UC Merced
Proceedings of the Annual Meeting of the Cognitive Science Society

Title
Cognitive Principles In The Design Of Computer Tutors

Permalink
https://escholarship.org/uc/item/0d07z172

Journal
Proceedings of the Annual Meeting of the Cognitive Science Society, 6(0)

Authors
Anderson, John R.
Boyle, C. Franklin
Farrell, Robert
et al.

Publication Date
1984

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

COGNITIVE PRINCIPLES IN THE DESIGN OF COMPUTER TUTORS

John R. Anderson
C. Franklin Boyle
Robert Farrell
Brian Reiser
Carnegie-Mellon University

This paper will identify and justify a set of principles derived from ACT (Anderson, 1983) for designing intelligent computer tutors (Sleeman & Brown, 1982). In doing this we will be drawing on our studies of high school students learning geometry and college students learning to program in LISP. We have observed four students spend approximately 30 hours studying beginning geometry and three students similarly spending 30 hours learning LISP. We recorded these sessions and have analyzed them to varying degrees. Some of these analyses have been reported in a series of prior publications (Anderson, 1981; Anderson, 1982; Anderson, 1983a; Anderson, Farrell, & Sauers, 1984; Anderson, Pirolli, & Farrell, in press). This data base has served as a rich source of information about the acquisition of problem-solving skill and has heavily influenced our design of computer tutors. We have used this data base to develop tutors both the LISP and geometry. These tutors are described elsewhere (Boyle & Anderson, 1984; Farrell, Anderson, & Reiser, 1984).

Principle 1: Identify the Goal Structure of the Problem Space

According to the ACT theory, and indeed most cognitive theories of problem-solving, the problem solving behavior is organized around a hierarchical representation of the current goals. It is important that this goal structure be communicated to the student and instruction be cast in terms of the goal structure. It is not communicated in typical instruction in courses like geometry.

Proofs in geometry are almost universally in a two-column form. It is basically a linear structure of pairs where each pair is a statement and justification. Typical instruction encourages the belief that the goal structure of the student should mimic this linear structure--that at any point in the proof the student will have generated an initial part of the structure and the current goal is to generate the next line of the structure.

There are two serious flaws with using linear proofs as goal structures. First this practice denies the validity of problem-solving search. It encourages the idea that the correct next line should be obvious, but finding the next line often involves considerable planning and search. Students engage in search but feel bad about themselves because they do. Second, search in such a linear structure is doomed to be hopelessly unguided. If the only constraint is to generate a legal line, the search space for the correct proof is hopelessly large.

We have observed students flail at solving geometry problems because they try to work within this linear goal structure. We have evidence that successful students represent proofs to themselves as hierarchical structures of implications that start with the givens of a problem and end

in the conclusion to be proven. It needs to be emphasized that conventional instruction does not communicate this structure and students hardly find it obvious. This deficit is particularly grievous because the successful student's goal structure is much more closely related to this hierarchical proof structure than it is to the linear structure of a two-column proof. Basically, the successful student engages in a forward search from the givens and a backward search from the to-be-proven statement.

Principle 2: Provide Instruction in the Problem-Solving Context

Students appear to learn information better if that information is presented during problem solving rather than during instruction that is apart from the problem-solving context. There are a number of reasons why this should be so:

First, there is evidence that memories are associated to the features of the context in which they were learned. The probability of retrieving the memories is increased when the context of recall matches the context of study (Tulving, 1983; Tulving and Thomson, 1973). An extreme example of this was shown by Ross (1984) who found that secretaries were more likely to remember a text-editor command learned in the context of a recipe if they were currently editing another recipe.

Second, it is often difficult to properly encode and understand information presented outside of a problem context and so its applicability might not be recognized in a problem context. For instance, students may not realize that a top-level variable is really the same thing as a function argument even though they are obliquely told so. As another example, many students reading the side-angle-side postulate may not know what included angle means and so misapply that postulate.

