DOCUMENT RESUME

ED 078 648	EM 011 178
AUTHOR	Koffman, Elliot B.
TITLE	Generative Computer Assisted Instruction: An Application of Artificial Intelligence to CAI.
INSTITUTION	Connecticut Univ., Storrs. Dept. of Electrical Engineering.
SPONS AGENCY	National Center for Educational Research and Development (DHEW/OE), Washington, D.C.; National Science Foundation, Washington, D.C. Office of Computing Activities.
PUB DATE	72
GRANT	OEG-0-72-0895
NOTE	8p.; USA-Japan Computer Conference, 1972
EDRS PRICE	MF-$0.65 HC-$3.29
DESCRIPTORS	Algorithms; *Artificial Intelligence; *Computer Assisted Instruction; *Computers; Problem Solving; Programed Texts; State of the Art Reviews
IDENTIFIERS	Generative Computer Assisted Instruction; Natural Language Communication

ABSTRACT

Frame-oriented computer-assisted instruction (CAI) systems dominate the field, but these mechanized programed texts utilize the computational power of the computer to a minimal degree and are difficult to modify. Newer, generative CAI systems which are supplied with a knowledge of subject matter can generate their own problems and solutions, can provide extensive drill and tutoring, and can diagnose learner problems and prescribe remedial feedback. These systems reduce the amount of instructor effort required to teach new material and encourage students to assume the initiative in studying new topics. Information-Structure Oriented (ISO) systems have capabilities for natural language communication and are useful in the humanities and social sciences. Quantitative systems rely on algorithms and are oriented toward providing students with practice in problem-solving. While the existing generative systems are still experimental it is expected that their use will become more widespread as research continues and as CAI facilities become more available and economical. (PB)

First USA-JAPAN Computer Conference, 1972

GENERATIVE COMPUTER ASSISTED INSTRUCTION: AN APPLICATION OF ARTIFICIAL INTELLIGENCE TO CAI

Elliot B. Koffman

(University of Connecticut, Storrs, Connecticut)

I. INTRODUCTION

Limited progress has been made in software for computer-assisted instruction. Frame-oriented CAI systems have dominated the field. These systems are classically 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.

Recently, generative CAI systems have appeared. These are systems for CAI which are capable of generating their own questions or problems and deriving their own solutions. Hence, they can provide unlimited drill and tutoring with little effort required from the course-author to define an instructional sequence.

These systems are supplied with knowledge of their subject matter. Therefore, they can often interpret and answer questions or problems posed by the student. They are also capable of diagnosing the degree of inaccuracy in a student response and providing remedial feedback on an individual basis. Most of these systems incorporate techniques and concepts which are outgrowths of research in Artificial Intelligence.

This paper will describe some of the generative systems which have recently been designed. A generative tutor under development by the author, which has been used to teach digital computer concepts, will also be examined in detail.

These systems can be divided into two classes. Those which are oriented towards the humanities and textual material and those which are more concerned with numerical manipulations and quantitative material. The former have been termed Information-Structure Oriented systems (ISO) and will be discussed next.

II. ISO GENERATIVE SYSTEMS

Research in the area of Artificial Intelligence has provided most of the background for the ISO Generative Systems. These systems are intelligent in the sense that they have knowledge of the subject matter they are designed to teach. This data base of facts is stored in memory in an information network.

This research is sponsored by the U.S. Office of Education under Grant OEG-0-72-0895. Computer services were supported by NSF Grant GJ-9.

In an ISO system, there must be a capability for natural language communication between teacher and student. The system must be able to determine the degree of correctness of a student's response by comparing it with information stored in its data base. The system should be able to understand and answer questions which are posed by the student. Simmons (1) provides a very comprehensive review of research in the areas of natural language communication and question-answering.

Carbonell (2,3) has designed an extremely versatile system called SCHOLAR. SCHOLAR is an example of a "mixed-initiative" system in that the student can interrupt the flow of questions generated by SCHOLAR at any time and interrogate SCHOLAR. SCHOLAR relies heavily on prior research by Quillian (4,5) in the area of semantic memory and utilizes a semantic network for storage of a data base of facts concerning South America.

A semantic network is an organization of units of information in terms of their meaning and mutual interrelationships. Each unit or node in the memory contains pointers to other related units. Starting from a given unit, one can trace out a set of relationships by following the diverging trails of pointers. See Figure 1.

