Behavior Research Methods, Instruments, & Computers
1992, 24 (2), 205-212

Using tutoring systems to study learning: An application of HyperCard

ADRIENNE Y. LEE
University of Colorado, Boulder, Colorado

HyperCard was used to develop a simplified tutoring system whose principles were based on a learning theory, and a genetics tutoring system was evaluated experimentally. Learning was studied by examining immediate versus delayed feedback after an error was made. Such tutoring systems aid in psychological studies of learning, because experimental variables can be easily manipulated. HyperCard provides a good vehicle for tutoring system development, since it requires no extensive programming skills.

The application of learning theories in the creation of teaching machines can be traced as far back as Skinner's work in the 1950s (see Skinner, 1958). This work, along with other researchers' early use of computers for teaching, gave rise to the term *computer-aided instruction* (CAI). Given the technology of the time, most CAI systems could not present difficult problems to students or maintain a record of responses over many problems. Even today, most of the educational software that is commercially available has been low in quality and has not been based on any kind of learning theory (Anderson, Boyle, & Reiser, 1985a).

With the availability of inexpensive personal computers for use in the schools, research on the use of computers in teaching has earned more interest in psychology and computer science (Burns, Parlett, & Redfield, 1991; Polson & Richardson, 1988; Psotka, Massey, & Mutter, 1988; Wenger, 1987). Computers have become powerful enough for difficult problems to be presented. Furthermore, computer technology has allowed for simulations of the knowledge needed to solve problems in a domain and the addition of system responses to an individual student's particular needs (Anderson et al., 1985a). These additions have created the terminology of "intelligent" computer-aided instruction (ICAI) or intelligent tutoring systems (ITS). These systems incorporate the knowledge of human tutors and experts in the domain being taught. In this paper, I will refer to ICAI and ITS systems as tutoring systems.

I gratefully acknowledge the assistance of Peter Foltz and Nancy Pennington for their invaluable aid in the preparation of this paper and advice on the psychological aspects of the project. I also wish to thank Michele Lee and Jane Bock for assistance on the biological aspects of the tutoring system. In addition, I wish to thank Peter Polson, Clayton Lewis, Jan Gehrig, Philip Thompson, Paula Messamer, Keith Hannon, Jean McKendree, and Matthew Lewis for their help, and the anonymous reviewers for comments on a draft of this article. Requests for reprints should be sent to Adrienne Lee, Department of Psychology, University of Colorado, Boulder, CO 80309.

Although tutoring systems have often been focused toward the effective use of computers in the teaching of domain topics, researchers can also use tutoring systems to study learning. Early learning theories were based on simple tasks, such as list learning, but recent studies reflect the opinion that learning theories should be examined by means of more difficult tasks, such as learning how to use oscilloscopes or learning a full course in some classroom material (Anderson, Boyle, & Yost, 1985b; Brown, Collins, & Duguid, 1988; Lee, Polson, & Bailey, 1989). Since today's tutoring systems are able to trace series of students' actions and to adapt the level of difficulty of instruction, they provide the perfect tool for studying learning theories with more difficult tasks. Thus, tutoring systems do not have to be focused solely on presenting problems. They can be used to test learning theories, and in addition, models of learning can inform the creation of a tutoring system (Anderson, 1982; Anderson, Boyle, Corbett, & Lewis, 1986). Anderson (1987) makes the point that tutoring systems can provide the vehicle for examining learning while the student is actually learning some academic topic. The tutoring system can also provide an experimenter with the ability to test experimental manipulations in a realistic context.

Clearly, tutoring systems can be created on the basis of learning theories, although going from the basically descriptive nature of cognitive theories to a more prescriptive theory can be difficult (Singley, in press). For some researchers, the development of a tutoring system may be difficult for very fundamental reasons. For example, the researcher may not have had extensive computer training and may not be able to program the system. In such cases, the researcher may choose to avoid tutoring system development in favor of an easier mode of presentation, such as booklets of materials or simple computer programs already developed by others. In this paper, I describe an alternative with which one can utilize the prescriptive ideas presented by Anderson and colleagues, without having to know complex computer graphic or programming techniques.

ANDERSON'S PRINCIPLES FOR TUTORING SYSTEMS

The work by Anderson and colleagues presents the most comprehensive application of a learning theory to the development of tutoring systems. Their set of principles for the development of such systems is shown in Table 1 (Anderson, Boyle, Farrell, & Reiser, 1984a).

