6 is truly remarkable. The first expression, (to the left of the first "=") was arrived at by "multiplying through" L5, by 2. This is an instance of the more general difficulty, operating on <u>expressions</u> as if they were <u>equations</u>. One wonders whether the "=" was written first (so that its presence triggered the error), or, whether the invocation of the (inappropriate) rule, (since its correct use occurs in equations), triggered the sequence of "=" signs on this line! The second expression, 2A-1+(2A-1), could be either a "multiplying through", but an abortive attempt to combine into a single fraction, using an LCD-like rule but forgetting the denominator. The boxed answer, 4A-2, must have seemed strange as the subject continues on to "factor" it, introducing yet another (sign) error.

[L7] $$\frac{1}{2(2A-1)} + \frac{1}{\ }$$

What can be made of L7? Clearly, L6 seemed incorrect, and the subject returned to L5. (Notice the need for the computer to be sensitive to clues for backtracking: L7 could not have been gotten from L6 by any plausible sequence of operations; the boxed answer, followed by additional work; and the global history information, repetition of a previous line.)

[L8] $\frac{1}{(2A \quad 1)} \quad + \quad A \quad 1$

[L9] $\frac{A}{(A \quad 4B) \quad \frac{A}{2})} \quad + \quad \frac{1}{2}$

L9 tends to corroborate the claim that the subject has serious problems with combining fractions! Probably, he recognized the similarity to L6 and abandoned the effort without completing the line. This analysis is supported by L10: the subject has backtracked as far as L2. His lack of confidence in the subsequent work (L3-L8) is sound -- he has returned to a correct solution path, with <u>no</u> <u>errors</u> above this branch of the "problem behavior graph".

[L10] $\frac{4AB}{(2A \quad 4B) \quad 2A} \quad - \quad + \quad \frac{1}{2}$

[L11] $\frac{4B}{4A^2 \quad 8AB} \quad + \quad \frac{1}{2}$

L10 is then a positive step forward; L11 is legal, if counterproductive. The subject was apparently aware of the relationship to L3 -- he (deliberately?) chooses a different path forward. His meandering suggests that his <u>strategic</u> difficulties are more fundamental than the matter of illegal operations. By backtracking, he indicates at least a sense of uncertainty about the illegal moves.

[L12] $\frac{A}{2A(A \quad 2B)} \quad + \quad \frac{1}{2}$

The next line, L12, is instructive. His meandering apparently has led him to a re-factoring of the denominator, albeit with a careless error of omission. A good tutor would point out the parallels between L12 and L10: the student might even be led to an understanding of some higher level heuristics about problem solving, though such a global examination of the solution path.

[L13] $\frac{3B}{2(A-3B)} + \frac{1}{2}$

[L14] $\frac{3B+(A-3B)}{2(A-3B)} \quad \frac{A}{2(A-3B)}$

The result is a correct solution of the altered problem, and a fairly common wrong answer at that. The above provides a realistic view of the difficulties encountered in interpreting the implicit operations that the student is performing and which a tutor must be able to decode.

Experimental Data on Student's Doing Algebra:

We have collected a large body of experimental data, from three main sources: (a) a growing number of individual protocols for a (relatively) complex simplification problem, (b) final exam papers from two college level remedial math courses (c) experimental protocols from two tests of our own creation (with many problems borrowed or adapted from the other two sources, in order to facilitate comparisons), administered to a class at a local teachers college. A considerable amount of effort has been devoted to analyzing this data.

A Simplification Problem

In order to investigate the range of student generated solution paths to a mathematical problem which requires both decision making heuristics as well as algebraic manipulation, we chose a simplification task involving partial fractions. Approximately 40 subjects comprising college students in a teachers college, secretaries, and [illegible] students were asked to simplify (i.e. reduce) the following expression to simplest form:

[illegible] $A-3B)]] + 1/2$

Each person was told to work at his own rate (i.e. no effort was made to [illegible] any time pressure) and to show all his work.

Each solution protocol was carefully annotated by hand and these [illegible] were used to construct hypothetical [illegible] models of how the subject might have been [illegible] and how a tutor might have responded.

Qualitative impressions

One overriding impression we got from this experiment was a sense that each person approached solving the problem as if by "rote", with most people failing to see explicit decision points in the solution space. Many people found this task very difficult. For example, out of ten college students presented with the simplification task, three just "threw up their hands" and said they couldn't do it at all. The other seven tried, but none succeeded in correctly solving the problem. Nevertheless, the problem is well within the bounds of high school algebra in difficulty, especially for students who do not instantly panic at the sight of such expressions, realizing, instead, that the problem can be decomposed into a sequence of rather simple steps.

It was also interesting to discover the wide range of answers to the problem (indicating the futility of attempts to specify ahead of time a comprehensive set of incorrect answers). Also the few people that solved the problem correctly (approximately 25%) generated an incredible variety of correct <u>solution</u> <u>paths</u>. The high error rate (or the low mean free time to an error in a solution) also indicated the ineffective use of student time, when the only feedback concerns the validity of the <u>answer</u>, as opposed to commentary on the intermediate steps which were performed.

Figure 4.2 below shows the range of answers generated for the problem and Figure 4.3 shows a typical solution path leading to the correct answer. Figure 4.4 shows an annotated solution to the problem which points out some of the decisions that could lead to solving this problem in an "optimal" way.

[insert Figures 4.2, 4.3, 4.4]

Final exam data:

The analysis of the final exam data from two college level remedial math courses involved using several strategies. For both groups of exams, every problem on every exam was re-graded (they had already been graded by the teacher) in order to understand some of the heuristics used (and errors made) by graders in real world situations. The solutions for three randomly chosen students were analyzed in detail for every problem, carefully describing and accounting for each error. Furthermore, a single problem was selected and analyzed across all students, in order to estimate the range and variety of rules and errors which can occur in even a short exercise.

FIGURE 4.2

Some Typical "Answers"

$$\frac{A}{2A-3B} \qquad \frac{1}{2-3(\frac{B}{A})} \qquad \frac{1}{2(1-\frac{3B}{2A})}$$

$$\frac{1}{A}+\frac{1}{2} \qquad \frac{3B+1}{4A-6B} \qquad \frac{1}{12AB}-\frac{1}{18B2}+\frac{1}{2}$$

$$\frac{3B+1}{4A-6B} \qquad \frac{3B}{2(2A-3B)}+\frac{1}{2} \qquad \frac{A}{4(A\cdot A)-9(B\cdot B)}+\frac{1}{2}$$

$$\frac{1}{2} \div \frac{-A}{9B} \qquad \frac{A}{2A\frac{(1-3B)}{3B}}+\frac{1}{2} \qquad \frac{A}{2(A-3B)}$$

$$\frac{2A+3B}{4A-6B} \qquad \frac{A}{2A-6B} \qquad \frac{1}{2}\left(\frac{1}{1-\frac{3B}{2A}}\right)^{-1}$$

$$\frac{A}{2A\frac{(A-3B)}{3B}}+\frac{1}{2} \qquad \frac{1+(2A-3B)}{2(2A-3B)}$$

98

Figure 4.3

This is a typed replica of a subject's solution path

Reduce:

$$\left\{A \div [(2A-3B)(2A\div 3B)]\right\} + \frac{1}{2}$$

$$\frac{2A}{3B}(2A) - \frac{2A}{3B}(3B)$$

$$\frac{4A^2-6AB}{3B}$$

$$\cancel{A} \quad \frac{3B}{\cancel{A}(4A-6B)}$$

$$\frac{3B}{4A-6B} + \frac{1}{2}$$

$$\frac{3B}{2(2A-3B)} + \frac{1}{2}$$

$$\frac{3B}{2(2A-3B)} + \frac{2A-3B}{2(2A-3B)}$$

$$\frac{2A}{2(2A-3B)}$$

$$\frac{A}{2A-3B}$$

Figure 4.4

$A \div [(2A-3B)(2A \div 3B)] + \frac{1}{2}$

We first transform the problem into a two-dimensional format.

$$\frac{A}{(2A-3B)(\frac{2A}{3B})} + \frac{1}{2}$$

At this point we could either distribute the $(\frac{2A}{3B})$ across the (2A-3B) term or cancel the A's. Cancellation is chosen.

$$\frac{1}{(2A-3B)\frac{2}{3B}} + \frac{1}{2}$$

Again we could distribute the 2 or $\frac{2}{3B}$ but we decide not to since the 2 matches the $\frac{1}{2}$ on the other fraction. Hence to accentuate this global match we move the 3B up.

$$\frac{3B}{(2A-3B)2} + \frac{1}{2}$$

We are now in a position to combine these partial fractions into one fraction. Taking advantage of the 2 in the denominator of each fraction we can combine them as:

$$\frac{3B+(2A-3B)}{(2A-3B)2}$$

which leads to $\frac{2A}{(2A-3B)2}$

and then to $\frac{A}{(2A-3B)}$

Some global observations about this data are in order:

(A) People are <u>incredibly</u> <u>bad</u> at algebraic manipulation. In some sense, we were aware of this before we started -- but our estimate of <u>how</u> bad was an order of magnitude too naive. An example may help to illustrate:

On their final exams in the college remedial algebra course, sixteen students were given the following problem:

$$(-3X)^2(2X)^3$$

<u>Not</u> <u>one</u> <u>student</u> in the course obtained the correct answer, "$72X^5$". The most common answers (four students each) were "$72X$", and "$-65X^5$". Other variations were: "$-36X$", "$17X^5$", "$-216X^2$", "-72", "$-72X^2$", "$-72X$", "$-108X^2$", and "$-16X^2+6X^3$". Results for this problem reached by students in other courses, from other schools, were similar.

