

AD A132 457

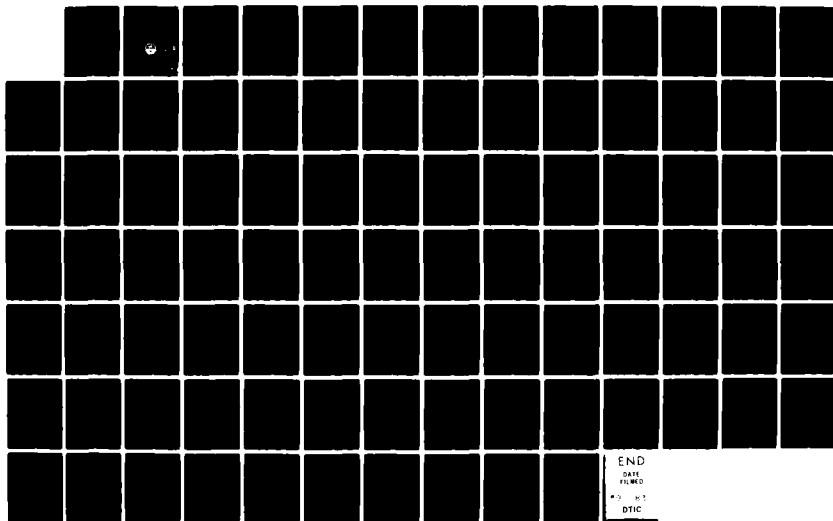
A MULTIMETHOD GENERATIVE CAI (COMPUTER-ASSISTED
INSTRUCTION) SYSTEM FOR R..(U) NAVY PERSONNEL RESEARCH
AND DEVELOPMENT CENTER SAN DIEGO CA J W COTTON ET AL.
MAR 80 NPRDC-TN-80-12

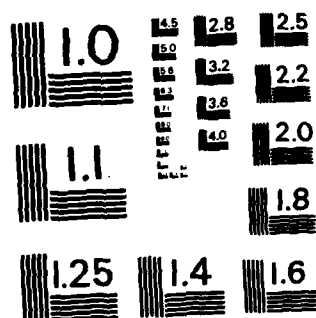
1/1

UNCLASSIFIED

F/G 5/9

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ADA132457

NPRDC-TN-80-12

MARCH 1980

**A MULTIMETHOD GENERATIVE CAI SYSTEM
FOR REMEDIAL MATHEMATICS**



**NAVY PERSONNEL RESEARCH
AND
DEVELOPMENT CENTER
San Diego, California 92152**

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

SEP 14 1983

H



DTC FILE COPY

83 08 05 047

March 1980

A MULTIMETHOD GENERATIVE CAI SYSTEM FOR REMEDIAL MATHEMATICS

John W. Cotton
John P. Gallagher
Marc Hopkins
Sandra P. Marshall

University of California
Santa Barbara, California

Reviewed by
John D. Ford, Jr.

Released by
Richard C. Sorenson

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
<i>by</i>	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
<i>A</i>	

DTIC
COPY
INSTRUMENT

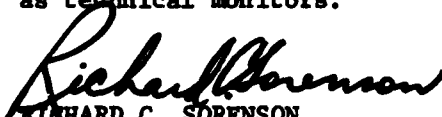
THIS TECHNICAL NOTE IS A WORKING DOCUMENT AND MAY NOT
BE CITED OR QUOTED WITHOUT PRIOR APPROVAL OF THE DIRECTOR
OF PROGRAMS, NAVY PERSONNEL RESEARCH AND DEVELOPMENT CENTER,
SAN DIEGO, CA. 92152, TELEPHONE: AUTOVON 933-2232,
COMMERCIAL (714) 225-2231.

Navy Personnel Research and Development Center
San Diego, California 92152

FOREWORD

This research and development was conducted in response to Navy Decision Coordinating Paper, Education and Training Development (NDCP-20108-PN), under subproject PN.32, Advanced Computer-based System for Instructional Dialogues, and under the sponsorship of the then Director of Naval Education and Training (OP-99). The objective of the subproject is to develop and evaluate advanced techniques of individualized instruction. The present study is concerned with the development of lesson materials that can be used to evaluate the effectiveness of computer-generated instruction in comparison to conventional frame-oriented instruction.

The work was performed under Contract N00123-77-C-0087 by the University of California, Santa Barbara. Dr. J. D. Fletcher and Mr. John Wolfe served as technical monitors.


RICHARD C. SORENSON
Director of Programs

SUMMARY

Problem

The Navy needs modern and efficient instructional systems to meet required levels of operational readiness. Recent work in this area has relied heavily on computer-assisted instruction (CAI). Because there are several methods of CAI, some of which are still under development or have received little empirical assessment, it is important to determine which methods are most efficient with certain curricula and student groups.

Purpose

The purpose of this effort was to develop a CAI system for use in assessing the relative efficiencies of four modes of teaching remedial arithmetic and algebra. The four modes result from orthogonal variation of two binary dimensions: (1) ad hoc, frame-oriented (AFO) instruction versus information-structure-oriented (ISO) instruction, and (2) didactic, or traditional, instruction, versus affective instruction.

Approach

Following a literature review regarding generative CAI systems, a computer-assisted tutorial system for remedial mathematics was written in modified LISP programming language and was installed on the PDP 11/45 computer in the Computer Systems Laboratory at the University of California, Santa Barbara. The new system was then compared to five generative systems and one problem-solving system to determine its suitability for assessing the four modes for teaching remedial mathematics to navy personnel.

Results

A computer-assisted instruction system in remedial arithmetic and algebra is almost complete. It provides (1) training for students on the use of a cathode ray tube plus typewriter keyboard terminal as a medium of instruction, (2) a series of 20-minute instructional sessions using one or more of five modules (negative numbers, factors and prime numbers, fractions, inequalities, and simple linear equations), and (3) a pair of on-line tests. Training sessions and tests alike employ randomly generated problem types and numerical values. Paper-and-pencil pre- and posttests of mathematical performance and attitudes toward mathematics and computers have been prepared. Modules have two or three levels of numerical problems plus word and story problems.

The AFO condition requires that students reach a correct performance criterion on each module in the order listed above. This criterion includes the meeting of subcriteria in order on the first two levels of difficulty. Separate criteria for Level 3 (if present), word, and story problems are used; but items from these sections of the curriculum are intermingled during training. The ISO condition allows students to ask for a variety of special aids not available to AFO students. In addition, ISO students need not meet specific performance criteria or study modules or levels within modules in a prearranged order.

The didactic condition, which is combined with either AFO or ISO instruction to form two of the four instructional options, gives instructions in a straightforward manner with no special emphasis on the emotional reactions of the students. The affective condition, also combined with either AFO or ISO instruction, personalizes the instruction by calling the student by first name and referring to the computer by a name selected by the student. Students are asked how they feel at the beginning of most sessions. Short dialogues, puzzle sessions, or off-line conversations between two affective students in the same experimental condition are common procedures before regular problem sessions begin.

A comparison with the earlier systems shows that the present system lends itself readily to the gathering of empirical data on its teaching effectiveness. Among its strengths are its diverse curriculum, the complex information structure of its teaching modules, its considerable flexibility in the kinds of student requests it can accept, and its wide range of possible dialogue.

Recommendations

The Navy should consider the use of mini- or super-minicomputers (e.g., PDP 11/45, PDP 11/70, VAX-11/780) in simple generative CAI systems as a means of reducing permanent memory demands and operating overhead.

CONTENTS

	Page
INTRODUCTION.	1
Problem	1
Purpose	1
Background.	1
UCSB COMPUTER-ASSISTED INSTRUCTIONAL SYSTEM	3
Design and Implementation	3
Curriculum.	4
System Operation.	6
ISO Condition	6
APO Condition	12
Test Items and Test Construction.	14
Pretest and Posttest with Domain-defined Item Selection	14
Testing During Training	15
The Off-line Reference Text	15
Design of Experiment Based on Present CAI System.	16
Computer-specific Features.	18
CONCLUSIONS	19
RECOMMENDATIONS	25
REFERENCES.	27
REFERENCE NOTES	33
APPENDIX A--REVIEW OF RELEVANT LITERATURE	A-0
APPENDIX B--STRUCTURAL UNITS OF THE UCSB COMPUTER-ASSISTED INSTRUCTIONAL SYSTEM.	B-0
APPENDIX C--DESCRIPTION OF CURRICULUM MODULES WITH SAMPLE PROBLEMS.	C-0
APPENDIX D--FIRST SESSION PROCEDURES.	D-0
APPENDIX E--ANALYSIS OF HINT PROGRAM STRUCTURE FOR THE FRACTIONS MODULE	E-0
APPENDIX F--SOLVE PROGRAM STRUCTURE FOR THE FRACTIONS MODULE.	F-0
APPENDIX G--DETAILS OF DESIGN OF CONFLUENT (AFFECTIVE) PROCEDURE.	G-0

LIST OF TABLES

	Page
1. Criteria to be Met by AFO Students on Successive Curriculum Modules	13
2. Test and Training Sessions and Curriculum Topics (Ordering of Curriculum and Estimated Times Apply Primarily to AFO Students) . . .	17
3. Comparison of the Present System to Other Information-Structure-Oriented or Generative CAI (or Problem-solving) Systems	20

INTRODUCTION

Problem

The Navy needs modern and efficient instructional systems to meet required levels of operational readiness. Recent work in this area relies heavily on computer-assisted instruction (CAI). Because there are several methods of CAI, some of which are still under development and have received little empirical assessment, it is important to determine which methods are most efficient with certain curricula and student groups.

Purpose

The purpose of this effort was to develop a multimode computer-assisted instruction (CAI) system for use in assessing the relative efficiencies of teaching remedial arithmetic and algebra. These modes result from orthogonal variation of two binary dimensions.

One dimension is information-structure-oriented (ISO) instruction versus ad hoc frame-oriented (AFO) instruction. AFO instruction may be considered traditional CAI, with little control exercised by the instructor or curriculum developer. In contrast, ISO instruction permits a somewhat unstructured discourse between computer and student, and permits student initiative in the selection of topics and methods of learning.

The second dimension is affective versus didactic, or traditional, instruction. Both of these modes emphasize cognitive elements. Affective instruction, however, attempts to facilitate learning (and, possibly, personality development) by providing an instructional experience that explicitly considers attitudes and emotions.

Background

ISO instruction has its origins in computer science, where artificial intelligence specialists such as Carbonell (1970b) and Brown and Burton (1978) have made important innovations. For example, they have developed systems that permit intelligent dialogue between computer and student as well as a good deal of student initiative in selecting instructional topics and methods.

Both AFO and ISO instruction have substantial merit but, because of the greater information base and consequent flexibility of ISO systems, they may replace AFO instruction in a few years. Before advocating primary use of either kind of instruction, psychologists need to know something about their relative efficiencies and palatabilities (to students), as well as how that efficiency and palatability relate to the features that differentiate them. As noted previously, this effort deals with a working CAI system that can produce either AFO or ISO instruction, thus permitting the necessary experimentation.

Brown (1971) has called for the merging of cognitive and affective emphases in instruction, terming this merger "confluent" education. In the present project, it is hypothesized that holding somewhat informal conversations with a computer may help

students to develop efficient learning strategies, possibly by preventing fixation on inappropriate activity. Similarly, encouraging students to express their feelings about mathematics and about computer-assisted instruction may be beneficial. The other side of this instructional dimension, didactic teaching, is defined by default—it is instruction focused upon explicit intellectual tasks to be mastered, without emphasis upon emotional factors.

A literature review providing further background for this research is presented in Appendix A.

UCSB COMPUTER-ASSISTED INSTRUCTIONAL SYSTEM

Design and Implementation

The present CAI system, which was developed in the past year, can provide on-line instruction for the following aspects of arithmetic and algebra: negative numbers, factors and prime numbers, fractions, inequalities, and simple (linear) equations.

The system is programmed in a modified version of Harvard's UNIX LISP (obtained in part from the Navy Personnel Research and Development Center). It has an expanded list of hardwired functions plus LINEREAD, SLEEP, ASCII, and CHECKF (like the LISP 1.6 LOOKUP at the University of California, Irvine (UCI) as well as a new output-formatting function written in LISP). It runs under the UNIX operating system on the Computer Systems Laboratory and Computer Center PDP 11/45 computers at the University of California, Santa Barbara (UCSB).

In the problem-generating, responding, and feedback cycle, instructional items are generated by programs specific to the modules in which they are contained; for example, a fraction problem would be generated by the fractions module. Numerical fraction problems may be from one of three levels, as in the following examples:

- Level 1: $1/2 + 2/5$.
Level 2: $1/2 \times (1/6 + 1/3)$ or $(1/6 + 1/3) \times 1/2$.
Level 3: $(1/6 \times 1/3) - (1/8 \times 1/3)$.

Two other kinds of problems exist for most modules: word problems and story problems. Word problems are almost identical to numerical or algebraic problems but use more words to ask the same questions. For example, a Level 1 fraction numerical problem might ask:

$$\underline{1/3} \times \underline{3/4} = ?$$

(with $1/3$ being the first term, $\times$ being the operator, and $3/4$ being the second term), whereas the corresponding word problem might ask:

What is the product of $1/3$ and $3/4$?

Story problems involve real-world quantities and events; for example, a story problem for fractions might say, "The gas gauge on your car indicates that the tank is $1/4$ full. If a full tank holds 20 gallons, how many gallons are in the tank?"

(The remainder of this section will be easier to follow if the reader first examines Appendix B to learn the system's principal structural units.)

Once the structure of a level of numerical problems has been established, it is possible to generate quasi-random instantiations of that class of problems. The top-level monitor and curriculum driver for the system specify what type of problem (numerical, word, or story) is to be presented, and a specific problem is then generated. With the most routine response possible, the student uses scratch paper as necessary to develop an answer, and then types that answer into the computer via a Lear ADM-3 terminal keyboard. The monitor routes this response to an English interpreter to be sure it is an answer rather than a

request for help, translates it into list notation, and refers it to the solution analyzer. The solution analyzer codes the responses as correct, incorrect, or partially correct; and returns them to the monitor, which arranges for appropriate feedback to the student and for storage of the coded response. Depending upon the specific teaching method to be employed, the monitor and curriculum driver next check the student's history, decide whether a member of the same or a new class of items is to be presented next, and have the problem generator devise an appropriate new item.

This basic cycle may be interrupted or elaborated on by certain kinds of student requests. In all teaching conditions, students may end the session early, request a general set of help instructions, request the answer to the current problem, or request display of the solution to the current problem. In the ISO conditions, students may also request a new module or submodule, request the answer to a calculation that may be of help in the larger problem, or request hints as to how to proceed with the current problem. Data on requests for new modules or submodules and on requests for hints are stored in the student histories. Once a special request has been satisfied, instruction continues in a natural sequence, except that early logout using QUIT or BYE leads to nothing new until the next session.

Curriculum

The curriculum of UCSB's CAI system is intended to include higher-level arithmetic processes and lower-level algebra. Suppes, Goldberg, Kanz, Searle, and Stauffer (1971) reported that the types of items used in such curriculum have been rated as low as Grade 1.5. For example, the item

$$9 - \underline{\quad} = 4$$

is a problem of first grade arithmetic, whereas the item

$$(9 - X = 4. \text{ Solve for } X.)$$

is comparable to algebraic equations used in the UCSB remedial mathematics course (Math IX, Individualized Instruction in College Algebra), making a definitive statement of level of problem somewhat difficult. Suppes et al. based their grade level ratings on the grade level of the textbook in which problems were found, the performance of CAI students on those problems, the advice of teachers, and the experience of the research staff.

The ten modules that will be included in the UCSB system are listed below:

1. Negative numbers
2. Factors and Prime Numbers
3. Decimals
4. Absolute Values
5. Fractions
6. Exponents
7. Square Roots
8. Inequalities
9. Simple Linear Equations
10. Polynomials

In the present report, no attempt is made to state grade levels of item classes within these modules. Rather, the modules are listed in an order that ensures

that inferred level prerequisites needed in later modules have been covered in earlier ones. Unfortunately, only five modules--those on negative numbers, factors and prime numbers, fractions, inequalities, and simple linear equations--were completed or nearly so at the time of this report. The principal reasons for the inability to complete all modules were repeated computer system breakdowns, and garbage collection problems. The latter required (1) the institution of a new function (SUPERG) to ensure that LISP atoms are erased when commands such as PUTD FILENAME NIL are used to swap one file out before loading a new one and (2) modification of LISP to be sure that the garbage from executing various output functions is routinely eliminated.

Appendix C, which provides sample problems for the 10 modules, shows that problems of more than one level of difficulty are commonly presented within one module, and that both word and story problems are frequently employed. Word problems have little real-life reference and use words or groups of words such as "the product of" in place of "x" in restating problems. Story problems typically involve real-life situations and require knowledge of concepts such as miles per hour. The flavor of the curriculum is conveyed by the following examples of word and story problems from the fractions curriculum. (Underlined terms are variables that change when new items are generated, subject to internal consistency requirements, and parenthesized items are substantive alternatives to underlined terms. Possible variations in numerical entries are not shown, but random generation of such numbers is employed in setting up calculations to be performed.)

Word Items:

1. Which one is not equal to the others:
 $\frac{1}{6}$, $\frac{4}{5}$, or $\frac{4}{24}$?
2. Which is larger: $\frac{1}{4}$ or $\frac{2}{3}$?
(smaller)
3. Convert $\frac{3}{2}$ to a mixed number
(an improper fraction)
(its reduced form)
4. What is the product of $-\frac{2}{3}$ and $-\frac{5}{6}$?
(difference between) (sum of)
5. What is the smallest common denominator of $\frac{2}{3}$ and $\frac{1}{5}$?

Story Items:

1. Isabel worked for $2\frac{1}{4}$ hours on Friday and $\frac{2}{3}$ hours on Saturday. How many hours did she work?
2. A motorcycle travels 60 m.p.h. How many miles will it travel in $\frac{3}{4}$ hour?
3. Rachel had a piece of wire $3\frac{2}{3}$ feet long. She cut $\frac{5}{6}$ feet from the piece to use in hanging a picture. How many feet of wire were left?

4. On a hike Stanley walked $1\frac{1}{6}$ miles the first hour. At this rate, how many miles can he walk in $2\frac{3}{5}$ hours?

Examples of such items for the other modules are included in Appendix C.

Problems within a module are normally generated at random, subject to being at the desired level (e.g., Levels 1A and 1B are considered variants on the same level). Typically, students begin with Level 1 of a module, progress to Level 2 problems intermixed with word and story problems, and finish with Level 3 problems intermixed with word and story problems of the same difficulty as before.

System Operation

ISO Condition

Since most features of the CAI system are present under the ISO condition, ISO system operation is described first (Figure 1). In addition to the actual instruction, the system includes a series of tests. The first and last days of the instructional sessions are testing days, using paper-and-pencil tests consisting of items generated randomly from curriculum modules and submodules. Measures of attitude toward mathematics and toward the teaching method are also included. On-line tests of mathematics achievement are also conducted on the session immediately after the 6th and 12th sessions of actual instruction. These tests also use random generation of items and cover the entire curriculum; they do not include attitudinal items.

The English Interpreter. A parser was originally developed for this system but proved unnecessarily complex, given the degree of natural language usage required. Therefore, it was modified into an English interpreter for handling a moderate variety of student input. This interpreter first changes input from the terminal into list notation where required, as in the case of requests for the computer to perform a preliminary calculation with the COMPUTE routine. Then it checks for alternate forms for basic input words, accepting, for example, "y" and "n" and other variants in place of "yes" and "no." The interpreter also has a misspelling routine that accepts alternate spellings of words such as "answer" and "compute." Then it performs a keyword check to recognize student calls for routines such as COMPUTE, and routes such calls to the appropriate utilities.