For example, the unit Argentina, denoted as an object, might have attributes such as superconcept, superpart, location, capital city, etc. The values of these attributes would be other units in the network and, hence, there would be pointers to the nodes representing the units country, South America, latitude and longitude, Buenos Aires, respectively. Also, associated with each attribute is a tag indicating its irrelevancy with respect to the parent unit.

The Semantic network is used by SCHOLAR both for generating and answering questions. The initial context of the questions is determined by the teacher. The teacher's input "South America" specifies that the initial unit to be investigated is the node South America. An attribute is selected and a question is generated concerning the value of that attribute. At the same time, the correct answer is retrieved for comparison with the student response.

For example, given the initial context "South America" and the attribute "Countries" SCHOLAR might ask one of the following:

What are the countries in South America?

Is it true that Argentina is a country in South America (T/F)
A Country in South America is --- (fill in).

There is a certain probability that the context will change from the initial unit to a related unit. Hence, a subsequent question might be:

Select an Alternative from the list:
Paysardu
Rio De Janerio
Buenos Aires
Uruguay River

In the question:

What is the capital of Argentina?

The actual generation of questions is handled by a set of procedures which are available for forming each of the above question types. The question type to be asked is determined probabilistically. As has been mentioned, the correct answer to a question is derived when the question is generated. Since all expected student responses are numbers or object names, the problem of recognizing a correct student response is minimized. The teacher can specify the acceptable degree of inaccuracy in numerical answers and the degree of misspelling allowed for object names.

Since SCHOLAR is a mixed-initiative system, the student can interrupt with questions of his own such as "Tell me about Argentina" or "What is the capital city in Argentina". SCHOLAR analyzes the student's question and determines which of the elements in the set (attribute, object, value) are missing. SCHOLAR then generates an appropriate reply.

In the case of "Tell me about Argentina," SCHOLAR would trace through the portion of the network originating at the node for the unit Argentina and generate a set of statements. SCHOLAR keeps track of the irrelevancy tags to make sure that all statements generated are pertinent. If the student comes back with "Tell me more about Argentina," then SCHOLAR will generate statements which are somewhat more remote than the initial statements.

It is obvious that the burden on the instructor for specifying a CAI session is minimal. He need only specify an initial context, the length of the session in terms of maximum time and minimum number of questions, the probability of switching to a subcontext for question generation, and the degree of irrelevancy allowed for a subcontext.

The major task for the instructor is supplying the semantic network. Currently, the semantic network is input manually. Carbonell is working on an interface to simplify this procedure while Quillian (5) is concerned with the general problem of automatically constructing semantic networks.

The original implementation of SCHOLAR was in BBN LISP (6) on the XDS-940 time-sharing computer. It utilized approximately 144K 24-bit words. This includes 35K words for the LISP system, 90K words for SCHOLAR itself, and 12K words for the semantic network. The time required to answer a student's question is about a minute. It is anticipated that this can be cut by a factor of 15 if compiled code is used.

Another example of an ISO Teacher is the system designed by Wexler (7,8). Wexler uses an information structure that consists of classes, objects within those classes, and links between objects. Sample classes from the geography of Canada might be "Provinces" and "Cities" with "Newfoundland" and "Quebec" examples of objects in the former class and "St. John's" and "Quebec City" examples of objects in the latter class. There could be a link between the objects St. John's and Newfoundland and between the objects Quebec City and Quebec. The course-author constructs a chain of links in the network by specifying an object together with its class name and the next object (with associated class name) in the chain. Figure 2 gives an example of an information structure from this system.

The information structure utilized might be thought of as a skeletal representation of relationships. It is considerably more streamlined than the semantic network and, consequently, additional data must be supplied in order to interpret it.

This data consists of a set of skeleton segments and their interpretation strings. A sample segment might be the following:

```
STATE *V1 LINK TO CITY *V3
```

Associated with this segment might be the interpretation string

```
*V3 is a city in *V1
```

This statement pair means that if there is a link in the information network between V1, a member of the class "state", and V3, a member of the class "city", then the assertion V3 is a city in V1 is valid. Skeleton segments can be combined into skeleton patterns of the form "If A then B" where A is a boolean expression formed from a set of skeleton segments and B is the explanation string for the complete skeleton pattern. An example follows:

```
#SP1 IF (STATE *V1 LINK TO CITY *V2)
AND (*V2 LINK TO CAPITAL *V3)
AND (*V3 LINK TO *V1)
THEN THE CAPITAL OF *V1 IS *V2
```

The skeleton patterns are used by the system to generate statements and remedial aid and to verify student responses. The statement generated would take the form of the explanation string. In skeleton pattern one (SP1) above, either V1 or V2 must be specified. The other variable would be determined from the network and the appropriate information statement presented to the student.

