MODELLING STUDENT ACQUISITION OF PROBLEM-SOLVING SKILLS

Robert Smith
Department of Computer Science
Rutgers University
New Brunswick, N. J. 08903

ABSTRACT

This paper describes the design of a system that simulates a human student learning to prove theorems in logic by interacting with a curriculum designed to teach those skills. The paper argues that sequences in this curriculum use instructional strategies, and that the student recognizes* these strategies in driving the learning process.

I. INTRODUCTION

A central issue in the design of learning systems (LS's) is the classification of the sources of information that the system uses for its acquisition. The general notion is that an LS begins with certain knowledge and capabilities, and then extracts information from training sequences or experiments in the acquisition process. Implicit within this general characterization is the idea that the LS enforces some kind of interpretation on the training sequence by way of driving the acquisition process. Often the nature of the interpretation is left implicit. An example of an LS that makes somewhat explicit the interpretation of its training sequence is Winston's program for learning structure descriptions, where the program interprets the near miss example as providing key information about the structure being learned [4].

We speculate that much human learning takes place in a more richly structured environment, wherein the human learner is interpreting the instructional sequences provided to him in a richer way than LS's typically envision. Indeed, most LS's have made few if any explicit assumptions about the structure of the environment in which the training sequence occurs. One particularly rich environment is learning by teaching. We suggest that teachers use certain instructional strategies in presenting material, and that students recognize these strategies.

This paper describes the motivation for an LS called REDHOT. REDHOT is a simulation of a student acquiring the skill of constructing proofs in elementary logic. We characterize this skill as consisting of (1) primitive operators in the form of natural-deduction rules of inference, (2) "macro moves" consisting of several rules of inference, and (3) application heuristics that describe when to use the rules. The central theme of this research is to model the acquisition of these skills around the recognition of instructional strategies in a curriculum designed to teach the student.

II. CURRICULUM FOR REDHOT

We are using the curriculum from the computer-assisted instruction (CAI) course developed at Stanford University by Patrick Suppes and co-workers. (See [3] for details.) This CAI system is used as the sole mode of instruction for approximately 300 Stanford students each year. We chose a CAI curriculum because we thought that the explicitness inherent in a successful CAI system developed and tested over a number of years might make the instructional strategies relatively clear.

The curriculum contains explanatory text, examples, exercises, and hints. The explanatory text is rather large, and uses both computer-generated audio and display as modes of presentation. The presentation strategy used by the actual CAI system is linear through the curriculum. For use with REDHOT, we have developed a stylized curriculum in an artificial language CL. It contains the examples, exercises, partially formed rules, and hints.

The exercises are the central part of the curriculum. There are approximately 500 theorems that the student is asked to prove, with about 200 in propositional logic, 200 in elementary algebra, and 100 in the theory of quantification.

The human student performs these exercises by giving the steps of the developing proof to an interactive proof checker. This proof checker is the heart of the original instructional system. We developed a version of this proof checker for use with the REDHOT student simulation.

III. THE DESIGN OF REDHOT

REDHOT learns rules for proving theorems. These rules are initially the natural deduction

---

* I would like to thank Phyllis Walker for analysis of the curriculum and student protocols; Saul Amarel, Tom Mitchell, Don Smith, and N. Sridharan for many ideas and assistance. The research reported here is sponsored by the Office of Naval Research under contract N00014-79-C-0780. We gratefully acknowledge their support for this work.

rules of many logic systems. The student improves upon these rules by building macro operators and by adding heuristics to existing rules--i.e., giving strategic advice* in the left-hand-sides of the production rules.

For example, the rule AA ("affirm the antecedent", the system's version of modus ponens) can be stated as the following rule:

```
┌──────────────────────────────────────┐
│              Rule AA                 │
│  GOAL:  Derive  Q                    │
│  Prerequisites:                      │
│         P already on some line i     │
│         P -> Q on some line j        │
│  Method:                             │
│         AA command on lines i and j  │
│  Heuristics:  None (yet)             │
│  Effectiveness:  Perfect             │
└──────────────────────────────────────┘
```

In the above, we have adopted a style of rule presentation that is similar to the rules of a production system. The letters P and Q stand for arbitrary formulas, and i and j for arbitrary lines of the already existing proof. The goal tells what the rule will produce. The prerequisites tell us that two formulas of the appropriate forms are needed, and the method gives the schematic command to the proof checker that will accomplish this. The heuristics associated with a rule are learned by the system, and indicate how the rule should be used. Effectiveness of the rule is also learned, and indicates how effective the rule will be in achieving its goal given its prerequisites. The effectiveness of this rule is "perfect" since the rule is given as such in the curriculum.

The underlying problem solver for REDHOT is a depth-first, heuristic problem solving over the existing rules. It is assumed that: (a) the system has sufficient memory and CPU facilities for this search, if necessary; and (b) that the underlying pattern matchers are sufficient to match arbitrary formulas, line numbers, etc. Both of these assumptions are open to criticism on the grounds of being psychologically unrealistic. One of the goals of the construction of REDHOT is to decide on plausible ways to restrict the problem solver and pattern matcher to make a more realistic system.

REDHOT learns the heuristics for the application of the rule. These heuristics are stated in a heuristic-language HL, which is strongly tied to the curriculum language CL. The heuristics associated with a rule guide the student as to when to try that rule or not to try it.

For example, the curriculum appears to teach the student that the AA rule is a plausible thing to try when the prerequisites are available (whether or not the goal is immediately desired). This is one of the primitives of the HL heuristics language.