By following these principles, Anderson and his colleagues have developed several different types of systems: a LISP tutor (Anderson, Farrell, & Sauers, 1984b; Farrell, Anderson, & Reiser, 1985; Reiser, Anderson, & Farrell, 1985), a geometry tutor (Anderson et al., 1985b), and an algebra tutor (Lewis, Milson, & Anderson, 1987). In these instances, students were examined as they solved problems in the various domains (Anderson et al., 1984a); the information gained was then used to create hierarchical representations of goals and production system models. The implementation is generally accomplished with the use of LISP machines and graphical representations for the tasks. For these systems, Anderson et al. (1984a) estimated that approximately 10 man-years would be required for one to create a complete course for geometry or LISP. Subsequent work has shown that these systems can be quite effective in instructing students in the domain of interest (Lewis et al., 1987; McKendree,1986; Singley, 1987, in press).

Although many of the principles have not been tested directly to see what modifications, if any, need to be made to them, several of Anderson's students have attempted to test pieces of the theory (Singley, 1987, in press). Such studies enable one not only to extend and test the principles, but also to evaluate the theory itself.

Even though many of these principles appear at first glance to represent the prescriptions of common sense, other developers have challenged specific parts of the principles. For example, several researchers have attached the idea of immediate feedback after an error (see, on ALGEBRALAND, Foss, 1987a, 1987b; on SOPHIE, Brown, Burton, & de Kleer, 1982; Burton & Brown, 1979). In spite of this and other challenges, these principles represent a starting point for the development of a tutoring system, if a researcher is concerned about basing the system on some learning theory.

Table 1
Anderson's Principles for Tutoring Systems

1. Identify the goal structure of the problem space.
2. Provide instruction in the problem-solving context.
3. Provide immediate feedback on errors.
4. Minimize working memory load.
5. Represent the student as a production set.
6. Adjust the grain size of instruction according to learning principles.
7. Enable the student to approach the target skill by successive approximation.
8. Promote [the] use of general problem-solving rules over analogy.

USING HYPERCARD TO DEVELOP A TUTORING SYSTEM

Developing a tutoring system can be daunting for psychologists who are not experienced programmers. First, the developer must consider the programming of the tutoring system. Second, the programmer must consider creating a user interface that is easy to learn and use. The Macintosh, with its HyperCard application, provides simple answers to both issues.

HyperCard provides the tools for such experimenters to be able to develop tutoring systems. It is easy to use and its programming language, HyperTalk, is easy to learn. In addition, the Macintosh interface is easy to master, and therefore, a student can begin to use tutoring systems created in HyperCard with minimal instruction.

Applying HyperCard

HyperCard is a hypermedia toolkit available for the Apple Macintosh (Goodman, 1987, 1990). Developers apply their ideas to stacks of cards in which each card can represent the presentation of some ideas. Thus, HyperCard developers can easily create simple, static presentations of textbook material by entering the material directly.

Multiple choice questions can be created by placing the text of a question on a card and then using buttons for the choices. When a user selects a button, the program can evaluate whether the response is correct. If the answer is incorrect, the program can jump to an appropriate feedback card. In this way, HyperCard can be used to develop a CAI system.

More interesting, however, is the creation of intelligent CAI systems. For example, can the tutoring system principles presented in Table 1 be applied with HyperCard?

Production System Models and Tutoring Systems

Of primary importance to Anderson's learning theory is the development of a production system model of the domain. A production system is a way to model and to understand human knowledge (Neches, Langley, & Klahr, 1987). It describes the cognitive architecture of the human information-processing system, and it can be implemented in a variety of special programming languages (such as OPS5). Production systems explicitly describe the goal structure of the problem-solving domain. In addition, they can be used to simulate the behavior of the student when the student is given various options. Such a production system can be considered as the student model, because it is based on the student's actions.

In order to create a production system, a task analysis must be performed. This analysis can provide insights into human problem-solving behavior, because the student will respond to the elements in the task that he or she is asked to perform (Card, Moran, & Newell, 1983). The task

analysis can then provide information about likely goals of the student in solving the problem, the operators that can be used to affect the problem state, the methods or procedures for accomplishing the goals, and the selection rule that will aid the student when more than one method can be used to reach a given goal. This model can be translated into a production system and used to simulate behavior. Although the production system can be created to perform error-free behavior (or the ideal student behavior), it can also be expanded to include incorrect or buggy behavior. The important issue is whether or not a trace of the current student's behavior can be made in order to be compared with the ideal student's behavior.