In order to generate questions, a question construct must be supplied.

```
#Q1 NAME THE CAPITAL OF *P1
@OK = SP1 (V1 = P1, V2 = A1)
```

The parameter P1 in the resulting question would be a member of the class state and could be prespecified by the instructor or selected at random. The student response, A1, would be considered correct if the three conditions of SP1 were satisfied within the information network when A1 and P1 were substituted for V2 and V1 respectively.

If the objects A1 and P1 applied to SP1 did not lead to a valid "trace" then two types of prompting information could be supplied to the student. The system could search for a correct trace of SP1 using V2 = A1 and V1 unspecified. Finding a correct trace would set the value of V1 in the explanation string. Thus, if the student had responded "Providence" to the question "Name the capital of Connecticut", the prompting information "The capital of Rhode Island is Providence" would be generated.

An alternate form of help is available. The system could substitute the correct value of V2 when V1 = P1 and cycle through the skeleton segments of SP1 and their associated interpretation strings.

The result might be

Hartford is a city in Connecticut
Hartford is a capital
One city in a state is the capital
The capital of Connecticut is Hartford

As soon as the student realizes his mistake, he can halt this sequence.

The teacher has considerable range in specifying his instructional program. An instructional sequence will consist of a series of information statements followed by one or more questions concerning these statements. The entire teaching strategy can be specified by the teacher using appropriate command statements. More interesting modes of instruction are the parameterized and fully generative mode.

In the parameterized mode, the teacher plans the main line and remedial sequences of instruction by specifying the order of information statements and questions. However, he can utilize skeleton patterns to simplify this task and can either specify parameters for these patterns or have them selected at random. In the fully generative mode, the teacher specifies only which skeleton patterns he wishes exercised and the system will generate a sequence of information statements and questions using these patterns.

There is also the capability for a student to ask questions of the system. The system attempts to recognize the objects and class-names in the student's question and then searches for skeleton patterns that would produce traces which contain these objects. The actual traces are then presented to the student in a sequential fashion. If a student is interested in the resulting trace produced by a particular skeleton pattern, he can request more traces from this same pattern.

This system is implemented in ALGOL on a Burroughs B5500 with 32K words of core memory. The system was operated in a time-sharing environment which can handle up to 12 users at one time. It was found necessary for a single user to utilize the full capacity of the machine in order to obtain reasonable response times (2 minutes or less).

Simmons and Melton (9,10) have recently described an experimental ISO Generative System. Their system, called IT1 (Interpretive Tutor 1), is similar to Carbonell's scholar in that it uses a semantic network to store its data base. Their system; however, does have the capability to automatically construct the semantic network from text supplied by the course-author. The text is the same as that which will be seen by the student. It is necessary for a linguist (or the lesson designer) to monitor this process and occasionally intervene to select from a set of possible alternative representations constructed by the system.

The system functions as a tutor in the following way. The student reads the text which has been prepared by the course-author. If he cannot understand a complex sentence, he types in "EXPLAIN" followed by the number of the sentence. The system will then generate a set of simpler sentences from that portion of the semantic network derived from the original sentence. If the student does not know the meaning of a word in a particular sentence, he types in "DEFINE" followed by that word and the sentence number. The particular meaning required is retrieved from the lexicon (also supplied by the linquist) and displayed to the student. Table 1 depicts a result of the "EXPLAIN" command.

The remaining teaching function is similar to SCHOLAR'S (though less powerful) in that the system generates TRUE-FALSE and FILL-IN type questions.

The student selects the section of text on which he wishes to be quizzed. The system then generates a set of questions from the corresponding portion of the semantic network. FILL-IN type questions are produced by omitting a word from the generated sentence; TRUE-FALSE questions are produced by substituting (or not substituting in the TRUE case) for a word in the generated sentence.

