HENRY M. HALFF

Instructional Applications of Artificial Intelligence

Computer programs that can tutor, discuss subject matter with students, and diagnose student errors will profoundly affect educational practice.

Photographs by Gregory E. Dixon

Computers play many educational roles in today's schools. Drill-and-practice programs let students exercise with more efficiency and less pain than older manual methods. Tools such as word processors have brought signal increases to students' productivity. Programming languages and other problem-solving environments allow for intensive practice of cognitive skills. Computers are also used for a rudimentary form of automated instruction that provides individualized routing through a curriculum.

But, with a few exceptions, computers don't really teach. They don't know the material the way that teachers do, so they can't discuss it with students the way teachers can. Although computers can observe how students perform different tasks, these observations are commonly used to tell *how much* and not *exactly what* a student knows or does not know. Nor do computers have much instructional expertise. Rather, they rely on programs that may or may not incorporate principles of good teaching.

Teaching requires intelligence, and if computers are to teach, then they must be given the intelligence needed for teaching. To describe research on computers with the intelligence needed to teach, I survey artificial intelligence (AI), the discipline dedicated to making machines intelligent. Then I use the notion of a computer-based tutor to explore the main issues involved in intelligent computer-based teaching. Finally, I speculate on the future of artificial intelligence in education.

AI for Instructional Applications

Artificial intelligence is a field dedicated to discovering how computers can be made to exhibit intelligence, including the intelligence involved in teaching and learning. To understand what the future holds for AI and education, we need to look at how the AI enterprise views issues associated with teaching and learning. AI is concerned primarily with two such issues: what is the fundamental nature of intelligence, and how can we make computers do the things that we consider intelligent?

Intelligence and Knowledge

Most of the AI community's questions about intelligence concern the nature of knowledge and its use in cognitive tasks. We can speak of three kinds of knowledge: conceptual, procedural, and imaginal. The AI community has worked with computer representations of all three kinds of knowledge and, more to the point, has relied on them for instructional applications of artificial intelligence.

Conceptual knowledge. Conceptual knowledge is knowledge of concepts and the relations among them. Knowledge of facts and definitions is conceptual. Conceptual knowledge can be represented in AI systems using a device known as a *semantic network*.

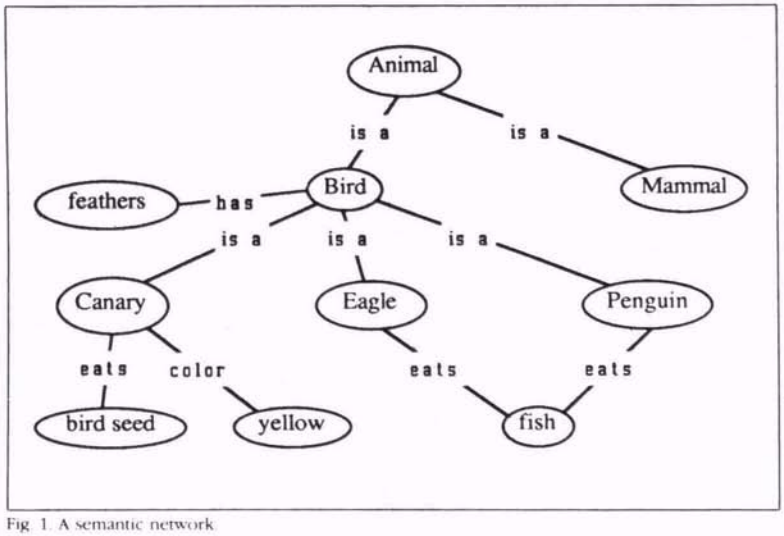


Fig. 1. A semantic network.

Working Memory

Problem		
23 +78	Current Column	ones
	Current Column Sum	11

Rule Set

No.	Condition	Action
1	Current column is blank.	Record the problem in working memory, and set current-column to ones.
2	Current-column-sum is blank.	Sum the digits in current-column, and put the result in current-column-sum.
3	Current-column is ones, and current-column-sum > 9.	Write units part of current-column-sum in sum row of ones column, write tens part of current-column-sum in carry row of tens column, set current-column to tens, and make current-column-sum blank.
4	Current-column is ones.	Write current-column-sum in sum row of ones column, set current-column to tens, and make current-column-sum blank.
5	Current-column is tens, and current-column-sum > 9.	Write units part of current-column-sum in sum row of tens column, write tens part of current-column-sum in sum row of hundreds column, and stop.
6	Current-column is tens.	Write current-column-sum in sum row of tens column, and stop.