(B) A close look at examinations which had already been graded by algebra teachers has indicated fundamental inadequacies of traditional grading techniques. By this is meant that the ranges of scores which would typically be assigned on the basis of right, wrong, and partial credit answers are only crudely correlated with any reasonable metric one might wish to apply to the <u>knowledge</u> <u>structures</u> in question. Suppose we describe in detail the knowledge which we desire for the student to acquire and use. This might be formalized as a set of procedures, operators and heuristics. Perhaps the student has accurately learned all but one of, say, fifty procedures. But in applying this one "buggy" procedure he consistently makes some simple mistake. If there are only a few problems which require the buggy procedure, the score will be high. However, if the faulty procedure applies at a "low level" or -- is called by higher level procedures - (which by contrast implies that the student's high level knowledge may be quite sound), the procedure will be needed in many problems, and the resulting score will be low. Similar themes are elaborated upon by the following examples:

(1) A common test grading heuristic (necessary if there is a poor faculty/student ratio) is to examine the student's work in detail only when the final answer is incorrect (usually to determine whether any partial credit should be assigned). Those problems for which the correct answer has been obtained are only cursorily examined (usually to ensure that no cheating has occurred). This claim is based on

careful regrading of tests previously graded by algebra teachers for the purpose of determining final credit for a course. We can provide examples of students who were passed on the basis of this heuristic, whose algebraic skill appears to be inferior to that of some students who did not. (Other grading techniques also led to the opposite situation.) Two trivial instances should help to illustrate how this can occur. According to a rough statistical analysis, one student always guessed when assigning the sign after an algebraic transformation. Since the correct answers reflected even parity about as often as odd parity, this strategy was remarkably successful on short problems with few intermediate results. This student did almost as well as several others who systematically applied "almost right" procedures. A second, very common confusion which, in more difficult situations can lead quickly to compound errors, involves the relative precedence of the various arithmetic operations. Students who consistently misapply use of precedence and consistently make inappropriate use of (or fail to use) parentheses frequently end up with correct answers, even though many of their intermediate steps are incorrect!

(2) Very often, multiple errors will occur in the solution of a single problem. This reflects a number of weak areas in the student's knowledge. For example, the work may indicate use of the following faulty rules: "X -:- 0 => X", "(XY+Z) -:- Y => X+Z", "A/B + C/D => (A+C)/(B+D)", and so on. But the exam score may be only slightly worse than that of a student who only manifests the last difficulty, since the first rules may only be invoked in problems where the last is also needed.

Additional points that have emerged from our empirical analysis include:

Some of the exams were "uniform graded" by algebra teachers. This means that the same problem was graded by a given teacher across all of the exams, but different teachers would have graded different problems by the same student. This provided an excellent opportunity to assess the effectiveness of analysis which can be achieved by humans in the absence of a detailed student model (on the assumption that constructing an adequate model would require analysis of more than a single problem). In fact, there were many instances where a more global view of the student's strengths and weaknesses would have facilitated, not only the assignment of credit for the problem, but the utility of remedial comments which could have (or ought to have) been made. Whereas many students would actually be required to repeat an entire course because of their poor scores, a global view of their <u>work</u> on the examination indicated that a few sessions of precisely planned remediation might have sufficed.

It would be fruitful in teaching algebra to maintain a distinction between two separate <u>phases</u> of transforming an expression. The first phase consists of procedures for parsing the two-dimensional notation into an unambiguous internal representation (i.e. perceiving the structure of the expression). The second phase consists of the application of "algebraic" (as opposed to "perceptual") procedures to the internal representation. Many student errors which we have seen can be more succinctly characterized as the result of <u>misparsing</u> or misperceiving the external notation, than as incorrect applications of algebraic procedures per se. Consider the following very simple example. The student is asked to evaluate "XY", when X=-3, and Y=-5. The student writes, "-8". This can be most simply understood (especially in the context of many similar errors) by assuming that the student performed (perhaps only in his head): "-3-5 => -8", not having understood that the notational distinction between concatenated literals and concatenated signed numbers constrains one to employ additional parentheses during substitution. Note that little is ever explicitly taught about how to proceed in <u>parsing</u> (or, for that matter, carefully and unambiguously <u>synthesizing</u>) algebraic <u>notation</u>! As a corollary, when understanding student moves, the system's internal representation must not lose the (geometric) information needed to propose that such an incorrect parsing may in fact have occurred. Another example of an error that most likely relates to a <u>perceptual-learning</u> problem rather than as a difficulty with the <u>mathematical</u> procedures per se is that manifested by a subject's work shown in Figure 4.5.

[insert Figure 4.5]

It must be emphasized that the tutoring system must embody sufficient <u>algebraic</u> expertise to deal with fairly complex, unanticipated situations. It is quite common for a simple error to convert an elementary problem into a relatively difficult one. For example, one student, when confronted with:

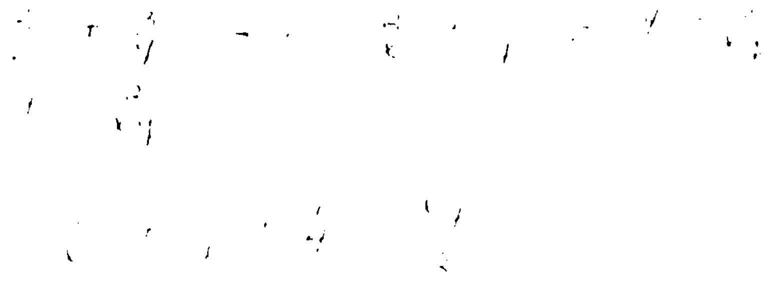
Solve for X: $\frac{3}{x+1} + 2 = \frac{4x}{x+1}$

wrote:

$$\frac{3}{x+1} + \frac{2x+2}{x+1} - \frac{4x}{-x-1} = 0$$

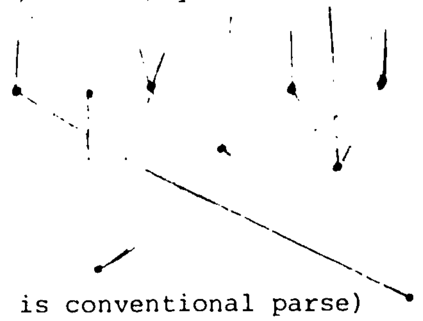
Figure 4.5

(A) Student's work: (#9F)



(B) Explanation as a description/perceptual-learning problem:

2 / x + 3 / y ÷ 4 - 2 / xy



(This is conventional parse)

(This is student's parse.)

101

in which a single sign error has converted an elementary linear equation into a quadratic, which could easily overtax the student's abilities. Such transformations occur frequently, and, moreover, provide the contexts in which student expectations lead to further mistakes (which tend to restore the situation to something "within normal boundaries"). The following student's calculation is a good example:

$$\begin{array}{llrl}
\text{Solve for x:} & x + 9 - 3x & = 2x + 1 & \\
 & 2x + 9 & = 2x + 1 & \text{(commits a sign error)} \\
 & \underline{-2x \qquad\qquad} & \underline{-2x \qquad} & \\
 & x + 9 & = 1 & \text{(his subtraction is in-} \\
 & & & \text{correct but is explain-} \\
 & \underline{\qquad -9 \qquad} & \underline{-9 \qquad} & \text{able - see text)} \\
 & & = 8 &
\end{array}$$

In order for the system to intelligently field such events, its algebraic competence must be extensive enough to recognize the student's dilemma, and not limited, say, to hand coded solutions for pre-arranged problems. This is only one of the ways in which intelligent CAI needs to be "generative". In this particular case this expertise must, indeed, be fairly extensive, encompassing a "theory" of how <u>expectations</u> -- established by the input patterns for a procedure -- can affect one's <u>quick</u> <u>perception</u> of the problem. In this problem, there is little doubt (having viewed the students' previous work etc.) that he "knew" that 2X-2X=0. That is, if he had been given the problem to evaluate "2X-2X" in isolation, he would have given the correct answer without hesitation. However, this same problem in a context which sets up powerful expectations (concerning the pattern for solving such equations) led him on the one hand to conclude 2X-2X was X and on the other hand that it was 1.

A very large proportion of the errors which we have encountered can be accounted for by at most several dozen error schemas; in other words, a relatively tight error taxonomy can be employed. Our categorization of common errors seems to be almost universal. Our data strongly suggest that the occurrence of the various error types is independent of the student's particular background (such as which high school was attended), as well as of the examination score, within a fairly wide range. Of course, very poor students turned in virtually blank exams, which did not manifest these errors at all. Likewise, those students whose scores were almost perfect (there were one or two) made only those errors which could be described as "simply careless". But for the overwhelming majority of students in the middle range, the following sorts of errors were typical:

(a) [anything * zero => anything]. e.g.,

$$5(-5)^2\ (-4)\ -\ (-5)(-4)(0)\ =\ ?$$

$$-500\ -\ 20\ \text{-->}\ -520$$

(b) [a + bc => (a + b)c]. e.g.,

$$2A\ -\ 3B\left(\frac{2A}{3B}\right)\ \text{-->}\ \frac{4A^2\ -\ 6AB}{3B}$$

(c) [anything -:- zero => either zero or anything]. e.g.,

$$\frac{(7(-5))^2}{5(-5)(0)}\ \text{-->}\ \frac{1225}{0}\ \text{-->}\ 1225$$

(d) $[\ \frac{a}{b}\ -:-\ \frac{c}{d}\ \ =>\ \frac{a}{b}\ \cdot\ \frac{c}{d}\ ]$. There are several subvarieties of this form, but this is typical:

$$\left(\frac{1}{2}\right)\ +\ \left(\frac{3}{5}\right)\ \div\ \frac{5}{6}\ \text{-->}\frac{1}{2}\ \cdot\ \frac{5}{6}\ +\ \frac{3}{5}\ \cdot\ \frac{5}{6}$$

(e) Illegal cancellations of the form,

$$\left[\ \frac{xy+zw}{y+w}\ \ =>\ \ x+z\ \ \right].$$

of which a typical instance was:

$$\frac{-24x^5y^6z^8}{3x^3y^2z^8-9x^3y^2z^8}\ \text{-->}\ \frac{-8x^2y^4}{-3}$$

(f) incorrect versions of fractional addition, taking forms such as:

$$[\ \frac{a}{b} \quad \frac{c}{d} \quad => \quad \frac{a+c}{b+d}\ ].$$

This is one of the most common difficulties, and so there are many variations; apparently they reflect an attempt to reconstruct the forgotten, correct form on the basis of primarily syntactic considerations. Student example:

$$\frac{2}{x-1} - \frac{3}{x+2} \;-->>\; \frac{-1}{1}$$

(g) Partial distributions such as: [A(B +/- C) => AB +/- C], which includes the special case:

$$-(3x-w) \;-->>\; -3x-w$$

(h) Confusion of arithmetic operators. Example:

$$(-1)^3 \;-->>\; (-3)$$

Note that several of these (such as b, e, and h) can be more easily explained as attempts to provide interpretations for notation with which the student is unfamiliar (or has forgotten). This list is illustrative, but by no means exhaustive. A more complete list should provide the domain-specific basis for the system's remedial strategy.

