COGNITIVE PSYCHOLOGY AND INTELLIGENT TUTORING

John R. Anderson
Department of Psychology
Carnegie-Mellon University

This paper is an attempt to think through the relationship between intelligent tutoring and cognition. This is a particularly crucial issue to me because I have both worked on the ACT theory of cognitive psychology (Anderson, 1983) and on tutors for geometry and LISP (Boyle & Anderson, 1984; Farrell, Anderson & Reiser, 1984). The paper is in two parts. First, I will start from the goals of intelligent tutoring and reason to how cognitive psychology might serve these goals. Second, I will start from the goals of cognitive psychology and reason to how intelligent tutoring might serve these goals. I found the implications in both directions to be quite surprising.

Implications of Cognitive Psychology for Intelligent Tutoring

Instruction in general, and computer-based instruction in particular, is practiced pretty much as a black art. Students are exposed to various experiences in some vague belief that they will become more capable of dealing with certain vaguely conceived situations in later life. There is both failure to specify what the to-be-learned behavior is and how the experience will affect the learning. While people do learn from such poorly conceived instruction, there is every reason to believe that there is a lot of wasted motion. Clearly, cognitive psychology has a major potential contribution to make in adding precision to the art of education generally, and more specifically to intelligent tutoring. An interesting question concerns which aspects of cognitive psychology are relevant to achieving intelligent tutoring. It turns out that there is a rather interesting demarcation between those aspects of cognitive psychology which are relevant and those aspects which are not.

The first contribution of cognitive psychology would be to provide a well-specified model of the target behavior to be tutored--the goal to which the instruction is directed. In the areas we have thought of tutoring, mathematics and science, this amounts to developing a problem-solving model of the ideal student. Such a model involves a specification of the problem-solving goals, the representation of the relevant knowledge, and the operators that control the transition among goals. Thus the basic ingredients are goal structures, representation, and control. Such student models can be used to represent both the state of the current and the state desired for the student at the end of the instruction.

Creating such models is no mean feat, but given that we had such models, what would we do with them? This requires a theory of the acquisition of problem-solving skills that will specify what the consequences of various experiences will be on the state of the student model. So, we can add a theory of skill acquisition to the list of potential contributions of cognitive psychology to intelligent tutoring.

Exactly how the student model is used in an intelligent tutoring system depends on the learning theory, but our ACT learning theory leads us to

build the tutoring system very directly around the model of the ideal student. Our tutors guide the students through the problems trying to make their steps correspond to those of the ideal student model. At each next step the student's step is compared with the range of acceptable steps and, if it does not correspond, immediate explanation is generated which tells the student what the correct step is and why it is correct. This is a very powerful role for a student model in the tutoring. It effectively structures the whole tutoring interaction. I refer to this mode of tutorial interaction as model-tracing.

Another important consequence of the conjunction of a learning theory and a student model is a prescription for problem sequence. One can look at weaknesses in the student model and construct problems which the learning theory predicts will give the best opportunity to repair the weaknesses.

Another important consequence of the conjunction of a learning theory and a student model is a prescription for problem sequence. One can look at weaknesses in the student model and construct problems which the learning theory predicts will give the best opportunity to repair the weaknesses.

All forms of instruction require that one correctly interpret the student's behavior and successfully impart information to the student. In our model-tracing paradigm this requires that we understand why the student makes moves and that we can correctly describe goals to the student, the operators to apply at these goals, and why these operators should apply. These communication needs place further demands on cognitive psychology:

To the extent that the interaction involves natural language, this brings up issues of natural language processing. These issues of communication turn out to involve issues of knowledge representation. For instance, a major problem turns out to be that of the student acquiring the right representation of domain concepts. In geometry students have to learn the meaning of terms like premise and consequence. In LISP they need to understand the meaning of terms like tail of a list and evaluation of an expression. Part of the tutoring goal becomes teaching of domain concepts. Also one needs to be able to refer to the student's internal goal structures. This requires designing an efficient way of representing the goal structure to the student. For example, in geometry we teach students a graph representation for forward search from the givens and backward search from the to-be-proven statement.

Another issue in communication concerns the serious problems of working memory overload. One has to have an accurate estimate of how much information the student can hold at any one time. It is very easy for instruction to fail because the student cannot process it all.