The first step is to produce a list of goals, ranging from the most general goal to the most specific of subgoals (Kieras, 1987). Then one should go back through the listing of goals and attach methods for accomplishing those goals. The methods usually contain the subgoals of the higher level goals. The easiest way to formulate these methods is to simply list what would need to be done to accomplish the current goal. After one has listed the methods, one should check for accuracy. Selection rules, or places where more than one method could apply, should also be noted as one is describing the methods.

Figure 1 shows an example of how to write a model that can be later translated either directly into HyperTalk or into a production system that can be run to test different problems as input. In Figure 1A, the overall goal is set out. Then the subgoal is written (Figure 1B). A selection rule is provided to illustrate how a selection rule might look.

To meet most of Anderson's prescriptions, a production system, based on the domain of interest, should be created; it should be either integrated with or independent of the tutoring system. In fact, since HyperTalk permits IF-THEN statements, an experimenter can make a direct translation between a working production system and HyperTalk. The knowledge gained by the experimenter in producing the production systems can be used to generate instruction and feedback. In addition, by examining the production system, the experimenter can determine the appropriate division of the knowledge to be learned. Using these small incremental steps, students can learn by successive approximation.

Implementing Other Andersonian Principles

The main assumption is that students will learn some material and then be able to solve some problems by using the tutoring system. Instruction is necessarily provided during problem solving, since feedback is given when a mistake is made. Since HyperCard can capture a student's actions, feedback can be provided immediately after an error. Another principle, minimizing working memory load, is a prescription to display as much of the previous student actions as possible. Although the Macintosh Plus and Macintosh SE do not provide much screen space, there is enough room to present a problem, the student's responses to parts of the problem, and the student's scratch work.

Adding Extras to the HyperCard Tutoring System

Although the tutoring system development principles do not explicitly state the necessity of a help system, all of Anderson's tutoring systems provide one. A help system can be easily provided, either by placing a help system in each stack or by having a separate stack for the help system. In addition to a help system, HyperCard and the Macintosh interface allow for graphics to be created and incorporated into the program easily. For example, one can scan images into the system, touch them up with a paint program, and then paste them into HyperCard (in the same manner as text can be cut and pasted with a word processor).

Summary

HyperCard provides ease of development based on: (1) graphically oriented programming, (2) a programming language, HyperTalk, that is simple to learn, and

a) IF the goal is to solve the problem THEN
Step 1. Set the goal of identifying the type of inheritance.
Step 2. Set the goal of identifying the genotypes of the parents.
Step 3. Set the goal of identifying the genotypes of the offspring.

b) IF the goal is to identify the type of inheritance AND
the type is not determined THEN
Step 1. Accomplish the goal of running through the rules for determining inheritance.
Step 2. Set the goal of verifying that the correct type is chosen.

c) Selection rule set for the goal of running through the rules and determining inheritance.
IF there is no pattern in the pedigree, THEN
set the type to autosomal.
...etc....
Report goal accomplished.

Figure 1. An example of rules from a model of genetics problem solving.

(3) HyperTalk's IF-THEN statements, which allow the direct translation of production systems models of behavior.

APPLYING IDEAS TO DEVELOP A GENETICS TUTORING SYSTEM

A simplified tutoring system for genetics was developed with HyperCard, and an evaluative study was performed on that system. The system is an example of how the ideas presented in the previous section can be applied to a specific topic. The evaluative study examined Anderson's prescription that feedback must occur immediately after an error is made (Lee, 1989). It provides an example of how learning issues can be investigated with the use of tutoring systems.

The genetics system will be described in the following sections. However, since the paper was designed to demonstrate the ease of application of tutoring system ideas in HyperCard, it will only be discussed briefly.

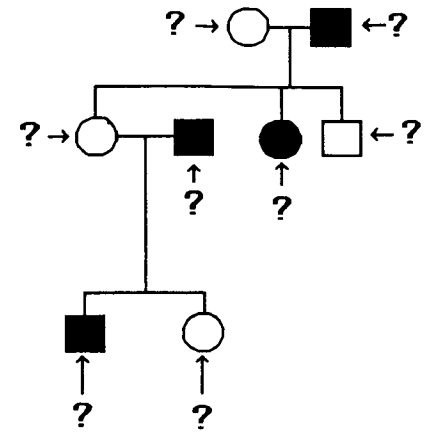

Figure 2. An example question from the tutoring system: Question 4 of Problem Set 4. Squares represent males, circles represent females, affected individuals are colored, and unaffected individuals are not colored.