Answers to mathematical questions are routed to the solution analyzer for scoring and feedback. Individual match programs exist in the solution analyzer for each mathematics module. In most modules an equivalence program exists for finding alternate acceptable answers, or answers that are basically correct but need minor improvement such as reducing a fraction to simplest terms. Answers to conversational questions in the confluent condition are evaluated to some degree on the basis of keyword matching. The results of this matching may then affect the direction of the conversation.

External Bookkeeping, Signin, Signout, etc. When called by an experimenter, MAKFIL permits the establishment of files for one or more students, calling for name, date of birth, and teaching conditions (ISO or AFO and didactic or confluent). An actual teaching session can begin when the experimenter has logged into the computer, called LISP, and loaded a start-up program. The executive program, MONITOR, activates SIGNIN, which causes the terminal to display:

Please type (hello).

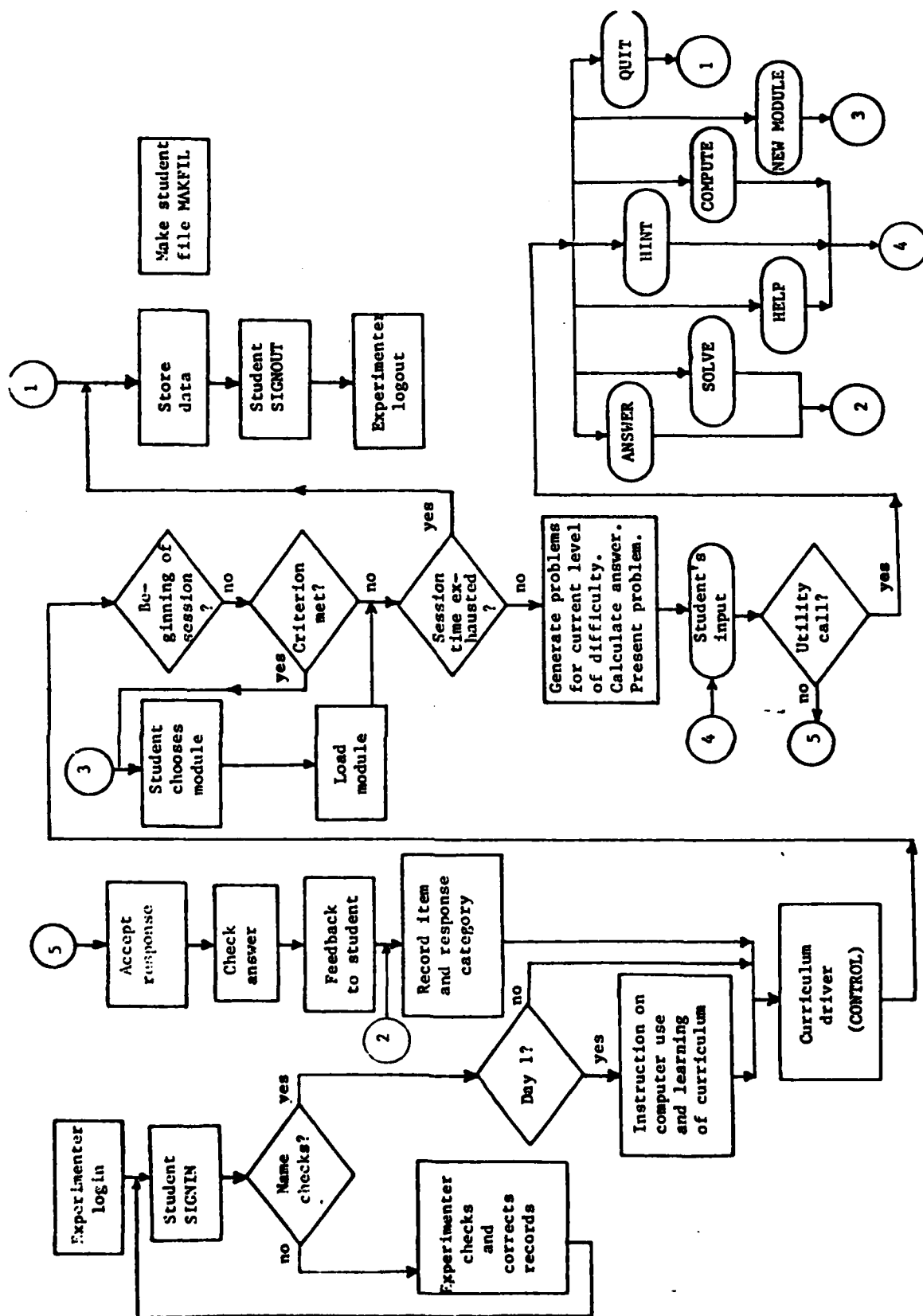


Figure 1. Flowchart of ISO system operation.

MONITOR is involved in most aspects of the system, including bookkeeping, signin, and signout. Once the student has typed "(hello)," SIGNIN asks for the student's name and gives a welcoming message. If the student's name is not recognized, SIGNIN asks for it to be repeated, checking the first name and then the last. If still unrecognized, the student is not allowed to continue until the experimenter has discovered the reason for the discrepancy. The filename has the form johndoe.

SIGNOUT simply ends a session with an appropriate closing message plus a complete logging out from the computer. At the end of each problem, MONITOR asks if another problem is desired. If not, SIGNOUT is activated. Also, a timing program, TIMER, activates SIGNOUT at the end of the time allotted for the session.

Item, Response, Feedback Cycle. When it is time to present items, MONITOR calls CONTROL to decide whether to continue with an earlier category of items or to begin a new module or submodule. CONTROL checks the student file and current day's data to see the status of the student's progress. In the ISO condition after Day 1, MONITOR asks if the student wants to continue with the previous day's module. The student can stay on a chosen module throughout the experiment, but it may be that a criterion of a certain number of successive correct responses on a module has been reached, representing substantial over-learning. In that case, CONTROL notifies the student that the module has been mastered and that a move to new material should be considered.

Once the current module and level of items to be presented are selected, MONITOR loads that module. The curriculum generator then generates an appropriate problem, presents it on the terminal screen, and calculates its answer. The students either answer the question and receive feedback (right or wrong), or call a utility program (ANSWER, SOLVE, HELP, HINT, COMPUTE, LEVEL, NEW MODULE, SKIP, or QUIT). If an answer was given, then Unit 5 (see Appendix A) codes the item and response, keeping a full record of item characteristics and all responses to each item.

A response is stored as correct, correct except for needed simplification, or incorrect. If simplification is needed, the student is told the type of error and is given an opportunity to correct the original answer. After an incorrect response, a student is asked some variation of, "Do you want to try this problem again?" Therefore, a student may offer several answers before moving on to a new problem.

If the criterion has been met, the student is asked about changing to new material. When a different module is selected, its set of programs is loaded and the problem generator for that module is called. Otherwise, the current problem generator is called for another problem from the current module; and the item, response, feedback cycle is repeated.

Before each new cycle begins, there is a check to see if the scheduled time for the session has been exhausted. If so, then the TIMER program automatically instigates the SIGNOUT process. The session's data are stored at this time. SIGNIN and SIGNOUT can both be student activities, with the standard login process reserved for the experimenter. The reason for having the experimenter do login and logout is to reduce the complexity of student activities.

Student-activated Routines (Utility Calls). The routines are described in the following paragraphs.

1. ANSWER

Students are normally told only whether their answers are right, right with minor modifications, or wrong. Before responding to a problem, however, students can type "answer," and Unit 6 (see Appendix B) will arrange for the correct answer to be displayed. Such a response is recorded as "Give up," and MONITOR arranges for another item to be presented in the usual fashion. If the "answer" request is made after an incorrect answer and after the student has asked to try the problem again, then "Give up" is recorded as a further response following an error and MONITOR arranges for another item to be presented.

2. COMPUTE

This routine (also housed in Unit 6) permits partial solution of a problem before the student offers a full answer. Suppose a problem were:

$$3 \times (1/4 + 2/3) = .$$

The student could say:

*compute $1/4 + 2/3$

and watch the screen for the resulting partial answer before multiplying by 3. Should the student type the complete problem of

*compute $3 \times (1/4 + 2/3)$,

the computer will refuse to do the computation. This aid helps to reduce the purely mechanical demands of the curriculum and conforms with the recent emphasis upon the use of hand-held calculators in elementary mathematics courses.

3. HELP

This routine from Unit 6 is essentially an introduction to other student aids. When called, it will display the same information about recommended study procedures and about ANSWER, COMPUTE, HELP, HINT, LEVEL, NEW MODULE, SKIP, and SOLVE as were given in the first-day instructions. Following a "help" command, the initiative remains with the student. Typically, a student will follow that command by calling for one of the aids just mentioned.

4. HINT

The "hint" command is made after seeing a problem but before attempting an answer. Progressively more complete hints are possible by calling HINT repeatedly, if early hints are inadequate. Appendix E displays a story problem from the Fractions module, followed by eight calls of the "hint" command, with seven successive commands being honored and the eighth command leading to a report that no more hints are available for this problem. The HINT routines are stored in Unit 6.

The HINT routines for different modules are separate but similar in design and operation. This routine for fractions has the formalized goal structure given below; other modules use intuitive approximations or simulations

of that structure, or somewhat more intelligent, conversational routines for generating hints. For the fractions problem of Appendix E,

$$1/2 \div (2/3 + 1-1/6) = ,$$

(where / is presented in lieu of ÷ for computer-dictated reasons), HINT checks the format of the problem to see that the only top-level goals required are given as follows:

TOPLIST = (NEST LAST2).

TOPLIST defines the general plan by specifying that a solution requires the satisfaction of two basic goals: First, the nested expression, $(2/3 + 1-1/6)$, must be simplified by NEST. Then LAST2 must be satisfied; this is the combining of the two remaining terms to obtain the final solution. A term in a fraction is defined as follows:

Term : : = digit+ or digit* SP digit+/digit+.

That is, a term is either an integer of any nonzero length, the ratio of two integers of any nonzero length, or an integer of any length plus the ratio of two integers of any nonzero length: SP is used here to denote a space. (In Backus Normal Form grammars, "+" is used as a symbol to permit one or more than one of the elements it follows, and "*" functions in the same way but allows the nonoccurrence of the symbol it follows.)

The second line of the goal structure for our example problem is given by SOLIST:

SOLIST = ((NEST CONVERT 2 2 LCM ADD RETAIN))
(LAST2 INVERT MULTIPLY)).

In the LISP terminology, the CAR of SOLIST is (NEST CONVERT 2 2 LCM ADD RETAIN), giving NEST and its subgoals. The CADR of SOLIST is (LAST2 INVERT MULTIPLY), giving LAST2 and its subgoals. CONVERT 2 2 is the subgoal of converting to improper form (meaning of the first digit 2) the second term (meaning of the second digit 2) of the problem. A more complete description of the HINT routine's goal structure appears in Appendix E.

After receiving all hints requested or available, the student gives an answer to the problem and receives feedback. Unit 5 then records the number of hints requested and the correctness or incorrectness of the answer.

5. LEVEL

When a student types "level," MONITOR arranges a display of the options "easy," "hard," and "mixed" difficulty so that the student can choose the difficulty of items to be presented. Level 1 items are easy items; Levels 2 and 3 and word or story problems are hard items; and a 50:50 random mixture of easy and hard items constitutes mixed difficulty.

6. NEW MODULE

When a student responds "module," the monitor has the terminal display the list of all modules. The student is then given the choice of a new module. While it is possible to choose to review a previously studied module, the usual choice will probably be for a new one. NEW MODULE is also part of Unit 6.

7. QUIT

At any time in the session, a student can type "quit" and will then activate the SIGNOUT routine.

8. SKIP

A student who finds a problem too hard (or otherwise undesirable) may simply type "skip." In that case, a new problem from the same module is presented and a "skip" response is recorded for the previous problem.

9. SOLVE

Either at the time of first display of a problem, or just after feedback about one's own answer to that problem, the student can type "solve." Then Unit 6 arranges for the terminal to display all steps in the solution of the problem. A request for solution before attempting the problem is scored by Unit 5 as "give up," and a request for solution after an attempt is scored as "give up" on a second trial; the first response is scored on the basis of its own correctness.

The operation of SOLVE is closely related to that of HINT. SOLVE uses a production system (Newell, 1973) to set goals and subgoals for the solution of fraction problems and most other modules. In other parts of the present system, SOLVE often behaves in a rote fashion because it is easier to program answers to specific classes of problems than to develop a systematic way of generating solutions. SOLVE has not yet been developed for the inequalities and negative numbers modules.

Appendix F displays the first 26 productions required for solving fraction problems. (Actual programming conformed to this system but was not written in production system notation.)

Recordkeeping. Recordkeeping and curriculum control in either AFO or ISO instruction requires categorization of item and response types. Whenever an item is presented to a student, the subsequent response is characterized as falling into one of the following categories:

- 0 = Correct.
- 1 = Correct except for sign or a need to be further simplified.
- 2 = Incorrect.
- 3 = Skip.

A = Answer requested.
 C = Compute command given.
 F = Hint command given after hints exhausted.
 G = Gave up (can only be a second response to a problem).
 H = Hint command given.
 L = Level command given.
 M = Module command given.
 P = Help command given.
 Q = Quit command given.
 S = Detailed solution requested.

More than one response category may be scored on an occurrence of a problem because of student use of various aids to instruction. The rules for these options are as follows: After seeing a problem presented, one can begin with either A, C, H, L, M, P, S, O, 1, 2, or 3. The occurrence of A, S, O, or 3 ends the trial. A 1 or a 2 can lead to A, G, H, or S or may end a trial. An F can lead to A, C, G, S, O, 1, or 2, but not to any later path that will ever contain an H (since F means there are no more hints available).

This discussion implies that the computer does not store specific student responses. The answers stored by the rules just given will permit considerable analysis of the quality of student answers and of the frequency of use of student-initiated utilities such as ANSWER and SOLVE.

The response record is stored as the value of a variable called RCAT, together with a characterization of the item, the variable called QCAT. The QCAT has as its value the level and type of item, the curriculum module involved, and other relevant information. For example, the fractions module has six elements on the QCAT tag for an item. Those elements specify:

1. That fractions are involved.
2. The level and sublevel of a numerical problem (if present).
3. Whether or not mixed fractions are involved.
4. Whether or not negative fractions are involved.
5. Whether the item is a numerical, word, or story problem.
6. Whether or not the item is an immediate repetition of the previous one (e.g., because of a partially correct answer leading to another try at the problem).

AFO Condition

Differences from ISO System. The present ISO system permits the student to select specific modules at the beginning of a session (or later) by use of the MODULE command. This option is not available to students in the AFO condition with its fixed order of modules. Whereas most AFO systems have a relatively small number of items, the generative feature of the present system leads to a much larger pool from which training items are selected. This should increase the generality of teaching effects in the AFO conditions somewhat; further research on the effects of this change may be desirable.

The current AFO system not only prevents student selection of specific modules to study; it also prevents the use of the "compute," "level," and "hint" commands and the full SOLVE routine. None of these aids is frequently used in other AFO systems. In place of the original SOLVE routine, the AFO system uses a modified SOLVE for a sample problem rather than for the current problem. Thus the student must infer how the model solution method can be modified for use with the specific problem presented. The AFO condition provides for exclusion from the HELP routine any mention of the COMPUTE, LEVEL, and HINT routines. These routines are rendered inoperative by a series of flags activated by the characterization of experimental condition on the student's data file at the time of signing in for a session.

Use of Performance Criteria in Item and Topic Selection. To complete a curriculum module, an AFO student must have a performance criterion on each part of that module. Also, criteria (Table 1) must be met before moving from one section of a module to another. One cannot receive instruction in the AFO condition for a higher-level part of the module until criterion has been reached on all lower-level parts. Word and story problems are somewhat independent of the numerical problems with specific level assignments: Typically, word and story problems are not presented until criterion has been reached at Level 1. Thereafter, word and story problems, if used for that module, are randomly interspersed with the numerical problems. The performance criterion for word problems or for story problems may be reached either before or after the criterion for Level 2 or Level 3 numerical problems.

Table 1

Criteria to be Met by AFO Students on Successive Curriculum Modules

Curriculum Module	Numerical or Algebraic Problems			Word Problems	Story Problems
	Level 1	Level 2	Level 3		
Negative Numbers	2	2	-	2	2
Factors and Prime Numbers	2	2	-	2	2
Fractions	2	2	2	4	2
Inequalities	2	3	3	3	2
Simple Linear Equations	2	3	3	2	2

Test Items and Test Construction

Pretest and Posttest with Domain-defined Item Selection

Following Glaser and Nitko (1971) and Hively, Patterson, and Page (1968), the testing program for this system is tied to the curriculum, employing domain-defined item selection. The domain from which test items can be selected may be viewed as the set of all items that could be generated from the existing implementation of UCSB's CAI system. Definition of the domain used here seems to combine aspects of what Shoemaker (1975) calls the item-banking and item-forms approaches. In a strict use of the item-banking method, a teaching objective could be tested by a variety of items that might look quite different. In the item-forms method, a teaching objective would be so specific that only a few of the problems would be subject to variation.

Variations in basic forms of word or story problems like those on pages 5 and 6 are compatible with the item-banking approach; variations in names, numerical values, and verbal quantifiers within one of those basic forms are compatible with the item-forms approach. Some test constructors using these approaches might have explicit statements of objectives or justifications of those objectives, rather than letting the items speak for their own instructional validity as is done here. Note that the validity of the items used in tests stems from instructional validity plus an assumption of random sampling from the total domain or within subdomains.

The pretest on Session 0 and the posttest on Session 19 have each been designed to last 45 minutes, with an average of about 1 minute allowed for each of 46 items drawn at random from the 4 or 5 subdomains for each of the 5 curriculum modules of Table 1. These subdomains are the three levels of numerical or abstract problems plus the word and story problems associated with each module (see Appendix B). This gives two items from each of the 23 subdomains being taught in this experiment. Items on the two forms of the test are matched in the sense that Item 1 is a Level 1 negative numbers problem, for example, on both Forms A and B. Because of variations of format for story or word problems of a given module and because there are sometimes subcategories within levels of the other modules, items are not necessarily matched in the item-forms sense. A 5-minute attitude test dealing with attitudes toward both mathematics and computer-assisted instruction precedes each pretest and posttest. The development of the pretest and posttest forms is nearly complete; the attitude test remains to be developed.

Pearlman (1977, p. 99 and p. 145) reported that high correlations were sometimes found between scores on pairs of items in arithmetic tests for addition, subtraction, multiplication, and division, with r values as high as .75 for multiplication items. Accordingly, high correlations may be expected between corresponding pairs of items of Forms A and B of the present test. The correlations may be reduced when content is not matched in an item-form test. Nonetheless, the classical effort of increased reliability with increased numbers of items is expected to yield a substantial parallel-forms reliability for the total tests.

In either Form A or Form B, all items from one curriculum module appear before those from another. The use of Forms A and B is balanced within the four

teaching conditions and between pretests and posttests. For example, 10 students in the ISO didactic group will receive Form A for their pretest and Form B for their posttest while the other 10 students in that group will receive Form B for their pretest and Form A for their posttest. This will permit comparison of average difficulty of the two forms and an unbiased measure of mean improvement from pretest to posttest. Forms A and B are paper-and-pencil tests administered without the computer.

Testing During Training

During 16 of the 18 days that the computer will be used in this experiment, there will be continuous instruction and feedback to the students. This permits measurement of performance on specific classes of items and measurement of the stage of curriculum reached (for AFO students only). However, overall learning measures during the instructional phase require special procedures.

On Sessions 7 and 13, an on-line test will be given covering aspects of all modules. Since the computer sessions are scheduled for no more than 15 minutes, an approximately 15-minute test with 15 items will be presented on each of these sessions. For each of the five modules taught, one Level 1 and one Level 2 numerical or abstract symbol problem and one story problem will be generated randomly, with modules represented in the same order as on pretests and posttests. Alternate forms are not used for these tests. Rather, a new test will be developed at random for each student on Session 6 and again on Session 12.