Experimental Exam

The results of analyzing the final exams of the two remedial math courses raised some serious doubts as to whether the kinds and frequency of errors we discovered were due, in part, to the lack of any semantic basis for the mathematical procedures (or "how to do it" type material) used in these courses.

Partly to explore this possibility, we designed an experimental test comprised of problems from each of the two exams used in the special remedial classes. This test was then given to a group of college students (from a different college) who came from diverse backgrounds and high schools in which a variety of teaching styles and text books had been used.

The data of this experiment were subjected to the same kind of analysis as was used in analyzing the remedial math exams. From this analysis we discovered that, in fact, the kinds and to a considerable extent, the relative frequencies of the errors encountered on this experimental exam were the same as the others, thereby indicating that our error classification was surprisingly independent of the teaching style and material used.*

Figure 4.6 is the exam used in this experiment and Figure 4.7 provides a summary of the answers encountered in this experiment. (These figures are included at the end of this chapter.)

A Preliminary Theory of Hints, Bugs, and Remediation for Procedural Skills in Algebra

From the above experiments, in conjunction with a preliminary study of a procedural knowledge base for algebraic skills, we believe that a theory of hints and remediation will probably need to be closely tied to a study of causality and purpose ("teleology"). Many of our insights into these issues are also derived from recent research on understanding, planning, and debugging procedures. From this latter work, it appears that many hints are geared to structural features of procedural representations: preconditions, intrinsic commentary ("specifications") intentions ("inductive assertions" interspersed within code segments), extrinsic commentary (purposes -- what higher level goals are being achieved), causal (planning) commentary (how the procedure achieves its objectives). Likewise, remediation is closely linked to procedural "caveats" which warn about typical bugs, environments in which the prerequisites may fail to be satisfied, etc. In fact, our study of the teaching and acquisition of such procedural knowledge promises to lead to fundamental insights into such issues as how "bugs" arise from progressively extending procedures to handle a wider range of inputs and environments.** Arithmetic operations, for example, are first learned with respect to the natural ("counting") numbers, then the (signed) integers, then polynomials (as well as the progression through rational numbers to real numbers, etc.). Many of the common algebraic errors may be derived from this sequence of modifications and extensions.

Conclusion

From the results of our analytic and synthetic study of this domain of knowledge, we now are in a position to complete a practical ICAI system for providing highly adaptive intelligent tutoring for algebraic skills. The above studies clearly indicate the need for the tutoring system to contain a sophisticated problem-solving component in order as to make explicit all the implicit steps that a

----

(*)Alternatively, one might conclude that an inability to successfully integrate the procedural skills and (deductive) semantics is ubiquitous; this requires further investigation.

(**)This idea originated, in part, from Ira Goldstein's compelling theory of Rational Bugs as developed in his doctoral thesis.

student performs in his head while solving a problem, and b) to characterize and generalize the underlying cause of an encountered error. These preliminary studies indicate that a problem-solving component with such capabilities is well within our reach given the kinds of errors and error types we have encountered and classified. Our next step is therefore to build this component and to couple it to our various tutoring strategies.

Figure 4.6

INSTRUCTIONS FOR EXPERIMENTAL ALGEBRA QUIZ

This quiz is being given to you as part of an experiment to understand how to teach algebra more effectively. Your "score" will not affect your average in any courses you may happen to be taking. SIMPLY DO THE BEST YOU CAN. No particular weighting will be assigned to the problems, although they range (in order) from easy to fairly difficult. You will have one hour to work on the exam. You are not expected to be able to complete it in this time. Try not to spend a great deal of time on any one problem - if it seems too hard, go on to the next. There are four pages, not counting these instructions. Although not everyone will have the exact same version of the quiz, they are all about equal in length and difficulty. Do not attempt to work the last two problems unless you have considerable time left after completing the others, and checking your work. Please turn in all of your work! Thank you for participating!

1. $(-3X)^2(2X)^3 = ?$

2. $(-5)(-3)(0) = ?$

3. $(-2)^4 - 2^4 = ?$

4. $\frac{X}{0} = ?$

5. TRUE OR FALSE: When $X=2$, $Y=1$, and $Z=-3$,

   $X(Y-Z)(Y+Z) = -16$?

6. TRUE OR FALSE: $(3+4)^2 = 9+16$?

7. Solve for <u>X</u>:

   Ⓐ $3X + 24 = 18 - 3X$

   Ⓑ $\frac{-3X}{5} = 12$

   Ⓒ $\frac{-2X-7X}{-3} - 1 = 2$

8. Multiply:

   Ⓐ $XY(-2X+3)$

   Ⓑ $(X-1)(X+5)$

   Ⓒ $(X-3)^2$

9. Factor completely:

Ⓐ $X^2 + 5X + 4$

Ⓑ $3X^2 - 4X + 1$

10. Reduce to lowest terms:

Ⓐ $\frac{XY^2Z}{X^3YZ}$

Ⓑ $\frac{4X-4}{X^2-X}$

Ⓒ $\frac{-24X^5Y^6Z^8}{3X^3Y^2Z^8-9X^3Y^2Z^8}$

11. Combine:

Ⓐ $\frac{2X+W}{3Y} - \frac{3X-W}{3Y}$

Ⓑ $\frac{2}{X-1} - \frac{3}{X+2}$

12. Evaluate, when $X = -5$, $Y = -4$, $W = 7$, and $Z = 0$

Ⓐ $5X^2Y - XYZ$

Ⓑ $\frac{(WX)^2}{5XYZ^2}$

13. Perform indicated operations and simplify:

Ⓐ $\frac{15}{7} \div \frac{3}{4}$

Ⓑ $\frac{\frac{1}{2} + \frac{3}{5}}{\frac{5}{6}}$

112

14. Solve for X:

(A) $\frac{3}{X+1} + 2 = \frac{4X}{X+1}$

(B) $\frac{3X+1}{2X} = \frac{5}{2}$

15. (OPTIONAL) Reduce:

$\{A \div [(2A-3B)(2A \div 3B)]\} + 1/2$

16. (OPTIONAL) Solve for X and Y.

$X-Y = 2$

$Y^2-2X = 4$

113

FIGURE 4.7

STUDENT	PROBLEM #								
	#1	#2	#3	#4	#5	#6	#7	#8	#9
1	-6	$72X^6$	$-6X^2-2X$	1	1	FALSE	TRUE	3	3
2	-6	$72X^5$	$-2X(3X+8)$	-5	1	FALSE	TRUE	1	$-2X^2Y+3X$
3	-6	$72X^5$	$-2(3X^2+8X)$	-3-1/2	1	FALSE	FALSE	1	$-2X^2Y+3X$
4	-6	$72X^5$	$-6X^2-16X$	-5	1	TRUE	TRUE	1	$-2X^2Y+3X$
5	-6	72X	$-3-(118+7X)$	$\frac{9}{-1} - 2$	1	TRUE	18	2=2	$-2X^2Y+3X$
6	-6	$54X^5$	$-3X^2+8X+3$	1	a^0 or 0	FALSE	-9	$\frac{-10}{27}$	$-2X^2+3X$
7	-6	$-6X^5$	$-6X^2-16X$	$\frac{5^2-4^2-3^2+1}{-2-1-5}$	1	FALSE	TRUE	1	$-2X^2Y+3X$
8	-6	72X	$-(6X^2+16X)$	-5	1	TRUE	FALSE	1	$-2X^2Y+3X$
9	-6	$-6X^5$	$-6X^2+2X$	-5	1	TRUE	TRUE	1	X^3Y^2
10	-6	$8X^3 9X^2$	$-2X(3X+8)$	-7	1	FALSE	TRUE	0	$-2X^2Y+3X$
11	-6	$54X^5$	$16X+6X^2$	$\frac{4^2-3}{15}$	1	TRUE	FALSE	1	$-2X^2Y+3X$
12	-6	$-6X^5$	77[illegible] $-49X^3$ $-21X^2$	$\frac{7}{-3}$	1	TRUE	FALSE	2	$+1X^3Y^2$
13	-6	0	25X-9X+2	1	a	TRUE	FALSE	-2X-7X	$-2X^2Y+3X$
14	-1^3+6	$-6X^5$	$15X^2$	$\frac{1^2}{-2-1} - \frac{4^2}{5}$	$1a^3$	TRUE	FALSE	2	-----
15	6	72	-21X+-6	$\frac{-3^2}{-8}$	a	TRUE	FALSE	1	$6X^2Y^2$

SECTION III - Related Research on Uses and Representation of Knowledge in "Intelligent" CAI

CHAPTER 5
TOWARD A MORE FLEXIBLE INPUT MEDIUM

Existing constraints on how a student can communicate his thoughts to a machine have unwittingly shaped our views on the range of tutorial interactions and strategies that are possible in CAI systems. The effects of these limitations are abundantly clear in the various mathematical teaching systems in use today. Simply stated, algebra is not naturally done on a typewriter! The kinds of algebraic expressions that a student manipulates cannot be conveniently represented (or even thought about) in a one-dimensional medium. Mathematical expressions are inherently two-dimensional: exponentiation is represented by superscripts, fractions are represented by vertical arrangements of numbers and the fraction bar, etc. In addition many of the manipulations in algebra are described in terms of a two-dimensional layout of expressions. Consider cancellations. Without a two-dimensional representation the rule for cancelling the same factors above and below the fraction line would have to be expressed more awkwardly and then relearned to be applied on expressions written in standard two-dimensional language.

