SESSION VIII INSTRUCTION AND MEASUREMENT STUDENT COMPETITION AWARD

A very general problem-oriented CAI system

CONAL ELLIOTT

University of California, Santa Barbara, California 93106

A very general problem-oriented CAI system is described that allows an individual with no programming knowledge to control the nature and content of student-machine interaction. A functioning stand-alone system is outlined for instruction in linear equations of variable complexity.

Computer-aided instruction (CAI) systems have recently gone beyond their original function as tools for disseminating scripted information. Now they are also used as devices for informing the researcher about fundamental cognitive processes (e.g., Bregar & Farley, 1979; Brown & Burton, 1975). These systems take on more of the form of a "tutor" (e.g., Millward, 1979; Millward, Mazzucchelli, Magoon, & Moore, 1978) than a workbook." However, as currently configured, they sacrifice generality of applicability for generality of psychological process: They do not travel well from one situation to another. So while CAI offers the psychologist a powerful research tool, it is a burdensome one. The programming effort needed is considerable.

The system described here attempts to encourage use of CAI by providing a general structure for problem-oriented instruction that can be tailored to specific use, thus eliminating much of the work involved. It is also an immediately functioning stand-alone system for instruction in linear equations of variable complexity. The system is written in C and is implemented on a VAX-11/780 under Version 7 UNIX at the University of California at Santa Barbara computer systems laboratory.

The rationale of the system is to allow a person with no programming knowledge to have great control over the nature and content of the student-machine interaction. The resulting system provides a researcher or instructor with a facile tool appropriate to a broad range of topics and situations.

OVERVIEW

The functioning system consists primarily of three interactive components, illustrated in Figure 1. The

Requests for additional information or reprints should be addressed to the author at 703 Bolton Walk No 204, Goleta, California 93117, or The College of Creative Studies, University of California, Santa Barbara, California 93106.

Figure 1. Functioning system with three interactive components: interactive lessons, problem maker, and director.

first of these is a collection of interactive lessons. They consist primarily of a list of events to transpire in the teaching process. These events might include instructional text, dialogue frames for interaction with students, and a specification of the type of problems that the student will attempt.

The second component is the problem maker. The problem maker constructs a problem, presents it to the student, and helpfully interacts while the student tries to solve the problem, all within limits specified by the lesson designer.

The third component is the director. This component provides continuity between lessons by keeping a list of lessons that will be called up in sequence, or according to a specific request from a currently active lesson. This flexibility in lesson control adds to the system's generality by allowing the instructor to specify a number of lessons that are to be covered in the course of a learning procedure, while still permitting on-line judgment for remedial or enrichment instruction.

PREPARATION OF THE SYSTEM

To use the system, the instructor proceeds in three phases, illustrated in Figure 2. First she/he designs a learning procedure by making up one or more lesson descriptions. Second, the instructor selects one or more problem makers that will generate the problems and give

Figure 2. Preparation of the system: making up lesson description, selecting problem makers, and creating interactive lessons.

the appropriate help as a function of the Phase 1 specifications. Third, the instructor invokes a program that makes use of the Phase 1 lesson descriptions to create the interactive lessons. The following provides a bit more detail of the three phases involved in the preparation of a learning procedure.

Phase 1

To begin, the instructor makes a collection of files, the lesson descriptions, using a simple language that encompasses (1) enumeration of the instructional text that the student will see during the lessons, (2) specification of the type and sequence of problems to be given, and (3) constraints on the environment in which the student will be solving problems (e.g., how many tries or hints to allow, what kind of problem manipulations to allow, etc.).

Phase 2

Next, the instructor writes a problem maker or chooses one that is already written. This program will trade off with the lessons in interacting with the student. Because of its generality, one problem maker may be used by many different lessons and in different studies.

Phase 3

The program makelesson is used to convert the lesson descriptions into programs, called interactive lessons, which are run during the learning procedure. Then the instructor makes a list of the lessons involved in a given learning procedure. At this point, the system is ready to be used. The learning procedure is started by running the program director, giving it the lesson list. Director starts up the first lesson and waits for it to finish. Other lessons are started up as a consequence of the instructor's specified sequence or an overriding judgment made by a lesson.