Fig. 2. A production system for two-column addition. (The rule interpreter [not shown] on each cycle, executes only the first applicable rule. Working memory is shown as it would appear after the first two cycles.)

Semantic networks consist of nodes representing concepts and links representing the relationships between concepts. Figure 1 shows a simple semantic network describing common knowledge about birds. Computer programs can interrogate the network to accomplish a number of tasks associated with intelligence. They can answer questions such as "What color are canaries?" or "Is a penguin an animal?" They can compare and contrast concepts: penguins and canaries are both birds, but penguins eat fish and canaries eat birdseed. They can make inferences: eagles are birds, and birds have feathers; therefore, eagles have feathers.

One of the first instructional applications of AI used a semantic network to represent the subject matter to be taught. Carbonell's (1970) SCHOLAR used a large semantic network of concepts and facts about South American geography to answer students' questions, to ask them questions, and to evaluate their answers.

Procedural knowledge. Procedural knowledge is the kind of how-to knowledge needed for particular cognitive tasks such as solving mathematics problems, understanding a spoken sentence, or writing a computer program. Many AI programs use a *production system* to represent such knowledge. A production system typically has three parts: a *working memory*, a *rule set*, and a *rule interpreter*. Figure 2 shows a production system for two-column addition problems.

The working memory contains information about the solution as it evolves. In the case of two-column addition, working memory holds the results of intermediate calculations and notes the problem-solver's focus of attention. The rule set consists of a collection of rules, each with the potential for advancing the solution by modifying working memory. Each rule has two parts, a *condition* and an *action*. The condition tells what must be present in working memory for the rule to apply, and the action specifies what is done to working memory when the rule is used. A production system operates in cycles. On each cycle the rule interpreter executes one or more of the rules that apply. The system in Figure 2 executes only one rule on each cycle—the first applicable rule in a fixed precedence order.

Production systems have been used with considerable success to model a

number of human cognitive skills, including basic arithmetic (Young and O'Shea 1972), computer programming (Anderson, Farrell, and Sanders 1984), and problem solving in physics (Simon and Simon 1978), to name a few. Current instructional applications of AI rely heavily on production systems to represent procedural knowledge at all levels of skill acquisition.

Imaginal knowledge. Common sense, anecdotes, and even some compelling research results (Kosslyn 1980), tell us that imagery is an important part of human intelligence. We can think about imagery as the ability to produce, in the mind, the results of some sensory experience. We can imagine our mother's face, the smell of burning leaves, or what it might be like to catch a baseball hit into the stands at a major league game. These imaginings seem to be different in character from the facts about such phenomena that might be captured in a semantic network. Imaginal knowledge seems more like perception than a network of facts.

The popularity of computer graphics tells us that the importance of imaginal knowledge has escaped no one concerned with educational computing. The most influential computer graphic traditions originated in projects to bring computing power to children, namely Papert's (1980) Logo project at MIT and Kay's Smalltalk project at Xerox's Palo Alto Research Center (Kay and Goldberg 1977).

Applied Artificial Intelligence

Making AI work in instruction requires not only theory but also applied science, and the most popular applications currently go by the name of *expert systems*. Expert systems use the rule-based systems described above not to model cognition but rather to solve tough technical problems, such as medical diagnosis or oil exploration.

An expert system is an automated consultant. Given a problem, it requests data relevant to the solution. After analyzing the problem, it presents a solution and explains its reasoning. Expert systems are relevant to education because they can represent problem-solving expertise and explain to students how to use it.

A brief description of an archetypal expert systems, MYCIN (Shortliffe 1976), illustrates how these systems work. MYCIN diagnoses various forms of meningitis based on clinical data that it gathers in consultation with a physician. Its expertise is captured in a set of rules, each of which consists of a premise and a conclusion. Figure 3 shows one of MYCIN's rules.