Description of Genetics System

A single specific area of genetic problems, that of pedigree problems, was chosen (see Figure 2). Pedigrees are diagrams showing relations among individuals (either by blood or marriage) and the inheritance of a trait or traits by those individuals. A trait could be hair color or a disease. Figure 2 shows unaffected males and females (squares/circles not filled in) and affected males and females (squares/circles filled in). For example, a man with colitis will be represented by a filled-in square. A pedigree problem will show a pedigree without the type of inheritance indicated next to each individual. Question marks indicate where the subject was asked to identify the genotype of the individual. (The genotype is a two-letter notation, in which each gene is represented by a single letter. A pair of genes work together to represent a trait, such as pea pod color. For example, a yellow pea pod, where yellow is recessive, would be represented as "yy".)

The tutoring system consists of panels, which present basic material from introductory genetics books. These materials include basic information, information on how to solve pedigree problems, and examples. An example showing the inclusion of graphics is given in Figure 3.

After certain sections, a series of problems were presented. An example card from the problem set is shown in Figure 4. Individual production system models were created for each problem. In addition, an ideal student model and bug catalog were also created. All problems for all problem sets were tested in a pilot study to determine whether the sequences for the production system models were correct. The production system models were used to program the buttons and provide information for the feedback screens. Some additional programming was needed to provide the proper interaction between what a production system would produce and what was desired from HyperCard. For example, the production system may indicate that an error a student makes should be corrected by giving the student specific feedback. The additional programming would send the student to the feedback screen.

When subjects were asked to solve problems, they chose their answers from the sets given to them. In the original tutoring system, subjects received a box with a limited number of choices. In a later version of the tutoring system, subjects selected their answer from all possible choices. (Compare Figures 4 and 5.)

An aid to solving the genetics problems is called a Punnett Square (Figure 6). This functions as a scratch area where subjects can calculate what the possible genotypes of offspring can be. Notice that the Punnett Square acts as a multiplication square. Since the problems the subjects were solving did not involve more than one type of disease/trait, a small scratch area or Punnett Square was provided (see Figures 4 and 5).

A help system was also provided. The help button appears at the bottom of each problem screen (see Figures 4 and 5); by selecting it, subjects can access the help information screen, which in turn allows them to select what type of help they need. Thus, for example, subjects can find out the definitions for major genetics terms. Furthermore, depending on what part of the problem the student is working on at the time of request, the help system is also able to provide the next step that the subject needs to take. However, the help system never provides the answer to any part of a problem.

Since a tutoring system can act as both a tutor and an experimental presentation vehicle, the tutoring system was programmed to record subject number, date, and time spent on each panel. Responses (including whether or not the subject answered the question correctly) were re-

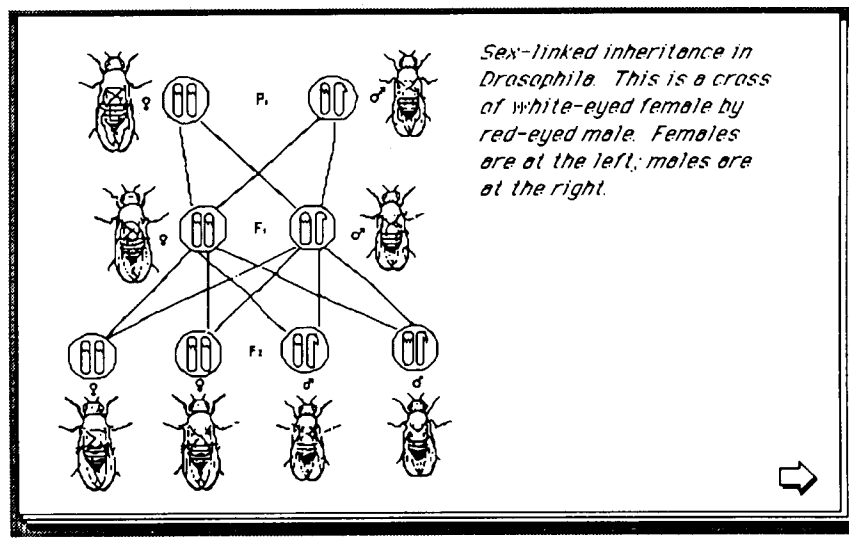

Figure 3. An example card showing the use of graphics to illustrate a principle.

corded. All answers to questions, including responses to problem aids (Punnett Squares) were recorded.