In summary, three systems have been presented which make use of information networks. These systems are experimental in nature and would not yet be practical for large-scale use. They are oriented towards the "soft-sciences". They combine a question-answering or information retrieval capability with question generation; thus, allowing the student to assume the initiative when he desires to explore interesting points at greater depth. The Wexler system, especially, has the capability to generate prompting information relevant to a student's incorrect response.

Once the semantic network has been formed, very little information need be supplied by the teacher in order for SCHOLAR to generate a rich teaching sequence. Due to the relative sparseness of its information network, the Wexler system requires additional information in the form of skeleton patterns and question constructs in order to generate a teaching sequence. However, it allows the instructor to assume more control over what form this teaching sequence will take. Also, the information network is somewhat easier to construct in the Wexler system.

III. GENERATIVE CAI IN TEACHING QUANTITATIVE MATERIAL

The potential for incorporating generative CAI in the "hard sciences" is extensive. These courses are quantitative in nature and teach techniques of problem solving. Problem solving competence is often acquired through a process of "learning by doing" in which a student is required to solve a representative sample of problems.

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 (11) or proving theorems (12). In any event, CAI systems organized around a set of algorithms would have the capability to generate and solve a wide range of problems.

Several generative programs have been written by Uhr and his associates which deal with elementary arithmetic or algebra (13,14,15). These programs generate a series of examples for drill and practice. The difficulty of the problem generated is determined by the magnitude of the numbers used. The system keeps track of a student's progress and attempts to provide him with a suitable problem.

Lists of correctly answered and incorrectly answered problems are maintained. A student will normally be presented with a question which he previously missed. If there are no problems which he has yet to solve correctly, a new problem will be generated for him.. There is also a possibility he will be asked to review a problem which is on the correctly answered list.

If he answers incorrectly, an appropriate remedial comment is generated. As an example he might be told: "Wrong. 3+4 is not equal to 6. Your answer is too small".

Uhr (13) describes a possible extension which would define a set of complex operations in terms of a smaller set of primitive operators. In determining an answer to a complex problem, the computer would carry out the prescribed set of primitive operators. If the student's answer was incorrect, the system could backtrack through its solution process. It would determine the point at which the student's solution diverged from the correct path and explain the procedure that should have been followed.

The Peplinski (15) system generates either linear or quadratic equations. These equations are separated into classes such as linear equations with terms in "X" on the left side only, "X" on both sides, quadratic equations of the form $aX^2-b^2=0, aX^2-b^2=0, aX^2+bX+C=0$, etc. The student chooses the type of problem he would like. The system sets a series of indicators depending on the problem type chosen. The indicators tell the system which problem option has been selected and guide the solution process. For example, one indicator states that both answers are to have the same absolute value; another states that the problem will require simplification as the coefficient of the "X^2" term is not 1. The parameters in the general equation (aX+b) (cX+d) are selected at random within the constraints imposed by the indicators.

The problem is then presented to the student and his answers are checked. If he is incorrect, the solution process is explained to him. The form of this explanation depends on the type of problem being solved. A sample is shown in Table 2.

Siklóssy (16) describes a methodology for the design of a computer tutor which teaches the concepts of set union and intersection. This tutor can randomly generate problems or accept sample sets from the student. If a student's response is incorrect, it is analyzed and the appropriate remedial feedback given. The types of errors which can be detected for a set intersection problem are missing elements, elements are present which belong to only one of the sets, elements are present which belong to neither set, etc.

Hoffman and Seagle (17) describe a methodology for a generative teaching system in which the instructor specifies an individual course for each student. The material to be covered, the problem areas and models to be explored, and the tools which are to be made available are determined from a data base that describes the student's background. The student is quizzed on his ability to select the correct tools for a problem solution as well as his ability to use them.

The authors claim that a wide variety of phenomena in physics, economics and various social sciences can be described by analytical models using polynomial expressions. Specifying the details of the process to be modelled is equivalent to setting a series of linear constraints on the polynomial coefficients. By using the techniques of linear programming, suitable sets of coefficients satisfying these constraints can be obtained. These coefficients would be used in the formulation of the problem to be solved. This procedure has been applied to the generation of economics problems.

An extensive project in the subject area of analytical geometry has been described by Uttal (23). 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 also obtained from a general quadratic equation of the form: $AX^2+BY^2+CX+DY+E=0$.

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.