To avoid introducing these artificialities we have been designing a two-dimensional tablet based input system for the algebra tutoring system. It is our intent that with this system the student working on an algebra problem can use this tablet input mechanism just as he would se scratch paper for working out his intermediate results. This would allow a student to work in a natural way (i.e. just as he would without the computer) This increased view of the student's work improves the bandwidth of information being passed to the student modelling system. It actually allows the computer to look over his shoulder and watch what he is doing in order to provide hints and develop a better understanding of the processes that the student employs. For this to work, however, the recognition system must be robust and natural to use or it will interfere with the thought process of the student.

The task of recognizing handwritten algebra expressions can be thought of in two parts: 1) the recognition of the individual characters, symbols, numbers, and digits, and 2) the putting together of these symbols into larger expressions, that is, parsing the string of characters into mathematical expressions. As mentioned above, this parsing must take into consideration two dimensional information, since many of the algebraic operations are expressed as two-dimensional relationships.

Before going on, a word of caution may be in order. The concept, and some of the techniques, for using a tablet to do limited character recognition of mathematical expressions is not new. Indeed one of our prototype systems (to be described later) draws heavily upon some of these existing techniques. However, nearly all existing systems have suffered from two constraints: first, they were designed during a period in which dedicated computers of the power of LSI-11 (PDP-11 costing around $700) were merely pipe-dreams and hence the designers of these systems were forced to make many compromises solely because of the lack of computational resources. Secondly, and much more importantly, none of these systems had any effective way of using higher-order knowledge such as semantics and pragmatics of the given domain to help resolve ambiguities that inevitably appear in the input. In fact, simply segmenting a stream of character strokes into meaningful characters often requires knowledge about the likely intent or meaning of the expressions, as does coping with any sloppiness in the way these characters are written. The net result has been tablet based input systems that are awkward to use and require a substantial amount of human adaptability.

Another reason for our exploring how to design more flexible and context-sensitive input mechanisms, concerns our belief that a knowledge-based CAI system must be able to use its knowledge to perform various tutorial tasks as well as regurgitate factual information. It seemed to us that an elegant and natural spin-off from having systems that can engage in tutorial reasoning would be that these same skills could be focussed inward to hel provide a dramatically more flexible and "forgiving" input medium. Indeed many of the same techniques for isolating, describing and even correcting student errors are precisely what is needed for deciphering ambiguous scribbles.

Experimental Input System:

During this contract we have devoted approximately one man month to constructing a prototype system (that is currently demonstrable) and the design and partial implementation of a second more ambitious system. Both of these systems use the IMLAC, a Computek tablet which is interfaced to the IMLAC and a TENEX computer. The IMLAC does the local and real-time processing of the character strokes as they are drawn on the tablet. The IMLAC then passes the parameterized stroke information off to the TENEX where nearly all the computation is performed. Our prototype systems are programmed mostly in LISP which means we have sacrificed real-time performance for an extremely flexible and powerful programming environment to develop and experiment with our ideas. It is our intent that after we have refined our algorithms and control strategies we shall then worry about implementing them on a system something like LSI-11.

Our first prototype system was written by Daryle Lewis. This system uses a character recognizer based on the Ledeen algorithm which is described in Principles of Interactive Computer Graphics <Newman & Sproull 73>. This character recognizer is also trainable. During the learning session the recognizer extracts simple features from each stroke and stores the character in a decision table under these features. Very briefly, during recognition it determines the set of characters that the input character might have been by comparing the features of the input character with the features in the decision table.

This initial system also used a top-down syntax-driven parser described in Anderson's <Anderson 68> thesis as a technique to use some higher-order knowledge in interpreting the input. In the appendix of this thesis, Anderson describes an efficient parser for mathematical expressions which is a specialization of his more general parser taking advantage of certain features of expressions, namely that the principal operator of an expression is always the left-most character. It also uses the stronger assumption that arithmetic expressions are almost linear, and that they can be linearized. In fact, the scheme begins by linearizing the expression. It does this by first sorting the characters by their spatial location, then assembling strings with the notation for superscript and subscript. Arithmetic expressions can be described as linear with the exception of fractions and as previously mentioned, subscripts and superscripts. Then a linearized expression is parsed in a conventional top-down, left to right way with backtracking. Lewis's prototype system currently recognizes arithmetic operators, integral signs, summation signs, fraction bars, numbers and variables. Example of expressions it's capable of parsing are shown in Figure 5.1.

This system demonstrates that a very simple system with a minimal understanding of two-dimensional expressions can do reasonably well in a highly constrained environment. However the system has definite flaws and probably cannot be extended to adequately fit the needs of our algebra system. Some of its problems are as follows: Anderson's parser never really had noisy real-world data. Instead he gave it letter-perfect data: two dimensional coordinates of characters, perfectly prerecognized characters, and correctly assembled integers. Unfortunately, probably no character recognizer can be good enough to present student generated data in this clean a form. Since the string

$$\int_{0}^{i+5}\int_{i+5}^{23} SI\!'X^{2} COS Z\, dX\, dZ$$

$$FG - B_{2}^{3A} + D$$

This is a faithful copy of the actual written input expression done by the user at the Computer Tablet.

```
 I + 5    23
/\       /\
 |        |          2
 |        |     SIN(X  ) COS(Z) dX dZ
 |        |
\/       \/
 0        I + 5
------------------------------------
                   3 A
        F G - B        + D
                 2
```

This is how the input expression gets printed out on the TI or teletype terminals.

```
_PP FOO
  (DIV (INTEGRATE 0 (ADD I 5)
                  (INTEGRATE (ADD I 5
                             23
                             (MPY (SIN (POWER X 2))
                                  (COS Z))
                             X)
                  Z)
       (ADD (SUB (MPY F G)
                 (POWER (SUBSCRIPT B (2))
                        (MPY 3 A)))
            D))
```

This is the semantic interpretation of the input expression.

Figure 5.1

parser is nondeterministic, it could be extended to cover the uncertainties present in the character recognition but the linearizer is inherently deterministic and is very sensitive to poor noisy data and therefore limits the robustness, flexibility and strength of this kind of recognition system. The character recognizer of Ledeen has its limitations too. For example it does not address the segmentation problem, that is, partitioning the stream of strokes into groups of strokes making up each character. To circumvent this problem our initial system required the student to pause a half a second between characters in order to unambiguously and correctly segment the strokes into the characters. This technique works but represents an unnatural and disturbing constraint for a student using the tablet.

Because of the psychological ramifications of these constraints, the system was extended to the point where it collects a stream of strokes <u>without</u> explicit indication of the inter character segmentation and then attempts to fit characters over the stream. Spatial information is used in this segmentation. The overlap relationship between adjacent strokes is used to strongly suggest that this pair of strokes belong to the same character. Likewise a longer sequence of strokes that makes up a character is preferred, and a number of other heuristics help the segmentation. Although these changes improved its likelihood of success, we decided that a fundamentally different approach should be explored.

New Views of Parsing

One view of parsing is that of translating strokes or other input into a tree or other meaningful structure. In this view, a grammar describes the bridge between two representations. The meaning of an input travels over the bridge into new, structured representations. Parsing is grouping and rearranging information.

A more recent and more powerful view is that parsing is not rearranging meaning but rather finding meaning, and just as importantly, finding intention in a communication. Consequently, an input may not be a correct realization of the intended meaning, but may be full of mistakes, sloppiness and other noise. To translate the input without any consistency or plausibil [illegible] checks will further obscure the meaning by introducing [illegible]ore erroneous structure. It is only by knowing what the c[illegible]unicator could want to say and what could be said that [illegible] utterance can be understood. A parser and a grammar must be much more a filter of what inputs make sense. What the input seems to mean may be nonsensical or irrelevant to the situation in which it is said. It is only by understanding the situation that the right interpretation can be found. The importance of the situation of an input has only recently come to be recognized. Communication always has a context and can only be understood within it. Linguists have become painfully aware of the weakness of syntax without semantics. What we hope will distinguish our tablet understander from previous ones is the constructive use of semantic knowledge as a powerful constraint or filter for parsing.

There are many levels of knowledge to be applied to input that will potentially help the parser disambiguate. At the simplest level, algebraic expressions have a syntax. For example, the knowledge that parenthesis come in pairs can help distinguish a "C" from a left parenthesis which can be realized by the same input stroke. Only context identifies which was meant. Natural hand printing is not precise enough to distinguish between these two characters or between many other pairs of characters. For this reason many previous tablet systems have required the user to learn new printing conven.ions such as slashing zeros. These conventions shou[illegible] not be necessary where the meaning is "obvious" (to the tudent anyway). At another level of knowledge, the [illegible]stem knows which algebraic manipulations are being taught [illegible]ierefore, by knowing what exercises are being performed, [illegible] student's input can be related to them by the operations and manipulations being taught. The system might even know of faulty manipulations that students commonly use. For example if the distributive operation were applied to (A+B)X and the parser found AX+13X it would be able to see it as AX+BX. On yet another level the system could use its model of the student to understand the input. By being aware of the student's vocabulary of algebraic symbols and manipulations some interpretations of input become less likely. 31 might look like 3!, but if the student doesn't know about "!", the factorial sign, then the correct interpretation is easy to make.