Evaluation and Experimental Results

This genetics system was used to evaluate Anderson's third prescription that immediate feedback should be provided after an error is made. Two systems were therefore developed: one that provided immediate feedback and one that provided delayed feedback. For the immediate condition, feedback was given when the subject made an error; for the delayed condition, feedback was given at the end of the problem. Since the focus of this paper is to demonstrate the usefulness of HyperCard in the development of tutoring systems, the description of the evaluative study will be brief. A more complete description may be found in Lee (1989).

The experiment took place over 2 days. On the 1st day, subjects used the tutoring system, and on the 2nd day, subjects took a written test on the material learned and completed a questionnaire. While the subjects used the tutoring system, the computer recorded all their actions and times to complete tasks. The written test consisted of a multiple choice quiz, pedigree problems, a questionnaire on the tutoring system, and a questionnaire on the subjects' backgrounds. Some problems were taken directly from the tutoring system; other problems were more difficult, with the most difficult found in tests taken from tests given to a genetics class at our university. The test was scored for correctness.

The immediate condition subjects learned more quickly than did the delayed condition subjects [$F(1,19) = 31.95$, $p < .01$]. However, on a posttest, although both the im-

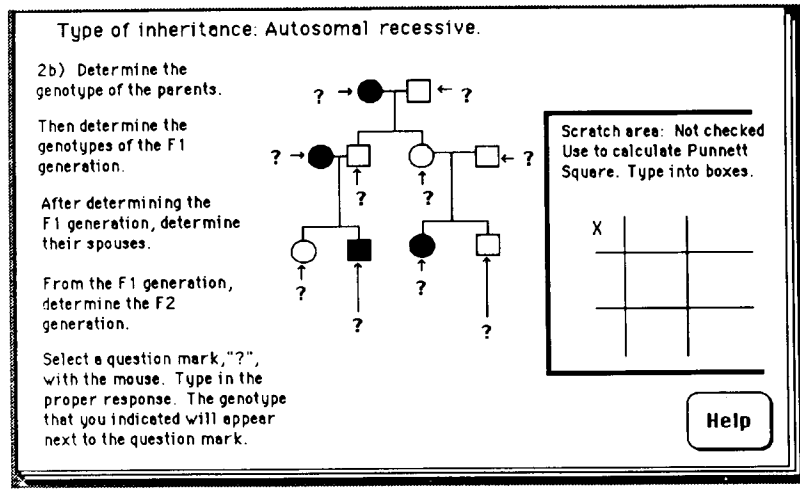

Figure 4. Problem-solving panel from original tutoring system.

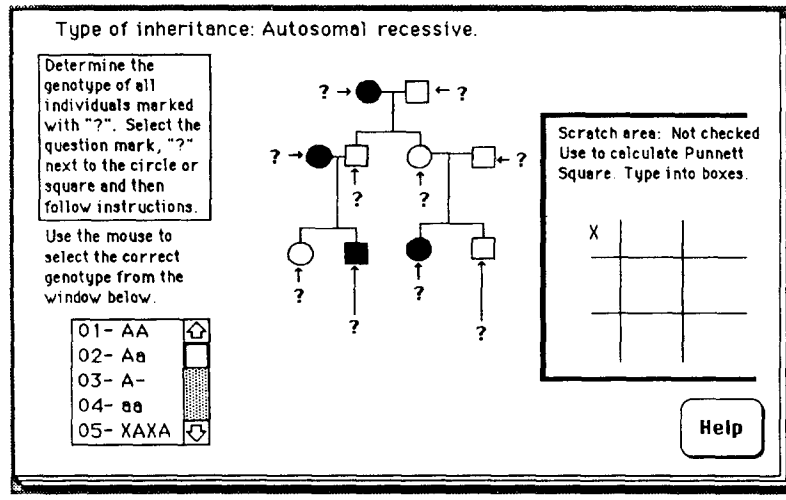

Figure 5. Problem-solving panel from tutoring system developed to replicate original experiment.

mediate and the delayed condition subjects performed equally well on problems that were similar to problems seen in learning, the delayed condition subjects performed better on a far transfer task [$F(1,19) = 9.13, p < .01$]. This experiment is currently being replicated with additional conditions to test hypotheses about the reasons for the performance of the delayed condition subjects.