To evaluate the correctness of a student response a "binary branching tree partitioner" may be utilized. This algorithm is applied to trace algebraic and arithmetic errors by successively decomposing the expression into smaller left and right branches. If the error is judged to be in the left branch, this branch is further decomposed until the source of the student error has been pinpointed.

The last system to be discussed here, "An Associative Learning Project by Rochart, et al., (24), could also have been included in the section on ISO Generative Systems since it uses a simplified form of a semantic network to answer student questions. However, it has been designed to teach accounting, a course which is quantitative in nature.

This system has no capability to generate questions and, in fact, normally operates as a conventional frame-oriented system. However, if the student desires, he can interrupt the lesson sequence to ask a question. The question is scanned for any keywords (words essential to the subject matter), question words, (what, when, where, and how used) and relation words. Stored with each key word is a "property list". Each of the question words is a property whose value is a statement relating the question word and keyword. After the keyword and question word have been identified, the appropriate explanatory statement is printed out.. An example follows:

EXAMPLE of Property List for ASSET:

WHAT = is tangible for intangible property
USE = can be used to generate future value
WHEN = exists if there is value remaining at balance sheet time
WHERE = is noted on the left-hand side of a balance sheet
REL = (is a form of capital)(is increased in value by credit)(is equivalent to property)(is measured by value)...

A student's question such as "What is an asset?" would be answered by the statement: "Asset is tangible or intangible property."

In addition, if there are two keywords recognized in the question, a statement (or set of statements) expressing their relationship will be retrieved and printed out. This statement will be stored with each of the keywords under the property "relationship". If the keywords are not directly related, a path through intermediate keywords is searched for. For example, if the keywords are "assets" and "wealth" the path would be "WEALTH IS ECONOMIC VALUE. VALUE MEASURES AN ASSET."

The authors feel that such a system, while not generative, still presents a useful alternative to frame-oriented CAI. It provides the student with the capability to ask questions when they arise and to temporarily divert from the normal sequence of frames.

The following describes a generative tutor which is utilized in conjunction with an intelligent monitor of student progress. The objective is to provide a greater degree of individualization of instruction than is currently available as well as enable effective teaching of a variety of computer science fundamentals through generative CAI.

IV. GENERATIVE CAI IN DIGITAL SYSTEMS

The author is developing a generative CAI system which will be capable of generating and solving a wide range of problems in the area of digital systems. (18,19) This CAI system is currently being used in the introductory computer science course offered by the Electrical Engineering Department at the University of Connecticut. The course coverage includes an introduction to combinational and sequential design as well as machine and assembly language programming as described in (20).

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. He can also increase or decrease the amount of explanation and monitoring he receives while working on a problem.

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.

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. 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.

The system uses this information to determine how quickly a student 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.

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 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 real number between 0 and 3) in the concept and the difficulty of the sub-task.

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. Table 3 presents examples of different degrees of interaction with a student in an octal addition problem. The first character of all student inputs is understored.

The magnitudes of the increment and decrement for correct and incorrect answers respectively are also individualized to fit the student's past performance in a concept. The mechanism for accomplishing this is described elsewhere (21).

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. Currently, each student is arbitrarily assigned an initial master average of 2. His master average is updated after the completion of a concept.

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.

Once the student's plateau has been determined, the system chooses one concept from among the candidates at that plateau. 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 weighed more heavily), and its relevance to the 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.

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 4 gives some examples of problems which may be generated .

In order to generate problems, a probabilistic grammar must be specified for each problem class. A probabilistic grammar is a formal language in which each production rule is assigned a probability of being applied. These probabilities are defined in such a way that the production rules leading to more difficult problems have higher probability as a student's level increases. In this way, the difficulty of the problem generated is tailored to fit each individual student.

The system has been implemented on the IBM 360/65 at the University of Connecticut. It is programmed in the Conversational Programming System (CPS) (22) CPS is a dialect of PL/I and includes some string processing features which have been extremely useful in programming this system.

V. CONCLUSIONS

This paper has attempted to survey a wide range of generative CAI systems. The purpose of these systems is to reduce the amount of effort required by the instructor to produce new instructional material. Also, these systems are intended to free up the instructional process and provide the student with the capability to assume the initiative and investigate topics which interest him.

Some of these systems have extensive capabilities for natural language communication. They are, thus, able to interpret a student's questions and generate a meaningful, well-phrased response. These systems are provided with a data base of factual information and would be applicable to instruction in the "soft-sciences".