Requirements from Cognitive Psychology

It is interesting to consider those things that intelligent tutoring needs that cognitive psychology does not offer, those things which it needs that cognitive psychology offers, and those things which it does not need that cognitive psychology offers. It should be recognized that many things go into successful tutoring that have nothing to do with cognitive psychology. This tends to involve the computational aspects of the medium.

In our own work this has included principles of graphics, design of highly modularized systems, efficient implementation of production systems, principles for automatic problem generation, and principles for inducing student models from surface behavior. Each of these is a major concern in our work but a concern that is totally devoid of any psychological content.

Then there are a set of psychological issues that seem key. To review, these were:

1. Theory of goal structures,
2. Theory of control,
3. Theory of knowledge representation,
4. Theory of skill acquisition,
5. Theory of natural language understanding, and,
6. Theory of working memory limitation.

What is most interesting, however, is the large segments of cognitive psychology that seem irrelevant. Specifically, there is no role for a theory of the speed or probability with which knowledge is stored, retrieved, or applied. I cannot think of one timing result from experimental psychology that has implications for tutor design. We would have our tutors engage in the same behavior independent of how long the students studied that information, how long it took them to retrieve it, and whether they could remember it. No matter what, we would test for the knowledge at a later point and provide remedial instruction if the students are wrong. Issues about the exact times and probabilities of performance are irrelevant. We could build into our tutor the ability to make accurate predictions about times and probabilities, but it could not use these predictions to further its tutoring. These are predictions at the wrong level of analysis.

It is useful to make a distinction between two levels of cognitive theory--the level of process specification and the level of process implementation. This can be done in analogy to the distinction between programming language and machine implementation. Although there is some controversy about the matter (e.g., Anderson & Hinton, 1981), it seems that there is a programming language of the mind (compare to Fodor, 1975) in which cognitive processes are specified. Such a language is at a level analogous to the LISP programming language. In our work, we take the ACT production system as this language. The issues listed as relevant to tutoring--goal structure, control, knowledge representation, skill acquisition, and working memory--are all concerned with aspects of this language. The one other relevant issue, natural language understanding, is concerned with a process implemented <u>in</u> this mental language.

Much of a cognitive theory like ACT is concerned with how this mental language is implemented. This is like asking how LISP is implemented on a computer. It is these aspects of implementation which seem not to be relevant to tutoring. This is not to say, of course, that they are uninteresting.

So, the conclusion is that the division between issues of mental language and issues of its implementation corresponds to the division between those aspects of cognitive psychology which are relevant to

intelligent tutoring and those aspects which are not. To the extent that intelligent tutoring work can progress without consideration of the implementation, this is evidence for the distinction between the mental programming language and its implementation.

Implications of Intelligent Tutoring for Cognitive Psychology

It is natural to speculate that the relationship might be symmetrical and that work in intelligent tutoring might have implications only for the mental language level and not for the implementation level. However, this is not so. By looking at the history of latency and success in working with a tutor one may be able to make numerous inferences about cognitive implementation. However, it is hardly the only way to make inferences about cognitive implementation and for many issues, traditional experiments are more effective. On the other hand, I am prepared to argue that the only methodologies that can get at the language level are the tutoring methodology and variants on that methodology.

To make this argument, I will consider an analogy: Suppose a programmer presented to us a fairly complex program, and we wanted to test whether it was a LISP program. In the analogy, LISP corresponds to our theory about mental programming language and the programmer is the learning system that changes the mental program. To make the analogy work we will have to assume we cannot ask the programmer what the language is nor can we physically inspect the program. All we can do is look at the input-output behavior of the program. Now it is well known that any computationally universal programming language can produce any input-output behavior. Similarly, any "reasonable" theory of the mental language could produce any behavior. It is unlikely that implementations in two languages will differ interestingly in their timing behavior, particularly if all we can look at is relative time to solve two problems, not absolute time. Similarly, it is unlikely that we can choose between two theories of mental language by looking at processing times or probabilities of correct responses.

How in fact are decisions made about the mental language? The argument is usually that a particular type of input-output behavior is characteristic of the "programming style" appropriate to a particular mental language. A classic example of this kind of reasoning is the paper by Hayes-Roth & Hayes-Roth (1979) arguing for opportunistic planning. Similarly, one might argue that a program that printed out

```
=>(factorial 4)
1<Enter> factorial (4)
|2<Enter> factorial (3)
| 3<Enter> factorial (2)
| |4<Enter> factorial (1)
| | 5<Enter> factorial (0)
| | 5<EXIT> factorial 1
| |4<EXIT> factorial 1
| 3<EXIT> factorial 2
|2<EXIT> factorial 6
1<EXIT> factorial 24
24
```

was written is LISP. In certain contexts, this might be a reasonable inference. However, in the human case the inference from surface behavior to mental programming language is quite perilous.