The subjects were also given a questionnaire about the tutoring system. Generally, the students found the tutoring system useful for solving problems on the posttest. The students would have liked to have had a break in the tutorial, in addition to color, more sound, more graphics, and more examples. These suggestions are easy to implement, and in the continuation studies currently being performed, the only option not adopted was the addition of color and sound. However, color can be added by using HyperCard on a Macintosh II.

DISCUSSION

This experiment was designed to illustrate how one could develop a tutoring system based on a learning theory and then test the learning theory through testing the tutoring system. The results from this experiment can shed light on the prescriptions for tutoring system development and on the learning theory itself.

The learning theory hypothesizes that people create productions automatically from their representations of examples when they are asked to solve problems. An error would indicate that an incorrect production had been created and that correction would be needed in order to remove that incorrect production. Thus, the theory would indicate that immediate feedback was preferable, but the theory itself does not indicate exactly how much time is needed before a production becomes permanent.

Immediate feedback does not allow for self-correction, which is preferable to correction by intervention of an instructor or a computer (Anderson et. al., 1984a). In fact, delayed condition subjects changed many incorrect responses in this experiment. These results are consistent with those of Lewis and Anderson (1985), whose subjects in delayed conditions were better able to identify incorrect paths. Self-correction is not inconsistent with the idea of immediate feedback. Instead, the term *immediate* should probably be clarified. If the problem is small, like these problems or single LISP expressions (Anderson et al., 1984a), then *immediate* could mean "at the end of the problem." Thus, this discussion illustrates how an evaluative study can feed information back to both the theory and the prescriptions for tutoring systems derived from that theory.

	Y	y
Y	YY	Yy
y	Yy	yy

Figure 6. Example of a Punnett Square.

Considering Another Learning Theory to Address This Issue

Although the results from this experiment can be explained by correcting the prescriptions derived from Anderson's theory, we can compare the prescriptions derived from another learning theory in order to obtain insight into the results from this experiment. For example, let

us consider VanLehn's (1988) impasse-driven theory of learning. In a test situation, a student might reach a problem spot, and in the absence of help, the student would generate a solution to get past that spot, a repair, which might or might not be correct. After an error has occurred, the incorrect production rule has already been formed when the impasse is reached and the action taken. One idea that follows from impasses and repairs is that the teacher wants to catch an impasse before a student will make the incorrect repair. This would translate into two possible prescriptions: (1) to provide a means for students to find out a detailed (rather than general) next step for a problem before an error can be made, or (2) to provide feedback when an error is made that includes all steps from the beginning of the problem to the point where the impasse has occurred.

This example illustrates that the issue of immediate versus delayed feedback may not be important when one is deriving prescriptions for tutoring systems in another learning theory, such as VanLehn's (1988) impasse-driven learning theory. It also indicates the simplicity of going from a learning theory to prescriptions for a tutoring system, given that learning theory specifies its conditions explicitly. In addition, the examination of a single learning theory can lead to an interest in comparing it with other learning theories.

Comparison With Other Tutoring Systems

In addition to comparing the learning theory examined with other learning theories, the tutoring system can be compared with other tutoring systems. This tutoring system does provide a way to include all of the basics found in other systems, such as a student module, an expert module, and an instructional module. It also modeled students' behavior as a production set. In addition, HyperCard allows an experimenter to develop an interface that is easy to use, can visually display the goal structure of the problem, and can ease the student's work load by providing a display of the problem on the screen. As far as work load is concerned, HyperCard, in comparison with other systems, allows for ease in programming (Anderson & Skwarecki, 1986). Finally, a HELP system was developed by creating both a separate HELP stack for each problem set and direct access on each problem screen.

There are several differences between the simplified tutoring system described in this paper and other tutoring systems. Anderson's tutoring systems are developed to tutor for complete courses; the present tutoring system is focused on one section of a genetics course. In addition, Anderson has created full production system models for complete courses on particular topics (LISP, algebra, geometry). For complete use of his prescriptions, it would be necessary to build a similar production system for the topic to be tutored; however, to test a learning principle, it may be sufficient to build production system models only for the problems to be used. Such an approach has been used to study novice learning of oscilloscopes (Lee et al., 1989). This simplifies the task considerably and allows one to start the creation of the tutoring system with HyperCard more quickly. Finally, a running production system in the background is not easily feasible and was not used for the genetics tutoring systems. However, students' actions can be followed and recorded. These actions can be compared to an ideal student model and a bug catalog.

