A generative CAI tutor for computer science concepts*

by ELLIOT B. KOFFMAN

University of Connecticut
Storrs, Connecticut

INTRODUCTION

Limited progress has been made in software for computer-assisted instruction. Frame-oriented CAI systems have dominated the field. These systems function as mechanized programmed texts and utilize the computational power of the computer to a minimal extent. In addition, they are difficult to modify and tend to provide a fairly fixed instructional sequence.

The conventional frame-oriented CAI system is organized as a series of frames. A frame may present a piece of information and/or ask a question. The questions are normally objective and often are of the multiple-choice type. The frames are usually linked in a sequential fashion and a student will cycle through them one at a time. Frames may be presented on a teletype, a graphical display, a slide projector, via an audio track, or any combination of the above.

There are severe problems inherent in systems of this type. All questions must be specified by the course-author as well as a set of anticipated student responses to each question. If branching is to occur, explicit instructions must be given indicating the performance criteria for a branch and the new continuation point in the program.

Since everything must be specified in advance, extensive time must be spent in preparing course material for presentation. Furthermore, once programmed this material has very little flexibility. Modifying the set of questions to be asked of the student or the material to be presented is a major undertaking and much reprogramming must be done.

This type of system is not very useful in teaching quantitative courses. Subject areas such as engineering or the physical sciences are concerned with teaching techniques of problem solving. Problem solving competence is often acquired through a process of "learning by doing." Consequently, it is essential that the CAI system be capable of presenting a wide variety of problems and solutions to the student. Reprogramming each problem and its solution in a manner suitable for presentation by CAI would be extremely inefficient.

It is precisely for these reasons that generative CAI systems have recently become of great interest. Generative systems are capable of generating a series of questions (and answers to these questions) as a function of the student interaction. These systems can be divided into two classes. Those which are oriented toward the "soft-sciences" and textual material and those which are more concerned with numerical manipulations and quantitative material.

Carbonell1 and Wexler2 have designed generative CAI systems which have been used to teach concepts in geography. These systems are organized around an information structure or network. Carbonell uses the semantic network developed by Quillian.3

Once the information network has been specified, these systems are capable of generating a sequence of questions for a student. As each question is generated, the answer is retrieved for comparison with the student's response. If the student is incorrect, Wexler's system is capable of providing individualized remedial comments. This would consist of either a correct and relevant statement using the student's incorrect answer or a systematic presentation of the steps performed by the system in deriving the correct solution. Both these systems allow the student to interrupt and pursue topics which interest him at greater depth.

The potential for incorporating generative CAI in the "hard sciences" is extensive. Algorithms for solution of classes of problems could be incorporated into CAI systems. In some cases, solution techniques might be sufficiently complex that heuristic programs would be necessary. Examples of the latter case would be teaching symbolic integration or proving theorems. In

---

* This research is sponsored by the Connecticut Research Commission under Grant RSA-71-7.
any event, CAI systems organized around a set of algorithms would have the capability to generate and solve a wide range of problems.

An extensive project in the subject area of analytical geometry has been described by Utta1. His system is capable of generating twelve problem types which are representative of the problems found in an analytical geometry course. These problems usually involve an expression or graphical representation of a particular conic section. The expression is obtained from the general quadratic equation: \( AX^2 + BY^2 + CX + DY + E = 0 \).

The required expression is obtained by setting certain coefficients to 0 and selecting the others at random. The complexity of the equation generated depends on the constraints imposed on the coefficients. For example, to generate circles centered at the origin, \( A = B \) and \( C = D = 0 \).

Associated with each of the twelve problem types is an answer routine. The routine which determines if a randomly generated point \((x,y)\) falls on the locus represented by a randomly generated equation simply plugs this point into the equation. The expression generator itself is used as the answer routine when the student is asked to supply the equation for a conic section with given standard characteristics.

The following sections will describe a generative tutor that has been used in an introductory computer science course. It has been used to teach concepts of digital circuit design as well as to introduce students to machine language programming. Because of the large number of concepts covered, an intelligent "concept selector" has been designed which attempts to tailor the current instruction each student receives to fit his past performance record.

**GENERATIVE CAI IN DIGITAL SYSTEMS**

The system is designed to be extremely flexible in that it can completely control the progress of a student through the course, selecting concepts for study on an individual basis and generating problems. Alternatively, the student can assume the initiative and determine his own areas of study and/or supply his own problems.

In addition, the system also operates in a "problem-solver" mode. In this mode, the student specifies the concept area and his problem, and the system will crank out the solution without further interaction. It is anticipated that students in later courses and the digital laboratory will utilize this mode for solving complex minimization problems and determining the relative merits of different state assignments.