The quantitative systems have less of a need for understanding natural language, though, they could be provided with this capability. Their "data base" consists of a set of algorithms for problem generation and solution. They are oriented towards providing a student with practice in problem solving.

A generative system for teaching digital computer concepts has been described in detail. This system includes an intelligent executive 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 are 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.

All of the systems discussed are somewhat experimental in nature. It is expected that the use of generative CAI will increase as research in this area continues and facilities for CAI become more available and more economical.

REFERENCES

1. Simmons, R. F., "Natural Language Question-Answering Systems: 1969," Communications of the ACM, Vol. 13, No. 1, January 1970, pp. 15-30.

2. Carbonell, J. R. "AI in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction," IEEE Transactions on Man-Machine Systems, Vol. MMS-11, No.4, Dec. 1970 pp.190-202.

3. Carbonell, J. R., "Mixed-Initiative Man-Computer Instructional Dialogues," Bolt Beranek and Newman Report No. 1971, Cambridge, Mass., May 1970.

4. Quillian, M. R., "Semantic Memory," in Minsky, M. (ed) Semantic Information Processing, M.I.T. Press, Cambridge, Mass. 1968, pp. 227-270.

5. Quillian, M. R., "The Teachable Language Comprehender: A Simulation Program and Theory of Language," Communications of the ACM, Vol. 12, No. 8, Aug. 1969, pp. 459-476.

6. Bobrow, D. G., Murphy, D. L. and Teitleman, W., "The BBN-LISP System," Bolt, Beranek and Newman, Cambridge, Mass. 1968.

7. Wexler, J. D., "A Generative, Remedial and Query System for Teaching by Computer," Ph. D. Dissertation, Univ. of Wisconsin, Madison, March 1970.

8. Wexler, J. D., "Information Networks in Generative Computer-Assisted Instruction," IEEE Transactions on Man-Machine Systems, Vol. MMS-11, No. 4, Dec. 1970, pp. 181-190.

9. Simmons, R. F., "Natural Language for Instructional Communication," In Artificial Intelligence and Heuristic Programming, Edinburgh Univ. Press 1971, pp. 191-198.

10. Melton, T. R., "ITI: An Interpretive Tutor," Technical Report No. NL-4, Dept. of Computer Sciences and CAI Laboratory, Univ. of Texas, 1971

11. Moses, J., "A Program for Drilling Students in Freshman Calculus Integration Problems," Project MAC Report MAC-M-369, M.I.T., March 1968.

12. Robinson, J. A., "A Review of Automatic Theorem Proving". In Schwartz, J. T. Proceedings of 19th Symposium in Applied Mathematics, American Mathematical Society, Providence,,1967, pp 1-18.

13. Uhr, L., "Teaching Machine Programs That Generate Problems as a Function of Interaction With Students," Proceedings of 24th ACM National Conference: 125-134, 1969.

14. Wexler, J. D., "A Teaching Program That Generates Simple Arithmetic Problems," International Journal Man-Machine Studies, Vol. 2, No. 1, January 1970, pp. 1-27.

15. Peplinski, C., "A Generating System for CAI Teaching of Simple Algebra Problems," Technical Report 24, Computer Science Dept., University of Wisconsin, Madison, May 1968.

16. Siklóssy, L., "Computer Tutors That Know What They Teach," Proceedings Fall Joint Computer Conference 1970, pp. 251-255.

17. Hoffman, R. B., and Seagle, J. P.,"A Problem Oriented Computer-Based Instructional Procedure," Proceedings of 24th ACM National Conference: pp. 97-110, 1969.

18. Koffman, E. B., "Individualizing Instruction in A Generative CAI Tutor," Communications of the ACM, June, 1972, pp. 472-473.

19. Koffman, E. B., "A Generative CAI Tutor for Computer Science Concepts," Proceedings of the AFIPS 1972 Spring Joint Computer Conference.

20. Booth, T. L., Digital Networks and Computer Systems, Wiley and Sons, New York, 1971.

21. Hall, J. N., "Generation: A Computational Tutor," M.S. Thesis, Department of Electrical Engineering, University of Connecticut, May, 1971.