Expert systems apply their rules to a problem using an *inference engine*. MYCIN's inference engine considers each possible diagnosis in turn, trying to prove it to be correct. It constructs proofs by looking for rules that include the diagnosis under consideration in the conclusion and then trying to verify one of the rules thus found, either on the basis of clinical data or other rules. MYCIN is said to use a *backward-chaining* strategy since it works backward from diagnostic conclusions to clinical data.

Almost as popular in applied AI communities as expert systems are techniques for *natural language understanding*. Programs using these techniques can understand ordinary typed language. They can answer questions (Harris 1977), summarize documents (Cullingford 1978), and converse with students (Brown, Burton, and de Kleer 1982). Programming a computer to understand natural language is much more difficult that it might seem, especially in the light of the apparent ease with which people accomplish this task, because of the ambiguities present in almost any utterance (Winograd 1984). Most computer programs that are successful in natural language comprehension owe their success to so restricting the context of the discourse that ambiguities can be resolved quite easily.

I mention natural language techniques here because of their obvious potential in instructional systems. Teaching machines that can converse with students are clearly more flexible than those supporting more restrictive interaction. In addition, just as expert systems can represent a body of technical knowledge, so natural language systems can embody linguistic conventions and standards. Students may one day be able to get explanations of linguistic productions just as users of expert systems can get explanations of problem solutions.

IF the gram stain of the organism is gram negative, and
the morphology of the organism is rod, and
the aerobicity of the organism is anaerobic,

THEN there is suggestive evidence (0.6) that the genus
of the organism is Bacteroides.

Fig. 3. A diagnostic rule from MYCIN. (Adapted from Clancey (1982), Fig. 1.)

Anatomy of an Automated Tutor

How can the AI tools described above be brought to bear on instructional problems? For most researchers inter-

ested in this question, the answer lies in the concept of an *Intelligent Tutoring System* (ITS). An ITS is a computer system that provides intelligent instruction on a one-to-one basis. ITSs, as commonly conceived, have three components: an articulate expert that knows and can work with the subject matter to be taught, a student model that represents the tutor's moment-to-moment view of the student, and an instructional system that can apply instructional principles to the tutoring task.

Researchers dealing with AI and instruction have usually focused their efforts on one or another of these components. Let's take a closer look at some of the exemplary research on each component.

An Articulate Expert in Diagnostic Skills

Seven or eight years ago William Clancey of Stanford University began thinking about using the expert system MYCIN to teach diagnostic skills. On the face of it, MYCIN has just what instruction requires. It can solve diagnostic problems, and it can explain its solutions. Clancey (1982) implemented an ITS, called GUIDON, based on MYCIN, but his experience with GUIDON convinced him that MYCIN's approach to diagnosis had little in common with human experts' methods and that MYCIN was, therefore, fundamentally unsuited to tutorial tasks. Based on an intensive study of expert diagnosticians, Clancey (1984) created a new expert system, NEOMYCIN, that explicitly embodies human diagnostic expertise. NEOMYCIN's expertise differs from that of MYCIN and most other expert systems in three important ways.

Forward reasoning. People using goal-directed, backward-chaining inference methods like MYCIN's usually are novices who have not yet learned to solve problems in their domain. Experts are more inclined to reason forward from data, drawing out the important implications of known facts (Simon and Simon 1978). In contrast to human experts, a MYCIN-like articu-

The Geometry Tutoring Project in Action

A Student in Peabody High School's special geometry classroom brings up a problem on the computer screen. The student sees the problem diagram in the upper left hand corner, the givens across the bottom, and the goal at top center. Using a mouse, the student selects a set of statements, and the system prompts the student to enter a rule of geometric inference that takes these statements as premises. Next, the system prompts the student to type in the conclusion that follows from the rule. The Xerox 1109 Advanced Scientific Information Processor (Dandetiger) is updated with each sequence of premise, rule of inference, and conclusion.