The Off-line Reference Text

A reference booklet for remedial arithmetic and algebra is to be kept beside each computer terminal during training (but not testing) sessions. This booklet provides material to aid the student in learning new material or in reviewing old topics. The five modules of the curriculum are covered in the booklet in the same order as in Table 1. The printed text will consist primarily of sample problems and SOLVE routine outputs like that of Appendix F. It is expected that students needing the help given by the text will transfer techniques from these sample problems to the problems facing them on the terminals.

One problem from each level and category of Table 1 is to be presented in the reference booklet. Problems are ordered by level as well as by module. A minimum of other material, such as topic headings, is also included. The booklets will first be developed, stored, and formulated on a PDP 11/45 computer, using the UNIX NROFF system. Then they will be printed on 8-1/2 by 11-inch white paper by a Printronix lineprinter.

Design of Experiment Based on Present CAI System

This section describes the design of an experiment that is to be conducted as part of the current research project.

A 2-by-2 design will differentiate 4 groups of 20 UCSB students. Two of these groups will receive training with the ISO system and two with the AFO system. One ISO group and one AFO group will receive confluent instruction, while the other ISO group and AFO group will receive didactic instruction.

Each subject will be assigned randomly to one of four experimental conditions. A computer file will be created for each student, initially consisting of the student's name, experimental condition, and date of birth. During the first computer session after the pretest, the student will be taught how to type messages on the Lear terminal and how to use the special aids provided within the appropriate teaching condition. This first lesson will also include a description of the way course material will be presented and advice on how to learn it. Special informal conversation is generated in the confluent condition even though the system does not permit true understanding of natural language.

Students used in the experiment will be paid volunteers recruited from courses and campus centers (such as the Educational Opportunity Program units for Chicanos and Blacks). Students currently taking a course in the UCSB Department of Mathematics are not allowed to serve in the experiment. Students using mathematics in other courses, such as psychological statistics and introductory physics, are allowed to be subjects if they believe themselves to need remedial assistance of the sort offered in this experiment. These courses do not directly cover the curriculum material of this experiment.

Table 2 presents a day-by-day list of sessions and activities for the experiment. The testing days are not subject to change for any students, and the ordering of the curriculum units is not subject to change for AFO students. However, AFO students will not necessarily spend the same proportions of instructional time on the different units. Some increase in instructional time per session may be required because of slow computer operation.

Table 2
Test and Training Sessions and Curriculum Topics
(Ordering of Curriculum and Estimated Times
Apply Primarily to AFO Students.)

Session Number	Time Alloted (Minutes)	Activities
0	51	Paper-and-pencil pretest (46 min. math; 5 min. attitude).
1	12	Computer training (7 min.), Negative numbers (5 min.).
2	15	Confluent students; two-person dialog (9 min.), negative numbers (6 min.); didactic students: negative numbers (15 min.).
3	10	Negative numbers.
4	10	Factors and prime numbers.
5	10	Factors and prime numbers.
6	10	Factors and prime numbers.
7	15	Test.
8	10	Fractions.
9	10	Fractions.
10	10	Fractions.
11	10	Inequalities.
12	10	Inequalities.
13	15	Test.
14	10	Simple linear equations.
15	10	Simple linear equations.
16	10	Simple linear equations.
17	10	Simple linear equations.
18	10	Simple linear equations.
19	51	Paper-and-pencil pretest (46 min. math; 5 min. attitude).

Computer-specific Features

The CAI system described herein is normally implemented on the UCSB Computer Systems Laboratory's PDP 11/45 computer. Much of the system was originally programmed on the PDP 10 computer at UCI using the LISP 1.6 LOOKUP language. When the addition of the Harvard UNIX system to the UCSB computers made their use feasible for this project (as well as essential in view of troublesome telephone line noise from UCI), project personnel developed a better LISP interpreter than had previously been available at UCSB (See page 3).

Next, the system material from the UCI PDP 10 was translated and reprogrammed. In the process of such reprogramming, successful attempts were made to reduce the memory demands of the system because of the small-core memory of the UCSB computers (96K of 16-bit memory plus 2K in cache memory, with 32K available to an individual user) compared to the UCI PDP 10 (192K of 36-bit memory with about 50K normally available to an individual user). The principal means of such reduction was a procedure not readily possible on the PDP 10 (because of the need to swap large blocks of programs in and out of core to minimize I-O charges), but easy to implement on the PDP 11/45 UNIX operating system. One example of the nature and effects of this change concerns the program FRACTION HINT, which required the constant storage of 44,307 characters in core memory with the PDP 10; with UNIX on a PDP 11/45 it requires only 4100 characters or less to be resident at any one time. The new version of FRACTION HINT then calls small LISP files as needed; these other files need not be swapped out after use since they are simply executed without defining and storing functions in core. The space they occupy during execution is then reclaimed by using SUPERG, a new LISP garbage collection function described on page 5, when subsequent files are loaded.

One problem with a 16-bit machine such as the PDP 11/45 is that its operating system normally allocates no more than 32K of core memory to a single user. This is probably enough for the present system once some further garbage collection problems (see p. 5) are solved. If a larger interactive computer such as the VAX-11/780 does not become available at UCSB shortly, however, local computer managers may be asked to replace UNIX with the RSX11 operating system on at least one local PDP 11/45 to have Program Logical Address Space capabilities for accessing a larger amount of core memory. It would also be desirable to have a LISP compiler with overlay capabilities to improve the speed of the system. This would be expensive to develop but could greatly improve performance on a machine such as the PDP 11/45.

CONCLUSIONS

Although empirical conclusions await the analysis of data obtained with the CAI system described herein, it is possible to draw conclusions about the system itself in comparison with other ISO or generative systems. Table 3 summarizes a variety of property values held by (1) STUDENT, an algebraic word problem solver (Bobrow, 1968); (2) SCHOLAR, an ISO system for CAI about south America (Carbonell, 1970b); (3) SOPHIE, an ISO system for CAI about electronic troubleshooting (Brown & Burton, 1975; Brown, Burton, & Bell, 1975); (4) BIP, an ISO system for CAI about programming in BASIC (Barr, Beard, & Atkinson, 1975a, 1975b); and (5) four generative CAI systems for teaching mathematics. The latter are the closely related ones of Koffman and Perry (1976) and Gilkey (1974); that of Uttal, Rogers, Hieronymous, and Pasich, (1970); and the present UCSB system.

Despite the use of a variety of techniques, Table 3 shows that there is a good deal of commonality among these systems. All of the teaching systems are capable of generating a wide number of questions rather than retrieving a limited number of previously stored questions. They also permit a greater range of student choices or student aids than do AFO systems. Also, both the teaching systems and the problem-solving system STUDENT contain substantial information in relatively integrated structures, making it possible to think of them all as ISO systems. The seven systems break into two groups, however, when one asks whether or not each has a semantic network of knowledge: SCHOLAR and SOPHIE clearly do. The others, whose subject matter is more mathematical, have, at best, weak versions of semantic networks even though they do have complex data structures. Interpretation of this difference in system types requires clarification of the definition of a semantic network, and analysis of the way a semantic network either contains or could be coordinated with a network of mathematical knowledge.

Woods (1975) stated:

When one tries to devise a notation or language for semantic representation, one is seeking a representation which will precisely, formally, and unambiguously represent any particular interpretation that a human listener may place upon a sentence (p. 45).

He terms this the requirement of "logical adequacy" of a semantic representation. Two other requirements mentioned by Woods are that there must be an algorithm for translating the original sentence into this representation and algorithms that can use this representation for future inferences and deductions. It is not clear that such a semantic representation is possible, although SCHOLAR approximates it for South America geography. SOPHIE comes close for electronic troubleshooting, and several psychological modules such as Norman, Rumelhart, and the LNR Research Group's (1975) MEMOD are promising attempts to build general systems of natural language storage, comprehension, and question answering. Note that Woods' three requirements come close to specifying a total language system for a human; this effort focuses on the requirement of logical adequacy as being closest to the subproblem of building semantic networks.

Table 3

Comparison of the Present System to Other Information-Structure-Oriented or Generative CAI (or Problem-solving) Systems

Property	BIP	(a) Koffman & Perry				SCHOLAR	SOPHIE	STUDENT	UCSB	UTTAL
		(1976)	(b) Gilkey (1974)	(a) Digital systems design. (b) High school algebra story problems.	(a) Question answering about South American geography.					
Type of Problems:	Techniques of programming in BASIC.									
CAI	Yes.	Yes.			Yes.		Yes.	No.	Yes.	Yes.
Type of data gathered:	Computer aptitude battery, on-line performance, off-line achievement test.	(a) None. (b) On-line performance, off-line final test.			On-line protocols.		Quantitative and qualitative attitude data. Pretest and posttest on troubleshooting. On-line protocols.	Not applicable.	Off-line pretest and posttests, including attitude and mathematics achievement, on-line performance. (Projected.)	On-line performance? (Data gathering not discussed except incidentally.)
Kinds of student questions accepted:	Who, what, when.	See below.			Six kinds of questions in quantified or unquantified form, with English style variations permitted.		Present and normal voltage values at stated points. Is a certain hypothesis true? Hypothetical questions.	Not applicable.	See below.	See below.
Kinds of student requests accepted:	Help, plus most of the 27 commands listed in Table 1 of Barr, Beard, and Atkinson (1975b).	(a) Display old information. Display question and correct answer. Concept selection. Command student control if desired. (b) Calculation planned for development. Help (from human?).					Replace a specific component. Advice on what to do next.	Not applicable. (But STUDENT asks human operator for added information if solution otherwise impossible.)	Answer, compute, help, hint, new module, quit, solve.	Calculation, plotting, text display, definition, help from human tutor, quit, leave a utility early, save information on screen.

Table 3 (Continued)

Property	BIP	(a) Koffman & Perry (1976) (b) Gilkey (1974)		SCHOLAR	SOPHIE	STUDENT	UCSB	UTTAL
Information structure:	Curriculum information network summarizing problems and student knowledge. BASIC's grammar.	(a,b) Concept trees for certain types of problems. (b) Small grammar for age problems. General structure for study problems.	Semantic network about South American geography.	Semantic network focusing on electronic power supplies.	Semantic grammar using final sentence and their transformations.	Production systems for most module solutions.	Conic generators and some specific to certain types of problems.	
Semantic network?	Not natural language.	(a) No. (b) Very restricted.	Yes.	Yes.	Somewhat.	No.	No.	
Flexibilities in processing of answers:	Use of improved error messages compared to standard BASIC messages. Capability of execution of program one line at a time.	See below.	Look for unreasonable answers.	If hypothesis is vague, SOPHIE suggests a possible clarification.	Not applicable.	Scoring of answers as correct, correct except for minor changes specified, or incorrect.	Option of entering diagnostic routine after errors.	
Capability to determine relevance in selecting new text or question material:	Use of student interview after task plus task performance scoring to identify skills needing further training.	(a) None. (b) Use of a scoring polynomial to choose next concept. Solution monitoring can give partial problem and feedback prior to full problem.	Ask old questions in a new way. Ask questions related to previous one.	On receiving an unreasonable component replacement command, SOPHIE uses Socratic method to show the error to student. Points out redundancies in information requested. Points out contradictions and counter-examples to hypothesis.	Not applicable.	None other than ordering by module and within modules. HINT has memory for previously given advice which is used to structure current discussion.	Use of diagnostic process to lead into remedials, projected for further development.	

A semantic network is, first of all, a set of connected n-tuples that can alternately be represented in a connected graph. Second, the entries of these n-tuples are natural language words (and numbers) plus formalisms such as isa or superset (or x), which may or may not be considered part of the natural language. Parenthesized items are mathematical additions to the more commonly qualitative semantic networks.

Although STUDENT is a problem-solving system rather than a CAI system, it has inspired important research into human algebraic behavior (Hinsley, Hayes, & Simon, 1976; Paige & Simon, 1966). Being limited in the number and content of its kernel sentences, STUDENT nonetheless is almost competitive with SCHOLAR and SOPHIE in its natural language capability. STUDENT clearly shows the compatibility of including mathematical and natural language notions in the same grammar. With the addition of a SCHOLAR-type network or a SOPHIE-like parsing system, STUDENT could readily be expanded to the semantic level of SCHOLAR or SOPHIE. Similarly, Gilkey's (1974) semantic grammar for age problems could readily be developed to the complexity of STUDENT's.

BIP's grammar is highly complex, but its developers might have to start over if natural language were to be incorporated in its curriculum information network.

One reason for the differences between the information structure in the different systems follows from the nature of the school curriculum. In most school situations, the mathematics curriculum is relatively content-free even when word or story problems are used. Story problems tend to involve problem-specific (local) knowledge (e.g., "Eddy is 25 years younger than his father") more than general (global) knowledge (e.g., "A father is from 18 to 40 years older than his child") (Gilkey, 1974, p. 12 and pp. 28-30). Contrast this with SOPHIE, in which each new troubleshooting problem involves a change in symptoms but no change either in the laws of physics or in the design of the equipment to be repaired.

Other distinctions between the various CAI systems of Table 3 are also possible. The semantic capabilities of SOPHIE are not important so much because natural language can be employed by computer and student as because of a closely related reason—that extensive communication is possible between them. SCHOLAR is much less linguistically advanced than SOPHIE but is nonetheless somewhat "intelligent" in its communication ability. For example, Collins, Warnock, and Passafiume (1975) employed SCHOLAR as a computerized tutor to (1) arrange for the selection of topics according to a hierarchical outline, subject to modification based on the progress of student tutor interaction, (2) give feedback to students that is more than a simple "yes" and "no" (or "correct" and "incorrect"), simulating human tutors in elaborating on correct responses and making distinctions between incorrect responses and more desirable responses, and (3) ask comparable questions in a variety of related formats, thus providing more "human" instruction. Note that (2) and (3) are as much social features as intelligent features. They have some attributes of confluent education in the sense that they make instruction more acceptable to the students while maintaining or enriching its cognitive substance. A fourth characteristic, SCHOLAR's ability to answer a limited variety of student questions, is somewhat more relevant to the goal of developing intelligent CAI systems. No CAI system has been very successful in this area, however.

What is the difference in capability between the UCSB system responding appropriately to the command "solve" when a student finds a fraction problem too difficult, and SCHOLAR responding appropriately to the query? Each task requires a memory search and a deductive process, with the fraction problem being more complex than the geography problem. Superficially one might say that SCHOLAR takes the more complex input because a larger variety of words is used. However, the UCSB CAI system "understands" much more than "answer," "compute," "24," "2/3," or other student input. For example, "solve" means to solve a specific problem such as $(2/3 + 1 - 1/4)/(1/2) =$; the comprehension performed by the computer when "solve" is typed is not the meaning of that word alone but rather of the task to be performed. Whether or not the student types in the full message, a complex grammar in a high-level form not customarily encountered by a computer (because of the presence of special functions required in dealing with fractions) must be employed for the CAI system to analyze the problem, obtain a numerical answer, and display the steps of solution together with English text relevant to the computational steps that have been taken.

Does this mean that the UCSB system is "intelligent?" Is it intelligent only for computations such as fraction problems that cannot be solved routinely with languages such as FORTRAN? Or is it the use of explanatory text in a HINT or SOLVE routine that leads to any claim of intelligence for this aspect of the system?

Intelligence remains a fuzzy concept associated with (1) communicative capacity, (2) flexibility in the types of logically or psychologically equivalent input (or output) that can be accepted (or generated), (3) use of natural language either with or without a semantic network, and (4) degree of interdependence of computer output and student requests, statements, or question answering, both from problem to problem and within dialogues based on a single problem plus remedial or tutorial follow-ups to it. An IBM 360 computer acting on FORTRAN instructions from a student to solve a specific mathematics problem has some of the properties of Point 1, is somewhat capable on Point 2 because equivalent orderings of the same problem can be accepted by the computer, and fails on Points 3 and 4 until a CAI system of some complexity is added to the program. For this reason, FORTRAN mathematical computations are not considered to constitute intelligent CAI; if a LISP-machine or a natural language computer were developed and then were programmed to perform numerical computations, that would seem more intelligent but perhaps for no better reason than its novelty.

The UCSB system is strong on Point 1 in the sense that complex mathematical input can be accepted and understood and that a variety of one-stimulus, one-response dialogues are possible with utility functions such as HINT or with confluent material based on pattern matching. With respect to Point 2, there is flexibility in the acceptance of equivalent numerical problems and feedback generation (e.g., "good," "correct"), but not in making equivalent statements of the same word or story problem. This is related to the fact that the natural language interaction in the UCSB system lacks a semantic network base, making the system weaker on Point 3 than SCHOLAR or SOPHIE but nonetheless much stronger than a routine computation with FORTRAN. The addition of a semantic network of SCHOLAR's complexity but without a natural language parser like SOPHIE's would be feasible without greater effort than has already been expended on the UCSB system.

With respect to Point 4, the UCSB system's many utility commands, plus feedback and information about criteria reached, make it very strong so far as problem-to-problem interdependence is concerned. Within-problem dialogue is moderately strong, especially in the case of modules such as simple linear equations in which the hint given depends upon the student's report of progress on a problem. There is less remedial or tutorial follow-up to an individual problem than in SCHOLAR or SOPHIE.

It should be mentioned that any inventions from the field of artificial intelligence become potential techniques for use in intelligent CAI systems. Accordingly, Minsky's (1975), "frames" and Abelson's (1975) "plans" may shortly become components of ISO systems, both UCSB's and others.

What this means is that communication or intelligence is judged partly by the degree of window dressing (giving the verisimilitude of English dialogue to the system), partly by the ability of computer and student to evoke a variety of responses from the other, and (more formally) partly by stating the syntax and semantics of student and computer outputs.

This reasoning suggests that, if one were to build a semantic-mathematical network for use in a CAI system, one would want to do two things: (1) focus on alternate representations of the same items (and recognition of them as equivalent) by the computer, and (2) application of the system to the teaching of problems about a largely global body of real-life knowledge such as a specific engineering task or the operation of a specific financial institution. If the latter is not done, then there is the possibility of developing such a network as an investment in verbal computer-student communication marginal to the mathematical subject matter. It is an open question whether such an investment is better than developing a less sophisticated natural language communication system as has been done to date in the UCSB CAI system. One reason to think that the investment might be worthwhile is that diagnosis and remediation of student difficulties might proceed from intuitive student reports of confusion, or student queries, as well as from logical analysis of student error patterns and from statistical analysis of learning histories within a course or experiment. To some extent these student reports and queries are available in present mathematical CAI systems, but they deserve development with the aid of true semantic networks.

In view of this discussion, the existing UCSB CAI system for remedial arithmetic and algebra is essentially ready for experimental use and has substantial similarity to existing ISO and generative systems, particularly for teaching mathematics.

RECOMMENDATIONS

The Navy should consider increasing the use of mini- and supermini-computers (e.g., PDP 11/45, PDP 11/70, VAX-11/780) in simple generative CAI systems as a means of reducing permanent computer memory demands and overhead costs. The 32K single-user memory limitation on 16-bit machines (all but the VAX-11/780 just mentioned) may be a problem to some users and may dictate a move from UNIX to an operating system that can overcome that limitation (p. 18). Because of slow performance with interpretive systems, overlay LISP compilers may need to be developed for use with these computers if they are to be considered satisfactory for the purposes just mentioned.

Further recommendations are deferred pending the collection and analysis of empirical data on the relative efficiencies of instruction with the four CAI methods developed in the present study.