Figure 1 is a block diagram of the system functions. Subsequent sections of this paper will describe how these functions are accomplished. As has been mentioned, the student can assume the initiative and bypass the Concept Selector and/or Problem Generator (indicated by dashed lines from Student box). The bottom part of Figure 1 shows the student exercising full control in the problem-solver mode.

When the system is in control of the interaction, it attempts to individualize the depth and pace of instruction presented to each student. A model of each student is kept which summarizes his past performance in each of the course concepts. Table I shows the contents of a student record.

In addition, the system is supplied with a concept tree which indicates the degree of complexity (plateau) of a concept and its relationship with other concepts in the course. A sample tree for the concepts currently covered is shown in Figure 2. This, of course, represents the author's interpretation of the relationship between course concepts. There are alternate interpretations which are just as valid.

The system uses each student's record to determine how quickly he should progress through the tree of concepts, the particular path which should be followed, the degree of difficulty of the problem to be generated, and the depth of monitoring and explanation of the problem solution.

Figure 3 is a flow-chart of the overall operation of the system.

**PROBLEM-SOLVER/MONITOR**

The course is organized as a set of solution algorithms. Normally there is a single algorithm for each major concept of the course. The algorithms solve the problems much as a student would, breaking each problem
TABLE I—Student Record

<table>
<thead>
<tr>
<th>Concept #1</th>
<th>Concept #2</th>
<th>Concept #30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level</td>
<td>2.2</td>
<td>1.5</td>
</tr>
<tr>
<td>Last level change</td>
<td>.1</td>
<td>.5</td>
</tr>
<tr>
<td>Weighted Av. level change</td>
<td>.5</td>
<td>.6</td>
</tr>
<tr>
<td>Date of last call</td>
<td>3/15</td>
<td>3/17</td>
</tr>
<tr>
<td>Sequential order of last call</td>
<td>25</td>
<td>31</td>
</tr>
<tr>
<td>No. of times called in 0-1 range</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>No. of times called in 1-2 range</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>No. of times called in 2-3 range</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>No. of problems generated</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

down into a series of sub-tasks. After each sub-task is accomplished, a decision is made whether or not to question the student on this part of the problem solution. This decision is based on the student's current level of achievement (a number between 0 and 3) in the concept.

If the student is questioned, then his answer is compared with the system's solution. If the student is correct, his level is increased; if he is incorrect, he will receive a remedial comment explaining the correct solution procedure and his level for that concept will be decreased. The higher a student's level, the fewer questions he will be asked. When the student reaches a level of 3 in a concept, the system will solve subsequent problems dealing with this concept for him.

Figure 2—Concept tree

Table II presents examples of different degrees of interaction with a student in an octal addition problem. The first character of all student inputs is underscored. A single algorithm solves binary, octal, and hexadecimal addition problems. The decimal sum of each pair of digits and old carry is first calculated. The corresponding new carry and sum digits in the problem base is then determined.

A student with level less than 1 is quizzed on all details of this algorithm; while, the initial decimal calculation is skipped for students with level between 1 and 2. The student with level between 2 and 3 is asked only for his final result; whereas, the final result is generated for a student with level equal to 3.

Figure 3—System flow chart
TABLE II—Different Degrees of Interaction

Problem: Form the sum of 75.3, 2.7 in the base 8.

**LEVEL RANGE 0-1**
Modify 2.7 for addition to 75.3
02.7
Correct. The rightmost digit is digit #1.
What is the decimal sum of digit #1 of 75.3 and digit #1 of 02.7?
8
No. Their decimal sum is 3+7 or 10.
What is digit #1 of the sum in the base 8?
1
No. The decimal number 10 is equivalent to 12 in the base 8.
Digit #1 is 2.
What is the carry to digit #2?
1
Correct. What is the decimal sum of digit #2 of 75.3 and digit #2 of 02.7 and the carry from digit #1?

**LEVEL RANGE 1-2**
The rightmost digit is digit #1.
What is digit #1 of the sum in the base 8?
2
Correct. What is the carry to digit #2?
0
No. 3+7 = 12 in the base 8.
The carry to digit #2 is 1.
What is digit #2 of the sum in the base 8?
1
Correct. What is the decimal sum of digit #2 of 75.3 and digit #2 of 02.7 and the carry from digit #1?

**LEVEL RANGE 2-3**
What is the complete sum of 75.3, 02.7 in the base 8?
77.2
No. The sum of 75.3, 02.7 is 100.2 in the base 8.
**LEVEL RANGE ≥3**
The sum of 75.3, 02.7 is 100.2 in the base 8.

amount of interaction a student receives will vary as the problem solution progresses and his level changes from one range to the next.