THE LINEAR EQUATIONS PROBLEM MAKER

The system is currently equipped with a problem maker that generates linear equations of arbitrary form and complexity. It uses a general interface, the same that is available to other problem makers.

This problem maker, which I call "linear," deals with linear equations in 1-26 variables. One variable is to be solved for, and the others appear as part of the answer. As with all problem makers, it has a dual role. It must (1) create and present a problem when requested to by a lesson and (2) interact helpfully while the student tries to solve the problem.

A problem is described to linear by providing a set of criteria that must be satisfied, thus describing a whole class of problems. This class can be large or small, depending on the strictness of the requirements. For example, consider the following specification. The problem is to be in the variables r and s , to be solved for r , with one occurrence of each variable and two constants. The problem will contain fractions, but the answer will contain only whole numbers. One problem generated from such a specification is " $2/5r + 1/4 = 9/4 - 8/5s$. Solve for r . (answer: $r = 5 - 4s$)." Another problem generated from the same specification is " $9/5 + 9/5s = -9/5r - 9/5$. Solve for r . (answer: $r = -2 - s$)."

Once the problem is presented to the student, the current lesson is inactive for a period during which the problem maker handles the interaction. This period is devoted to solution of the given problem.

There are several options potentially available to the student. She/he may do any of the following: (1) suggest an attempted answer, (2) give an equation purported to be a simplification of the given one, (3) request a hint, (4) verbally suggest an operation to perform on the equation, (5) ask for help in a computation, (6) ask for a step-by-step solution of the problem, or (7) give up and request the answer. Since certain of these options are inappropriate in some learning situations, the instructor may restrict their use. For instance, hints may be disallowed, the number of tries may be limited to three, and so on. The student is told if she/he tries to use an option that has been disallowed or exhausted. As an example, Figure 3 shows a protocol segment. The

```

Solve for x:
-2/7 x - 2/5 y + 4/7 = 4/5 - 4/7 y
  add 2/5y to each side
-2/7 x - 2/5 y + 4/7 + 2/5 y = 4/5 - 4/7 y + 2/5 y
  combine the y terms
-2/7 x + 4/7 = 4/5 - 6/35 y
  compute 4/5-4/7
8/35
  -2/7x=8/35-6/35y
That is right. The problem is now
-2/7 x = 8/35 - 6/35 y
  hint
multiply each side of the equation by 7
  multiply by 7
-2 x = 8/5 - 6/5 y
  x = 4/5 - 3/5y
That is not right. Please try again. The problem is still
-2 x = 8/5 - 6/5 y
  x = -4/5 + 3/5y
That is correct.
  
```

Figure 3. Example of student/problem maker interaction. Protocol segment. Indented lines are entered by student; others are generated by linear.

indented lines were entered by the student, and the others were generated by linear.

DESCRIPTION OF THE INTERACTIVE LESSONS

The instructor describes the nature of the interactive lessons using a simple special-purpose language. A description in this language is interpreted line by line, resulting in the construction of a running program. Because it is simple to make or change these lessons, the system is freed from a burden common to CAI systems, namely, that of having the particulars of material presentation so imbedded in the system that it is impossible, impractical, or just undesirable to use the system in an altered context. The desired effect of this system, rather, is to encourage an instructor to add his/her own style and experience to his/her particular use of the system. For the same reasons, it offers the person interested in learning research a tool whose use lends itself to the investigation of varied or changing hypotheses. Figure 4 shows a lesson protocol and description exactly as given to makelesson. The lines beginning with a period have special meaning, and the others are display text. (For more details of the lesson description language, see Appendix A.)

SPECIAL FEATURES OF THE SYSTEM

Linear Equation Answer Recognition

Linear performs a semantic rather than syntactic analysis of a student's response. When the student enters an equation, linear parses the response into an internal canonical form. It then performs an algebraic comparison between the suggested answer and the correct answer. If the student has offered a simplification instead of a solution, linear makes sure that the asserted simplification has the same solution set as the original problem. The student then works on the simplified version. If the student enters some verbal response, such as "reverse the equation," linear picks out the words it needs to "understand" the request.

Starting and Stopping the System

The student may request to stop the learning procedure at any time the system is waiting for a response just by typing "stop" (unless disallowed by the instructor). When this happens, all relevant data are automatically saved in a continuation file. When the learning procedure is restarted, this information is reloaded and the system starts up either where the student left off or at some earlier specified point.