REFERENCES

- Abelson, R. P. Concepts for representing mundane reality in plans. In D. G. Bobrow & A. Collins (Eds.), Representation and understanding. New York: Academic Press, 1975.
- Aiken, L. R., Jr. Attitudes toward mathematics. Review of Educational Research, 1970, 40, 551-596.
- Anderson, J. Induction of augmented transition networks. Cognitive Science, 1977, 1, 125-157.
- Atkinson, R. C. Ingredients for a theory of instruction. American Psychologist, 1972, 27, 921-931. (a)
- Atkinson, R. C. Optimizing the learning of a second-language vocabulary. Journal of Experimental Psychology, 1972, 96, 124-129. (b)
- Atkinson, R. C. Teaching children to read using a computer. American Psychologist, 1974, 29, 169-178.
- Atkinson, R. C., & Paulson, J. A. An approach to the psychology of instruction. Psychological Bulletin, 1972, 78, 49-61.
- Barr, A., Beard, M., & Atkinson, R. C. A rationale and description of a CAI program to teach the BASIC programming language. Instructional Science, 1975, 4, 1-31. (a)
- Barr, A., Beard, M., & Atkinson, R. C. The computer as a tutorial laboratory: The Stanford BIP project (IMSSS Tech. Rep. 260). Stanford, CA: Institute for Mathematical Studies in the Social Sciences, Stanford University, 1975. (b) (AD-A015 092)
- Barr, A., & Beard, M. An instructional interpreter for BASIC. Proceedings of Computer Science and Education. SIGCSE Bulletin, 1976, 8, 325-333.
- Block, J. H. Student learning and the setting of mastery performance standards. Educational Horizon, 1969/72, 48-50, 183-191.
- Block, J. H., & Burns, R. B. Mastery learning. In L. S. Shulman (Ed.). Review of research in education 4: 1976. Itasca, IL: F. E. Peacock, 1977.
- Bobrow, D. G. Natural language input for a computer problem-solving system. In M. Minsky (Ed.). Semantic information processing. Cambridge, MA: MIT Press, 1968.
- Brown, G. I. Human teaching for human learning. New York: Viking, 1971.
- Brown, G. I. The training of teachers for affective roles. National Society for the Study of Education Yearbook, 1975, 74 (Part II), 173-203.
- Brown, G. I., Phillips, M., & Shapiro, S. Getting it all together: Confluent education. Bloomington, IN: Phi Delta Kappa Educational Foundation, 1976.

- Brown, J. S., & Burton, R. R. Multiple representations of knowledge for tutorial reasoning. In D. G. Bobrow & A. Collins (Eds.). Representation and understanding. New York: Academic Press, 1975.
- Brown, J. S., & Burton, R. R. A paradigmatic example of an artificially intelligent instructional system. In Proceedings of the First International Conference on Applied General Systems Research: Recent Developments and Trends. Binghamton, NY (in press).
- Brown, J. S., Burton, R. R., & Bell, A. G. SOPHIE: A step toward creating a reactive learning environment. International Journal of Man-Machine Studies, 1975, 7, 675-696.
- Brown, J. S., Burton, R. R., Miller, M., DeKleer, J., Purcell, S., Hausmann, C., & Bobrow, R. Steps toward a theoretical foundation for complex, knowledge-based CAI (BBN Rep. 3135). Boston, MA: Bolt, Beranek, and Newman, 1975.
- Brown, J. S., Collins, A., & Harris, G. Artificial intelligence and learning strategies. In H. O'Neil (Ed.). Learning strategies. New York: Academic Press, 1978.
- Brown, J. S., Rubenstein, R., & Burton, R. Reactive learning environment for computer-assisted electronics instruction (BBN Rep. 3314). Boston, MA: Bolt, Beranek, and Newman, 1976.
- Bunderson, V. C., & Faust, G. W. Programmed and computer-assisted instruction. National Society for the Study of Education Yearbook, 1976, 75 (Part I), 44-90.
- Burton, R. R., & Brown, J. S. A tutoring and student modeling paradigm for gaming environments. Proceedings for the Symposium on Computer Science and Education, Anaheim, CA: 1976.
- Cantlay, L. O. A study of the relationship between confluent teaching and mathematics self-conception in remedial math students (Doctoral dissertation). Santa Barbara, CA: University of California, 1975.
- Carbonell, J. R. AI in CAI: An artificial-intelligence approach to computer-assisted instruction. IEEE Transactions on Man-Machine Systems, 1970, MMS-9-11, 190-202. (a)
- Carbonell, J. R. Mixed initiative man-computer instructional dialogues (BBN Rep. 1971). Boston, MA: Bolt, Beranek, and Newman, 1970. (b)
- Collins, A. Comparison of two teaching strategies in computer-assisted instruction (BBN Rep. 2885). Boston, MA: Bolt, Beranek, and Newman, 1974.
- Collins, A. M. Processes in acquiring knowledge. In R. C. Anderson, R. J. Spiro, & W. E. Montague (Eds.). Schooling and the acquisition of knowledge. Hillsdale, NJ: Erlbaum Associates, 1977.
- Collins, A. M., Carbonell, J. R., & Warnock, E. H. Semantic inferential processing by computer. In J. Rose (Ed.). Advances in cybernetics and systems. London: Gordon and Breach, 1974.

- Collins, A., & Grignetti, M. C. Intelligent CAI (BBN Rep. 3183). Boston, MA: Bolt, Beranek, and Newman, 1975. (AD-A016 613)
- Collins, A. M., & Quillian, M. R. Retrieval time from semantic memory. Journal of Verbal Learning and Verbal Behavior, 1969, 8, 240-247.
- Collins, A., Warnock, E. H., Aiello, N., & Miller, M. L. Reasoning from incomplete knowledge. In D. G. Bobrow & A. Collins (Eds.). Representation and understanding. New York: Academic Press, 1975.
- Collins, A., Warnock, E. H., & Passafiume, J. J. Analysis and synthesis of tutorial dialogues. The Psychology of Learning and Motivation, 1975, 9, 49-87.
- Cotton, J. W. Models of learning. Annual Review of Psychology, 1976, 27, 155-187.
- Fletcher, J. D. Computer applications in education and training: Status and trends (NPRDC Tech. Rep. 75-32). San Diego: Navy Personnel Research and Development Center, April 1975. (AD-A009 800)
- Gilkey, T. J. Generative CAI in high school algebra (UCSE Tech. Rep. CS-74-6). Storrs, CN: University of Connecticut School of Engineering, 1974.
- Gilkey, T. J., & Koffman, E. B. Generative CAI in high school algebra. In A. Guenther et al. (Eds.). International computing symposium 1973. New York: North-Holland, 1974.
- Glaser, R., & Nitko, A. Measurement in learning and instruction. In R. L. Thorndike (Ed.). Educational measurement (2nd Ed.). Washington, DC: American Council on Education, 1971.
- Goldberg, A., & Suppes, P. A computer-assisted instruction program for exercises on finding axioms (IMSSS Tech. Rep. 186). Stanford, CA: Institute for Mathematical Studies in the Social Sciences, Stanford University, 1972.
- Goldberg, A., & Suppes, P. Computer-assisted instruction in elementary logic at the university level (IMSSS Tech. Rep. 239). Stanford, CA: Institute for Mathematical Studies in the Social Sciences, Stanford University, 1974.
- Good, T. L., Biddle, B. J., & Brophy, J. E. Teachers make a difference. New York: Holt, Rinehart, and Winston, 1975.
- Harris, D. E. Psychological awareness and moral discourse: A curriculum sequence for moral development (Doctoral dissertation). Dissertation Abstracts International, 1977, 37, 5562-A-5563-A. (University of Wisconsin, 1976, University Microfilms No. 76-28, 148).
- Heath, D. Humanizing schools. New York: Hayden, 1971.
- Hinsley, D. A., Hayes, J. R., & Simon, H. A. From words to equations: Meaning and representation in algebra word problems (CIP Working Paper No. 331). Pittsburgh: Carnegie-Mellon University, 1976.

- Hively, W., Patterson, H. L., & Page, S. A "universe defined" system of arithmetic achievement tests. Journal of Educational Measurement, 1968, 5, 275-290.
- Ivey, A. E., & Alschuler, A. S. (Eds.). Psychological education: A prime function of the counselor. Personnel and Guidance Journal, 1973, 51, 588-591.
- Jamison, D., Suppes, P., & Wells, S. Effectiveness of alternative instructional media. Review of Educational Research, 1974, 44, 1-67.
- Kahn, S. B., & Weiss, J. The teaching of affective responses. In R. M. W. Travers (Ed.). Second handbook of research on teaching. Chicago: Rand McNally, 1973.
- Keller, F. S. Goodbye, teacher . . . Journal of Applied Behavior Analysis 1968, 1 79-89.
- Kimball, R. B. Self-optimizing computer-assisted tutoring: Theory and practice (IMSSS Tech. Rep. 206). Stanford, CA: Institute for Mathematical Studies in the Social Sciences, Stanford University, 1973.
- Kintsch, W. The representation of meaning in memory. Hillsdale, NJ: Erlbaum Associates, 1974.
- Koffman, E. B. Design techniques for generative computer-assisted instructional systems. IEEE Transactions on Education, 1973, E-16, 182-189.
- Koffman, E. B., & Blount, S. E. Artificial intelligence and automatic programming in CAI. Artificial Intelligence, 1975, 6, 215-234.
- Koffman, E. B., & Perry, J. M. A model for generative CAI and concept selection. International Journal of Man-Machine Studies, 1976, 8, 397-410.
- Kohlberg, L. Moral stages and moralization: The cognitive-developmental approach. In T. Lickona (Ed.). Moral development and behavior: Theory, research, and social issues. New York: Holt, Rinehart, and Winston, 1969.
- Lecarme, O., & Lewis, R. (Eds.). Computers in education. Amsterdam: North-Holland, 1975.
- McCarthy, D. N. A confluent reading and English fundamentals curriculum: Derivation, description, and evaluation (Doctoral dissertation). Dissertation Abstracts International, 1976, 36, 5228-A-5229-A. Santa Barbara, CA: University of California, 1975. (University Microfilms No. 76-2736.)
- Minsky, M. A framework for representing knowledge. In P. Winston (Ed.). The psychology of computer vision. New York: McGraw-Hill, 1975.
- Newell, A. Production systems: Models of control structures. In W. G. Chase (Ed.). Visual information processing. New York: Academic Press, 1973.
- Norman, D. A., Rumelhart, D. E., & LNR Research Group. Explorations in cognition. San Francisco: W. H. Freeman and Company, 1975.

- Offir, J. Automation models of performance. Journal of Mathematical Psychology, 1973, 10, 353-363.
- Paige, J. M., & Simon, H. A. Cognitive processes in solving word problems. In B. Kleinmuntz (Ed.). Problem solving: Research, method, and theory. New York: Wiley, 1966.
- Pearlman, A. P. Linear relations between cognitive and affective test performance (Doctoral dissertation). Santa Barbara, CA: University of California, 1977.
- Rumelhart, D. E., & Norman, D. A. Learning, memory, and the acquisition of knowledge. In J. W. Cotton & R. L. Klatzky (Eds.). Semantic factors in cognition. Hillsdale, NJ: Erlbaum Associates, in press.
- Shiflett, J. M. Beyond vibration teaching. In G. I. Brown, T. Yeomans, & L. Grizzard (Eds.). The live classroom: Innovation through confluent education and gestalt. New York: Viking, 1975.
- Shiflett, J. M., & Brown, G. I. Confluent education: Attitudinal and behavioral consequences of confluent teacher training. University Center, MI: University Center Monograph Series, Saginaw Valley State College, 1972.
- Shoemaker, D. M. Toward a framework for achievement testing. Review of Educational Research, 1975, 45, 127-147.
- Sigal, J., Braverman, S., Pilon, R., & Baker, P. Effects of teacher-led, curriculum-integrated sensitivity training in a large high school. Journal of Educational Research, 1976, 70, 3-9.
- Smith, N. W. A question-answering system for elementary mathematics (IMSSS Tech. Rep. 227). Stanford, CA: Institute for Mathematical Studies in the Social Sciences, Stanford University, 1974.
- Smith, R. L., & Blaine, L. H. A generalized system for university mathematics instruction. SIGCUE Bulletin, 1976, 8, 280-288.
- Smith, R. L., Graves, W. H., Blaine, L. H., & Marinov, V. G. Computer-assisted axiomatic mathematics: Informal rigor. In O. Lecarme and R. Lewis (Eds.). Computers in education. Amsterdam: North-Holland, 1975.
- Smith, R. L., Smith, N. W., & Rawson, F. L. Construct: In search of a theory of meaning (IMSSS Tech. Rep. 238). Stanford, CA: Institute for Mathematical Studies in the Social Sciences, Stanford University, 1974.
- Stanford, G. Psychological education in the classroom. Personnel and Guidance Journal, 1972, 50, 585-591.
- Suppes, P. Computer-assisted instruction at Stanford. In Man and computer: Proceedings of International Conference, Bordeaux, 1970. Basel: Karger, 1972.

- Suppes, P., Fletcher, J. D., & Zanotti, M. Performance models of American Indian students on computer-assisted instruction in elementary mathematics. Instructional Science, 1975, 4, 303-313.
- Suppes, P., Fletcher, J. D., & Zanotti, M. Models of individual trajectories in computer-assisted instruction for deaf students. Journal of Educational Psychology, 1976, 68, 117-127.
- Suppes, P., Fletcher, J. D., Zanotti, M., Lorton, P. W., Jr., & Searle, B. W. Evaluation of computer-assisted instruction in elementary mathematics for hearing-impaired students (IMSSS Tech. Rep. 200). Stanford, CA: Institute for Mathematical Studies in the Social Sciences, Stanford University, 1973.
- Suppes, P., Goldberg, A., Kanz, G., Searle, B., & Stauffer, C. Teacher's handbook for CAI courses (IMSSS Tech. Rep. 178). Stanford, CA: Institute for Mathematical Studies in the Social Sciences, Stanford University, 1971.
- Suppes, P., & Morningstar, M. Computer-assisted instruction at Stanford, 1966-1968: Data, models, and evaluation of the arithmetic programs. New York: Academic Press, 1972.
- Uttal, W. R., Rogers, M., Hieronymous, R., & Pasich, T. Generative computer-assisted instruction in analytic geometry. Newburyport, MA: Entelek, 1970.
- Woods, W. A. What's in a link: Foundations for semantic networks. In D. G. Bobrow & A. Collins (Eds.). Representation and understanding. New York: Academic Press, 1975.

REFERENCE NOTES

1. Brown, J. S., Burton, R. R., & Hausmann, C. Representing and using procedural bugs for educational purposes. Unpublished preliminary manuscript. Bolt, Beranek, and Newman, 1977.
2. Kifer, E. The effects of school achievement on the affective traits of the learner. Paper presented at the annual meeting of the American Educational Research Association, New Orleans, February 1973.
3. Callan, M. F. The effects on teacher's moral judgment of a didactic and experimental training program in the methodology of cognitive moral development and self-knowledge of one's behavioral patterns (Doctoral dissertation). Santa Barbara, CA: University of California, in preparation.
4. Gollub, W. Affective education development program research report 1970-71. Philadelphia: School District of Philadelphia, 1971.
5. Gollub, W., & Mason, E. The affective education program: Evaluation report 1972-1973. Philadelphia: School District of Philadelphia, 1973.
6. Young, P. Affective education abstract. In Evaluation of Title I ESEA projects, 1973-1974: Abstracts. Philadelphia: Office of Research and Evaluation, School District of Philadelphia, 1974.
7. Loue, W. E., III, Leibovitz, J. G., & Sklar, L. R. Affective education abstract. In Evaluation of Title I ESEA project 1975-76: Abstracts. Philadelphia: Office of Research and Evaluation, School District of Philadelphia, 1976.
8. Loue, W. E., III. Personal communication. 23 November 1977.

APPENDIX A
REVIEW OF RELEVANT LITERATURE

REVIEW OF RELEVANT LITERATURE

Recent Sources of Information About CAI

This section provides a point of entry to the many hundreds of publications on computer-assisted instruction (CAI). Jamison, Suppes, and Wells (1974) examined the empirical evidence on the relative effectiveness of CAI and traditional instruction, as well as evaluating instructional radio and television and programmed instruction. Fletcher (1975) surveyed CAI developments in military and civilian education. One of the special strengths of the report is its description of the major CAI projects in the United States. Bunderson and Faust (1976) reviewed CAI, and included somewhat less unpublished material than did Fletcher. Cotton (1976, pp. 175-178) has discussed CAI in the context of mathematical models of learning and instruction. LeCarme and Lewis (1975) have edited a conference volume on computers in education that displays much of the newest thinking on CAI, such as the information-structure-oriented systems to be discussed below.

AFO and ISO Methods of Instruction

Early AFO systems were programmed to present preplanned text and questions in a linear order (or a linear order subject to branching arrangements contingent upon student responses) and to employ preprogrammed feedback to student answers. The student was limited to a particular format for these responses. The use of psychological models of learning and memory was limited to admonitions to course planners to provide immediate feedback after student answers, to increase difficulty of items by small increments as instruction progressed, and to present new material as performance on old items reached criterion.

A more sophisticated theoretical approach to AFO-CAI instruction is seen in various papers by Atkinson (1972a, 1972b) and his coworkers. In German-English vocabulary instruction, Atkinson (1972a, 1972b) compared the effects of sequencing item presentation according to optimization rules based on the linear model, the trial-dependent Markov model, or according to the student's choice of the current stimulus. With the maximization of average percent correct on the posttest as a criterion, the trial-dependent Markov model was much more effective than the linear model, especially if different difficulty estimates were used with each item under the former model. Student self-selection of items was superior to both models except when the trial-dependent Markov model was employed with separate difficulty parameters for each item. Similarly, Atkinson and Paulson (1972) found that a sequencing strategy based on optimization procedures with the all-or-none model produced higher posttest means following spelling instruction than did the corresponding procedure for the linear model. A third model's strategy, that of the random-trials-increment model with individual difference parameters for items and students, proved superior to either of the other two models on the posttest. Cotton (1976) conjectured that the all-or-none model strategy (i.e., selecting, for the next stimulus, one of the items that had shown the fewest successive correct responses since the last error) is approximated by the student self-selection procedure, explaining the good performance in that condition in the Atkinson (1972a, 1972b) reports.

Based on an important set of competing learning models, the aforementioned line of research dealt best with unstructured curriculum material; that is, with relatively independent items to be learned. A second trend in mathematical modeling for CAI tasks has been used for somewhat structured tasks such as reading (Atkinson, 1972a; Atkinson, 1974) and even more structured tasks such as arithmetic (Suppes, Fletcher, & Zanotti, 1975, 1976; Suppes, Fletcher, Zanotti, Lorton, & Searle, 1973). The exponential and power models involved in these studies are empirically successful but are molar in nature and, thus, say relatively little about the underlying processes of learning.

Suppes (1972) and Suppes and Morningstar (1972) have developed task-oriented models of arithmetic instruction. This represents an advance in psychological theorizing, expanding the sensitivity of psychological theory to the subject matter being learned. Suppes and Morningstar (1972), and Offir (1973) employed automaton models of the addition process, and Suppes (1972) dealt with both automaton models and register machine models of addition. The latter models have the learner make explicit movements to attend to relevant stimuli, such as the upper right-hand digit of a problem, and allow for computer-like subroutines to be followed by the student. This particular set of models is primarily concerned with performance rather than learning, although a sketch is given in Suppes (1972) for the development of a possibly hierarchical model of learning employing register machines.

The development of ISO instructional systems is both a direct and indirect consequence of recent progress in computer science. Direct influence comes from the pioneering use of semantic network theory by Carbonell (1970b) in his development of SCHOLAR, a system for teaching or reviewing information about South American geography by dialogue between a computer and a student. Brown, Burton, and Bell (1975), also computer scientists, developed a second ISO system called SOPHIE (for Sophisticated Instructional Environment) to teach methods of troubleshooting electronic equipment. Direct influence from computer science to ISO instruction also comes in the form of progress in the design of interactive computers and list-oriented computer languages compatible with the storage, processing, and retrieval of organized information such as English prose. The indirect influence comes from recent psychological theorizing using computer simulation or linguistic formalisms to express memory, comprehension, retrieval, and question answering regarding highly structured information (e.g., Anderson, 1977; Collins & Quillian, 1969; Kintsch, 1974; Norman, Rumelhart, & the LNR Research Group, 1975). This further example of a current trend toward cognitive rather than behavioral psychology helps to explain the interest of psychologists in ISO instruction.