In comparison with other tutoring systems in general, this genetics tutoring system provides neither dynamic creation of problems nor natural language dialogue between the system and the student, and the system cannot learn (Fischetti & Gisolfi, 1990; Kimball, 1982; O'Shea, 1982).

When these considerations are taken into account, it can be seen that HyperCard will not create a system that is as complete as one of Anderson's tutoring systems or the other systems in the literature. On the other hand, one can easily create a tutoring system that does test the learning principles involved.

CONCLUSIONS

Learning can be studied in many ways. One of the most interesting ways is to use a tutoring system. Subjects can learn material that is taught in schools and universities rather than nonsense materials, which they may not encounter in real life. Since tutoring systems record the process of learning as subjects are solving problems, experimenters can learn about learning.

In this paper, I have described principles for tutoring system development based on Anderson's learning theory, and I have shown how those principles can be applied in the development of a simplified tutoring system with HyperCard. Two such systems were developed, in which one of Anderson's tutoring system principles, immediate versus delayed feedback after error, was tested directly. The empirical evaluation provided information about how learning can be affected by feedback timing; it also provided direct information about the advantages of using HyperCard for the Macintosh.

Using a Macintosh allows a student to learn easily how to use the computer without instruction and to concentrate on the learning of the material presented. Using HyperCard provides an ease of entering a variety of material (text, graphics, sound), ease of collecting user data, ease of programming for the novice programmer, and ease of modifying existing HyperCard tutoring systems. In addition, since Anderson's learning theory and tutoring system concepts are based on production systems, one can directly program them in HyperTalk, which accepts an IF-THEN format. In this way, one can utilize psychologically based principles in a simplified tutoring system.

The key point of this work is that experimenters should not avoid tutoring system research because they lack programming skills. Instead, a simplified tutoring system based on the principles of a learning theory can be developed. HyperCard provides a way for many researchers to explore learning through the use of tutoring systems in their research. In conclusion, tutoring system research

can be rewarding, because it provides an experimenter with a method to explore ways of teaching and instruction and, in the process, provides information about the learning process itself.

REFERENCES

ANDERSON, J. R. (1982). Acquisition of cognitive skill. Psychological Review, **89**, 369-406.

ANDERSON, J. R. (1987). Methodologies for studying human knowledge. *Behavioral & Brain Sciences*, **10**,467-505.

ANDERSON, J. R., BOYLE, C. F., CORBETT, A., & LEWIS, M. (1986). *Cognitive modelling and intelligent tutoring* (Tech. Rep. No. ONR-86-91). Pittsburgh, PA: Carnegie-Mellon University, Department of Psychology.

ANDERSON, J. R., BOYLE, C. F., FARRELL, R., & REISER, B. (1984a). *Cognitive principles in the design of computer tutors. In Sixth Annual Conference of the Cognitive Science Program* (pp. 2-16). Boulder, CO: University of Colorado, Institute of Cognitive Science.

ANDERSON, J. R., BOYLE, C. F., & REISER, B. (1985a). Intelligent tutoring systems. *Science*, **228**, 456-462.

ANDERSON, J. R., BOYLE, C. F., & YOST, G. (1985b). The geometry tutor. In *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 1-7). Los Altos, CA: Morgan Kaufmann.

ANDERSON, J. R., FARRELL, R., & SAUERS, R. (1984b). Learning to program in LISP. *Cognitive Science*, **8**, 87-129.

ANDERSON, J. R., & SKWARECKI, E. (1986). The automated tutoring of introductory computer programming. *Communications of the ACM*, **29**, 842-849.

BROWN, J. S., BURTON, R. R., & DE KLEER, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II, and III. In D. Sleeman & J.S. Brown (Eds.), *Intelligent tutoring systems* (pp. 227-282). New York: Academic Press.

BROWN, J. S., COLLINS, A., & DUGUID, P. (1988). *Situated cognition and the culture of learning* (Tech. Rep. No. 6886). Cambridge, MA: Bolt, Beranek, & Newman.