In this manner, the amount of monitoring is dynamically adapted to fit a student’s current level of achievement. Often a solution algorithm will utilize an algorithm dealing with another course concept as a subroutine. When this happens, the student’s level in the sub-concept will control the depth of instruction. If he has mastered the sub-concept (level ≥3), this portion of the solution will be provided for him.

When a student response is incorrect, he receives an individualized remedial comment which explains the system’s derived answer. For example, in a problem utilizing the Flip-Flop Design algorithm, the student would be given the current state and desired next state and asked to supply the required values of the Flip-Flop excitations. Table III presents some of the alternative forms of feedback which might be presented depending on a student’s level and his answer.

The correct solution in this case is derived through a table lookup. System response A is automatically generated if one of the excitations match and the correct answer for the non-matching excitation is “D” (a don’t care). System response B is presented for all other types of errors. The additional information presented to students with level <1 is obtained by calling the Flip-Flop analysis algorithm in the problem-solver mode. This algorithm determines the next state which would result from the current state if the student’s values of the Flip-Flop excitations were applied.

The magnitudes of the increment and decrement for correct and incorrect answers respectively are also individualized to fit the student’s past performance. They are calculated for the particular concept which has been selected as follows:

\[
\text{Increment} = K_2 \cdot K_1 \\
\text{Decrement} = K_2 / K_1
\]

where

\[
K_1^2 = 1 + W_N + \Delta L \text{ and } 0.5 \leq K_1^2 \leq 2, K_2 = 1 - 0.5 R \\
\Delta L = \text{last change in level} \\
W_N = \text{weighted average level change for } N \text{ uses of a concept} \\
W_N = (N - 1) \cdot W_{N-1} + 2 \cdot \Delta L \\
R = (\text{Number of uses of concept at present level range}) / N
\]

If the student’s level in a concept has been consistently increasing, \( W_N \) and \( \Delta L \) will be positive and \( K_1^2 \) will tend to be large. \( K_1^2 \) represents the ratio of Increment to Decrement. Consequently, a student who has been performing well in a concept will be rewarded more for a correct answer than he is penalized for an incorrect one. Thus, he is receiving the benefit of the doubt for an occasional incorrect answer and progresses more rapidly to the problem-solver mode for that concept. The opposite is true for a student who has been performing poorly. This will enable him to drop quickly to a lower level at which he will receive further monitoring and instruction if he continues to answer incorrectly.

\( R \) represents the stability of the current level for the concept in question. \( K_2 \) assumes values between 0.5 and 1.0, the higher values representing a greater degree of instability. Consequently, when the current level is in a relatively unused or unstable level range, the Increment and Decrement will both tend to be larger than when the current level appears to be a stable one. This makes it easier for a student to move out of a new level range; whereas, he must answer more questions correctly (or incorrectly) if he is to move out of a level...
QUESTION

The current state of JK Flip-Flop 1 is 0.
The next state should be 1.
What are the values of J1, K1?

STUDENT ANSWER: J1 = 1, K1 = 1

SYSTEM RESPONSE A: Your answer is correct, however, a better answer would be J1 = 1, K1 = D.
(D symbolizes a "don't care" condition)

The remedial feedback generated for a wrong answer to the above question follows:

STUDENT ANSWER: J1 = D, K1 = 1

SYSTEM RESPONSE B: Wrong. To bring a JK Flip-Flop from the 0 to 1 state, set J = 1 and K = D.

ADDITIONAL RESPONSE FOR LEVEL <1: Your Flip-Flop excitations would cause the next state to be indeterminate.

range which has become established as typical of him for a particular concept.
The values of Increment and Decrement calculated in this manner will have a minimum of .35 and a maximum of 1.4. These are multiplied by a fraction associated with each question. A typical value of this fraction is .2 which means the magnitude of the change in level per question is normally between .07 and .28.
The entry level of a student during his initial use of a concept is set equal to .5+2-M, where M is his master average. The significance of a student's master average will be discussed in the next section.

CONCEPT SELECTOR

Since there are a large number of concepts available for study, the system attempts to select the next concept in such a way as to make optimal use of the student's time. The goal is to pace the student through the concepts quickly enough so that he does not become bored or unmotivated and yet not so fast that he becomes unduly confused.

There is no set order in which the concepts are selected nor is there a set level of achievement which every student must exceed in order to advance. The algorithm attempts to individualize concept selection through examination of the student's performance record.