It is convenient here to identify two types of ISO instruction: systems whose objective is tutoring over a specified subject area, and those designed to teach a set curriculum (sometimes called "drill and practice" units, especially in AFO systems). The tutorial systems are not designed to teach new material to students. Rather, they provide an environment of review and possible exploration of ideas about already known material. Often either the computer or the student can structure the session, and each question and response of the computer is tailored to the individual. In contrast are systems designed to teach a block of new material. By whatever means or

model, they determine sequences of material to approximate a non-CAI course of instruction. All stand-alone courses belong to this category. Individualization to a student's needs in these systems is limited to changes in the pace of learning and to changes in lesson structure achieved by feedback procedures. In general, such systems have a firm sequence of lessons designed by the developers, but problems in this sequence are usually generated individually and feedback is response-sensitive. Some systems within this category also contain a tutorial mode. For the most part, the tutorial components of these systems are not as fully developed as those designated as tutorial systems here. The student has little control in the tutoring sessions of the instructional systems, and these systems do not necessarily allow the student to ask questions or pose problems.

The prototype of ISO-CAI is Carbonell's (1970b) SCHOLAR, which was later extended by Collins (1974); Collins, Carbonell and Warnock (1974); Collins and Grignetti (1975); Collins, Warnock, Aiello, and Miller (1975); Carbonell (1970a); Collins and Grignetti (1975); and Collins, Warnock, and Passafiume (1975). Carbonell (1970b) originally intended SCHOLAR to be a functional CAI system, but it was later modified to represent artificial intelligence by simulation of human dialogue, with little attention given to instructional use. Currently, some instructional emphasis is found in Collins' (1977) attempt to formalize the Socratic method of instruction.

SCHOLAR is a simulation of a tutor that interacts with students on the subject of South American geography. By asking questions, either the student or the computer-tutor can initiate and structure the session. The information base of SCHOLAR is stored as a semantic tree structure; that is, the information is stored as hierarchical interrelations among bits of data (such as countries, cities, major regions, and their outstanding features). For example, concepts under Argentina include cities, rivers, and major regions.

The tutor draws inferences from the knowledge stored in this way as well as determining the correct responses to factual questions such as "What is the largest city of Chile?" Thus, the computer's responses to the student are generated by the system; these responses are tailored to fit the specific question or exchange of statements currently taking place.

There is little question that SCHOLAR represents a tremendous advance in CAI techniques. However, in terms of instruction, it is incomplete. Carbonell specifically omitted internal models of student learning, stating that since SCHOLAR functioned as a tutor, such models were unnecessary (Carbonell, 1970a). This seems naive; human tutors are continually forming informal models about what their students know and don't know. Moreover, confirmed hypotheses about their student's knowledge are not discarded but become part of the tutor's overall picture of the student. Without a model or record of the student's abilities, SCHOLAR cannot tailor its tutoring to the student's needs. By ignoring the learner's history, it can only fit single responses to the immediate question.

Other tutorial systems also deserve mention here. The best-known of these is Brown, Burton, and Bell's (1975) SOPHIE (see also Brown & Burton, 1975; Brown, Rubenstein, & Burton, 1976). For the domain of electronics troubleshooting, SOPHIE poses problems, answers questions, verifies hypotheses, and formulates theories. The student is encouraged to experiment. A model of the student's knowledge is created, and within the parameters of this model, SOPHIE simulates the changes and takes the measurements specified by the student. SOPHIE monitors the student's progress and provides information and advice when it becomes obvious that the student's current strategy will not work.

SOPHIE's objective is to create a reactive environment in which the student can experiment. The student is given the initiative in almost every exchange. SOPHIE does maintain a rough model of the student's troubleshooting attempts, but little is said about this model in system documentation. Although it records current student hypotheses and evaluates whether these hypotheses are consistent with known information about the circuit, nothing is known about its updating function. Like SCHOLAR, however, SOPHIE is not concerned with identifying particular skills; that is, there is no emphasis on the student's learning of particular troubleshooting techniques. No record of student strengths or weaknesses in the subject matter is kept, so there is no possibility of tailoring the instruction to the individual. In fact, SOPHIE is one of the least individualized CAI systems. It presents a similar problem to each student: one circuit with a flawed component. The basic circuit never changes, only the flawed component varies.

SOPHIE contains a clear model of the subject area. It was designed to troubleshoot one particular circuit, and it "knows" everything about that circuit. All components and their proper measurements are contained in its network of information. In addition, there exists a testing procedure within the system. Once a student claims to have found a faulty component and requests its replacement, the system asks questions to determine whether the student understands the consequences of such a request. Using an internal decision rule (not explained in published reports), SOPHIE either determines that the student understands, in which case it makes the requested replacement, or else it concludes that the student does not understand and refuses to change the circuit. In the latter case, the student is advised to continue working on the problem. Although SOPHIE can determine that the student does not understand the problem, it does not use this information to structure subsequent dialogue. No student gets special questions related to a specific misunderstanding. In essence, the student is told that there is a problem and is left to diagnose it.

The tutoring system of Burton and Brown (1976) for the game "How the West Was Won" is not precisely a CAI system, but it does tutor a student by teaching game strategies. Many of SOPHIE's unique ISO features are found in this system in addition to several new features. In particular, Burton and Brown have added an explicit model or record of a student's history in the game.

"How the West Was Won" is a PLATO game in which two players move along a sequence of squares. Some of the squares yield special points and some are designated as short cuts to future squares in the sequence. A player is given three simple numbers to combine into an optimal move. Possible operations on the numbers include such things as combining two and multiplying the result times the third, or multiplying any two and adding the product to the third. In all, there are sixteen possible moves for every trio. The player is expected to select the move that will advance him in the game. The largest number is not always the the best move; it is frequently more desirable to move to special squares.

The tutor plays the game with a student, and the tutor always makes the optimal move. The tutor also determines the student's optimal move and how far the student's actual move is from the optimal choice, but the student is unaware of these calculations. A record or model of all student selections is maintained. Within this model are the types of moves (maximum distance, special squares, short-cut squares) and the number of times these would be optimal as well as the number of times the student actually used them in the game. When the tutor determines that the student is weak in some particular type of move, the program so informs the student the next time that particular move is made.

Like SOPHIE, this system contains many diagnostic features. Its model of the subject matter is knowledge of the game; this feature is represented in terms of possible moves within the rules of the game. It does no explicit testing to determine whether the student is aware of all possible moves; the student has control of the session and the tutor asks no questions. Essentially, the tutor is invisible to the student and interrupts the game only rarely to provide hints to the student. However, the tutor does maintain a model of the student and does make an attempt to identify the student's weaknesses by simulating each move of the student. After the student has made the same mistake a certain number of times, the tutor suggests an alternative move. With aids like these, the student learns to play the game better. A by-product of this process is drill and practice in arithmetic (Brown & Burton, in press).

Both SOPHIE and the tutor for "How the West Was Won" have one disturbing feature: They cannot be easily generalized to any other subject area. Recall the system of SCHOLAR. Given its structure, it could easily be changed into some other subject matter by alteration of the network. SOPHIE and the tutor cannot. Both are very large systems operating specifically upon very small curricula. Language processing and student models are both specific to the nature of the individual problem at hand and are completely context-dependent. For example, the tutor depends completely upon the use of three numbers that can be combined in particular ways. Changing to another game that did not involve this same possibility would render the tutor useless. Brown, Collins, and Harris (in press) have dealt with this problem theoretically, seeking a common theory of understanding such disparate areas as simple stories, elementary mathematics, and electronic circuits. They distinguish between surface traces summarizing exactly what has been happening, and deep structure traces summarizing the decision histories leading to certain responses.

It should be noted that Brown and Burton (in press) have emphasized the paradigmatic status of systems such as SOPHIE and "How the West Was Won." Another of their paradigmatic systems, BLOCKS, was developed to teach and analyze the process of making classifications based on combinations of block attributes, using complex forms of a rule-learning task widely employed by experimental psychologists. Monitor units detect inefficient responses by a student and activate a tutor to help the student rethink the problem. The tutor is reported to have proved oppressive, apparently giving more information than desired or than could be assimilated. This caused the authors to desire a psychological model of the learning process.

Kimball's (1973) tutor for problems of integral calculus contains archives of problems and their answers in the style of traditional (AFO) CAI, but it also has a network of possible techniques of each problem in ISO-CAI style. Little attention is given to this network in the documentation, and it is not completely obvious how the network is utilized in the system. The focus of the program is the model of the student. This is a Markovian model that is created by the system for each student. The number of states in the model varies according to the number of problem-solving techniques known to each subject. Kimball was interested in tracing the various theoretical states in which a student could be while solving one particular problem. His predictions are, in the main, predictions that a certain technique will be selected next given that the student is utilizing a particular technique at the moment. This is the prediction "where will the student go next?"

As a student solves a problem, progress is plotted as a path through different states in the model. Using the transition probabilities of the model together with the knowledge of techniques needed for a given problem, the tutor calculates the probable need for diagnostic intervention. The calculation is a function of the tutor's technique priorities together with the current state of the student model. When the number exceeds a fixed value, the student is informed that solution is unlikely via the current path and that another technique choice is advised. Upon completion of the task, the tutor compares its solution with the student's and allows the student to make the same comparison. Should the student's solution be superior to the computer's solution (e.g., contain fewer steps), the tutor "learns" from the student and changes the techniques and solution steps stored for that problem. In addition, the system reviews its model of the student at this point. If the student's path to solution is sufficiently inefficient or incorrect, the tutor can force the student into its "slave mode." This step is taken only when a measure of solution difficulty reaches a certain level. The measure is calculated as a function of the number of steps involved in the solution attempt and the difference between the tutor's and the student's technique selection at each step. The slave mode takes the student step-by-step through the archive problem that best utilizes the techniques in question.

Kimball's system makes use of its diagnosis; it selects problems based on the student's problem-solving abilities and known problem areas. It utilizes its curriculum to build a model of student knowledge.

The testing procedure here is based upon each attempt by the student to solve a single problem; when the transition probabilities in the model reach a certain point (determined by Bayesian inference), the tutor begins its diagnosis. It does not attempt to determine by testing and evaluation which skills are not being used properly. Instead, the system simply asks which techniques the student knows. Self-evaluation alone may be insufficient in diagnosis, for a student may be unaware that problem-solving skills are improperly formed.

Another tutorial system to be discussed here is BUGGY, created by Brown, Burton, and Hausmann (Note 1). It is part of a larger, though still incomplete system (Brown, Burton, Miller, DeKleer, Purcell, Hausmann, & Bobrow, 1975) that maps error patterns in high school algebra. BUGGY, however, deals only with simple problems of addition and subtraction. The objective of the overall project is to identify a student's specific error patterns; that is, to diagnose the pattern of errors given a set of answers to specific problems. BUGGY, however, does not do that. Instead, it teaches student-teachers to diagnose problems. BUGGY contains a large number of flawed algorithms for addition and subtraction problems. Then, much like SOPHIE, it inserts one of these algorithms into the solving routines, and the system answers questions according to this flaw. The student-teacher must diagnose the error from BUGGY's pattern of responses.

This seems incompatible with the authors' stated goals; the computer makes no diagnosis whatsoever. In effect, the student must discover the computer's error. Of course, since the error patterns are part of the system, it seems likely that simple modifications could be made to reverse the process and have the program identify flaws in the student's answers. However, this strategy does not appear feasible to deal with all high school algebra. It seems inefficient to program all possible error patterns; these could be quite numerous. Such an approach does not appear to differ from AFO-CAI with its anticipated wrong answers and specified branching. It is too early to determine how well such a technique will actually function; further research on the subject is currently in progress, and more information should be available soon.

Uttal, Rogers, Hieronymous, and Pasich (1970) developed a tutorial system for instruction in analytic geometry. This system permitted random generation of problems of several types and difficulty levels from each of 17 topics such as properties of circles. It also presented text material appropriate to the topic being taught, employed diagnostic programs to determine specific difficulties a student was having, and used remedial programs to eliminate those difficulties. The diagnostic package used a Binary Branching Tree Partitioner to decompose a problem into sequential subproblems, testing as many subproblems as necessary to identify the part of the larger problem that could not be solved by a certain student. Most of the planned remedial techniques had not been completed at the time of publication (Uttal et al., 1970, p. 54). Student records were kept with this tutorial system, but no data were reported in the reference cited.

In contrast, ISO-CAI systems are designed to teach new material. These systems differ from tutorial CAI in a number of ways; for instance, the curriculum is generally much larger, a greater variety of lesson formats

is supplied, and some means of performance evaluation is provided. None of these CAI systems function solely with semantic networks of simple concepts. In practice, the networks relate much larger blocks of information, such as entire lessons or complex concepts.

One large-scale ISO-CAI course is LOGIC 57A, Introduction to Symbolic Logic, currently being taught at Stanford University (see Goldberg & Suppes, 1972, 1974). The course consists of 35 lessons, each of which contains a varied number of problems. The problems usually involve simple proofs. An individual's grade for the course is based upon the number of lessons successfully completed; there is no final examination. The system's record or model of student performance is updated after each problem, and includes the student's history of both lesson sequence and problems completed within the current lesson. The problems themselves are not stored but are generated from stored skeletal frameworks in response to the current state of the student record. A single, linear sequence of lessons is intended for all students.

One unusual feature of the logic system is the lack of a formal theorem-prover or set of stored solutions. As the student writes a proof, the program examines it line by line with the internal theorem-analyzer. The machine takes the working premise of the student, and if it is feasible, the system then constructs an unseen logical proof with which to check the student's progress. When the student constructs illogical proofs, the system's feedback and corrective statements are based upon the student's own approach and premise rather than one stored in the system as the best solution.

However, little diagnostic help is available in the course. The model of student knowledge is limited to a record of correct and incorrect responses. It keeps track of each individual's current lesson but does not provide a sequence of problems based upon a diagnosis of student difficulties. The program has no way to recognize or provide special assistance in any method or technique used in solving problems. Thus, the course does not contain the components necessary in diagnosis, although it does provide some individualization in problem sequencing. It contains a highly structured curriculum but no network of skills or techniques. Without such a network, the other elements of a diagnostic system are impossible.

A second course offered at Stanford is a course in axiomatic set theory called EXCHECK (Smith & Blaine, 1976; Smith, Graves, Blaine, & Marinov, 1975). This course allows informal proof techniques, eliminating the necessity for fully explicit formal proofs. Its intent is to provide a semantic base for work in natural language processing in set theory. (For more information, see Smith, Smith, & Rawson, 1974; Smith, 1974.) Although self-contained, this program functions only as a proof-checker. A student submits a proof, and the system ascertains whether or not it is correct. As with the logic course, there is no final examination. Final grades are determined by the number of correct proofs submitted. EXCHECK contains no diagnostic capabilities whatsoever; it qualifies as ISO-CAI only because of its language flexibility; it has no other common

features with any CAI program described here. It would appear that EXCHECK functions chiefly as a timesaving device for checking proofs. No information is provided the student other than whether or not a given proof is correct.

Another course in ISO-CAI is given at Stanford University. This is the BASIC Instructional Program, better known as BIP (Barr, Beard, & Atkinson, 1975a, 1975b; Barr & Beard, 1976), and is perhaps the best generative CAI course currently available. Designed to provide individualized instruction, BIP contains a remarkable model of student knowledge. BIP teaches elementary programming in the BASIC programming language. It stores about 100 fully written problems in its curriculum information network (CIN). Solutions to the problems are not stored; indeed many problems have several correct solutions, each utilizing a different programming skill. Within the CIN, each problem is associated with the necessary skills for solution(s), and the relations among skills is also represented. There is no fixed curriculum of lesson; BIP scans the information in the CIN and selects the problem that best utilizes a particular skill or set of skills.

Following every lesson, the student is questioned by the tutor about confidence in and problems with the lesson solution. Based upon the student's self-evaluation and responses during the lesson, BIP creates a model of the student, an internal representation of the student's current knowledge. In contrast to the models of other generative systems, BIP's curriculum network is used to model progress in terms of developing skills instead of the student's history of right or wrong answers. This model is used in selecting the next problem for the student. BIP is concerned with programming skills rather than exact solutions to problems, and the model contains the cumulative record of the student's use of various programming skills. Thus, two students might be shown the same programming problem and be expected to write very different solutions because of differences in past performance.

Note that BIP models its curriculum in terms of programming skills, and it uses these skills in its model of student knowledge. The only aspect in which it is lacking is the generation of problems to test specific skills. Currently, the authors of BIP are attempting to implement a more complete diagnostic model that includes generative features.

Machine Language Tutor (MALT) (Koffman & Blount, 1975) and its variants (Gilkey, 1974; Gilkey & Koffman, 1974; Koffman, 1973; Koffman & Perry, 1976) teach either automatic programming in machine language or digital systems design at the University of Connecticut. It has as its foundation (1) a probabilistic grammar that generates and solves problems, and (2) a detailed model of student knowledge. It is both a tutor and a stand-alone course that takes naive subjects through a step-by-step sequence in elementary machine language programming.

As a teacher of machine language, MALT currently has 26 concepts, which can be used individually or combined in groups of 2 or 3. These concepts can be used more than once in a single program. When a problem is generated by MALT, it is automatically divided into subprograms using the basic concepts.

There is little variation in this part of the system; even though a student might structure the problem differently, he must solve MALT's subproblems rather than his own. In this respect MALT is inferior to the Stanford logic course, which accepted the student's premise and solved the problem accordingly. For beginning subjects, feedback is provided by MALT at every line of input. The amount of feedback in later tasks varies, depending upon the current state of the student information model.

Although the developers of MALT emphasize the need for diagnosis, little is present. MALT maintains a model of the student's abilities in basic concepts, and it contains a tree structure for possible problems in utilizing these concepts either individually or in combinations. Based upon a student's progress in the concepts, MALT generates appropriate problems using those concepts. Although no other system has this capability, MALT is unable to make good use of it. Basically, it interprets a student's input program one line at a time. If the program fails to run, MALT cannot isolate the error and the student must do so. Complex programs of more than one line cannot be diagnosed at all. Thus, MALT can employ its model and testing procedures only at the most elementary level; it corrects syntax but not logic. It cannot diagnose the student's programming ability in complex tasks.

Koffman and Perry (1976) evaluated the effectiveness of MALT's digital systems design curriculum by a one-semester experiment with two randomly assigned (possibly quasi-random, in view of the use of a university scheduling algorithm) classes of about 30 students. Previous mean grade point averages and mean pretest scores favored the control group, but the mean final exam scores showed a small, nonsignificant difference in favor of the CAI group, despite the fact that its members spent one-third less time in class.

The final system to be mentioned here is Gilkey and Koffman's (1974) and Gilkey's (1974) algebra system. Although it is a generative system, it cannot be classified as either didactic or tutorial. Based upon features of the Koffman and Blount (1975), Gilkey (1974), Gilkey and Koffman (1974), and Koffman and Perry (1976) MALT system, this program also depends upon a probabilistic grammar for both problem and answer. Gilkey and Koffman are working primarily with word problems, and their system contains skeletal frames that the grammar fills to generate sentences. The generated problem is presented to the student, who is asked to translate the sentence into an equation. When this is done correctly, the student then solves the equation. The system monitors both the equation transformation and the solution.