At any time in the process, the student can ask the system for help with definitions, postulates, and theorems appropriate to the problem. In addition, if the student is not on a proof path, the tutoring part of the system (that is, that part that keeps track of the student's strategic choices) will guide the student back onto a proof path. Should the student make a logical error in inference, the system recognizes the error and tutors accordingly. The system functions as coach *or* as tutor, depending on need.

The teacher, Rick Wertheimer, plays an active role in computer-assisted teaching. Using a problem editor, Wertheimer draws a problem diagram and inputs the givens and the statement to be proved. The geometry expert "finds" all the possible solutions and prepares the problem to be tutored, all in time for the student's next class.

The goal of the Geometry Tutoring Project is to develop a theory of student problem solving and skill acquisition in high school geometry, and to contribute, thereby, to a general theory of problem solving and skill acquisition as well as to a general theory of intelligent computer-assisted instruction. As constructed environments, the tutors allow researchers to embed and test a theory within the educational system; the control population is all other students taking geometry.

The geometry tutor follows the student's every problem-solving move. An "ideal" student model embedded within the tutor represents the skill knowledge necessary to solve the kinds of problems that students will have to solve. A "buggy" model allows the tutor to recognize patterns of student error. As the student works through the curriculum, an "individual" student model records how well he or she has learned particular geometry skills. These three models make possible both immediate feedback and individually tailored instruction.

The Geometry Tutoring Project differs from many other software programs in several important respects: (1) It embodies principles derived from one of the most complete theories of cognition, John Anderson's Act* Theory; (2) it uses artificial intelligence technology; (3) the problem-space representation, displayed on screen as a *proof graph*, makes explicit the goal structure of the problems as well as the search that is required to solve them; (4) the system is not an add-on to the curriculum; it comprises about 50 percent of the curriculum this year and is expected to comprise between 60 and 70 percent next year; and (5) the system tailors instruction to individual students.

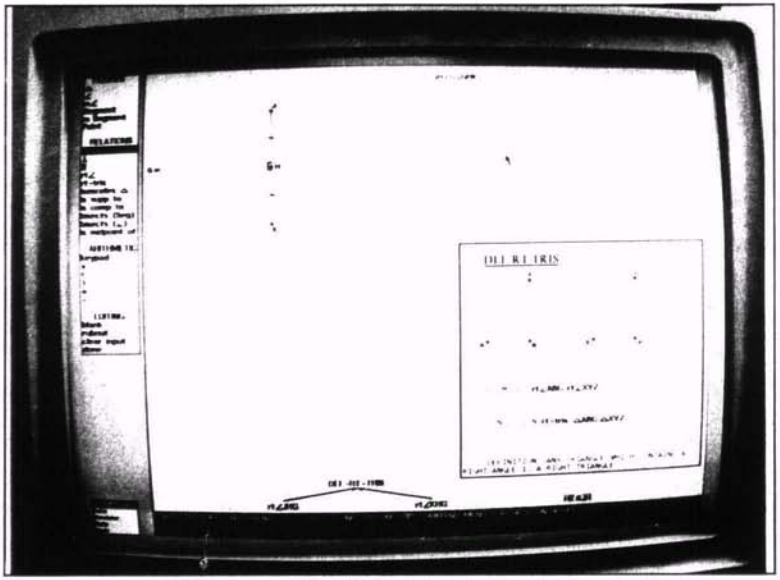
Thus, three major components in the geometry tutor work toward better student learning: (1) the expert system that embodies the ideal, buggy, and individual student models; (2) the tutor that controls strategic interaction between student and expert, and (3) the interface that communicates knowledge to the student on the terminal screen, focuses the student's attention, and serves as an external memory. In the near future, a math laboratory could become a standard high school facility, and the sophisticated interactive environment of intelligent computer-assisted tutoring could enable students to work productively on their own time at school or at home.

—By C. Franklin Boyle, Geometry Project Leader, Advanced Computer Tutoring Project, Carnegie-Mellon University, Pittsburgh, PA 15213.

late expert substitutes weak, backward-inference methods for stronger, forward-reasoning strategies. In a sense, MYCIN works against the acquisition of expertise.