So we have seen that there are many levels of knowledge that can be applied to parsing. What makes this system a challenge to design is that each of these sources of knowledge is imperfect and will introduce errors. Collectively they must correct their errors and arrive at the best interpretation of imprecise input.

Chart Parser for Parsing Subexpressions

Using a straightforward top-down parser for rejecting anomalous character identification is subject to serious combinatorial problems because a purely backtracking parser throws away the good information gathered under an unsuccessful hypothesis. To overcome these and other problems, another approach is being tried by Steve Purcell. The second system uses the same character recognizer but will employ a substantially different parsing algorithm. The parser will be a chart parser like those developed by Martin Kay and Ronald Kaplan. The chart is a data structure to hold the fragments of successfully parsed expressions. All the phrases that are ever noticed are remembered and are available for building larger phrases. The chart holds the entire utterance or expression. This allows parsing to proceed from whatever islands of certainty there are. The parser is not constrained to proceeding in chronological or left to right fashion! The chart would correspond to the notion in Woods' ATN grammars of the well-formed substring table. This is a table of subexpressions which are saved when the parser makes a mistake and backs up. The well-formed substring is a successful parse of a group of words. It may be used by another path in the grammar other than the one that first requested the phrase. This helps the top-down parser prevent the repeated parsing of lowest level expressions.

The basic concept of the chart must be adapted to two dimensions. In addition, there are strong built-in assumptions that the parser is parsing a one-dimensional utterance. In one dimensional strings a phrase is constrained to cover or include one interval (in time) of input data. In two dimensions a phrase is, roughly, some continuous neighborhood of a simple-connected set of the two-dimensional plane. In one dimension the simple-connected set is merely an interval and can be described by its end points. In two dimensions there is no such simple description of simple connected sets because the sets can be very complicated. There can be very many different connected sets.

This second system will not linearize the characters into a string for the parser. It is felt that this would destroy many of the two-dimensional relationships. A mapping of two dimensions into one is always going to be fragile and unstable. Small differences of interpretation in two dimensions could lead to radically different one dimensional strings.

The structure of a one-dimensional grammar or parser rises from the notion of phrases, coupled with the notion of concatenation. In one-dimensional strings there are two kinds of concatenation: symbols are concatenated to the left and to the right. Similarly, larger phrases are viewed as concatenated with their left and right neighbors so the constraint for forming a phrase in a string is that the constituents be contiguous and non-overlapping. For a phrase to become part of a larger phrase it must combine either with its left or its right neighbor. In two dimensions there are more possibilities; a phrase could combine with another phrase in any direction. For our purposes (in algebra expressions) we consider nine different neighborhood concatenation relationships: above, below, left, right, overlapping and the 4 diagonals. The model of parsing will be to take the characters in their enclosing rectangles, the coordinates of their centers, and to compute the two-dimensional relationships between neighboring characters to propose neighbors as potential phrases. It will record the neighborhood relationship of the larger constituents and grow larger and larger phrases. The constraints on this parser are similar to the string parser. Two subphrases of a phrase must be neighboring and they must be disjoint sets of strokes. The parse will be a tree with one root and the leaves will be all the strokes or all the characters of the input.

Some phrases may cause expectations to be set up which will affect interpretations of neighboring phrases. For example, the presence of an integral sign would cause the character "d" to be interpreted as part of the integral derivative. The parser will reach decision points as it builds phrases. This reflects the local ambiguity that will not be resolved until hopefully later in the parse. The chart has to hold all the possibilities or the representation of the possibilities so far encountered.

The chart is constructed so that decisions or ambiguities in one area of the expression will not recombine with the ambiguities of another, as long as these phrases are disjoint. This is in contrast with the back-up parser. If two independent assumptions or decisions are made and the first proves false, then the second is needlessly undone and remade. In this way the backtracker multiplies out ambiguities even though they are essentially in phrases that do not overlap or interact.

But even a chart parser still has the potential of exploding. There may be an ambiguity in a subphrase of a much larger phrase. This ambiguity could be preserved and larger phrases constructed which may be similar in every way except in one subphrase of a subphrase, etc. To prevent this kind of explosion we hope that the parser can converge its idea of the parse and merge alternative parses as soon as there is a resemblance.

For example there is the ambiguity between the letter B and the number 13 which participates in a large phrase 4(A + 13) versus 4(A + B). The parser would build two independent large structures differing only in the B versus 13. The Purcell parser as planned will merge these two structures into the same structure where the representation, the B or the 13, will have a decision marker on it. When alternative phrases arise which do not merge soon, the parser will stop growing the phrase in that direction and wait for expectations to be established to give it better guidelines for choosing between ambiguous phrases.

The algebraic context is another source of information that the parser must make use of. In a series of expressions written in the course of solving an algebra problem, there is much similarity from line to line as expressions are manipulated and rewritten. For example, there are many expressions that the parser should be able to recognize once it encounters the phrase 4(A + B). If 4(A + 13) is then input, the 13 should be forced back to B or the original B should be reworked as a 13.

In other words the parser will attempt to recognize expressions that it has encountered before. This is an example of how the semantics of algebra augment the syntactic grammar in the parser. To do this, equivalence relations on several levels will be recognized. There are many graphical realizations of any given number. For example, 4 can be written with an open or closed top. There are equivalences in the algebraic vocabulary. Division can be indicated by a minus sign with a dot above and below, or it can be realized by a diagonal slash as in a fraction or as a horizontal bar as in a large fraction. Any occurrence of one of these will have associated with it the canonical form of division so that, in successive expressions, division realized in one way will correspond to division realized in another. This correspondence can guide the parser to the right interpretation.

There is also the notion of a canonical expression. For example, let's say the parser has already encountered the expression A + B and now encounters A +. It maps A to the canonical A which has already appeared in previous expressions. It will also see, among other things, "A + B". This will motivate the parser to look for B on the other side of the +. This is one way that expectations are generated. The advantages of looking for common subexpressions are that an expression which has been written clearly and carefully once, can then be written more sloppily and the parser will be more robust and able to recognize it.

There are even stronger semantics to guide parsing. Another level of equivalence that the parser can look for is algebraic equivalence under the legal transformations of algebra: cancellation, solving equations, dividing through both sides of an equation, etc. Through the series of steps in solving an algebra problem, the variables and expressions can be traced from line to line, transformed slightly but largely remaining the same. There is hope that the parser can do this in conjunction with the understander which is monitoring a student. The trouble that the parser has may be trouble that the student has, and may be worth commenting on.

We've said that the character recognizer used in the Lewis system will probably be used in the Purcell system. It has, however, a few defects. The recognizer is a feature extracter and the features that it extracts are not sufficient for distinguishing some characters that it should distinguish. The recognizer confuses u's and v's, 2's and z's, c's and parentheses, 7's and parentheses, 1's and parentheses, integral signs and 1's, etc. One feature that probably could be added to the recognizer that could be very useful in distinguishing many of these otherwise ambiguous characters would be some kind of inflection point recognizer, some second derivative of the curve, or some measure of curvature to recognize sharp corners of v's, 7's, etc. Extra feature extraction may be desirable after an initial guess at the letter has been made. If the letter falls into a certain kind of ambiguity, a second feature detector might be invoked. Alternatively all the features might be extracted initially on all the strokes. Either method would work.

Where are we in the construction of the Purcell system? Most of the system is still on the drawing board. What has been implemented is a program to grow a network of two-dimensional neighbor relations between characters. By using the enclosing rectangles of pairs of characters which are near each other and locations of their x and y centers, the pairs are put into spatial relations such as above or below. The first relation builder is being interfaced with a tablet to see how it can handle simple expressions. The relations are easy to find for very simple, carefully drawn data and it is expected that it will degrade gracefully, still finding most relations correctly. Some of the partial strategies have been experimented with by hand. Samples of data from a large number of subjects have been used to design the system described above. A lot of implementing lies ahead, but there is every reason to expect that the parser will be able to understand algebra expressions from the tablet. The open question is how much the user will have to adapt to the computer to make the task easier for it. We hope that as few conventions as possible will be necessary so that the tablet can be as natural and unobtrusive as possible. We hope, for example, that 0's or z's will not have to be crossed to distinguish them from o's and 2's. Perhaps this convention can be used as a backup in the absence of context, but we hope the user will not view the tablet as any different than paper.

There are many things that can be done, with the dynamic medium of a graphics display and a tablet, that can't be done with pencil and paper. For the time being, we plan to avoid these so that the tablet can be as similar to paper as possible. It is tempting to put in some kind of editing facilities to the tablet, but this will not be pursued.

We have high hopes that the tablet will be a large improvement over typewriter input of algebra. We think that tablet understanding will touch on many interesting issues in its own right; issues of using context to disambiguate a noisy environment, and developing more general notions of parsing. Related to this graphical input is the graphical output of expressions. The Lewis system also included an expression "pretty printer" that transformed expressions into two-dimensional text. It handled layout successfully, but had trouble expressing superscripts and subscripts because characters had to lie on a line or clearly up on the next line above and subscripts could not be made smaller. Perhaps characters could be scaled and translated to form nice expressions. Some expressions or characters could be transformed by rule, not merely by shrinking or stretching. Square roots for example have a definite width and height and could merely have a rule to extend the root sign. We don't see it as important to rewrite the student input for feedback. If the parser works, there's no need for it. The original input is most easily recognizable to the user; after all it's what he entered. But there is a time when the tutor would like to give examples, and give them in as similar a way as possible to the way the student would perform the same example. For instance, a demonstration of cancelling fractions needs an expression printer.

In conclusion, we see the graphical medium for both input and output as having the potential for greatly improving the communication between computer tutor and student.