Each student is assigned a master average when he first logs onto the CAI system. This could be a function of his I. Q. or class standing. (In the past, each student has been arbitrarily assigned an initial master average of 2.) This value changes as the system gains experience with a student.

A student's master average controls the speed with which he jumps from one plateau of the concept tree to the next. In order to jump to the next higher plateau, the average of his levels of achievement in all concepts at and below the current plateau must exceed his master average. Consequently, the lower a student's master average, the faster he will progress.

Each student's master average is updated after the completion of a problem as follows:

\[ \Delta M = -K(\Delta L+W_N)(1.1-R) \]  \hspace{1cm} (2)

where \( \Delta L \), \( W_N \), and \( R \) have been defined previously. (See Equation 1.)

\( K = 1 \) if the concept is from the student's current plateau
\( K = 2 \) if the concept resides at a higher plateau and \( \Delta L > 0 \).
\( K = 2 \) if the concept resides at a lower plateau and \( \Delta L < 0 \).
\( K = 0 \) for all other cases.

Since \( R \) cannot be > 1, \( \Delta M \) and \( (\Delta L+W_N) \) will be opposite in sign.

If the system selects the concept from the student's current plateau, \( K \) will be 1. If the student's level increases and \( W_N \) is also positive, his master average will decrease. If the student has selected the concept from a higher plateau and \( \Delta L > 0 \), the magnitude of the decrease is doubled as this indicates the student is ready to progress more quickly. If the concept is above his current plateau and he does poorly \( (\Delta L < 0) \), he is not penalized by an increase in his master average \( (K = 0) \). However, if the concept is a remedial one (below his current plateau) and \( (\Delta L+W_N) \) is negative, the increase in his master average is twice what it would be for the same performance in a concept from his current plateau.

The effect of the term \((1.1-R)\) is to cause those changes, which occur when in a relatively new level range, to have a greater influence on the master average. This is reasonable since a student who performs
TABLE IV—Sample Problems Generated

1. Convert the decimal number 65.75 to the base 8.
2. Calculate 75.3 – 24.56 in the base 8 using 8’s complement subtraction.
3. Derive the truth table for \((PV(\neg Q)) \uparrow (\neg (RAQ)))\)
4. Design a combinational circuit such that:
   - Input word A is 2 bits long. Input word B is 2 bits long.
   - The output \(X = A + B\). The output \(Y = 1\) if \(A < B\).
5. Minimize a function which has 0, 1, 3, 4, 5, 10, 11 as minterms and 2, 8, 14 as don’t cares using the Karnaugh Map Method.
6. Minimize a function which has 2, 4, 5, 10, 11 as minterms using the tabular method.
7. Find the excitation equations for a JK Flip-Flop such that:
   - Input word A has maximum value 7.
   - The Flip Flop is in state 1 if \(A = 0, 2, 4, 6\).

well after reaching a new level range is indicating that he is not disturbed by the decrease in interaction and is ready to advance at a faster pace. While a student who does poorly may be in over his head and might prefer it if the pace were slowed down. In any event, the magnitude of the change in master average may not exceed .2.

After the master average is updated, the student’s plateau is determined by comparing his new master average with the average level of all concepts at and below his current plateau. If this average level exceeds his master average his plateau is increased by one and the comparison is repeated.

Once the student’s plateau has been determined, the system selects a set of candidate concepts from this plateau and those below it if necessary. In order to qualify as a candidate concept, the average of the student’s levels of achievement in all prerequisites for this concept (as determined from the concept tree) must exceed his master average. If this is not the case, the prerequisite concept in which the student has the lowest level is selected as a candidate in its place. This provides for automatic review of selected concepts at lower plateaus.

The system then chooses one concept from among the candidates. Each concept is evaluated based on a number of factors such as the time elapsed since its last use, the “stability” of its current level, the sign and magnitude of its most recent level change (negative changes are weighted more heavily), and its relevance to other concepts as determined by the number of branches of the tree connected to it. The highest scoring concept is selected for presentation to the student.

The student always has the option of vetoing this selection and choosing his own concept or accepting the system’s second best choice. The selection process is described more formally in the Appendix.

PROBLEM GENERATION

After a concept has been decided upon, the system must generate a problem within this concept area unless the student prefers to supply one. The system attempts to tailor the problem difficulty to suit the student’s level in that concept. Table IV gives some sample problems generated by the system.

As an example of the problem generating procedure, the algorithm used to form logical expressions for teaching about truth tables (see problem 3 in Table IV) will be presented in some detail. The basis of this algorithm is the probabilistic grammar shown in Table V which produces expressions utilizing binary operators with \(P, Q, R, S\) as variables. A probabilistic grammar is a formal language in which each rewrite rule is assigned a probability of being applied.