An example of a macro operator that is not a primitive rule is the "multiple AA" macro move. A paraphrase of this macro operator might be:

```
┌──────────────────────────────────────┐
│          Multiple-AA Macro Move      │
│  IF you want to prove Q              │
│     AND                              │
│     have P, P -> P1, P1 -> P2, ..., Pn -> Q │
│  THEN                                │
│     make multiple AA applications,   │
│     which is guaranteed to succeed   │
└──────────────────────────────────────┘
```

We discuss below the training sequence that teaches this macro move.

IV. THE RECOGNITION OF INSTRUCTIONAL STRATEGIES

REDHOT bases its acquisition of application heuristics and macro operators on its recognition of instructional strategies in the training sequence. For example, consider the sequences of exercises in Figure 1, taken from the actual curriculum. The sequence, which is at the beginning of the whole curriculum, gives the primitive rule of inference AA, then shows successive elaborations of the use of that rule of inference. REDHOT detects this to be a use of a strategy called focus and elaborate, in which a rule is first focussed, and then a particular elaboration is given.

Teacher: Here is a rule called AA.

Teacher: Here are some exercises involving AA:

1.	Goal:	Q
	Premises:	S -> Q, S
2-5	[Several more exercises with different formulas.]	
6.	Goal:	W
	Premises:	S -> Q, Q -> W, S
7.	Goal:	S
	Premises:	R -> S, Q -> W, W -> R, Q
8,9	[Two more similar exercises involving multiple applications of AA.]	

Figure 1
Sequence of Exercises for Learning Multiple Application of AA Command

In the above training sequence, REDHOT takes steps 1-5 as focussing on the AA rule, and steps 6-9 as providing a useful elaboration of that rule, in the form of the macro operator for multiple application.

---

* See [1] for a conceptual discussion of the levels through which this process might proceed. One way to regard this research is a suggestion of the mechanism for this acquisition of heuristics and macro moves.

A second example of the use of an instructional strategy concerns removing possible bugs in learned heuristics and macro operators. We illustrate this with the macro rule for conditional proof, a common strategy in logic and mathematics, which we paraphrase as follows:

```
Conditional Proof MACRO MOVE
IF you want to prove P -> Q
THEN
      ASSUME P as a working premise
      PROVE Q (likely from P);
      APPLY "CP" rule, removing premise
```

The actual instructional sequence goes to great length to teach this principle, and students typically have a difficult time with the principle; one defective version of the rule that students seem to learn is the following:

```
"Defective" MACRO MOVE
IF you have a formula P -> Q
   AND you want to prove Q
THEN
      ASSUME P as a working premise;
      PROVE Q using AA;
```

This is a very poor strategy; but the evidence suggest that over half of the students learn it. The following exercise seems to help students debug the rule "Defective".

Derive: S -> (Q OR R)
Premise: (R -> R) -> (Q OR R)

In examining student protocols, we see that many students will try several times to apply the "Defective" rule to this exercise. Finally, (we speculate) they realize that (R -> R) is already something that they know how to prove, using a previously learned macro operator. Then, the actual proof becomes obvious, the student corrects the defective rule, and goes on to handle similar exercises correctly. We call this instructional strategy "focus and frustrate", wherein a student discovers--somewhat traumatically--that a rule he learned is defective.

Therefore, an exercise such as the above is not just randomly selected, but instead tests possible student "bugs" in an exact way. Notice that it is one of the simplest exercises that will discriminate between the correct and defective formulations of the macro rule for conditional proof. (See [2] for a discussion of "debugging" student skills.)

V. REDHOT AND LEARNING SYSTEMS

Like many LS's, REDHOT starts with the ability to state everything it will "learn", in some sense at least. The initial rules for solving the problem (the natural deduction rules for logic) are complete with respect to the underlying problem solver--unless it is restricted in time/space (in practice it is). The heuristic and macro languages are also given in advance, and they of course define a space of the possible rules that might be learned. So, the object is to select among heuristics and macro rules in this space. One way to formulate doing this is by experimentation or exploration. REDHOT selects objects from this meta-space by being taught.

Learning by being taught consists of the "teacher" laying out exercises in an organized and structured way, and the student recognizing something of that structure. The student makes--believes that he is entitled to make--fairly bold hypotheses about the rules he is learning, and relies on the training sequence to contain exercises that will check for common errors that he the student may have made in formulating these rules. REDHOT compares somewhat radically to many LS's that rely on a somewhat slow, computationally coherent delimitation of the rule (or concept) involved.

We speculate that learning by "discovery" or "experimentation" is a slow process for even humans, done over the eons of time and through social interaction. Most human learning is by being taught, and one can argue that AI should give attention to the relation between learning and teaching, in terms of modelling the acquisition of concepts, problem-solving skills, and natural language. We further speculate that learning by "discovery" will be aided by extracting as much information as possible from the structure of the environment in which the LS operates.

REFERENCES

[1] Amarel, Saul, "An Approach to Problem Solving and Theorem Proving in the Propositional Calculus", in Systems and Computer Science, (Hart and Takasu, eds.), Toronto: University of Toronto Press, 1967.

[2] Brown, John Seely, Burton, Richard R., and Larkin, Kathy M., "Representing and Using Procedural Bugs for Educational Purposes", in Proceedings of the 1977 Annual Conference of the Association for Computing Machinery, New York, 1977.

[3] Suppes, P., Smith, R. L., and Beard, M., "University-level Computer-assisted Instruction at Stanford: 1975", in Instructional Science, 1977, 6, 151-185.

[4] Winston, Patrick Henry, "Learning Structural Descriptions from Examples", Ph.D. thesis, in The Psychology of Computer Vision, (Patrick Henry Winston, ed.), McGraw-Hill, New York, 1975.