CHAPTER 6
SYSTEMATIC UNDERSTANDING

Synthesis, Analysis and Contingent Knowledge in Specialized Understanding Systems*

A fundamental challenge in the design of useful knowledge-based CAI systems is to find ways to embed knowledge and procedural skills into these systems in a manner which renders them both efficient and robust understanders (e.g. problem-solvers, question-answerers, student modellers, answer evaluators, etc.). The designers of nearly all generative and/or AI-based CAI systems have avoided facing the complexities inherent in substantive domains of knowledge by tackling only small, well-defined and closed subject domains. Indeed there are few, if any, knowledge-based systems in existence that honestly face the challenge of completely modelling a realistic body of knowledge! This is due in part to the shortcomings of current computer technology (which is rapidly and dramatically improving) and in part to the lack of any comprehensive and viable theory of how large and complex bodies of knowledge can be represented in computer-based systems.

As an attempt to remedy this deficiency we have combined the collective experience gained in building various knowledge-based CAI systems, and have constructed the beginnings of a theory for how to represent complex bodies of knowledge. It is our intent that a fully developed theory will not only establish some guiding principles for how to build practical CAI systems that really can "act" as intelligently as a human tutor, but that such a theory will also provide us with a new and more powerful methodology for understanding the structure and content of the particular knowledge needed to carry out a collection of tasks. If so, we will have a new handle on how to view the skills that must be imparted to a student and how the student is to use this knowledge in executing his assigned tasks!

(*)This chapter is to appear, in a slightly altered form, as a chapter in Representations and Understanding -- Studies in Cognitive Science edited by D. Bobrow and A. Collins, Academic Press 1975.

In the design of a knowledge-based system, careful attention must be paid to the choice of representations for different components of that knowledge. Clearly, the best representation for a body of knowledge depends on how that knowledge is to be used by the program, and thus better characterization of the uses of knowledge is likely to lead to better ways of designing knowledge representations. In this chapter we present the SCA model, a framework for describing the structure of "conceptually efficient" understanding programs, based on a characterization of three fundamentally different ways in which knowledge is used in such programs. Even though the SCA model is not full-developed, we feel that it can be of use both to those who are designing understanding programs, and to those who wish to study existing programs to develop insights into different approaches to representing knowledge.

The version of the SCA model described in this chapter applies to a class of programs that we refer to as specialists or expert understanders. These programs understand the world in the sense that they can take in a collection of data describing some situation, and then answer questions about that situation. The programs are referred to as specialists because they are only able to deal with a limited class of situations, and can only answer questions in some limited domain of specialization.

> We are still in the process of developing the SCA model, and many ideas are still in the form of speculations and intuitions. In order to present the essential aspects of the SCA model in the clearest possible form, we have described a simplified version of the model which does not deal with a number of crucial issues in the design of understanding systems. In particular, though the systems to be discussed "learn" about their environment in the simple sense of taking in descriptions of the current state of the environment, they do not learn how to improve their performance, nor do they extend their initial knowledge about those properties of their environment which hold in all states. Additionally, though we believe strongly that complex problems in understanding will be solved by programs built from many specialized modules that communicate and interact within a complex control structure, the model we describe consists of two programs that interact by simply creating and accessing a common data-structure.

Part of our intent in articulating the SCA model is to present a point of view on the design of expert understanders which provides insight into how knowledge of the expert's domain can be effectively used to design a conceptually efficient understander. The SCA model provides a framework for studying the structural similarities of a variety of superficially dissimilar high-performance understanding systems including perceptual understanding systems, natural language data base management systems and question answering systems. It also provides a basis for discussing many current knowledge-based research issues such as the meaning of analogical representations, the pros and cons of different inference schemes (e.g. computational, rule-driven...), ad hoc versus general knowledge represen1ations, and the integration of such tools as simulation into knowledge-based systems.

The remainder of this chapter is divided into two major sections. The first section is an account and explanation of the concepts involved in the SCA model. This section starts with a brief description of the general concepts of the SCA model, and then gives three examples of systems so organized, in order to provide an intuitive feel for the meaning of the concepts and their intended breadth. The second section describes various ways in which the SCA model can give insight into problems in the design of knowledge representations.

Section 1:

THE SCA MODEL

Brief Description of the SCA Model

For the purpose of providing an overview of this model, let us consider a hypothetical expert understander who obtains information about the world in the form of a collection of basic uninterpreted "sensations" or raw input data. The expert must answer questions about the world on the basis of this raw data. Much like a human expert, our hypothetical expert uses his specialized knowledge to combine, organize and augment this data, and thereby synthesize a substantially enriched world model or _contingent knowledge structure_ (CKS).

> In its simplest form, this CKS might just be a data base of assertions describing the expert's current knowledge. In general, however, the CKS is a _complex data structure_ that represents the expert's current model of the world. This model may include analog representations of some aspects of the world, as well as semantic networks and other knowledge representations.

The SCA model extends this idea and specifies that an understander should be designed as two separate experts, one to _synthesize_ a CKS from the collection of raw input data and the other to _analyze_ the information in the CKS in order to answer the questions posed to the understander. The raw input data are the result of the operation of other programs or input devices, and are not usually represented in terms of concepts that are directly usable by the understander. The first expert, or _synthesist_, converts this raw input data into a form suitable for the understander to act on.

> In addition to simply augmenting the information contained in the raw data, and reorganizing this data, the synthesis operation often changes the elementary concepts in which the information is expressed (e.g. changing from a representation of a visual scene as an array of intensities to a collection of boundary lines, or from a collection of boundary lines to a set of three-dimensional objects).

The second expert, or _analyst_, retrieves knowledge explicitly represented in the CKS, and uses world knowledge to infer facts not explicitly contained in the CKS, perhaps using procedures as complex as general theorem provers.

The effectiveness of the total understander depends critically on the designer's ability 1) to provide the correct balance of skills (and computational load) between the synthesist and the analyst, 2) to design a class of CKS's which represent just those aspects of the world state needed to facilitate the operation of the analyst, and which can be directly represented as efficient combinations of procedures and data-structures, and 3) to use special purpose techniques to enable the synthesist to fill in the CKS in an efficient way.

Section II:

GENERAL DESCRIPTION OF THE ELEMENTS OF THE SCA MODEL

The Synthesist

The synthesist describes or accounts for a collection of input data in terms of some acceptable structure which is an instance of a class of structures specified by the designer. It must impose the "best" acceptable structure on the input data (which may already be organized in some fashion). The imposed structure accounts for the original input data in terms of concepts useful to the analyst, and provides the only mechanism through which the analyst is permitted to obtain information about the state of the world. In general, the analyst never has direct access to the information contained in the raw input data. In more elaborate versions of the SCA model the analyst can request the synthesist to synthesize a new CKS based on the current needs of the analyst. This may change the effective information that the analyst obtains from the raw input data, by means of the CKS, but it does not remove the CKS as a necessary intermediary.

There are three inter-related operations that may be performed by the synthesist:

1) structure determination - determine which of the potential structures (e.g. configurations of blocks, parse trees, etc.) it knows about should be used to represent the given raw data,

2) matching - determine the correspondence between the raw data items and the parameters of the chosen structure, and,

3) augmentation - determining parameters of the CKS not directly corresponding to raw data items, but which must be chosen to satisfy some set of constraints.

These operations may require a coordinated search through both the entire set of facts and the possibly infinite set of potential structures, in a matching or parsing phase. The synthesist may put the given facts in a canonical form, for example by algebraic simplification, by reduction to a structure composed of semantic primitives, or by using hash-coding and search to reduce an expression to a unique internal structure <Reboh 73>. Finally, the synthesist may fill in parameters of the structure in a manner determined by the raw input data and a set of constraints. The synthesist may only have to perform one or two of these operations (for example, when the matching operation and structure determination are trivial but the augmentation operation is difficult).

It is often possible to distinguish repositories of knowledge corresponding to an active synthesis agent, a description of the class of possible contingent knowledge structures, and a goodness measure which evaluates how well a structure matches up with the raw input data. In the LUNAR parser <Woods 73>, for example, the description of the class of possible contingent knowledge structures (as well as rules for searching for the "best" structure) is given as an Augmented Transition Network and the active agent is the general ATN parser. In many cases, however, it is impossible to separate the synthesizing agent from the description of the structures to be synthesized. In these cases the synthesist is implemented as a specialized procedure for instantiating a particular class of structure - this is often more efficient, though less flexible.

It may seem possible to build in accessing functions in the analyst and do away with synthesis, particularly where the information used by the analyst only depends on a small amount of raw input data. However, the choice of structure, even for a small portion of the raw input data, often depends on the entire collection of raw input data. In these cases one cannot do without the synthesist by simply putting extra processing in the analyst - the extra processing needed is actually the complete synthesis operation. For example in the blocks world the analyst need only look at a structure determined by a small part of the input to answer questions about a single block, but the very concept of a block depends on a synthesis process that takes into account the entire set of line segments in the scene.

Contingent Knowledge Structures

The contingent knowledge structure, (CKS) is a data-structure which represents the understander's knowledge of the state of the world. The CKS forms the interface between the synthesist and analyst, and thus determines the way in which the characteristics of the world are interpreted by the specialized question-answering and problem-solving capabilities of the analyst. Because of this, the capability and efficiency of an SCA understander depends critically on the design of the CKS and its computer representation.

There are two distinct aspects to the design of the CKS:

1) the conceptual structure which the CKS imposes on the world

2) the data structures and procedures chosen to represent the entities, properties and relations which make up the conceptual structure of a CKS.

Like a language, the class of possible CKS's define a set of conceptual entities, relations and structures which determine the way the analyst can most easily express its questions and operate on its model of the world. It determines the distinctions which the analyst can possibly make between states of the world. For example, in the blocks world understander the conceptual entities are blocks with sizes, shapes, positions in three-dimensional space, and relations depending on their positions - rather than corners, connected sequences of line segments, or any other possible structural entities.