Third, even if a student can recall the information and apply it correctly, they are often faced with many potentially applicable pieces of information and do not know which one to use. We have frequently observed students painfully trying dozens of theorems and postulates in geometry before finding the right one. The basic problem is that knowledge is taught in the abstract and the student must learn the goals to which that knowledge is applicable. If the knowledge is presented in a problem-solving context its goal-relevance is much more apparent.

Principle 3: Provide Immediate Feedback on Errors

Novices make errors both in selecting wrong solution paths and in incorrectly applying the rules of the domain. Errors are an inevitable part of learning, but the cost of these errors to the learner is often higher than is necessary. They can severely add to the amount of time required for learning. More than half of our subjects' problem-solving sessions were actually spent exploring wrong paths or recovering from erroneous steps. Relatively little is learned while students are trying to get out of the holes they have dug for themselves.

In addition, errors often confuse the picture and make it difficult to determine which steps were right or wrong. The classic example of this is the student who finally stumbles onto the correct code but does not

understand why it works. Students often progress in this trial and error mode with respect to LISP evaluation: they don't know when an element will be treated as a function, a variable, or a literal but play around with parentheses and quotes until they get something to work. It is particularly difficult to learn from errors when the feedback on the errors comes at a delay. We (Lewis & Anderson, submitted) have shown that subjects learn more slowly in a problem-solving situation where they are allowed to go down erroneous paths and are only given feedback at delay. Also, the importance of immediate feedback has been well documented in other learning situations (Bilodeau, 1969; Skinner, 1958).

Another cost of errors is the demoralization of the student. In these problem-solving domains errors can be very frequent and frustrating. We believe that much of the negative attitudes and math phobias derive from the bitter experiences of students with errors.

Principle 4: Minimize Working Memory Load

Solving problems often requires holding a great deal of requisite information in a mental working memory. If some of that requisite information is lost there will be errors. It surprised us to find in our LISP protocols that most of the student errors appear to be due to working memory failures. A frequent and disastrous type of error is "losing a level of complexity". One way this manifests itself is that subjects lose track of one level of parentheses. Another way this occurs is when subjects plan to use function1 within function2 within function3, but forget the intermediate function and write function3 directly within function1.

A good human tutor can recognize errors of working memory and typically provides quick correction (McKendree, Reiser, and Anderson, 1984). Tutors realize that there is little profit in allowing the student to continue after making such errors. However, human tutors really have no means at their disposal to reduce the working memory load. This is one of the ways we think computer tutors can be an improvement over human tutors--one can externalize much of working memory on the computer screen. This involves keeping partial products and goal structures available in windows.

Principle 5: Represent the Student as a Production Set

All of our work on skill acquisition has modelled students' behavior as being generated by a set of productions. There is a fair amount of evidence for this view of human problem-solving (e.g., Anderson, 1983; Newell & Simon, 1972). It is also the case that numerous other efforts in the domain of intelligent tutoring have represented the to-be-tutored skill as a production set (e.g., Brown and Van Lehn, 1980; O'Shea, 1979; Sleeman, 1982).

Productions in ACT represent the knowledge underlying a problem-solving skill as a set of goal-oriented rules. Some representative examples for LISP and geometry are:

IF the goal is to insert an element into a list
THEN plan to use CONS and set as subgoals
1. To code the element
2. To code the list

IF the goal is to code a function that calculates a relation on a list
THEN try to use CDR-recursion and set as subgoals
1. To code the terminating condition
2. To code the recursive condition

IF the goal is to prove $\langle XYZ \overleftarrow{=} \langle UVW$
and $\overline{XY} \overleftarrow{=} \overline{UV}$
and $\overline{YZ} \overleftarrow{=} \overline{VW}$
THEN plan to use side-angle-side and set as a subgoal
1. To prove $\langle XYZ \overleftarrow{=} \langle UVW$

Such rules not only enable the system to follow student problem-solving but they define an appropriate grain size for instruction. Basically, our tutoring systems monitor whether a student uses each rule correctly and corrects any incorrect or missing rules. As emphasized by Brown and Van Lehn, student misconceptions or bugs can be organized as perturbations of correct rules.

Human tutors seem to intuit an appropriate grain size of rules for instruction but often their intuitions are wrong. This is one place where a system based on careful analysis of student problem-solving may be able to outperform the typical human tutor.