Lesson Answer Checking

While making up a lesson description, the instructor may take advantage of a rudimentary answer-checking mechanism. This is done by giving a list of allowable answers along with the question to be asked of the student. The lesson variable "match" is automatically set to tell which answer was matched. For instance, if the lesson description includes the lines

```
.input ^operation "What operation would you perform now?"
    "add*" "subtract*" "combine" "reverse"
.if (match == 3 or match == 4)
    Probably it would be more helpful to add or subtract something.
.else
    I think I would too.
```

then the following might occur in a lesson. The indented lines are student responses.

```
What operation would you perform now?
    divide by 7
Please answer combine or reverse, or begin your answer with add
or subtract.
    add 2/3y
I think I would too.
```

Note that the student's first response, "divide by 7," was not one of the four allowable responses, so the student was asked to answer again. Notice also the special meaning implied by ending the option with an asterisk.

Recording the Interaction

The system is equipped with a simple and flexible mechanism for recording the interaction. By setting a

<p>What is your name? Marianne Welcome to the algebra project, Marianne. I think you will find it instructive and enjoyable. Let's begin with some simple problems. Solve for x: $\frac{3}{2}x - 4 = -4x + \frac{3}{2}$. . . That is correct. You got 3 out of 5 right. Good bye for now Marianne.</p>	<pre>.set protocol "exampleout" .input ^lname "What is your name?" Welcome to the algebra project, ^name. I think you will find it instructive and enjoyable. Let's begin with some simple problems. .set pmaker "linear" .set ^30linearoptions "tries 3 hints 4 giveup 0" .set tomaker "linearoptions" .problem (solved<3) "u,c: b2 d1..3;x:s+;" You got 3 out of #problems right. Good bye for now ^name.</pre>	<p>Start recording into file "exampleout". Save answer in the variable "name". Give welcoming text. Notice the variable substitution.</p> <p>Choose problem maker. Allow three tries, four hints and no giving up. Send this variable to linear. Present a problem as long as fewer than three have been solved. The unknown and constant terms show up twice with denominators from one to three. The answer is to be positive.</p>
Protocol	Lesson description	Notes

Figure 4. Example of lesson and description language.

lesson parameter to the name of some file, recording begins in the named file. This may be stopped, restarted, or rerouted at any time, so that only the desired events need be recorded.

Specifying the Problem Solving Environment for Linear

Once linear has presented a problem, the student has many options for working on it. The instructor may wish to disallow or restrict the use of some of these. This may be done by setting the lesson parameter "linearoptions" to reflect a set of limits to hold simultaneously while the problem is being worked on. Each limit can refer to the use of one option or the combined use of several. For instance, if linearoptions were set to "hints 1 (add subtract multiply divide) 4 tries 3," then the student would be allowed at most one hint, three tries, and four requests of the form add, subtract, multiply, or divide.

Format of Linear Equation Presentation

When a problem is presented, the variables appear in lowercase and coefficients appear as improper fractions instead of mixed numbers. This may be changed by setting appropriate parameters in the lesson description.

REFERENCES

- BREGAR, W. S., & FARLEY, A. M. Toward the development of intelligent CAI. *Proceedings of the Association for the Development of Computer-Based Instructional Systems*, 1979, 1, 120-128.
- BROWN, J. S., & BURTON, R. R. Multiple representations of knowledge for tutorial reasoning. In D. G. Bobrow & A. Collins (Eds.), *Representation and understanding studies in cognitive science*. New York: Academic Press, 1975.
- MILLWARD, R. Teaching a computer to teach. *Behavior Research Methods & Instrumentation*, 1979, 11, 102-110.
- MILLWARD, R., MAZZUCHELLI, L., MAGOON, S., & MOORE, R. Intelligent computer-assisted instruction. *Behavior Research Methods & Instrumentation*, 1978, 10, 213-217.
- NILSSON, N. J. *Principles of artificial intelligence*. Palo Alto, Calif: Tioga, 1980.