Given several alternative conceptual structures for the class of CKS's, it is tempting to choose the "most expressive" one, to facilitate the description of the states of the world and the questions of the analyst. However, the analyst must be capable of dealing with any CKS within the chosen conceptual framework, and more subtle and complex frameworks require more complicated (therefore usually less efficient) analysts. Thus there is a tradeoff between expressiveness and efficiency, and a good conceptual structure is one that can readily express those properties of the world that are relevant to the analyst, but does not lend itself to describing irrelevant details or impossible states. An example of an unproductively rich conceptual structure for the blocks world CKS would be one that could represent all connected sets of line segments, including collections of lines not corresponding to the boundaries of a set of blocks. Such collections are impossible in the blocks world and hence are irrelevant to the blocks world analyst.

We believe this tradeoff is closely related to one hypothesized by Dr. Frederick Thompson of Caltech <Thompson 72, Randall 70, Greenfeld 72>. Thompson's conjectured "Fundamental Theorem of Information" says, roughly, that given a collection of observations of the world, and a sequence of progressively richer languages, there is an intermediate language in which the descriptions of the observations carry the greatest information. This most informative language is not rich enough to express all the details of every observation - the concepts that make up its semantics are broader and more abstract than the details of the observations, and thus it captures the important properties of the observations without allowing the expression of unnecessary or irrelevant detail.

By fiat, the CKS provides the only mechanism for the analyst to refer to the properties of the world state. The CKS reduces the complexity of the analyst by providing a form of canonical representation for world states. States of the world which can be treated similarly by the analyst, are mapped into similar CKS data structures. Thus, the conceptual structure must be designed to facilitate both the expression of the questions which the analyst must answer and the operations which the analyst must perform in order to answer the questions.

We believe that in most cases the CKS is best viewed as a <u>model</u> for the state of the world, rather than a description of the state, so that the operations of the analyst correspond more to observations of the world than to manipulation of the representation of assertions. We use the term "observation" in the sense of an operation on the world that produces information as a result, and which does not change the state of the world (this is in accord with the simple version of the SCA model, but is not a general restriction). We do not mean to imply that such observations are simple operations, or that the analyst is simply an information retrieval mechanism that observes what is explicitly present in the CKS.

Much of the understander's knowledge of the state of the world is not contained explicitly in the CKS, but is embedded in a set of tacit agreements between the synthesist and analyst as to the way in which the data-structures that form the CKS are to be interpreted. For example:

a) A linear sequence of links between several items can be interpreted as a transitive relation if the analyst determines that two items are related by seeing if there is a chain of links joining them.

b) A set of pointers to objects from elements of a three-dimensional array is sufficient to represent many of the three-dimensional relations among a group of physical objects if both the synthesist and analyst interpret the existence of a pointer from an array element A(I,J,K) to an object O as meaning that O is lies within a box centered at coordinates (I,J,K).

c) A list structure can be used to represent a procedure for answering a question, given a set of rules such as those used to interpret LISP forms.

Given a particular choice of conceptual structure and a set of operations (defined in terms of the conceptual structure) which the analyst must perform, the designer must choose a collection of computer data-structures to represent the CKS, which maintains the comprehensibility of the final program and its overall efficiency. The distinctions we draw between the conceptual structure of the CKS, the implementation of the conceptual constructs in terms of programming language constructs, and the eventual implementation of these constructs in terms of machine-level primitive operations are an attempt to deal with the problem that Hayes <Hayes 74>. poses:

> "a representation which appears to be a direct model at one level ... may ... be itself represented in a descriptive fashion, so that it becomes impossible to describe the overall representation as purely either one or the other. It seems essential, therefore, to use a notion of <u>level</u> of representation in attempting to make this distinction precise."

<u>The Analyst</u>

An analyst derives information from a CKS in order to answer questions posed by some other process. Informally, if the CKS represents a state of the world viewed from a perspective defined by the conceptual structure of the CKS, then the analyst infers answers to specific questions about the state of the world using information in the CKS and a set of rules. These rules include laws of the world and laws of logical inference, so that the answers provided by the analyst correspond to true propositions about the state of the world. The CKS is of necessity a finite collection of information, but the set of questions one can ask about the context usually come from an infinite set defined by linguistic rules. Thus, <u>the analyst is an inference mechanism for bridging the gap between the finite set of the properties of the world which are explicitly represented in the CKS, and the infinite set of valid assertions</u> (answers to questions) <u>about the world, which are implicitly determined by the CKS</u>.

The simplest analytic operations consist of the application of compositions of functions and relations to elements of a CKS, for example forming the sum of the lengths of several lines in a geometric structure, or comparing such a sum with the distance between two points. More complicated analytic operations might consist of using the results of such simple operations, along with general world knowledge to deduce further properties of the world state represented by the CKS. <u>In many cases the understander need not explain or explicitly justify its answers, so explicit logical deduction can be replaced by other forms of inference or computation</u>. For example, in a CKS which consists in part of a semantic network, properties of an element can be inferred by tracing a path to some other element and then applying simple computational rules to a description of the relations in the path and the properties of objects on the path.

> The fact that "John's uncle is Jane's grandfather" could be derived from a chain "John SON-OF Peter HUSBAND-OF Mary SISTER-OF Isaac FATHER-OF Ellen WIFE-OF Jack FATHER-OF JANE", by the application of a set of simple rewriting rules to the sequence "SON-OF HUSBAND-OF SISTER-OF FATHER-OF WIFE-OF FATHER-OF".

Section III:

COMMENTS ON REPRESENTATION ISSUES

Universal and Ad Hoc Contingent Knowledge Representations

A theory such as the one which we hope underlies our SCA model would make it possible to discuss rationally the representation of knowledge at the interface of the synthesist and analyst, as well as the design and operation of synthesists and analysts for particular problem domains. It would permit us to phrase meaningful questions about the relative merits of different contingent knowledge representations from the point of view of efficiency of synthesis, analysis and original design, and could thus clarify the debate over the relative merits of "general purpose" and "ad hoc" representations of knowledge.

Let us take a superficial look at this debate in terms of the SCA model. Given that all understanding systems must convert their raw input data into data structures used to meet goals of the understander, the more goals that can be met effectively with a single structure, the fewer times must a synthesist be invoked to create another one. However, building a single structure to serve many independent goal-oriented procedures may be more difficult than building several different specialized structures. In addition, the improvement in efficiency of the goal-oriented analytic operations may be brought about by the availability of specialized contingent structures to make up for the extra time spent in building them.

One must also take into account the resources necessary to maintain consistency and compatibility among multiple representations, or allow for the problems of potentially inconsistent actions by different analysts*. These problems are particularly difficult to resolve in understanding systems which generate and deal with multiple contexts, such as planning and problem solving systems.

One must also evaluate the impact of "generalized" and "ad hoc" representations on the problem of designing systems. Clearly, having a single, highly efficient and effective knowledge representation would substantially reduce the time necessary to design new understanding systems. Even a unified conceptual framework like "the omega order predicate calculus" (as in QLISP) can ease the designer's task. On the other hand, the usefulness of engineering handbooks attests to the fact that an organized collection of specialized structures with capabilities and limitations clearly spelled out can be quite as good a

----

(*)There is certainly evidence that human understanders have inconsistent representations of knowledge, and that they can come to inconsistent conclusions by using different techniques for solving problems or answering questions.

design aid as a single generalized technique.

Contingent Knowledge Structures and the Antecedent/Consequent Boundary

In building understanding systems with procedural representations of knowledge, there is a serious design problem in the distribution of expertise between the antecedent ("if-added") and consequent ("if-needed") procedures (as in PLANNER, <Hewitt 72>, CONNIVER, <Sussman 72> or QLISP <Reboh 73>. Roughly speaking, if-needed procedures are triggered by the introduction of specific goals and sub-goals to be met by the understander, while if-added procedures are triggered by addition of new facts to the contextual knowledge base.

> The if-added procedures clearly correspond to the operations of synthesis. It may not be clear that if-needed procedures correspond to a combination of synthesis and analysis, and not simply to analysis. Simply speaking, the if-added procedures correspond to data synthesis, while the if-needed procedures correspond to goal synthesis, since they replace a set of goals with a structure of goals and sub-goals that combine to satisfy the original goals.

One could conceivably avoid the use of if-added procedures entirely, by making all procedures goal-oriented, and "reasoning backward", so that nothing is done until it must be done to meet a specific goal. This runs into difficulty since it allows for little coordination between the goals and the context, so that it is possible to generate vast numbers of irrelevant, impossible and costly sub-goals. Additionally, unless the results of sub-goals are added to the knowledge base one can needlessly repeat many sub-goal computations.

Alternatively, one can "reason forward", taking the contents of the context and applying if-added procedures to derive all the goals which could be met given the context. I this approach is implemented in an unrestrained fashion one can end up swamping the data base in irrelevant results before getting around to meeting the specific goals posed to the system.

The concept of a CKS for a class of goals provides a handle on the problem of how far to let the if-added procedures run. In essence, the if-added procedures become "potential goal" directed, as compared to the "specific goal" directed if-needed procedures. The CKS to be produced by the if-added procedures stands in for the entire set of potential goals. While it may be said that if-added procedures are always directed towards satisfying a set of potential goals, the explicit design of a CKS for a given set of goals provides a mechanism for keeping track of decisions as to what knowledge is to be encoded in the if-added as opposed to the if-needed procedures. Given a set of goals, one needs only to define a class of CKS to organize contextual information and simplify the execution of the corresponding if-needed procedures. If-added procedures then implement a synthesis procedure to build such structures from any of the expected contexts. One can even package such compatible sets of if-needed and if-added procedures into "demon teams" (as in QLISP, <Reboh 73>)

There is a catch to the above suggestion - how does one find a compatible set of if-needed demons whose operations are facilitated by a single contingent knowledge structure? While we have no general answer to the problem, the technique used in system understanders is suggestive. Essentially, one replaces the search for a CKS and compatible set of if-needed demons with a search for a query language in which all the demons' information needs can be expressed. Starting with a rough idea of these basic semantic entities relevant to the demons (e.g. some "objects", "structures", "relations", etc.) one considers the types of questions about such semantic entities whose answers would help meet the demons' goals. This can often be refined to a well-defined set of primitive questions and composition operations which can be used to answer all of the needed questions. One can then design data structures and procedures that facilitate answering all the questions in the query language, and the data structure and procedures form the class of CKS's.