Principle 6: Adjust the Grain Size of Instruction According to Learning Principles

One of the reasons human tutors have difficulty with the grain size for instructing students is that the grain size changes as experience is acquired in the domain. According to the ACT learning theory, this change is produced by a knowledge compilation process that collapses a sequence of productions into larger "macro" production rules. Human tutors, being highly skilled in the domain, exemplify a large grain size in their problem-solving and have a considerable difficulty intuiting the appropriate grain size for the student.

An effective computer tutor will have to adjust the grain size of instruction as the student progresses through the material. Using a theory of production learning it will have to predict when the original productions become compiled into macro productions so that it can change the grain size of instruction.

Principle 7: Enable the Student to Approach the Target Skill by Successive Approximation

Students do not become experts in geometry or LISP programming after solving their first problem. They gradually approximate the expert behavior, accumulating separately the various pieces (production rules) of the skill. It is important that a tutor support this learning by approximation. It is very hard to learn in a tutorial situation that requires that the whole solution be correct. The tutor must accept partially correct solutions and shape the student on those aspects of the solution that are weak.

Generally, it is better to have the early approximations occur in problem contexts that are as similar to the final problem context as

possible. Skills learned in one problem context will only partially transfer to a second context. Students learn features from early problems to guide their problem-solving operators. If these features are different from the final problem space the problem-solving operators will be misguided. For instance, early problems in geometry tend to involve algebraic manipulations of measures. Consequently, the student learns to convert segment and angle congruence into equality. Later problems, such as those involving triangle congruence, do not involve converting congruence of sides and angles into equality of measures.

The advantage of a private tutor is that he/she can help the student through problems which are too difficult for the student to solve entirely alone. Thus, it is common to see a sequence of problems where the tutor will solve most of the first problem with the student just filling in a few of the steps, less of the second, etc. until the student is solving the entire problem.

Principle 8: Promote Use of General Problem-Solving Rules Over Analogy

There are two basic methods that we have observed students using to solve the first problems in a domain. One is to use analogies to earlier problems in the text or problems from other domains to help guide the problem solving. The basic strategy is to try to map the structure of a solution of one problem to another problem. Anderson (1981 tech report), Anderson, Farrell, and Sauers (1984), and Anderson, Pirolli, and Farrell (in press) discuss specific examples from our protocols on geometry and LISP.

The other method is to extract general problem-solving operators from the instruction and apply these to the problem. For instance, if the goal is to prove triangles congruent, one can apply postulates about triangle congruence. If the goal is to create a list structure, one can try to apply a function that creates list structures. The problem with such general operators is that in many domains the search space of the combinations of these operators becomes enormous. This is perhaps why only a little additional information tends to be introduced with each new section of a textbook. The student can restrict search to these new potential operations (cf. Van Lehn, 1983).

Another difficulty with the general problem-solving approach is that it is often difficult to encode the needed problem-solving operators. Often the instruction does not contain explicit statements of such operators. Rather the operators have to inferred from the instruction. Even on those occasions in which the operators are directly stated, students have a hard time understanding them because they are stated so abstractly. Students are often only able to encode the operators correctly when they see them applied to an example problem.

Students appear to prefer analogy as a method of solution in both geometry and LISP. The preference is not overwhelming in geometry and there are many episodes of problem solution by general problem-solving operators. In contrast, the preference is overwhelming in novice LISP programming. In almost every case where a student was writing a first instance of a particular type of LISP function, the student relied on analogy to example LISP functions.

Private human tutors differ as to whether they tend to guide the student to solution by analogy or by general problem-solving operators. We claim that solution with general operators would lead to the best long-term gains. This is because the student often successfully generates a solution by analogy but does not understand why the solution works. We have seen students work their way through problems by analogy and not learn anything of permanent value. What they often learn is how to do analogies. If we take away the problems from which to analogize and they are unable to solve problems. Halasz and Moran (1983) have also commented on the negative consequences of problem solving by analogy. They point out that students are prone to incorrect inferences in using the analogy. An analogy is frequently used in place of a deep understanding of the problem domain.