Two terms that are partially synonymous with information-structure-oriented CAI deserve mention: mixed-initiative CAI and generative CAI. Mixed-initiative CAI literally means computer-assisted instruction in which either the computer or the student can take the initiative in determining the course of the instructional session. Depending on the nature of the specific system, the student taking the initiative might select the subtopic of curriculum to be considered next, ask natural language questions of the computer, or ask for specific aids by using formatted responses. In generative CAI, specific questions need not be stored for presentation.

There is little variation in this part of the system; even though a student might structure the problem differently, he must solve MALT's subproblems rather than his own. In this respect MALT is inferior to the Stanford logic course, which accepted the student's premise and solved the problem accordingly. For beginning subjects, feedback is provided by MALT at every line of input. The amount of feedback in later tasks varies, depending upon the current state of the student information model.

Although the developers of MALT emphasize the need for diagnosis, little is present. MALT maintains a model of the student's abilities in basic concepts, and it contains a tree structure for possible problems in utilizing these concepts either individually or in combinations. Based upon a student's progress in the concepts, MALT generates appropriate problems using those concepts. Although no other system has this capability, MALT is unable to make good use of it. Basically, it interprets a student's input program one line at a time. If the program fails to run, MALT cannot isolate the error and the student must do so. Complex programs of more than one line cannot be diagnosed at all. Thus, MALT can employ its model and testing procedures only at the most elementary level; it corrects syntax but not logic. It cannot diagnose the student's programming ability in complex tasks.

Koffman and Perry (1976) evaluated the effectiveness of MALT's digital systems design curriculum by a one-semester experiment with two randomly assigned (possibly quasi-random, in view of the use of a university scheduling algorithm) classes of about 30 students. Previous mean grade point averages and mean pretest scores favored the control group, but the mean final exam scores showed a small, nonsignificant difference in favor of the CAI group, despite the fact that its members spent one-third less time in class.

The final system to be mentioned here is Gilkey and Koffman's (1974) and Gilkey's (1974) algebra system. Although it is a generative system, it cannot be classified as either didactic or tutorial. Based upon features of the Koffman and Blount (1975), Gilkey (1974), Gilkey and Koffman (1974), and Koffman and Perry (1976) MALT system, this program also depends upon a probabilistic grammar for both problem and answer. Gilkey and Koffman are working primarily with word problems, and their system contains skeletal frames that the grammar fills to generate sentences. The generated problem is presented to the student, who is asked to translate the sentence into an equation. When this is done correctly, the student then solves the equation. The system monitors both the equation transformation and the solution.

Two terms that are partially synonymous with information-structure-oriented CAI deserve mention: mixed-initiative CAI and generative CAI. Mixed-initiative CAI literally means computer-assisted instruction in which either the computer or the student can take the initiative in determining the course of the instructional session. Depending on the nature of the specific system, the student taking the initiative might select the subtopic of curriculum to be considered next, ask natural language questions of the computer, or ask for specific aids by using formatted responses. In generative CAI, specific questions need not be stored for presentation.

Rather, a procedure for generating members of a category of items exists, and presentation of items is random or quasi-random within the category or groups of categories from which generation is taking place at a given time.

Do these kinds of CAI differ greatly? Gilkey and Koffman (1974) call their system of teaching high school algebra a generative one, which it is. However, that system may also be called ISO in that it uses a grammar to generate equations or word problems, and uses a concept tree for the total curriculum as a means of ordering topics. Gilkey and Koffman permitted students to ask for help or ask to start a problem over. They also were developing a desk calculator mode in which the student could ask that specific calculations be performed. Apparently, their system began with little student initiative but was becoming a true mixed-initiative system.

While the original distinction made by Carbonell (1970b) was between ISO and AFO systems, it seems that AFO systems will be increasingly generative and at least partially information-structure-oriented, making a critical distinction in the future between mixed-initiative and computer-initiative systems, with the former corresponding to ISO systems and the latter corresponding to AFO systems. This is not to overlook wide differences in degree of information structure--SCHOLAR and SOPHIE clearly employ a semantic network of knowledge while other systems, like that of Gilkey and Koffman, employ a more restricted grammar than that of natural language and thus have a more specialized network of knowledge. Accordingly, four kinds of generative systems can be distinguished: mixed-initiative systems with or without semantic networks and computer-initiative systems with or without semantic networks.

Affective and Didactic Instruction

Affective education may be defined as instruction intended to influence students' affective behavior in specific ways in addition to developing their cognitive capabilities. Affective educators seem to believe that students' personal well-being is the prime goal. The attainment of intellectual skills can contribute to that well-being but may sometimes have to be sacrificed temporarily if it interferes with personal equanimity and feelings of self worth.

The theoretical focus of affective education differs from author to author, sometimes lacking specificity in terms of the processes by which certain predicted effects will be obtained when affective methods are employed. Shiflett (1975), however, is quite specific:

There is no topic or goal within conventional curricula which does not have an integral affective component. Loadings are those affective aspects of all learning tasks stemming from basic concerns or not, which, if taken into account, may enrich personal meaning, increase relevance, and broaden understanding in a manner not possible, or only haphazardly done, by focus on the cognitive dimensions alone. (p. 127)

He further mentions three loadings, which he calls orientation, engagement, and accomplishment. Orientation loadings, which are affective responses related to a desire to learn, include an affective readiness to learn specific material, tempered by affective responses to one's known capabilities to learn that material. Engagement loadings are affective responses associated with the classroom setting and with the learning experience itself. Accomplishment loadings are affective responses associated with the completion of a learning task, thus being the emotional counterparts of reinforcement and knowledge of results in traditional learning theory.

Given the foregoing analysis, affective education can be viewed as the attempt to further personal well-being in general with special attention being given to orientation, engagement, and accomplishment loadings in school situations. Affective educators assume that improved affect during orientation, engagement, and accomplishment will facilitate cognitive development.

Affective educators may be expected to engage both in activities that focus upon each of these loadings separately and in others relevant to two or more loadings at one time. So-called games and exercises are common activities in affective classrooms. For example, a "blind walk" in which one person is blindfolded and led around a building or campus by a fellow student has the apparent purpose of testing and building a trust between the two. A more important purpose from an academic standpoint is a possible increase in trust in the teacher and in contentment with the classroom experience. Thus, the blind walk may help to build more favorable orientation and engagement loadings, to use Shiflett's terminology. A second example would be the use of relaxation instructions when problems are encountered with classroom work. If successful, they would improve engagement affect and facilitate learning.

An a priori case in favor of or in opposition to these hypotheses is easy to make. Alternatively, one can frame an affective model with terms so theoretical that the model is ipso facto true and the only problem remaining is to find empirical referents for terms such as orientation loading. Some research on the effectiveness of affective education exists and will now be considered. However, it should be noted that most affective educators are clinically oriented and therefore tend to offer evidence from case studies or uncontrolled experiments rather than from studies with refined methodology.

For example, Good, Biddle, and Brophy (1975, p. 200) are not critical of Kifer's (Note 2) assumption that one can learn from correlational data (rather than experimental data) whether students who customarily succeed in school develop a favorable concept of their own ability because of that success. Kifer found positive relationship between achievement and affective measures such as self-concept, self-esteem, and locus of control; and found that this relation was greater in grade seven than in grade five. Good et al. (1975) realized that these data could also be interpreted as consistent with the assumption of a reverse effect from the affective domain to the cognitive or of reciprocal effects in both directions. However, they

do not mention the possibility of no causal effect whatever, and seem to indicate that a correlation between an earlier event and a later event implies that the former has influenced the latter. This example is one of many in which educators have let a correlation between affective and cognitive variables lead to a conviction that the development of certain properties on one have causal relations to the other.

Whether called affective education, humanistic education, confluent education, or psychological education, the field under discussion here embraces a variety of viewpoints, ranging from literary (Heath, 1971) through relatively nonempirical (Ivey & Alschuler, 1973), to academic and social psychological (Kahn & Weiss, 1973). The current review emphasizes the limited segment of literature on affective education, which deals with the short-term effects of such education upon progress in academic subjects or upon attitudes toward those subjects. The review is relatively critical of the field for frequently producing poorly controlled studies and poorly presented reports of those studies. Such criticism may be fair but is done gently because affective education is a new field and has not yet developed a strong research commitment. The reader should be aware that the present focus upon academic attainment gives only a partial picture of the domain of affective education, with much less emphasis on questions of personality development, achievement motivation, and personal values than most workers in the field would give. For general background the reader might well examine the sources already cited in this paragraph, plus Brown (1975) and Brown, Phillips, and Shapiro (1976).

In an experiment on the effects of affective education on high school students, Stanford (1972) assigned an unstated number of students at random (subject to scheduling limitations) to eight seminars, of which four were experimental and four were controls. Experimental groups received what were called psychological education activities, much like the methods of Brown (1971, 1975) and others who call themselves confluent educators. Control groups followed normal procedures for seminars in the innovative English program of the school under study. Experimental and control groups were paired by subject matter, with two pairs studying composition and grammar and two pairs studying nonfiction and science fiction.

Stanford's report presented the results only briefly, and often involved significance tests comparing pre- and posttest course performance within groups rather than between groups, thus making interpretation difficult. Social sensitivity and leadership, measured with the Project Talent Personality Inventory, may have been superior in the experimental groups, but all that is certain is that significant mean improvement occurred only in the experimental group. A pupil reaction questionnaire showed that a significantly greater percentage of students in the experimental groups than in the control groups judged themselves to feel close personally to their teacher and other members of the seminar. With teacher-developed tests there were no significant mean differences between intellectual course achievement of the two sets of groups. On department-wide tests, both experimental and control groups showed significant mean improvement from pre- to posttest, with a higher level of significance in the experimental groups. Stanford's research permits the tentative conclusion

that some affective results were superior with the experimental groups and that intellectual growth during the course was at least as good with those groups.

A second high school study (Sigal, Braverman, Pilon, & Baker, 1976) compared 87 students given sensitivity training in small (15 students) encounter groups to 270 students in larger (25 to 30 students) control classes and to 52 students in small (15 or fewer students) classes that were typically remedial or nonacademic. Assignments to groups were not random. On standardized personality tests, few significant differences appeared at the end of the experiment, with most of those obtained favoring the control groups. No cognitive measures were reported. Anecdotal reports, such as an indication of reduced trouble with potential delinquent students after special treatment in the experimental condition, make the sensitivity condition seem more successful than the objective data just cited.

Shiflett and Brown (1972) compared the results of confluent training and traditional training of students in an elementary school teacher preparation program. The confluent group consisted of 21 students; one control group consisted of 30 students in the same program (given traditional training); and a second control group consisted of 19 students in the same program, who were posttested only. Students were assigned to the three groups arbitrarily rather than randomly. Comparison of the confluent groups teaching behaviors showed significantly more informality in the teaching behavior of the confluent student teachers. Higher informality ratings were found to be significantly associated with higher self ratings of existential mastery (personal competence). Existential mastery was also significantly higher for the confluent group than for the two control groups. Nonsignificant group differences appeared on several other self ratings and teacher behavior ratings.

Callan (Note 3) hypothesized that moral reasoning could be facilitated by training in Kohlberg's (1969) theory of moral reasoning, with greater facilitation if affective methods as well as cognitive ones were employed. Forty volunteer teachers in a private girls' high school were randomly assigned to two teacher training programs of 20 teachers each; one with both cognitive and affective training in moral development and one with cognitive training only. A somewhat comparable group of 10 teachers in another private girls' high school served as a control group, which received no training. Callan found no significant group differences between them on any one (or the aggregate) of four measures of moral judgment. She also found no significant differences in profiles over the four measures for these groups. She did find significant differences between the control group and the other two groups on the aggregate of six behavioral measures from the Flanders Interaction Scale, with the two experimental groups having higher ratings on acceptance, use of student ideas, asking of questions, and student talk (responsive or self-initiated), acceptance of feelings, praise, or encouragement. However, there was no significant difference between the two experimental groups. Accordingly, one must conclude that cognitive training alone had as great an effect on behavior as cognitive training plus affective training.

Callan cites a study by Harris (1977) that compared changes in the mean moral maturity scores of 11th grade students as a function of type of training. Her report of that research indicates that a group receiving both discussion of moral dilemmas and psychological awareness education showed significant improvement in moral maturity; it appears that a group receiving only moral discussion either did not show improvement or did not show as much improvement as the former group.

Cantlay (1975) compared three intact (nonrandom) groups of ninth grade mathematics students. One group received confluent remedial instruction, and a second group received traditional remedial instruction. The third group received nonremedial traditional mathematics instruction. Using corresponding pretest and posttest scores as predictors and values to be predicted, respectively, Cantlay performed analyses of covariance showing significantly higher adjusted mean scores for the confluent group than for the other two groups on 3 out of 11 items in the Pro-math attitudinal section of the National Longitudinal Mathematical Abilities Battery. Similar results are shown on other attitudinal sections of the same battery. Significant superiority also appeared for 1 out of 12 items in the Self-concept of Ability section of the Brookover Self-concept Scale.

A nonstatistical finding was a trend over five tests given only to the confluent group: The percent correct of operational skills problems remained constant, but the percent correct of story problems increased. Any inferences from this finding must be considered unproven because different material was covered in each test. However, one possible explanation is that the overall level of performance improved on story problems but not on numerical problems because confluent education can tap a richer network of information with the former type of problem than with the latter. The experiment being conducted in the current project and summarized in the section of this report entitled Summary of Design of Experiment in Progress, includes design features necessary to test the robustness of the original finding and to provide a preliminary test of the hypothesis just stated.

At least four evaluation reports have indicated some of the effects of affective education upon certain forms of academic achievement in the Philadelphia public schools (Gollub, Note 4; Gollub & Mason, Note 5; Young, Note 6; Loue, Leibovitz, & Sklar, Note 7; Loue, Note 8). Matched but nonrandom groups appear to have been used most often. Significant mean superiority of affective groups at the elementary, middle, and high school levels was common but not universal, with significance being more likely on reading comprehension tests than on vocabulary sections of those tests.

As learning theorists might predict, Loue et al. (Note 7) and Loue (Note 8) found significant ($p < .001$) superiority of the affective groups in a study of Grades 3 through 6 when the affective teachers used specific strategies to help students develop histories of success in reading. (Note 7 reported only a $p < .10$ effect, clarified by Note 8 as a $p < .001$ effect.) This effect held for both vocabulary and comprehension sections of the Gates-McGinitie Reading Tests. In contrast, without such strategies being in use, Young (Note 6) found no significant differences between affective

and control students' reading scores. McCarthy (1976) failed to produce significant superiority on either of two reading comprehension tests for a confluent college group compared to a randomly assigned control group, each group having been taught for 2 hours per day for 6 weeks. The confluent group did prove significantly superior on two self-concept scores, both derived from the Bills-Vance-McClean Index of Adjustment and Values.

Affective aims may also be sought with teaching methods not labeled affective education, of course. Block and Burns (1977) provide relevant information in their review of mastery learning, a teaching technique in which students must pass a test on one unit of a subject at a stated level (e.g., 80% correct) before studying the next unit. Most of their summaries of mastery learning--or the closely related Personalized System of Instruction (PSI) advocated by Keller (1968)--showed favorable affective results compared to those of conventionally taught classes. Typically, these effects were significantly increased interest in, or more favorable attitudes toward, the subject matter; a more favorable attitude toward the teaching method; improved academic or general self concept; or a more cooperative attitude or more academic confidence. However, one study showed significantly greater anxiety for a mastery learning condition, and 8 out of 10 comparisons (usually employing PSI) showed higher rates of withdrawal from courses using mastery procedures than from control classes; significance tests were not performed in these 10 cases. Since Block and Burns (1977) also report general superiority of mastery groups on subject matter pretests, the mastery conditions seem generally favorable to both cognitive and school-related affective performance.

Block (1972) earlier pointed to an inverse relation between affective and cognitive effects at certain levels of mastery. He used a questionnaire to measure interest in matrix arithmetic (tendency to seek out activities, skills, and understandings associated with it) and attitude (emotional tendency to act in a positive or negative way toward it), taking these measurements immediately after a four- or five-session training series (short-term measures) and again 2 weeks later (intermediate-term). Comparison of the achievement of five groups (nonmastery, 65% mastery required, 75% mastery required, 85% mastery required, and 95% mastery required) showed a regular increase in average achievement test percent correct, with the nonmastery group scoring between the two lowest mastery groups. However, the corresponding monotonically increasing trends for short-term interest and attitude means were broken by the lowest means occurring in the 75 percent group. Mean short term attitude (but not interest) also declined slightly from the 85 percent to the 95 percent mastery conditions. Intermediate measures of both interest and attitude were monotonically increasing over mastery groups except for declines from the 85 percent to the 95 percent mastery conditions. Nonmastery group interest and attitude scores were low throughout both tests but were not perfectly correlated with their relative achievement scores. These findings are not wholly consistent with each other, but they do suggest, at least, that 85 percent may be as high a mastery level as should be sought with the subject matter age group (8th graders) in question unless one is willing to sacrifice affective gains for a final increment of achievement.

Further information about attitudinal factors in mathematics learning is given in a review article by Aiken (1970). That author places special emphasis upon teacher characteristics. If those characteristics can be precisely specified, then it seems possible that CAI systems with those characteristics may be at least partially realizable, possibly by combining aspects of cognitive and affective instruction.

APPENDIX B

STRUCTURAL UNITS OF THE UCSB
COMPUTER-ASSISTED INSTRUCTIONAL SYSTEM

STRUCTURAL UNITS OF THE UCSB
COMPUTER-ASSISTED INSTRUCTIONAL SYSTEM

1. A set of programs for external bookkeeping, student login, and instructions to the student not handled elsewhere.
2. A first-day unit of instruction on use of the computer and of the instructional systems.
3. A problem generator for numerical problems, word problems, and story problems. Selection of items within a module, such as negative numbers or fractions, is essentially random and subject to restrictions on difficulty level within various parts of the module. The calculation of correct answers, and coding of student questions, is also performed here.
4. A unit for coding student responses to specific questions and for keeping track of whether or not a performance criterion has been met for the current curriculum module.
5. A user-called system for generating and displaying ANSWER, COMPUTE, QUIT, HELP, HINT, NEW MODULE, and SOLVE aids.
6. An English interpreter to permit limited student inquiries via the routines listed above.
7. A special program package for use in a humanized education condition. This condition is intended to improve student rapport with the instructional system by introducing personalizing features and informalities into the basic procedure.
8. A monitor to route other programs and data in and out of core memory.
9. A control unit that aids in module selection by retrieving the previous session's data and examining how far the student had progressed at the end of that session.

APPENDIX C

DESCRIPTION OF CURRICULUM MODULES
WITH SAMPLE PROBLEMS

DESCRIPTION OF CURRICULUM MODULES WITH SAMPLE PROBLEMS

Level	Sample Problem	Restrictions
Negative Numbers Module ^a		
1A	$-2 \times 3 =$	Any operator. Positive and negative integers from 1 to 9.
1B	$-(2 + -3) =$	Any operator. Positive and negative integers from 1 to 9.
2A	$(-8 + 2) \times 7 =$	Any operator (except division within parentheses). Positive and negative integers from 1 to 9.
2B	$-3 \times (6 - 5) =$	Any operator (except division within parentheses). Positive and negative integers from 1 to 9.
Word	Is the product of a negative integer and a positive integer positive or negative?	No restrictions.
Story	On Monday, Andy had a bank balance of \$52. On Tuesday he wrote a check for \$23. On Wednesday he wrote a check for \$36. What was his balance after writing the two checks?	No restrictions.
Factors and Prime Numbers Module ^a		
1	What are 2 factors of 24?	Numbers from 2 to 50.
2A	The set of all factors of 36 is:	Selected numbers with at least 3 factors.
2B	What are the prime factors of 36?	Any nonprime number under 50.
Word	The complete factorization of a certain number is {3 2 2 1}. What is the number?	Any prime number under 50.
Story	A 6-by-8 rectangle is covered with squares that are all the same size. What is the largest square that can be used?	Select numbers from 6 to 50 that have a common factor other than 1.
Decimals Module		
1	$\begin{array}{r} \$2.43 \\ + .28 \\ \hline \end{array}$	Up to 2 digits before decimal point, and 2 digits after decimal point. Addition or subtraction. Dollar sign present or absent.
2	$\begin{array}{r} 4.21 \\ \times 7.2 \\ \hline \end{array}$	Up to 2 digits before decimal point of multiplicand or multiplier. Up to 2 digits after decimal point of multiplicand or multiplier. Values from 2 to 9.
Word	Find the equivalent fraction. Type the value of its numerator. $.23 = \frac{\quad}{100}$	Numerator from .01 to 1.99.
Story	Steel rods measuring 7.03 inches, 8.25 inches, and 6.5 inches were cut from a rod 25 inches long. How many inches were left after the three pieces of rod were cut off?	No restrictions.