The first decision to be made is the number of variables in the final logical expression. As currently implemented, the probability of two variables is .5 for \(0 \leq \text{level} \leq 1\) and decreases to 0 for \(\text{level} > 1\). The probability of 3 variables is .5 for \(0 \leq \text{level} \leq 2\) and increases to 1 for \(\text{level} > 2\). The probability of 4 variables is .5 for \(1 \leq \text{level} \leq 2\) and 0 elsewhere.

The incomplete logical expression is scanned from left to right for non-terminals. When the non-terminal symbol \(A\) is found \((P_a + P_b)\) represents the probability of increasing the length of the expression where:

\[
P_a(t) = P_b(t) = .75 c / (n(t) + 1.5c - 1.5)
\]

where \(n(t)\) is the current length and \(c\) is the number of variables in the expression \((c \geq 2)\)

Since \((P_a + P_b)\) is inversely proportional to the current length, the logical expressions do not become unwieldy. If the random number generated indicates that the

<table>
<thead>
<tr>
<th>TABLE V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probability: Rewrite rule</td>
</tr>
<tr>
<td>1: (3 \rightarrow A)</td>
</tr>
<tr>
<td>(P_a: A \rightarrow (A* A))</td>
</tr>
<tr>
<td>(P_b: A \rightarrow (\neg A))</td>
</tr>
<tr>
<td>(P_a: A \rightarrow P)</td>
</tr>
<tr>
<td>(P_a: A \rightarrow Q)</td>
</tr>
<tr>
<td>(P_b: A \rightarrow R)</td>
</tr>
<tr>
<td>(P_b: \rightarrow V ) (or)</td>
</tr>
<tr>
<td>(P_b: \rightarrow A ) (and)</td>
</tr>
<tr>
<td>(P_b: \rightarrow \uparrow ) (NAND)</td>
</tr>
<tr>
<td>(P_b: \rightarrow \downarrow ) (NOR)</td>
</tr>
<tr>
<td>(P_b: \rightarrow \oplus ) (exclusive or)</td>
</tr>
</tbody>
</table>

Constraints: \(0 \leq P_a, P_b, P_{eb}, P_{ea}, P_s \leq 1\), \(P_a + P_b + \sum P_{ei} = 1\), \(2P_b + 3P_a = 1\)
TABLE VI—Example of Student Interaction With System Executive

IS THIS THE FIRST TIME YOU HAVE USED CAI?
no
TYPE IN YOUR NAME
ebkoffman
AFTER REVIEWING YOUR RECORD, I HAVE SELECTED CONCEPT #9.
BINARY MULTIPLICATION
IS THIS OK?
yes
FORM THE PRODUCT OF THE BINARY NUMBERS 101.01, 1.001
IGNORING BINARY POINTS, WHAT IS THE PARTIAL PRODUCT DUE TO BIT 1 OF THE MULTIPLIER 1001?
10101
CORRECT
WHAT IS THE PARTIAL SUM SO FAR?
TYPE F TO GO FASTER (Instead of answering the previous question, the student has hit the console attention button.
S TO GO SLOWER The system attempts to determine his wishes.)
f
WHAT IS THE PRODUCT OF 101.01, 1.001? (The student's level has been increased by 1)
101.11101
CORRECT
YOUR LEVEL IN BINARY MULTIPLICATION INCREASED BY 1.2 TO 2.3
I HAVE SELECTED CONCEPT #12
KARNAUGH MAPS
OK?
no
WOULD YOU LIKE TO PICK YOUR OWN CONCEPT?
yes
DO YOU KNOW THE CONCEPT NUMBER?
no
HERE ARE THE CONCEPTS CURRENTLY COVERED (This printout would be omitted if the student had answered yes.)
1. REGISTER 'ANDING'
:15. STATE TABLE TO SEQUENTIAL DESIGN
WHICH CONCEPT NUMBER WOULD YOU LIKE?
15
FINE. THIS IS ABOVE YOUR CURRENT PLATEAU, BUT LET'S TRY IT.
DO YOU WANT ME TO GENERATE A PROBLEM?
no
HOW MANY STATES IN YOUR STATE TABLE? (Student specifies problem parameters and works through solution)
5
HOW MANY INPUT BITS?
:1
WOULD YOU LIKE TO CONTINUE PICKING CONCEPTS?
no
I HAVE SELECTED CONCEPT #12
KARNAUGH MAPS
OK?
yes
MINIMIZE A FUNCTION WHICH HAS 2, 5, 6, 7 AS MINTERMS USING THE KARNAUGH MAP METHOD
HERE IS THE KARNAUGH MAP (A student with level <1 in this concept would be tested on his ability to construct the map)

expression should not be extended, one of the variables $P, Q, R, S$ replaces $A$.