Diagnostic processes and goals. How then does diagnostic reasoning proceed in NEOMYCIN? NEOMYCIN views diagnosis as a cognitive task or mental discipline. To complete a diagnosis NEOMYCIN must accomplish certain goals, and, since it is articulate, it can discuss those goals with students in the context of particular diagnostic exercises. NEOMYCIN's diagnostic goals all revolve around a working-memory structure called the ***differential***, which simply lists potential diagnoses. The differential evolves as symptom descriptions and case data trigger certain suggestions. Further diagnostic goals direct the consultation to (1) narrow down the differential to one or a few diagnostic categories and (2) refine and elaborate the diagnosis within that category to account for all the clinical data.

A computer program that can discuss this process in particular cases is a powerful tool for teaching diagnostic skills. In addition, Clancey was able to create a language to represent diagnostic skills and possibly other mental disciplines as well. This language could prove to be useful in creating other articulate experts.



The computer screen illustrates a problem display in John Anderson's Geometry Tutoring Project.

Domain knowledge. A third important difference between a typical expert system and NEOMYCIN is its use of conceptual knowledge—in this case, the principles of medicine. MYCIN knows absolutely nothing about medicine. It knows only about the empirical associations between symptoms and disorders. NEOMYCIN, on the other hand, relies heavily on medical knowledge. It has a semantic network that describes the relationships among disorders, and particular diagnostic tasks are tied to this hierarchy. Thus, the task of confirming meningitis can be traced down the taxonomic hierarchy to acute or chronic meningitis. NEOMYCIN also possesses causal knowledge that allows it to leap across taxonomic categories. Knowing about the presence of brain pressure leads NEOMYCIN to consider the possible substances that might be causing this pressure.

NEOMYCIN's use of domain (medical) knowledge constitutes a valuable contribution to the connection between theory and practice. Instead of teaching theory in the abstract, NEOMYCIN is able to discuss theory with students as it applies to particular cases.

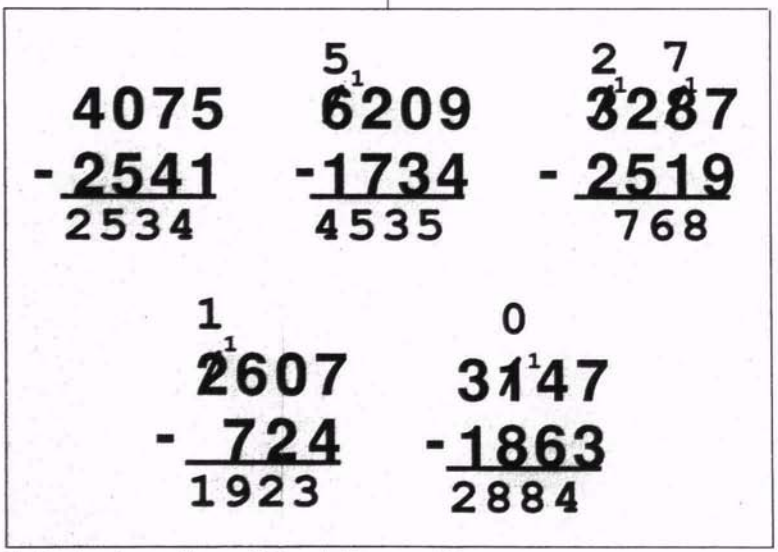


Fig. 4. Subtraction problems exhibiting the "$0 - n = n$" bug.

Student Models for Arithmetic Skills

A student model is a computer representation of an individual student's knowledge, ideally constructed solely on the basis of that student's performance data. One line of research on the modeling of children's arithmetic skills illustrates two general techniques for student modeling.

DEBUGGY. John Brown and Richard Burton (1978) began looking at children's performance in multicolumn subtraction in the late 1970s. They used data, such as that shown in Figure

4, consisting of worked subtraction problems from many students at many different stages of learning the skill, to examine the error patterns in individual students' work. In Figure 4, for example, the student's only problem was writing n when confronted with the problem of what to do with $0 - n$. Brown and Burton found that systematic error patterns accounted for a large proportion of the errors in their data. Sometimes students would not borrow across 0, on other occasions they would always subtract the smaller from the larger digit, irrespective of their rows.