Higher-Level Structures (Frames) and World Knowledge

The raw input presented to an understander is insufficient to tell the understander all it needs to know about the specific situation it is in. There is always a need for world knowledge in the understander to fill in the meaning of the input. Several chapters in this book discuss the problems of how to organize such higher-level knowledge, how to find knowledge relevant to a given collection of inputs, and how to use this knowledge to provide an interpretation of the input. We refer below to the process of combining higher-level knowledge with input information as frame-instantiation, and call the resulting structure an instantiated frame (even though the higher-level knowledge may be a script or scheme, etc.).

The SCA model presented in this paper does not say much about finding knowledge that is relevant to a given situation, but it does say something about frame-instantiation and the use and structure of instantiated frames. In fact, the process of frame-instantiation is a synthesis operation, and the resulting instantiated frame is a CKS. The primary use of the instantiated frame is to provide the information needed by analysis procedures, in the best organized form.

The view of parsing and perceptual processing as synthesists (and hence cousins to frame instantiation) leads to the realization that different instantiations of the same frame can be as different in structure as two distinct sentences, or two distinct arrangements of blocks. Many descriptions of frame instantiation give the impression that all instantiations of a given frame have similar structures, differing primarily in values that fill in slots in the frame. Since slots can be filled by instantiated frames this can indeed lead to a structure as complicated as a parse tree, but not obviously to a network-structured entity like a model for a collection of blocks. Simply expanding slots into subordinate frames is equivalent to top-down parsing. For complicated structures, a combination of top-down and bottom-up approaches may be advisable, and one might usefully apply many of the techniques of natural language parsing, including well-formed substring tables or charts for handling local ambiguities until they are resolved by more global constraints.

The use of synthesists for relaxation techniques and other methods for simultaneous constraint satisfaction extends the range of frame-instantiation operations beyond those commonly considered. Much attention has been given to the problem of determining which frame is to be instantiated, and how to switch from attempting to instantiate one frame to the instantiation of another frame when difficulties arise (see the working paper by S. Fahlman quoted in Minsky 1975). This suggests that choosing the correct frame is difficult, but instantiating it is simple. Once one realizes that frame instantiation may involve complicated simultaneous constraint satisfaction or expansion of structures as complex as natural language parse trees, it is clear that complicated synthesis techniques are needed for frame instantiation.

The literature on frame instantiation, and the structure of frames, gives little idea about the uses to which instantiated frames are put. The SCA model suggests that it is vital to know the questions which are to be answered with the help of the instantiated frame. A synthesist, CKS and analyst are designed together, as interdependent modules, suggesting that both frame instantiation and the structure of instantiated frames must _take into account the set of operations to be performed on the instantiated frame_. The conceptual structure and machine representation of a CKS depend heavily on the expected inputs to the synthesist, on the design of the synthesist, and _especially on the operations to be performed by the analyst_. Even when the inputs and their real-world meaning are fixed (e.g. where the inputs are always line segments corresponding to the edges of blocks) the information in the CKS must depend on the questions to be asked by the analyst (e.g. do they ever refer to position, volume, etc.).

Section IV:

CONCLUSION

The straightforward SCA model presented above is not a complete description of our current concept of the design of an understanding system. A more complete one is hinted at in the section on the LUNAR question-answering system, in which the synthesist is itself viewed as an expert which is broken down according to the SCA model. In general we hold a belief similar to that of Hewitt, in which programs are composed of interacting active procedural elements or ACTORS <Hewitt 73>. We feel that individual ACTORS should each be organized in terms of the SCA model, with separate synthesis, analysis and contingent knowledge structures. The synthesist (or analyst or CKS) of a more complicated ACTOR is built up by the activity of a collection of cooperating ACTORS. The crucial issue then becomes the design of the sociology of ACTORS, that is, the communication and control strategies used to organize the efforts of the independent ACTORS. The SCA model itself is a partial (very partial) answer to this organization problem, since it suggests that the ACTORS composing a complex ACTOR are organized into three separate groups that interact by well-defined means. We believe that another valuable source of ideas for the sociology is the work of R. Kaplan <Kaplan 75> on the GSP natural language system, in which the components that make up a parser are organized into consumer and producer modules that interact with one another.

Given the changing economics of machine architecture, in which it is becoming possible to think of machines with hundreds of interconnected micro-processors, the ability to view AI processes in a way which leads to parallel decompositions may be quite useful. Viewing synthesis as a constraint satisfaction operation leads naturally to implementing it by groups of parallel processes which cooperate to find the best structure to match the given set of constraints. We should point out that the economics that we are approaching is not a new one - it is the economics of genetic systems, the economics of constructing a brain. Given the information needed to define one type of neuron and its pattern of interconnections, it is not substantially more difficult to grow millions or billions of copies. Many of the underlying intuitions that led to the SCA model stem from a study of the interactions of neurons in terms of a model for neuron function suggested by Dr. J.Y. Lettvin <Lettvin, personal communication>. We hope that it will someday be possible to unify these disparate sets of ideas. Possibly some of the ideas arising from extensions of the SCA model to arrays of interacting SCA modules may be useful as way of viewing the operation of the brain.

REFERENCES

Anderson, Robert H. (1968) Syntax-Directed Recognition of Hand-Printed Two-Dimensional Mathematics, Ph.D. Thesis, Deparment of Engineering and Applied Mathematics, Harvard University, 1968.

Brown, A.L. (1974) "Qualitative Knowledge, Causal Reasoning, and the Localization of Failures -- a Proposal for Research", M.I.T. A.I. Lab, Working Paper 61,1974.

Brown,A.L., and G.J. Sussman (1974) "Localization of Failures in Radio Circuits a Study in Causal and Teleological Reasoning",M.I.T. A.I. Lab,Memo 319,1974.

Brown, John Seely, Richard R. Burton and Alan G. Bell (1974) SOPHIE: A Sophisticated Instructional Environment for Teaching Electronic Troubleshooting (An example of AI in CAI), Final Report,B.B.N. Report 279,A.I. Report 12,March,1974.

Goldstein, Ira P. "Understanding Simple Picture Programs." Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Technical Report 294, September 1974.

Greenfeld, N. (1972) "Computer System Support for Data Analysis", REL Project Report No. 4, Ph.D. Thesis, California Institute of Technology, 1972

Groen, and Atkinson (1966) "Models for Optimizing the Learning Process" Psychological Bulletin, Vol. 66, pp. 309-320, 1966.

Hayes, P. (1974) "Some Problems and Non-Problems in Representation Theory", Proceedings of the Summer School on Artificial Intelligence and Simulation of Behavior, Essex 1974

Hewitt, C. (1972) "Description and Theoretical Analysis (Using Schemata) of PLANNER: A language for Proving Theorems and Manipulating Models in a Robot", Ph.D. thesis, MIT Department of Mathematics, 1972

Hewitt, C. (1973) "A Universal Modular ACTOR Formalism for Artificial Intelligence", Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford Research Institute Publications Department, Menlo Park, California

Kaplan, R. (1973) "A General Syntactic Processor" in Natural Language Processing, R. Rustin (ed.), Algorithmic Press, N.Y., N.Y. pp 293-341 (1973)

Martin, W.A. (1971) Determining the Equivalence of

Algebraic Expressions, Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation 1971.

Minsky, M. (1975) "A Framework for Representing Knowledge", in Winston,P. (ed.) The Psychology of Computer Vision, McGraw-Hill, 1975

Newell, A. and H. Simon (1972) Human Problem Solving, Prentice-Hall, 1972.

Newman, W. and R. Sproull (1973) Principles of Interactive Computer Graphics, McGraw-Hill, 1973.

Randall,D.L. (1970) "Formal Methods in the Foundations of Science", Doctoral Dissertation, California Institute of Technology, 1970

Reboh, R. & E. Sacerdoti (1973) "A Preliminary QLISP Manual", Stanford Research Institute Artificial Intelligence Center, Technical Note 81, August 1973

Resnick, Cecily Ann (1975) "Computational Models of Learners for Computer Assisted Learning" Doctoral dissertation, Univeristy of Illinois at Urbana-Chanpaign, 1975.

Roberts, L. [illegible]63) "Machine Perception of Three-dimensional Solids" Technical Report 315, MIT Lincoln Laboratory, May 1963.

Sacerdoti, E.D. (1973) "Planning in a Hierarchy of Abstraction Spaces" Third International Joint Conference on Artificial Intelligence, 1973.

Sussman, G.J., and R.M. Stallman (1975) "Heuristic Techniques in Computer Aided Circuit Analysis",M.I.T. A.I. Lab,Memo 328,1975.

Sussman, G.J. and D.V. McDermott (1972) "Why Conniving is Better than Planning", MIT Artificial Intelligence Laboratory, AI Memo no. 255A

Thompson, F. (1972) "Dynamics of Information", The KEY Reporter, vol. xxxviii, number two, winter 1972-1973, Phi Beta Kappa, Washington, D.C.

Waltz, D. (1975) in Winston, P. (ed.) The Psychology of Computer Vision, McGraw-Hill, 1975.

Weizenbaum, J. (1966) "ELIZA -- A Computer Program for the Study of Natural Language Communication Between Man and Machine" Communications of the ACM, Vol. 9, 1966.

Winograd, T. (1972) "Understanding Natural Language", New York, Academic Press, 1972.

Woods, W. (1973) "Progress in Natural Language Understanding: An Application to Lunar Geology", AFIPS Conference Proc., vol 42, 1973 National Computer Conference and Exposition, pp 441-450