If the non-terminal symbol is $*$, one of the five binary operators is selected. $P_e$ increases with level, while, $P_d$ decreases. Hence, the more difficult operators are more likely in expressions generated for the students at higher levels of achievement.

Modifications of this technique are used to generate problems for combinational design and sequential design concepts. For the earlier number system manipulations, it is necessary to select a base for the number system and concatenate a set of valid digits from that base. As might be expected, the hexadecimal base appears more frequently at higher levels and the numbers to be manipulated are somewhat larger than those generated for lower levels.
IMPLEMENTATION

The system has been implemented on the IBM 360/65 at the University of Connecticut. It is programmed in the Conversational Programming System (CPS).4 CPS is a dialect of PL/1 and includes some string processing features which have been extremely useful in programming this system.

There are forty IBM 2741 terminals connected to the main computer. CPS operates in low-speed core and each user is allowed a maximum of 12 pages of core (48 K bytes). Currently, space is reserved for fifty student records on disk. There is room for thirty concepts on the tree. There is an instructor mode which permits easy modification of the concept tree. Student records can also be printed out while in the instructor mode.

Students can log onto the system any time from 9 A.M. to 11 P.M. The average student session is about an hour in length. The student's record is automatically updated every 15 minutes so that the results of a complete session will not be lost in the event of a system failure.

The course coverage includes familiarizing the students with the binary, octal, and hexadecimal number systems. The students also learn how to use truth tables to represent logical functions. They learn techniques of combinatorial design and how to minimize the logic elements needed for a design problem. They are introduced to Flip-Flops and the design of sequential circuits. They also learn about machine-language programming by analyzing short program segments and indicating the changes incurred in various registers as these segments are executed.

Reaction from the students using the system has been very favorable to date. It has been used as a supplement to the regular lectures and homework assignments covering the material up through plateau four of the concept tree (Figure 2). The problem generation capability was not fully implemented which required students to insert their own problem parameters as they were requested by the problem solution algorithms. The major complaint of the students was the slowness of the system. This has been alleviated considerably now that problems are generated for all system concepts.

The system responds to a student input typically in 5 seconds or less. In a couple of the more complex solution algorithms (tabular minimization for example), a student working in the problem-solver mode may have to wait up to three minutes for a problem solution. Students at lower levels very rarely have to wait more than thirty seconds as the solution algorithms are executed in a piecemeal fashion. This, of course, could be improved substantially if compiled code (CPS is an interpretive system) and high-speed core were available.

During the current semester, a student group is using the CAI system in lieu of regular homework assignments. The complete tree of concepts is available. Table VI is an example of the dialogue that occurs between the system executive and the student. The system's messages are capitalized. All student inputs are preceded by a "—." Table VII presents examples of the interaction that takes place during a problem solution.

CONCLUSIONS AND FUTURE WORK

A generative system for teaching digital computer concepts has been described. This system includes an intelligent concept selection routine which attempts to individualize the path taken by each student through the tree of course concepts. In addition, the instruction and monitoring provided by each solution algorithm is dynamically adapted to suit a student's current level of knowledge. The system can also operate as a problem-solver and will provide portions of the solution process which have already been mastered by a student.

Expansion of the CAI system is continuing. The majority of the portion which teaches logical design has been completed. Students have also been exposed to an on-line simulator of the Digital Equipment Corporation PDP-8 which teaches machine-language programming.

Future efforts in this area will lead to a generative system which produces programming problems. As the program is generated, a sequence of sub-tasks similar to a flow chart will also be produced. This will be used as the basis for teaching machine language programming and testing students' solutions. As before, portions of the program will be generated for the student and he will be required to supply the rest. The amount required from him will decrease as his level increases.

It is also anticipated that there will be further improvement of the concept selection algorithm. It is likely that performance can be improved by incorporating additional parameters into the algorithm as well as by adjusting the weights of the parameters currently used.