22. IBM Corporation, "Conversational Programming System (CPS) Terminal User's Manual," IBM Report GH-20-0758-1, 1970.

23. Utall, W. R., Pasich, T., Rogers, M. and Hieronymus, R., "Generative Computer Assisted Instruction," Communication No. 243, U. of Michigan, Mental Health Research Institute, 1969.

24. Rockart, J. E., Scott Morton, M.S. and Zannetos, Z. S., "Associative Learning Project-Phase-1 System," Working Paper, A. P. Sloan School of Management, M.I.T., Jan 1972.

Table 1

Use of "EXPLAIN"

```
SENTENCE: The first plant to appear on a newly formed
          tropical island is the stately and graceful
          coconut palm.

EFFECT OF EXPLAIN COMMAND:
          The plant is the coconut palm
          The plant appears first on a tropical island
          The tropical island is newly formed
          The coconut palm is stately and graceful
```

Table 2

Remedial Feedback for Algebra Problem

```
4X**2-25=0

YOUR ANSWERS ARE X = 2 1/3, X = -2 1/3

GENERATED ANSWERS ARE X = -2 1/2, X = 2 1/2


YOUR HAVE MADE AN ERROR. THE PROBLEM IS

4X**2 -25 = 0

THE EXAMPLE IS A DIFFERENCE OF TWO SQUARES. AND

THUS FACTORS INTO THE SUM AND DIFFERENCE OF THE

SQUARE ROOTS, AS FOLLOWS
```

(2X + 5) (2X -5) = 0

THEN 2X = -5 and 2X = 5.

DIVIDING TO SOLVE, S = -2 1/2 or X = 2 1/2

Table 3

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 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?

⋮

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.

Table 4

Sample Problems Generated

1. Calculate A3.E - 2.B4 in the base 16 using complement subtraction.
2. Derive the truth table for $(((P \oplus Q) \uparrow R) \vee (Q \downarrow R))$
3. Design a combinational logic circuit such that:

   Input A is 2 bits long. Input B is 2 bits long. There is a control bit I such that if I = 0, the output X = A + B; if I = 1, the output X = A * B.
4. Minimize a function which has 0, 1, 3, 4, 10 as miniterms and 2, 8, 14 as don't cares using Karnaugh Maps.
5. Design a modulo 6 counter with a single input line X.

   If X = 0 the count increases by 1, if X = 1 the count decreases by 2.
   The output should be 1 when the count is 0.
6. Design a sequential circuit to recognize the input sequence 0011.
7. Derive the transition table for a sequential network with 2 Flip-Flops such that $J1 = (Y2 \vee X)$, $K1 = (Y1 \vee Y2)$; $S2 = (\neg X \wedge (Y1 \vee Y2))$, $R2 = X$.

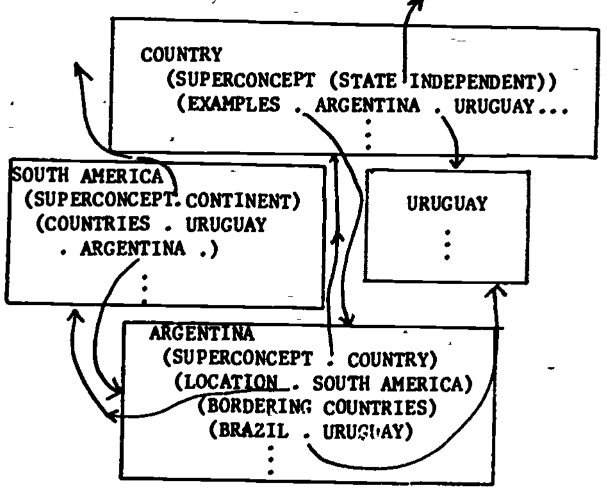

Figure 1
Part of a Semantic Network on South America

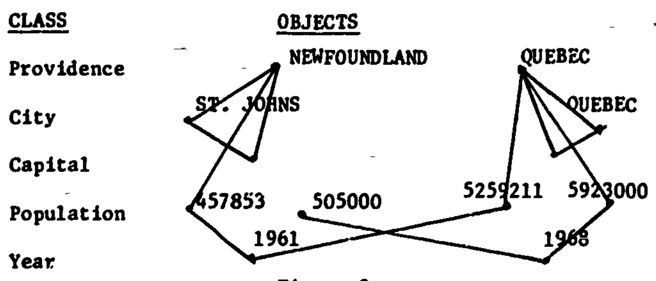

Figure 2

Part of an Information Network