^aModule actually used in present experiment.

Level	Sample Problem	Restrictions
Absolute Values Module		
1	$ -41 =$	Positive or negative integers from 1 to 99 in absolute value.
2	$ 24 - (-35) =$	Pairs of positive or negative integers from 1 to 99 in absolute value. Intervening operator either + or -.
3	$ -12 - -22 =$	Pairs of positive and negative integers from 1 to 99 in absolute value. Intervening operator either + or -.
Word	If the absolute value of a number is greater than the number itself, is the number itself negative or positive?	No restrictions.
No Story Problems		
Fractions Module ^a		
1	$1/3 + 1/2 =$	Addition, subtraction, multiplication, or division. Terms are either proper fractions with denominator from 2 to 9 or mixed fractions with an integer from 1 to 3 plus a proper fraction of the kind just stated. Terms may be negative or positive.
2A	$1-1/2 \times (1/4 - 2/3) =$	Same as for Level 1.
2B	$(1/2 + 1-1/5) \times 2/7 =$	Same as for Level 1.
3	$(1/2 + 1/4) \times (2/3 - 1/5) =$	Same as for Level 1.
Word	Which is not equal to the others: 1/6, 4/5, or 4/24?	All three quantities between 0 and 1 in value. One of the three is the reduced form of one other. The third has a different value.
Story	A turtle traveling from San Francisco to Los Angeles travels at the rate of 3/4 MPH. How many miles will it go in 1-1/2 hours?	No restrictions.
Exponents Module		
1	$3 ** 2 =$	Base 1 to 5. Exponent 0 to 6.
2	$2 ** (2 - 1) =$	Base 1 to 5. Each digit in exponent from -2 to +2. Zeroes not displayed. Base 1 to 4. Zeroes not displayed.
3A	$(3 ** 2) ** -1 =$	Base 1 to 4. Exponents -2 to +2.
3B	$(2/3) ** 3 =$ (Do not perform the division until after all exponentiation is complete.)	Multiplication or division of the base numbers which must each be an integer between -2 and +2.
Word	The third power of 3 is 27. What is the third power of 2?	No restrictions.
Story	A certain perfect cube has its height, width, and depth equal to 4 inches. How many perfect 1-inch cubes will fit inside it?	No restrictions.

^aModule actually used in present experiment.

Level	Sample Problem	Restrictions
Square Roots Module		
1	25 ** (1/2)	All exact squares from 1 to 225. Also all such squares divided by 100 or multiplied by 100.
2	250 ** (1/2)	Numbers in the range from .01 to 22,500. No more than two places after the decimal point for any problem. Answers rounded to the first digit after the decimal place.
3	25 ** (3/2)	All exact squares from 1 to 225.
Word	The square root of 16 is 4. What is the square root of 36?	No restrictions.
Story	A cube holds 27 perfect 1-inch cubes. How tall is the cube?	No restrictions.
Inequalities Module ^a		
1A	Please choose a or b below: a. $-2 > -3$ b. $-2 < -3$	Inequalities in opposite directions in a and b. The same two distinct numerical values in a and b, each from -1 to -9 or 1 to 9.
2A	If $A > B$, which is larger?	Either direction of inequality. Letters different but otherwise arbitrary. "Larger" or "smaller."
2B	If $A > B$, is $A + 4 < 4 + B$?	Same digit on each side of inequality. Different letters on each side of inequality. Same letters for the two inequalities. Either direction for each inequality.
2C	If $A < C$, is $-2A > -2C$?	Same positive or negative multiplier for each side of second inequality. Either direction for each inequality.
3	If $A > C$ and if $C > B$, is $A < B$?	First two inequalities of same direction (arbitrarily chosen) and third inequality unrestricted in direction.
Word	If X plus 2 is greater than 6, name about as small a value of X as you can think of.	No restrictions.
Story	If Bill is older than Jim and Jim is older than Ned, is Ned older than Bill?	No restrictions.
Simple Linear Equations Module ^a		
1	$5 + X = 10$. Solve for X.	Two not necessarily different integers, each from 1 to 10. Operator may be +, -, x, or /.
2A	$2X + 5 = 8$. Solve for X.	Three not necessarily different integers, each from 1 to 10. Operator may be +, -, x, or /.
2B	$5a - 3a = 6$.	No squared terms permitted. A unit coefficient for X will not be shown.
3A	$2a \times b = c$. Find a.	One integer from 2 to 9. Operators may be +, -, x, or /. A unit coefficient for a will not be shown.
3B	$2a + 5 = c$. Find a.	Two not necessarily different integers, each from 1 to 10. Operator may be +, -, x, or /. A unit coefficient for a will not be shown.
Word	If two times a number is increased by three, the result is four less than the number. What is the number?	No restrictions.
Story	Joe's coin collection contains only nickels and dimes. The total amount of money in his collection is \$2.50. How many nickels does he have.	No restrictions.

^aModule actually used in present experiment.

Level	Sample Problem	Restrictions
Polynomials Module		
1	If $Z = 3X + 2XY$ with $X = 1$ and $Y = 2$, what is the value of Z ?	Two terms on right-hand side of equation. Coefficients from 1 to 5; may be either positive or negative. XY may be replaced by a squared term X^2 .
2	There are two factors of $9X^2 + 5X - 2$. Type one of them. (After getting one correct) Now type the other factor.	Three terms on left-hand side of equation for quadratic equations. Four terms allowed if expressions involving X , Y , and XY (but not X^2 or Y^2 are used). Integer solutions only; no solution outside the range from -4 to $+4$.
3A	If $9X^2 - 6X + 1 = 0$, find two possibly identical factors on the left-hand side, but do not type them. Then type a solution from one of those factors. (After getting it correct.) If there is another distinct solution, type it. Otherwise, type none.	Middle term on left may be missing. Integer solutions only; none outside the range of -4 to $+4$.
3B	If $12 - 4X + 3Y - XY = 0$, find the solution for X . (After getting it correct.) Find the solution for Y .	Integer solutions only; none outside the range -4 to $+4$.
No word problems		
No story problems		

^aModule actually used in present experiment.

APPENDIX D
FIRST SESSION PROCEDURES

FIRST SESSION PROCEDURES

The student in the Didactic ISO condition receives the following training in computer use (hypothesized student responses are preceded by an asterisk, and explanatory additions are in parentheses):

Please type: (hello)
will show you how.

The experimenter

* (hello)

Please type your first and last names.

*Tim Jones

Thank you. Welcome to this series of lessons in arithmetic and algebra. The first thing to be done is to learn how to use the machine you are now looking at.

HOW TO USE THIS COMPUTER TERMINAL

1. This is very much like a typewriter. It has keys to press for letters, numbers, etc.
2. Whenever you see the symbol *, you are to type something. Never start typing until you see the *.
3. First you type the kind of message the computer asks for. Then you push the RETURN key to tell the computer you are finished. Never type a message longer than one line in this experiment.

Go ahead now and type any kind of a message you want.

* Hello world!

Very good!

4. When you see READY? *, do as much reading of the screen as you want. Then press RETURN, and new material will appear.

READY? *

5. For symbols like the + and the ; sign (on the right-hand side of the keyboard, second line from the bottom), you must hold down the SHIFT key while typing. Then release it.

Go ahead and type something like:

He's a good friend

or anything else you want. Remember to press RETURN when you are finished.

* He's a jolly good fellow

Fine!

6. To type a fraction such as 2/3, type the numerator 2. Then type the slash / (under the ? in the bottom right-hand corner of the keyboard). Then type the denominator 3. If that is the end of your message, be sure to press RETURN to let the computer know you're finished. Now go ahead and type a fraction. Then press RETURN.

* 2/5

Excellent!

7. If you want to type a 'one', be sure not to type a lower case 'L.' Use the '1' in the top left-hand corner of the keyboard.
8. Follow directions carefully. If you make a mistake and then press RETURN, there is no way to take your answer back. But, if you want to change something before pressing RETURN, here is what to do: First, find the CONTROL button (marked CTRL) on the left-hand side of the keyboard. Hold it down. Then type a little H and that will erase one symbol.

READY? *

You will see that the last symbol is erased on the screen and the underlining marker has moved back a space. If you to erase several symbols, type 'h' several times while holding CTRL down. Then go back to normal typing. You will learn this correction procedure now.

Don't type anything until these instructions stop and the * sign tells you to go ahead.

First type: run. Don't capitalize anything like this that is shown in little letters. Next hold CTRL down with a finger on your left hand. Then type: h three times with your right hand. Did 'run' disappear completely? If so, take your finger off the CTRL button. Then type: walk. Go ahead now. Always press RETURN when you finish.

* walk

That is just fine!

9. Sometimes the computer may do something really strange. In that case, ask the experimenter to help you.

10. We don't have a pretty symbol for multiplication, so we use a lower case x. But to name a quantity, we use capital letters such as X, Y, and Z.
11. Usually you won't have to type anything as complex as $2 - 3$. But sometimes you will need to do so. So try now to type:
 $2 - 3$. Be sure to leave one space on each side of a '+' or '-.' Then press RETURN.

* $2 - 3$

Very good!

12. Please don't ever use a capital letter except when you type the first letter of your first or last name. You can also use a capital letter for the name of a variable such as X as we discussed earlier.

READY? *

INSTRUCTIONS ABOUT LEARNING THE LESSONS

You can learn everything you need from the computer. However, there is a typed textbook on the table for your use during the experiment. If a problem is unclear to you, look at the textbook for help. The textbook also contains the computer instructions and these instructions about learning. Be sure to convert answers to their simplest form. If you are working with fractions, you will say $2/3$ rather than $4/6$. You will also say $1-1/3$ instead of $4/3$. The lessons will help you to remember this point.

Scratch paper is always provided so that you can work on a problem before typing your answer.

The way that you can learn with this computer is to answer questions and then see if you are correct. You will work on topics such as negative numbers, factors, prime numbers, and fractions.

You will control the speed at which you will move from one type of problem to another.

READY? *

Here are some ways to help you learn:

1. Watch what the computer says and think carefully about your answers.

2. If you want to know the answer to a problem, type: answer and then press the RETURN key. Let's try this.

$$-2 \times 3 =$$

Go ahead and type: answer

* answer

The answer is -6

Very good!

3. Another way to get help is to type in: solve and press RETURN. When you do this, you will be able to see how the answer you want is obtained.

Now you may try the 'solve' command. Wait for the problem to appear. Then type: solve and press the RETURN key.

$$-2 \times 3 =$$

* solve

This problem

$$-2 \times 3 =$$

is to multiply a negative number, -2, and a positive number, 3, together. When a negative number is multiplied by a positive number, the product is always negative. So the solution to

$$-2 \times 3 =$$

is -6

READY? *

4. You can also use the computer like a calculating machine to get answers to parts of a problem. If your current problem is:

$$-2 \times (3 - 7) =$$

you might type: compute 3 - 7 and press the RETURN key. This would give an answer to part of the total problem. Go ahead and type the line just suggested. Use the exact spacing as shown on the screen. One space before and after each number or minus sign.

* compute 3 - 7

The answer is -4

Beware of one problem: COMPUTE cannot answer questions using decimals. So please don't give it any decimal problems. Thank you.

5. Any time you are in trouble, you may type the word 'hint' and press the RETURN key. Read the hint and do what is suggested to solve the problem. If you still have trouble, ask for 'hint' again. Let's try this. Wait for the problem to appear. Then type: hint and press the RETURN key.

$$-2 \times 3 =$$

* hint

This problem

$$-2 \times 3 =$$

is to multiply a negative number, -2 and a positive number, 3, together. When you multiply a negative number and a positive number together, you always get a negative product.

6. Now you know most of the main ways to get help with your studies. To be reminded of those ways or to learn some new ways you can get help, type: help and then press the RETURN key. Go ahead and do this now.

* help

WAYS TO GET HELP OR TO CONTROL THE LESSONS

To learn how to respond on a current unit such as fractions, type: instructions.

To obtain an answer to a problem, type answer.

To obtain a partial answer such as the value of 3 - 7, type: compute 3 - 7 or replace 3 - 7 above by whatever else you need. Do not use any decimal points with this command.

To see how a specific problem is solved, type: solve.

READY? *

At the beginning of each new phase of this experiment, you will be asked to choose the difficulty level of the problems you will choose. Later on, if you want to change this difficulty level, type: level.

To obtain a hint for the problem you are trying to solve,
type: hint.

READY? *

To skip the problem shown to you and go on to a new problem,
type: skip.

To obtain a new kind of problem, type: module.

To stop this session before it would normally end, type:
quit.

To obtain these instructions, type: help.

Now you can begin to learn your first math lesson on the
computer.

READY? *

The preceding sets of instructions for the Didactic ISO condition come from the programs INSTR and SESS0. For the AFO condition, the only changes necessary in the previous three pages are to modify Number 3 in view of a change in SOLVE for that condition, delete Numbers 4 and 5 (deleting mention of COMPUTE and HINT), and to renumber Number 6 accordingly. The first session ends with about five minutes work on mathematics problems. When SESS0 is completed, MONITOR knows that no learning data are present. Therefore, in the ISO condition, MONITOR lists the different modules and asks the student to choose one. See Appendix G for changes in this session with students in the Confluent condition.

APPENDIX E

ANALYSIS OF HINT PROGRAM STRUCTURE
FOR THE FRACTIONS MODULE

ANALYSIS OF HINT PROGRAM STRUCTURE FOR THE FRACTIONS MODULE

The following displays all possible hints for a fraction problem, plus a list of goals and subgoals and an updating of HINTLIST development. The problem and hints are as displayed to students, but the hints have been numbered here for clarity. '*' indicates an experimenter's call of a function for display purposes.

*(fraction problem)

$$1/2 / (2/3 + 1-1/6) =$$

*hint

(1) the nested term,

$$2/3 + 1-1/6$$

should be solved first. You can then perform the division with the result and the remaining term.

*toplist

(NEST LAST2)

*solist

((NEST CONVERT 2 2 LCM ADD RETAIN) (LAST2 INVERT MULTIPLY))

*hintlist

((NEST LAST2))

*hint

(2) You must solve the nested term.

To do this I would suggest: The second term is mixed. It is a good idea to convert any mixed terms to fractions before you do the required addition.

*solist

((NEST CONVERT 2 2 LCM ADD RETAIN) (LAST2 INVERT MULTIPLY))

*hintlist

((NEST LAST2) (NEST CONVERT 2 2))

*hint

(3) You must solve the nested term.

To do this I would also suggest: Before the addition can be carried out, you must find the lowest common denominator for the two fractions you are working with. For instance, if you were working with the two fractions $1/3$ and $1/2$, you would have to convert them both to sixths. This type of conversion must be done when you add or subtract fractions but is not necessary if you are multiplying or dividing.

*hintlist

((NEST LAST2) (NEST CONVERT 2 2 LCM))

*hint

(4) You must solve the nested term.

To do this I would also suggest: After you have converted the two fractions to the least common denominator, you may add the fractions. To do this, you must add the two numerators (the top numbers) and place the sum over the denominator. You do not add denominators (the bottom numbers). So, $1/4$ plus $2/4$ would be $3/4$.

*hintlist

((NEST LAST2) (NEST CONVERT 2 2 LCM ADD))

*hint

(5) You must solve the nested term.

To do this I would also suggest: If you carry out the addition properly, your answer will be greater than 1. It would be 'correct' to write your answer as a mixed fraction; for example, $1-1/2$. However, because there are more terms in this problem, it will help not to do this extra step. Instead you will save effort by leaving the answer in its 'improper' form. In my example, $1-1/2 = 3/2$.

*hintlist

((NEST LAST2) (NEST CONVERT 2 2 LCM ADD RETAIN))

*hint

(6) After solving the nested term, your problem will have two terms remaining. You must solve the last two terms.

To do this I would also suggest: A simple rule to apply when dividing one fraction by another is to 'invert and multiply.' That is, change the second fraction such that the bottom number is on the top and the top number is on the bottom. Then change the operator from division to multiplication. You then multiply the fractions to get the proper answer.

*hintlist

((NEST LAST2) (NEST CONVERT 2 2 LCM ADD RETAIN) (LAST INVERT))

*hint

- (7) After solving the nested term, your problem will have two terms remaining. You must solve the last two terms.

To do this I would also suggest: As I described previously, you must invert one of these two terms and then multiply them. I have no more hints available. If you would like to watch me solve this problem step by step, just type 'solve.'

*hintlist

((NEST LAST2) (NEST CONVERT 2 2 LCM ADD RETAIN) (LAST2 INVERT MULTIPLY))

*hint

- (8) I have no more hints available. If you would like me to solve this problem step by step, just type 'solve.'

After the solution of the nested term and the substitution of that solution for the nested term in TOPLIST = (NEST LAST2), as discussed in the body of this report, the two subgoals of LAST2, INVERT, and MULTIPLY are sufficient to combine the remaining two terms of the problem into a single answer, which may not be in reduced form. Because the HINT program emphasizes getting a legitimate though not necessarily final answer, and because the scoring system allows partial credit for unreduced answers and leads to feedback specifying any failure to reduce an answer to its simplest form, a REDUCE subgoal is not included in LAST2.

The operation of HINT in generating the information of the preceding example may now be explained. The first hint is shown as (1). It advises the student to work with the nested expression first and then to do the rest of the problem. The specific hint given at any later stage depends upon the contents of HINTLIST, which tracks the progress of the advice given to the student. (It is not sensitive to other partial solutions developed by the student, assuming that a student will perform the process recommended by one hint before calling for another.) Before the second command, HINTLIST consists of

((NEST LAST2))

since the first hint has called attention to the two parts of the problem. So a hint about NEST is an elaboration of the previous hint about dealing with the nested term.

The student is told in the second numbered hint that it is a good idea to convert any mixed terms (integer plus fraction) to fractions (actually improper fractions) before doing the next step. Once this instruction has been given, HINTLIST becomes

```
( (NEST LAST 2) )  
(NEST CONVERT 2 2) ).
```

The third hint tells the student that, to solve the nested term, the lowest common denominator of the two component fractions must be determined. An example of such a determination is partially described. Now HINTLIST is changed to also include LCM:

```
( (NEST LAST2)  
(NEST CONVERT 2 2 LCM) ).
```

The fourth hint tells the student that solution of the nested term requires the addition of the numerators of the two fractions after they have been converted to the lowest common denominator. An example is given. Then HINTLIST becomes:

```
( (NEST LAST2)  
(NEST CONVERT 2 2 LCM ADD) ).
```

The fifth hint is the last concerned with the nested term. It advises the student not to convert the answer for the nested term into a mixed fraction such as $1\frac{1}{2}$ from its present improper form, such as $\frac{3}{2}$, since the reverse step would simply have to be performed later. This leaves HINTLIST in the form

```
( (NEST LAST2)  
(NEST CONVERT 2 2 LCM ADD RETAIN) ).
```

The sixth hint points out that all that remains to be done to solve the original problem is to deal with the two remaining terms. The student is advised to perform the required division by the rule "invert and multiply." HINTLIST now brings a new sublist involving LAST2 and a part of the information in this hint:

```
( (NEST LAST2)  
(NEST CONVERT 2 2 LCM ADD RETAIN)  
(LAST2 INVERT) ).
```

The final hint repeats the substance of Hint 6, advises the student that no more hints are available, and refers the student to SOLVE for a step-by-step demonstration of the solution. The final HINTLIST becomes

```
( (NEST LAST2)  
(NEST CONVERT 2 2 LCM ADD RETAIN)  
(LAST 2 INVERT MULTIPLY) ).
```

Notice that, in the ISO system, HINT and SOLVE (discussed more fully in the main text and in Appendix F) bear most of the responsibility of ordinary text in textbooks or of most previous CAI systems. Except for the reference textbooks available beside each terminal, students receive almost no information about a curriculum unit except in their solving of problems and in the feedback offered to them, the answer given when ANSWER is called, or in HINT and SOLVE.