The techniques discussed in this paper could readily be applied to other courses which teach techniques of solving problems. In fact, several courses could be administered simultaneously by the concept selection routine. A concept tree would have to be specified for each course together with the algorithms necessary for generating and solving problems.
The initial burden on the course designer is, perhaps, greater than in a frame-oriented system as he must program general solution algorithms rather than a representative set of questions and anticipated replies. It is possible that a “canned” solution program (such as ECAP for circuit analysis) could be converted into a teaching program by first segmenting it into course concepts. Each concept would be modified by inserting

### TABLE VII

Example 1—Sequential Circuit Analysis

Derive the transition table for a sequential network with 2 flip flops such that:

\[ J_1 = (Y_2 | X) \quad K_1 = (Y_1 | Y_2) \quad D_2 = (\neg X) \]

And the output \( Z = X \& (\neg Y_2) \)

The truth table is set up as follows:

<table>
<thead>
<tr>
<th>( J_1 )</th>
<th>( K_1 )</th>
<th>( D_2 )</th>
<th>( Y_1 )</th>
<th>( Y_2 )</th>
<th>( Z(t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(Note: Because level is now \( >1 \), additional information has been provided for row 1)

Type in \( J_1 \)

Type in \( K_1 \)

If \( J_1 = 0 \) and \( K_1 = 0 \), the FF state remains the same.

Type in the next state of flip flop 2.

Type in \( Z(t) \).

Now we must form the transition table. The heading follows:

<table>
<thead>
<tr>
<th>CURRENT</th>
<th>CURRENT</th>
<th>STATE</th>
<th>INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, Y_2)</td>
<td>(0, Y_2)</td>
<td>X=0</td>
<td>X=1</td>
</tr>
</tbody>
</table>

What is the next state/current output if the current state is \( (0, 0) \) and \( X=0 \)?

(Use format \((Y_1(t+1), Y_2(t+1))/Z(t)\))

(0, 0)/1

Wrong. \( Y_2(t+1) = 1 \)

Wrong. \( Z(t) = 0 \)

Wrong. \( Y_2(t+1) = 1 \)

Wrong. \( X=1 \)

Wrong. \( Z(t) = 0 \)

(Note: This line of questioning will continue until the problem is completed or the student's level becomes \( \geq 3 \).)
TABLE VII—Continued
Example 1—Sequential Circuit Analysis
Example 2—Sequential Circuit Design

DERIVE THE STATE TABLE FOR A SEQUENTIAL CIRCUIT WITH A SINGLE INPUT LINE X. THE OUTPUT WILL BE 1 WHENEVER THE LAST 4 VALUES OF X HAVE BEEN 1000.

HOW MANY STATES ARE NEEDED?
5
WRONG. SINCE THERE ARE 4 BITS IN THE INPUT STRING, 4 STATES ARE NEEDED.
Q 0 IS THE STARTING STATE. Q 1 CORRESPONDS TO THE FIRST 1 BITS BEING CORRECT.
Q 0 CORRESPONDS TO THE LAST INPUT 0 OR 1?
1
WRONG. BEING IN THE STARTING STATE MEANS THE DESIRED SEQUENCE HAS NOT BEGUN,

SO THE LAST INPUT WAS 0
THE STATE IS Q 1 IF THE LAST INPUT WAS ___?
1
THE STATE IS Q 2 IF THE LAST 2 INPUTS WERE ___?
11
WRONG
Q 2 CORRESPONDS TO THE PARTIAL SEQUENCE 10
THE STATE IS Q 3 IF THE LAST 3 INPUTS WERE ___?
100
IN STATE Q 0: IF X=0 WHAT IS THE NEXT STATE/OUTPUT?
Q 1/0
NO. THE LAST INPUT IS NOT PART OF A DESIRED PARTIAL SEQUENCE.
THE SEQUENTIAL CIRCUIT RETURNS TO THE INITIAL STATE Q 0
IF X=1?
Q 1/0
IN STATE Q 1: IF X=0?
Q 2/0
IF X=1?
Q 0/0
NO. THE LAST INPUT IS 1
THE NEXT STATE SHOULD BE Q 1
IN STATE Q 2: IF X=0?
Q 2/0
NO. THE LAST 3 INPUTS ARE 100
THE NEXT STATE SHOULD BE Q 3
IF X=1?
Q 1/0
IN STATE Q 3: IF X=0?
Q 0/0
NO. THE LAST 4 INPUTS ARE THE DESIRED SEQUENCE 1000
THE OUTPUT SHOULD BE 1
IF X=1?
Q 1/0
THE FINAL STATE TABLE FOLLOWS:

<table>
<thead>
<tr>
<th>CURRENT STATE</th>
<th>INPUT X=0 X=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q 0</td>
<td>Q 0/0 Q 1/0</td>
</tr>
<tr>
<td>Q 1</td>
<td>Q 2/0 Q 1/0</td>
</tr>
<tr>
<td>Q 2</td>
<td>Q 3/0 Q 1/0</td>
</tr>
<tr>
<td>Q 3</td>
<td>Q 0/1 Q 1/0</td>
</tr>
<tr>
<td>NEXT STATE</td>
<td>/OUTPUT</td>
</tr>
</tbody>
</table>

requests for student answers after the completion of each step in the solution process as well as remedial statements explaining the correct procedure to be followed for each step in the solution. In addition, a program statement would have to be included which would allow this question to be skipped by students above a specified level of proficiency.