Conclusions

We have stated a number of cognitive principles that seem important to designing intelligent tutors. Our specific geometry and LISP tutors (Boyle and Anderson, 1984; Farrell, Anderson, and Reiser, 1984) can be consulted for successful application of such rules. To the extent that such applications are successful, they are support not only for these cognitive principles of design, but also for the underlying ACT theory of cognition on which they are based.

References

Anderson, J.R. (1981). Tuning of search of the problem space for geometry proofs. In **Proceedings of IJCAI-81** (pp. 165-170).

Anderson, J.R. (1981). **Acquisition of Cognitive Skill.** ONR Technical Report 81-1, Carnegie-Mellon University, Pittsburgh, PA.

Anderson, J.R. (1982). Acquisition of proof skills in geometry. In J.G. Carbonell, R. Michalski & T. Mitchell (Ed.), **Machine Learning, An Artificial Intelligence Approach.**

Anderson, J.R. (1983). **The Architecture of Cognition.** Cambridge, MA: Harvard University Press.

Anderson, J.R., Farrell, R., & Sauers, R. (1982). **Learning to Plan in LISP.** ONR Technical Report ONR-82-2, Carnegie-Mellon University.

Anderson, J.R., Farrell, R., & Sauers, R. (1984). Learning to program in LISP. **Cognitive Science,** in press.

Anderson, J.R., Pirolli, P., & Farrell, R. Learning recursive programming. In forthcoming book edited by Chi, Farr, & Glaser.

Bilodeau, I. McD. (1969). Information feedback. In E.A. Bilodeau (Ed.), **Principles of Skill Acquisition.** New York: Academic Press.

Boyle, C.F., & Anderson, J.R. Acquisition and automated instruction of geometry proof skills. Paper to be presented at the Annual Meeting of the American Educational Research Association.

Brown, J.S., & Van Lehn, K. (1980). Repair theory: A generative theory of bugs in procedural skills. **Cognitive Science**, 4, 379-426.

Cohen, V.B. (April 20, 1982). Computer software found weak. **New York Times**, C4, Summary of a research.

Farrell, R., Anderson, J., & Reiser, B. Interactive Student Modeling in a Computer-Based LISP Tutor, 1984, submitted to **Cognitive Science.**

Halasz, F., & Moran, T.P. (March 15-17, 1982). **Analogy considered harmful.** Technical Report, Proceedings of the Human Factors in Computer Systems Conference, Gaithersburg, MD.

Lewis, M., & Anderson, J.R. The role of feedback in discriminating problem-solving operators.

McKendree, J., Reiser, B.J., & Anderson, J.R. Tutorial goals and strategies in the instruction of programming skills. Paper submitted to the 1984 conference of the Cognitive Science Society.

Newell, A., & Simon, H. (1972). **Human Problem Solving.** Englewood Cliffs, NJ: Prentice-Hall.

O'Shea, T. (January, 1979). A Self-Proving Quadratic Tutor. **International Journal of Man-Machine Studies**, 11(1), 97-124.

Ross, B.H. (1984). Remindings and their effects in learning a cognitive skill. **Cognitive Psychology**, in press.

Skinner, B.F. (1958). Teaching machines. **Science**, 128, 889-977.

Sleeman, D. (1982). Assessing aspects of competence in basic algebra. In D. Sleeman & J.S. Brown (Eds.), **Intelligent Tutoring Systems**, New York: Academic Press.

Sleeman, D., & Brown J.S. (Eds.). (1982). **Intelligent Tutoring Systems.** New York: Academic Press.

Tulving, E. (1983). **Elements of Episodic Memory.** London: Oxford University Press.

Tulving, E., & Ghomson, P.M. (1973). Encoding specificity and retrieval processes in episodic memory. **Psychological Review**, 80, 352-373.

Van Lehn, K. (1983). **Felicity conditions for human skill acquisition: Validating an AI-based theory.** Technical Report CIS-21, Xerox Parc, Palo Alto, CA.

INVITED ADDRESS: A FRAMEWORK FOR A QUALITATIVE PHYSICS

John Seely Brown, Xerox Parc