APPENDIX A HOW LINEAR WORKS

An explanation of how linear's problem maker works should prove enlightening, as it embodies general principles for finding objects satisfying given requirements. Although I will frame the discussion in terms of a linear equation in one or more variables, keep in mind its general applicability. First, let us consider how one might describe a particular class of problems: (1) The kind of terms involved—Besides the variable to be solved for, there may be constants and/or additional variables that will show up in the answer. By a "type" of term, I mean constant, unknown, or some additional variable. (2) For each type, where and how many times such terms will appear. (3) For each term of each type, what it (if constant) or its coefficients will look like (i.e., positive or negative, integral or fractional, how big or small the numerator or denominator should be). (4) For the constant and each additional variable, what the corresponding term in the answer will be like. (5) What the variables are to be called. (6) Possibly something about the order of the terms in the presented problem.

By specifying allowable ranges for the above criteria, one names a class of linear equations. The idea of the problem maker is to sequentially instantiate (i.e., guess a value for) each criterion within the allowed ranges in order to find a satisfactory problem. The method used is a nondeterministic programming technique, called "backtracking." The instantiation process is like searching a tree. The root node of the tree is the initially specified problem criteria, and from each node there is a collection of edges representing possible value decisions for some as yet uninstantiated range involved in the problem. Each edge, then, leads to another node that is a refinement of the previous criteria set. The search process is to follow some path from the root of the tree, so that with each node along the path, the range of possible problems is further restricted. This process stops either when all the ranges are chosen or when the problem maker detects that there is no problem satisfying the resultant criteria set. The latter case leads to the backtracking, which consists of "unmaking" the last guess and making a different one. In terms of the tree, this means backtracking up the last edge and choosing another one. The net result is a depth-first search on the tree of possibilities. The class of problems named by the problem description is the set of successful end nodes on the associated tree.

By itself, this method would be too slow for practical (especially on-line) use, since the associated tree could be very large. Therefore, it was necessary to introduce mechanisms to cut down on the number of bad guesses. This is known as "tree pruning," and its particulars depend strongly on the kind of object being constructed (here, a linear equation). For example, linear makes considerations of possible interrelating sizes of terms, prime power divisors of denominators, and solutions of linear diophantine equations.

This method is widely applicable. Interesting applications and extensions of these techniques may be found in Nilsson (1980).

APPENDIX B INTERACTION LESSONS

There is a special language for describing the sessions. Such a description is interpreted line by line, and a program in the language C is generated. If a line does not begin with a period, it is understood to be output text. It may be interspersed with variables and expressions that will be evaluated and displayed during the learning procedure. Lines beginning with a period are command lines, some of which are summarized below:

First Word	Description
.input	Print a question on the terminal and read in the answer. The answer is saved in some named variable and can hold a character string, whole number, or floating-point number. Optionally, a list of allowable answers may be specified, in which case the program will make sure the student gives one of these answers and keep track of which answer was matched.
.label	For interrupting the flow of the lesson.
.goto	
.if	Perform next sequence of instructions predicated on given condition.
.else	
.set	Set the value of a variable, which may be numeric or string. The value may be a constant or an expression. Instructions that are to be used in different situations may be consolidated into routines that can then be requested by name elsewhere.
.do	
.routine	
.problem	Ask problem maker to present a problem according to the given description as long as given condition is true. Optionally, routines may be named to be done before and/or after each problem.

Some variables have special meaning to make a lesson. Still others may be given special meaning by a problem maker.

Variable Name	Description		
pmaker	The name of the problem maker used (e.g., "linear").	stopword	The string a student may enter indicating she/he wants to stop the lesson.
restart	Determine what happens when the student asks to stop working. It may be set to "nowhere," "anywhere," or "here." The first means to disallow stopping in this area. The other two allow stopping. The second says to restart at the point at which the student left, and the last indicates the place to return to when the procedure is started up again.	match	After an .input line in which a list of permissible answers was given, this variable contains the number of the answer matched.
		solved	The number of problems the student has solved in this set.
		problems	The number of problems presented so far in this set.
		linearoptions	For linear, this determines what options to allow the student in solving a problem. It is set to a list of option/count pairs indicating that a certain option may be used at most the given number of times during the solution of each problem.
protocol	The name of the file in which recording of the interaction is to take place. One only needs to alter the value of this variable to turn recording on or off, or to direct certain output to certain files.	capitalvars	For linear, this says whether the variables are to appear as capital or lowercase letters.
		mixednumber	For linear, whether fractions are to appear as mixed numbers or improper fractions.