It is felt that the extra effort required to program
an algorithm based CAI system is worthwhile as the end product is a very versatile and flexible teaching tool.

APPENDIX—CONCEPT SELECTION

\(C_{ij}\) is the jth concept at Plateau \(i\)
\(L_{ij}\) is the level of \(C_{ij}\) where \(0 \leq L_{ij} \leq 3\)
\(n_i\) is the number of concepts at Plateau \(i\)
\(L_i\) is the average level of concepts at Plateau \(i\), where

\[
L_i = \frac{1}{n_i} \sum_{j=1}^{n_i} L_{ij}
\]

If \(I\) is the current plateau, calculate \(L_i\), where
\(L = I^{-1} \sum_{i=1}^{N} L_i\) and \(L\) is the average of all concepts at and below plateau \(I\).

If \(L \geq M\), where \(M\) is the Master average, then increase \(I\) by 1; otherwise, \(I\) does not change.

Now select the set of candidate concepts \(K = \{k_1, k_2, \ldots, k_{ni}\}\) as follows:

Let \(S_j = \{L_{xy} \mid (x \leq I) \land (y \leq n_x) \land (C_{xy}\text{ is a prerequisite of } C_{ij})\}\) for \(1 \leq j \leq n_I\)

Calculate \(\bar{S}_j\), the mean of \(S_j\)

If \(\bar{S}_j \geq M\), then \(k_j = C_{ij}\), otherwise, \(k_j = C_{ab}\) where \(C_{ab}\) is the prerequisite concept whose level, \(L_{ab}\), is the smallest element in \(S_j\).

Given the set of candidate concepts, \(K\), the candidacy value, \(V_j\), of each concept must be calculated. The concept, \(k_j\), which has the highest candidacy value for this student is selected.

\[
V_j = \sum_{i=1}^{n_i} w_i F_{ij}
\]

\[
F_{ij} = (2\Delta L_{ij})^2
\]

\(w_1 = 1\) if \(\Delta L_{ij} > 0\)

\(w_1 = 0\) if \(\Delta L_{ij} < 0\)

Factor 1 tends to favor repetition of concepts for which previous usage led to a large negative change in level. The student is experiencing difficulty with this concept and its reuse with more interaction and monitoring occurring may prove beneficial

\[
F_{ij} = \frac{(Q - Q_j)}{Q}
\]

where \(Q\) is the current sequence number and \(Q_j\) was the sequence number at the time concept \(k_j\) was last called. \((Q\) increases by 1 each time a new concept is selected.)

\(w_2 = 1\) if \(Q_j > 0\)

\(w_3 = 3\) if \(Q_j = 0\) (Indicates concept \(k_j\) has not been selected for this student.)

Factor 2 favors the recall of concepts which have not been used recently.

\[
F_{kj} = R_j
\]

where \(R_j\) is the stability of concept \(k_j\) as defined previously.

\(w_2 = -1\)

Factor 3 inhibits the use of a concept whose level is in a relatively stable range.

\[
F_{kj} = \frac{|\Delta L_{ij}|}{L_{ij}}
\]

\(w_4 = 1\)

Factor 4 is the percentage charge in level during the last usage of \(k_j\). Factors 3 and 4 bias the selection toward concepts whose levels seem most likely to change during an interaction

\[
F_{kj} = 2 \cdot D_j + E_j
\]

\(w_5 = .1\)

Factor 5 calculates the number of prerequisites for concept \(k_j\). Those prerequisite concepts which may be called as subroutines are counted twice. This factor gives emphasis to concepts which are more likely to result in the review of other concepts.

REFERENCES

1. J. R. Carbonell
   "AI in CAI: An artificial intelligence approach to computer-assisted instruction"

2. J. D. Wexler
   "Information networks in generative computer-assisted instruction"

3. M. R. Quillian
   "The teachable language comprehender: A simulation program and theory of language"
   Communications of the ACM Vol 12 No 8 Aug 1969 pp 459-476

4. W. R. Uttal et al
   "Generative computer-assisted instruction"
   Mental Health Research Institute Communication 243 1969 University of Michigan

5. IBM Corporation
   "Conversational programming system (CPS) terminal user's manual"
   IBM Report GH20-0758-0 1970