Probably a better method would be to ask the programmer to make some addition or modification to the program and see how long it took to accomplish that and with how much difficulty. Certain things are easy to implement in LISP and certain things are not. Analogously, in the case of the mental programming language it is very informative to look at what the effects are of various instructional manipulations. Certain instructional goals should be easy to achieve and certain should be hard.

Thus, the way to test hypotheses about the mental programming language is to perform instructional experiments. Of course, these need not take the form of constructing intelligent tutors, but there are advantages to that specific methodology. First, computer implementation operationalizes very precisely the instructional methodology. Second, interesting instructional manipulations take relatively long times. It is hard to motivate a subject population in such experiments unless one is trying to teach a useful domain. There are serious problems with trying to teach a useful domain in some perverse way to test cognitive theory. It might be informative to study the instruction of geometry without diagrams, for instance, but I could hardly use the scientific information gathered to mollify angry parents whose children were failing high school math. So for these reasons, one is naturally led to conduct instructional tests of cognitive theory as good faith efforts to have computers to teach real topics to real students. Beyond this there is something very informative about finding the optimal tutor for a topic--whether that optimum is just a very local one or a global one. The fact that the tutor is at an optimum places important constraints on the nature of the mental programming language.

Example Tutorial Experiments

It is not hard to think of tutorial manipulations that serve to test alternative theories of the mental system. Consider for instance the contrast between schema systems and production systems. The essence of a schema system is the notion that the critical knowledge structure is an abstract schema which is used for different purposes according to the current goal. Thus, there might be a recursion schema which could be used for coding, debugging, and evaluating. According to schema theory, it is necessary and sufficient to build up a rich schema representation of recursion. From that suitable performance would flow in different tasks. In contrast, production systems theory leads to the conclusion that there are task specific rules that must be acquired. Obviously, such theories lead to very different predictions about the relative merits of general versus task-specific instruction.

To consider a controversy more local to CMU, consider two types of productions system architectures. One (e.g., ACT) holds that information first comes into the system in a declarative form and then is complied into production form with practice. The other holds that all information is procedural, and there is no separate declarative representation. The first viewpoint would hold that there is a point to understanding and memorizing generally relevant facts in advance to actually using them. This would

establish the declarative encoding and permit the procedural compilation right away. The second viewpoint would hold that students might as well be told the specific rules as they are needed in problem solving without advance preparation.s

As a third and final contrast, consider the claim that goal selection is opportunistic and data-driven versus the claim that goal selection is in response to some hierarchical plan. The former would argue for a tutorial strategy that gave the student a lot of flexibility in what to do next whereas the second would promote a rather rigid flow of control through a problem.

Each of the above three contrasts are somewhat caricatures, and people may want to argue for other predictions. Rigorous predictions require careful interfacing of specific proposals with specific learning situations. However, the examples do serve to indicate how tutorial manipulation can go to the heart of fundamental issues about the mental programming language. (Also, it needs to be acknowledged that because these tutors require effective computational principles as well as correct cognitive assumptions, a particular tutor can fail even though the underlying theory is correct.)

Conclusion

So in conclusion, there seems to be a very intimate relationship between a subset of issues in cognitive psychology and intelligent tutoring. It goes beyond the relationship between a science and its applications. Research in intelligent tutoring is the natural methodology for study of the mental programming language. The relationship is this strong because the human mind is more than a naturally occurring object for scientific study; it is an object whose evolution was principly directed so it could be instructed in new problem-solving skills.

References

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

Anderson, J.A. & Hinton, G.E. (1981). Models of information processing in the brain. In G.E. Hinton & J.A. Anderson (Eds.), Parallel Models of Associative Memory. Hillsdale, NJ: Erlbaum.

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.

Farrell, R., Anderson, J. & Reiser, B. (1984). Interactive Student Modelling in a Computer-Based LISP Tutor. Submitted to Cognitive Science.

Fodor, J.A. (1975). The Language of Thought. New York: Thomas Y. Crowell.

Hayes-Roth, B. & Hayes-Roth, F. (1979). Modelling planning as an incremental, opportunistic process. Proceedings of IJCAI-709, 375-383.