A further interesting feature of this HINT system is that HINTLIST is generated by a process very close to the LISP operating of CARing items from a solution procedure list (SOLIST) and APPENDING them to HINTLIST. Each such pair of steps provides the occasion for writing a hint compatible with the subgoal being added to HINTLIST.

APPENDIX F

SOLVE PROGRAM STRUCTURE FOR THE FRACTIONS MODULE

SOLVE PROGRAM STRUCTURE FOR THE FRACTIONS MODULE

Partial Production System for Solving Fraction Problems Using SOLVE

- P1 SOLVE Solution=term
 ((problem = (term =) => (satisfy SOLVE)) .
- P2 SOLVE
 ((problem not = (term =)) (problem has nested term) =>
 (frame = nested term) (setgoal NEST))
- P3 SOLVE
 ((problem not = (term =)) (problem has no nested term)
 (problem has more than two terms) =>
 (frame = term1 operator1 term2) (setgoal FIRST2))
- P4 SOLVE
 ((problem not = (term =)) (problem has no nested term)
 (problem has only two terms) =>
 (frame = problem) (setgoal LAST2))
- P5 NEST
 ((frame = term) =>
 (problem = substitute frame for nested term in problem)
 (satisfy NEST))
- P6 NEST
 ((frame = (term op term)) => (setgoal COMBINE))
- P7 FIRST2
 ((frame = term) =>
 (problem = substitute frame for term1 operator1 term2
 in problem) (satisfy FIRST2))
- P8 FIRST2
 ((frame = term op term) => (setgoal COMBINE))
- P9 LAST2
 ((frame = term) =>
 (problem = (frame =)) (satisfy LAST2))
- P10 LAST2
 ((frame = term op term) => (setgoal COMBINE))
- P11 COMBINE
 ((frame = term) (term is in reduced form)
 (numerator less than denominator) => (satisfy COMBINE))
- P12 COMBINE
 ((frame = term) (term in reduced form)
 (numerator greater than or equal to denominator) =>
 (setgoal IM PROPER))

P13 COMBINE
 ((frame = term) (term not in reduced form) =>
 (setgoal REDUCE))

P14 COMBINE
 ((frame = term op term) (op = + => (setgoal ADD))

P15 COMBINE
 ((frame = term op term) (op = -) => (setgoal SUB))

P16 COMBINE
 ((frame = term op term) (op = X) => (setgoal MUL))

P17 COMBINE
 ((frame = term op term) (op = /) => (setgoal DIV))

P18 ADD
 ((denominator of term1 = denominator of term2) =>
 (frame = sum of term1 and term2) (satisfy ADD))

P19 ADD
 ((denominator of term1 not = denominator of term2) =>
 (setgoal LCM))

P20 SUB
 ((denominator of term1 = denominator of term2) =>
 (frame = difference of term1 and term2) (satisfy SUB))

P21 SUB
 ((denominator of term 1 not = denominator of term2) =>
 (setgoal LCM))

P22 MUL
 ((term1 or term2 is mixed) => (setgoal IMPROPER))

P23 MUL
 ((term1 and term2 are not mixed) =>
 (frame = product of term1 and term2) (satisfy MUL))

P24 DIV
 ((term1 or term2 is mixed) =>
 (frame = mixed term) (setgoal IMPROPER))

P25 DIV
 ((term1 and term2 are not mixed) => (setgoal INVERT))

P26 DIV
 ((frame = term) => (satisfy DIV))

The use of a production system will now be explained for readers to whom the idea is new, with reference to the above system. The productions of 27 LCM, 28 IMPROPER, 29 INVERT, 30 INVERT, 31 REDUCE, and new productions called by them are omitted to save space and complexity of display.

The use of a production system begins with the specification of a goal and a problem. For example, let the goal be SOLVE and the problem be the same one discussed in Appendix E:

$$1/2 / (2/3 + 1-1/6) = .$$

This allows the setting of SOLVE in Figure F-1, a history of successive goal stacks during problem solution.

At the beginning, the stack has an instruction to set SOLVE goal at the top of the figure. Each row below the top represents the stack at a later stage. The top of any stack is its rightmost goal.

The next step in analyzing the SOLVE process is to start at the top of the list of production rules (P1, P2, ...) and examine each to see if the goal at the top of the current goal stack is stated in capital letters at the beginning of the rule and if the condition or conditions given within the rule are met. If these requirements are satisfied, then the action following the arrow for that production rule is taken. In the second row of Figure F-1, the first goal is SOLVE. The first such rule activated by this problem is P2--the goal is SOLVE, the problem is not just to evaluate a single term, and the problem has a nested term. Therefore, the actions required by P2 are taken: a so-called frame becomes equal to the nested term $(2/3 + 1-1/6)$, and the goal NEST is set at the top of the goal stack. This brings us to the third row of Figure F-1, showing that the goal stack contains SOLVE and NEST, with NEST on top.

Once a production rule has been found to apply and the required actions have been taken, control passes to the top of the production list again. Now, a search takes place for an acceptable production with NEST at the top of the goal stack. P6 proves to be the appropriate production, for NEST is at the top of the goal stack and the condition (frame = (term op term)) is satisfied by the frame $(2/3 + 1-1/6)$, where op stands for operator and may be either +, -, x, or /, the four basic arithmetic operators. Therefore, the action required by P6 is taken: the goal COMBINE is added to the top of the goal stack. This establishes the fourth row of Figure F-1. Next, one must start again at the top of the production rule list and look for an applicable rule with COMBINE at the top of the goal stack. The same general procedure already outlined will continue until this problem is solved.

Further study of the Partial Production System would show that the complete sequence of productions to be applied (including the two already listed) is P2, P6, P14, P19, P27, P18, P12, P28, P11, P5, P4, P10, P17, P24, P28, P25, P29, P23, P30, P26, P13, P31, P11, P9, and P1. As each production is performed, a new row of Figure F-1 is generated, until the problem is solved by satisfying the top goal SOLVE and the figure is complete.

set SOLVE

SOLVE	set NEST					
SOLVE	NEST	set COMBINE				
SOLVE	NEST	COMBINE	set ADD			
SOLVE	NEST	COMBINE	ADD	set LCM		
SOLVE	NEST	COMBINE	ADD	LCM	satisfy LCM	
SOLVE	NEST	COMBINE	ADD	satisfy ADD		
SOLVE	NEST	COMBINE	set IMPROPER			
SOLVE	NEST	COMBINE	IMPROPER	satisfy IMPROPER		
SOLVE	NEST	COMBINE	satisfy COMBINE			
SOLVE	NEST	satisfy NEST				
SOLVE	set LAST2					
SOLVE	LAST2	set COMBINE				
SOLVE	LAST2	COMBINE	set DIV			
SOLVE	LAST2	COMBINE	DIV	set IMPROPER		
SOLVE	LAST2	COMBINE	DIV	IMPROPER	satisfy IMPROPER	
SOLVE	LAST2	COMBINE	DIV	set INVERT		
SOLVE	LAST2	COMBINE	DIV	INVERT	set MUL	
SOLVE	LAST2	COMBINE	DIV	INVERT	MUL	satisfy MUL
SOLVE	LAST2	COMBINE	DIV	INVERT	satisfy INVERT	
SOLVE	LAST2	COMBINE	DIV	satisfy DIV		
SOLVE	LAST2	COMBINE	set REDUCE			
SOLVE	LAST2	COMBINE	REDUCE	satisfy REDUCE		
SOLVE	LAST2	COMBINE	satisfy COMBINE			
SOLVE	LAST2	satisfy LAST2				
SOLVE	satisfy SOLVE					

Figure F-1. Goal stack trace for " $1/2 / (2/3 + 1-1/6) = .$ "

The utility command SOLVE is closely related to SUPERSOLVE, a program for finding solutions as needed in checking student answers and in facilitating the operation of ANSWER, HINT, and SOLVE. Now that the goal structure of SOLVE and the related goal structure of HINT have been shown, it is of interest to display an example of the computer's response to the student command SOLVE. The following example presents that response for the fractions problem under discussion.

$$1/2 / (2/3 + 1-1/6) =$$

This problem contains a nested term,

$$2/3 + 1-1/6.$$

It will be necessary to solve it first. The second term is mixed. This term, $1-1/6$, must be converted to sixths. There are 6 sixths in 1. 6 sixths plus 1 sixth is 7 sixths. So, $1-1/6$ can be converted to $7/6$.

So, now we must solve $(2/3 + 7/6)$. However, these terms must first be converted to their least common multiple. Terms must have the same denominators whenever you are adding or subtracting. It is not necessary if you are multiplying or dividing. The least common multiple for two fractions is the smallest number that both of the denominators will divide evenly. For example, the least common multiple for 2 and 3 is 6. This is because both 2 and 3 divide 6 evenly, and there is no smaller number for which this is true. As another example, the least common multiple for 5 and 10 is 10. Here, 10 divides itself evenly, as does 5. No other number which is smaller than 10 will satisfy this condition. The least common multiple for these two denominators, 3 and 6, is 6. We must convert $2/3$ to sixths. Since 6 divided by 3 is 2, both the numerator and denominator of $2/3$ must be multiplied by 2 to convert it to sixths. So, $2/3$ becomes $4/6$. Here $7/6$ must be added to $4/6$. We simply add the numerators and place the result over the denominator. So 4 plus 7 equals 11. Therefore,

$$4/6 + 7/6$$

is $11/6$. Because the numerator of $11/6$ is larger than the denominator, we could convert this fraction to a 'mixed' form. However, because we will need to operate on more terms in this problem, we will leave it in its 'improper' form. We have reduced the problem from

$$1/2 / (2/3 + 1-1/6) =$$

to two terms. We may now solve

$$1/2 / 11/6 =$$

directly. In order to divide $1/2$ by $11/6$, we must apply the 'invert and multiply' rule. That is, the second term must be changed such that the numerator and the denominator switch places. Then the two terms are multiplied. So,

$$1/2 / 11/6$$

becomes

$$1/2 \times 6/11.$$

In order to multiply fractions, the two numerators and the two denominators must be multiplied. To do this, multiply 1 and 6. This is the numerator of the answer. Also multiply 2 and 11. This is the denominator of the answer. So, $1/2 \times 6/11$ is equal to $6/22$. Notice $6/22$ can be reduced. To reduce a fraction, you must find the largest number which will divide both the numerator and the denominator evenly. For example, the fraction $5/10$ can be reduced. Both the numerator and the denominator can be divided by 5. When the division is carried out, 5 divided by 5 is 1 and 10 divided by 5 is 2. So, $5/10$ can be reduced to $1/2$. This makes sense because five-tenths of anything is the same as one-half. When we reduce $6/22$, we find that both its numerator and denominator can be divided by 2. When we do the division, we get 6 divided by 2 equals 3, and 22 divided by 2 equals 11. So, $6/22$ is reduced to $3/11$. So the solution to

$$1/2 / (2/3 + 1-1/6) =$$

is $3/11$.

A similar process is involved in the use of HINT and SOLVE routines in story problems. In an investigation of processing strategies for solving algebra word problems, Hinsley, Hayes, and Simon (1976) found evidence that people employ various "problem-type schemas" or "frames" that assist in translating story problems from prose to equations, or sets of equations, which may then be solved directly. More specifically, they found that:

1. People can categorize problems into types.
2. People can categorize problems without completely formulating them for solution.
3. People have a body of information about each problem type that is potentially useful.
4. People use category identifications to formulate problems in the course of actually solving them.

One might hypothesize that people solve story problems by the following process:

1. Identify the appropriate problem-type schema.
2. Identify and assign values provided by the problem to the proper nodes, or empty cells of the schema.
3. Carry out the operations on these values as specified by the schema.

Early assistance in solving story problems provided by HINT and the initial stages of the step-by-step solutions generated by SOLVE are largely structured by the theoretical process mentioned above. That is, the student is first helped to identify the appropriate problem-type schema and its values. If this assistance has not been sufficient, then it is assumed (depending upon specific characteristics of the problem) that either the student has failed to carry out arithmetic operations necessitated by the schema, or the schema was not available in the student's memory.

Rumelhart & Norman (in press) suggested that the most common method of producing new schemas may be the modification or combination of existing, nearly appropriate schemas. With this in mind, later HINTs and stages of SOLVE attempt to decompose the appropriate schema into combinations or modifications of schemas that are likely to be available to the student. The products of appropriate mathematical operations may also be provided.

Currently, the HINT and SOLVE routines are unable to decompose story problems, identify an appropriate schema, and then determine an appropriate sequence of instructional text. Story problems are generated from skeletons, or frames, of text containing variables. At the time of generation, values for the variables (both numerical and prose) are selected within the bounds of the story. Currently, a series of hints and a step-by-step solution also exist in skeleton form for each story problem type. When HINT or SOLVE is called for a story problem, these skeletons are filled by the variable values selected for the current story problem.

APPENDIX G

DETAILS OF DESIGN OF CONFLUENT (AFFECTIVE) PROCEDURE

DETAILS OF DESIGN OF CONFLUENT (AFFECTIVE) PROCEDURE

The Confluent procedure used in the CAI system makes little change in the substance of instruction or the mode of presentation of specific items. Most of the procedure is the introduction of additional material of an affective nature and a consequent small reduction in the amount of time available for presentation of mathematics problems to be solved. Part of the added material is intended to make the computer seem more like a friendly teacher who calls you by name, asks how you feel, and tells you how to relax when you are having difficulty. Another part is intended to allow face-to-face communication with another student to permit the sharing of experiences with the experiment and about the learning of mathematics. A third segment of the Confluent condition is pure entertainment, distracting the student from any anxiety introduced by the experimental situation. Details of the Confluent condition are now presented, beginning with Sessions 1 and 2, since they receive special emphasis in this condition.

Session 1

In the Confluent conditions, the initial instructions for signing on to the computer and using the computer terminal are slightly modified, personalized versions of instructions given in Appendix D. Next, a digression on astrology takes place.

Because the birthdate of each student was recorded when MAKFIL established a data file for that student, it is possible to determine the astrological sign for the student and to engage in the following sort of dialogue on Session 1, shown for a student born on March 22:

(Name), you are an Aries--are you really a pioneering and adventurous spirit?

*no

Are you courageous?

Are you direct in your approach?

*no

Are you highly energetic?

*no

Are you a hater of restriction and a lover of freedom?

*no

Since this exhausts the stored list of properties, the computer ends this episode dialogue with,

Well, so much for astrology!

If the student had said 'yes' to any of these queries, the computer would have closed the dialogue at that point with,

I knew that because you are an Aries.

Finally, a slightly modified section on how to use the computer is displayed, followed by about 5 minutes of mathematics instruction.

Session 2

Session 2 begins with each of two Confluent students signing into a separate terminal in the same experimental room. After a standard greeting, the first one to sign in then receives the following instructions:

Today you will work directly with the person at the other terminal in this room. Do not respond any more on this terminal until I ask you to, (name).

The second person to sign in receives these instructions:

(Name), now I would like you to get up and go over to your neighbor.

An additional message on the first terminal is given to the two persons to sign in (first name or mood of appropriate student goes in blanks):

Hi _____, glad you can join us. You were feeling _____ last time we met. And you, _____, were feeling _____ last session. How do you feel about working together? Please take turns telling me. First, _____.

*I don't know yet.

O.K., now what about you, _____?

*It's a change, so maybe it will be good.

Thanks, _____ and _____.

Now that you are both together, I would like both of you to talk together to review the first day's lesson. I want you to spend only a few minutes talking. Please start by asking each other what you are not quite sure of in using me. Some possible examples are:

Do you know how to log me in?

What happens when you don't press RETURN?

How do you write a fraction?

What kind of answers do I want?

How do you erase what you have written?

Go ahead and talk. When you hear me beep, stop talking and look at me again.

Now it is time to work on math problems. Thanks a lot, _____, for coming over and working with us. Please go back to your terminal _____.

Then both terminals say:

Please press RETURN if you are ready to continue when I give you the READY? message. Thank you.

After the student indicates that he is ready to continue, the computer concludes this portion of dialogue with the following:

O.K., _____, let's go on with our lesson.

General Procedure for All Sessions

Each session begins with the greeting,

Welcome to Session _____. You felt _____ last session. How do you feel today? I'm _____ and I'm here to help you again.

Whenever a student fails two successive items, he or she receives special instructions. Four options exist—two with advice about learning procedures and two with relaxation instructions. The first three options run sequentially across problems and repeat after each has been used. If a person makes eight successive errors, the fourth option used with that problem is Number 4 and is likely to lead to giving up on that problem.

1. Students are told that the task is unusual because information is provided during or after a problem rather than before it. Then they are reminded of the ANSWER and HELP routines and encouraged to try them.

2. The following instructions are given:

You're making mistakes now and that must be frustrating. Relax. You're doing fine. You'll soon find out what is going wrong. Take a minute to relax. Let me finish my instructions before you follow my next suggestions. Don't close your eyes until you have seen all my suggestions.

Please pay attention to your breathing. I want you to take deep breaths and let the tension out of your shoulders. Get in the most comfortable position you can in your chair. Close your eyes and continue breathing deeply. Imagine you are in your most favorite retreat, i.e., beach, mountains, living room

. . . . Find a nice comfortable spot and lie down. Let the warm sun soothe your body. Remember, this is just a fantasy, so anything can happen and everything is perfect. Just enjoy the setting around you.

READY?*

Imagine all the good things, sounds, smells, feelings that surround you. Continue to breathe deeply.

When you hear me beep, please open your eyes and come back to this room.

Please reread my directions as they appear again.

_____, it may be possible to do this exercise without closing your eyes. If you want to keep your eyes open while you fantasize, that is fine with me. Please begin your day dream now.

3. The following instructions are given:

Well, _____, things seem to be a little frustrating right now. I want you to take a few seconds and stand up and stretch. Don't worry about me. I will beep when I want you to come back to me to continue on to the next lesson. So keep stretching until you hear me beep.

For instructions 1 and 3, a beep recalls the student to the terminal and the session continues with:

Now we will go back to the problem that you were working on. I know it is frustrating when you can't find your mistakes right away. But if you relax, it will be easier on you.

The fourth set of instructions (following eight successive errors on a single problem) is substantially different.

Have you tried all the different aids I told you about? If not, go ahead and try everything listed when you type: help. Just take it easy and relax. Sometimes it takes a little time to figure out the problem--you are doing fine! So either type: 'stuck' if you can't think of anything to do or request one of the standards such as 'help; if you want to try them. (If answer was 'stuck,' continue with the following:) All right _____, since you have tried all the aids and the solution is not clicking for you right now, I'll give you a new problem. This will be a good time for you to stand up and take a breather for a second.

Other Features of the Confluent Procedure

Session 4 begins with a lesson on following directions and avoiding jumping to conclusions. Session 8 begins with a dialogue in which a pair of Confluent students discuss their early school experiences, emphasizing success and failure in mathematics as compared to other subjects. Session 11 begins with a lesson on abstract and concrete relationships.

END

DATE
FILMED

9 — 83

DTIC