BURNS, H., PARLETT, J. W., & REDFIELD, C. L. (1991). *Intelligent tutoring systems*. Hillsdale, NJ: Erlbaum.

BURTON, R. R., & BROWN, J. S. (1979). Toward a natural language capability for computer-assisted instruction. In H. O'Neil (Ed.), *Procedures for instructional systems development* (pp. 79-98). New York: Academic Press.

CARD, S. K., MORAN, T. P., & NEWELL, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Erlbaum.

FARRELL, R. G., ANDERSON, J. R., & REISER, B. J. (1985). An interactive computer-based tutorial for LISP. In *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 106-109). Los Altos, CA: Morgan Kaufmann.

FISCHETTI, E., & GISOLFI, A. (1990). From computer-aided instruction to intelligent tutoring systems. *Educational Technology*, **30**, 7-12.

Foss, C. L. (1987a). *Learning from errors in ALGEBRALAND* (Tech. Rep. No. IRL-87-0003). Palo Alto, CA: Institute for Research on Learning.

Foss, C. L. (1987b). *Memory for self-derived solutions to problems* (Tech. Rep. No. IRL-87-0002). Palo Alto, CA: Institute for Research on Learning.

GOODMAN, D. (1987). *The complete HyperCard handbook*. New York: Bantam Computer Books.

GOODMAN, D. (1990). *The complete HyperCard 2.0 handbook*. New York: Bantam Computer Books.

KIERAS, D. E. (1987). *A guide to GOMS task analysis*. Unpublished manuscript. University of Michigan, Department of Psychology.

KIMBALL, R. (1982). A self-improving tutor for symbolic integration. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 283-308). New York: Academic Press.

LEE, A. Y. (1989). *A preliminary look at timing of feedback in tutoring systems* (ICS Tech. Rep. No. 89-10). Boulder, CO: University of Colorado, Institute of Cognitive Science.

LEE, A. Y., POLSON, P. G., & BAILEY, W. B. (1989). Learning and transfer of measurement tasks. In K. Bice & C. Lewis (Eds.), *CHI'89 Conference proceedings: Human factors in computing systems* (pp. 115-120). New York: Association for Computing Machinery.

LEWIS, M. W., & ANDERSON, J. R. (1985). Discrimination of operator schemata in problem solving: Learning from examples. *Cognitive Psychology*, **17**, 26-65.

LEWIS, M. W., MILSON, R., & ANDERSON, J. R. (1987). The TEACHER'S APPRENTICE: Designing an intelligent authoring system for high school mathematics. In G. Kearsley (Ed.), *Artificial intelligence and instruction: Applications and methods* (pp. 269-301). Reading, MA: Addison-Wesley.

MCKENDREE, J. (1986). *Impact of feedback content during complex skill acquisition*. Unpublished doctoral dissertation, Carnegie-Mellon University, Pittsburgh.

NECHES, R., LANGLEY, P., & KLAHR, D. (1987). Learning, development, and production systems. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development* (pp. 1-45). Cambridge, MA: MIT Press.

O'SHEA, T. (1982). A self-improving quadratic tutor: A self-improving tutor for symbolic integration. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 309-336). New York: Academic Press.

POLSON, M. C., & RICHARDSON, J. J. (1988). *Foundations of intelligent tutoring systems*. Hillsdale, NJ: Erlbaum.

PSOTKA, J., MASSEY, L. D., & MUTTER, S. A. (1988). *Intelligent tutoring systems: Lessons learned*. Hillsdale, NJ: Erlbaum.

REISER, B. J., ANDERSON, J. R., & FARRELL, R. G. (1985). Dynamic student modelling in an intelligent tutor for LISP programming. *In Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 8-14). Los Altos, CA: Morgan Kaufmann.

SINGLEY, K. (1987). *Developing models of skill acquisition in the context of intelligent tutoring systems*. Unpublished doctoral dissertation, Carnegie-Mellon University, Pittsburgh.

SINGLEY, K. (in press). Improving operator selection through goal posting in a calculus word problem tutor. *Journal of Artificial Intelligence & Education*.

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

VANLEHN, K. (1988). Toward a theory of impasse-driven learning. In H. Mandl & A. Lesgold (Eds.), *Learning issues for intelligent tutoring systems* (pp. 19-41). New York: Springer-Verlag.

WENGER, E. (1987). *Artificial intelligence and tutoring systems*. Los Altos, CA: Morgan Kaufmann.