Brown and Burton call these patterns *mind bugs* or, more often, *bugs*, and they developed a catalog of bugs that now numbers over 100 (Burton 1982). Burton (1982) wrote a computer program called DEBUGGY that could look at a student's work and automatically determine which, if any, bugs were present in the work. This approach and its computer implementation is one method of student modeling based on bug catalogs. We also see *bug catalogs* used for student modeling in computer programming (Johnson and Soloway 1985) and elementary algebra (Sleeman 1982).

Repair theory. Not long after Brown and Burton's initial work, they and an MIT graduate student, Kurt VanLehn, noted that bugs manifest in one set of data might not be present in another set taken from the same child. In some cases the bug might disappear altogether; in other cases a different bug would replace it. This led Brown and VanLehn (1980) to formulate *repair theory*, a different approach to modeling students.

Repair theory, an example of an *overlay model* (Carr and Goldstein 1977), represents intermediate stages of learning as incomplete overlays on a fully competent model of the skill. Brown and VanLehn formulated a production-system model of multicolumn subtraction and suggested that individual students could be modeled by deleting rules or combinations of rules from that model. This suggestion is called the *deletion principle*. When a student confronts a problem requiring a deleted rule, he or she makes up a patch to repair the procedure (hence, the name *repair theory*). These repairs show up as bugs on subsequent problems.

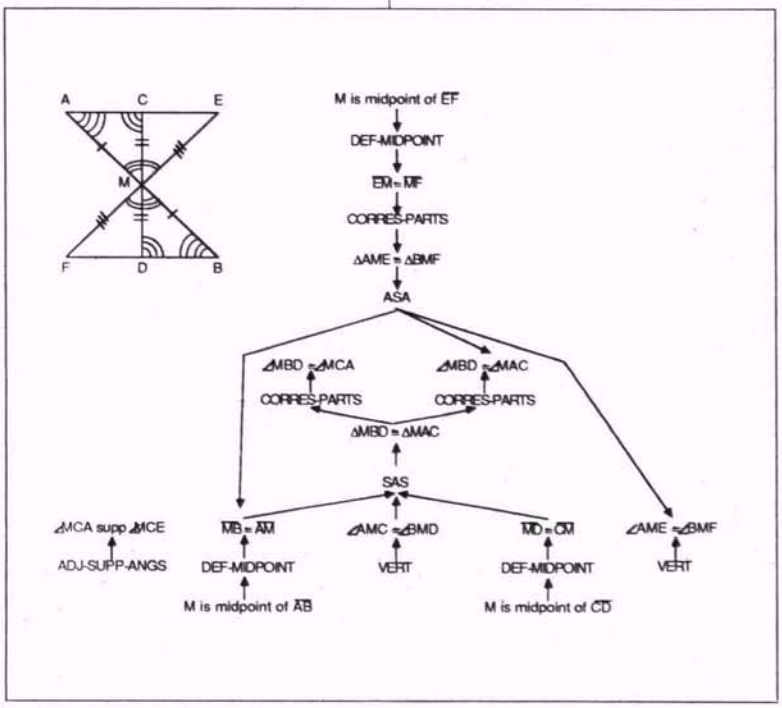


Fig. 5. Geometry tutor display after the student has completed the proof. (Givens are stated at the bottom of the screen, the goal is stated at the top, and all points that appear collinear in the diagram are assumed to be collinear. Adapted from Anderson, Boyle, and Reiser [1985], Fig. 1c.)

Overlay models are stronger than bug catalogs because one can derive all possible student models from a single "expert" model without having to determine the possibilities from case-by-case observations. On the other hand, it is difficult to construct a theory to which the deletion principle applies.

Instructional Systems for Problem-Solving Skills

Having discussed programs that know and can discuss subject matter and programs that can determine the state of a student's knowledge, we are now in a position to discuss the tutoring process itself. There are, unfortunately, very few functioning examples of automated tutors. Two of these examples (Anderson, Boyle, and Reiser 1985) are the creation of John Anderson of Carnegie-Mellon University.

One of Anderson's tutors teaches the programming language LISP. The other one, which we will draw on for this discussion, teaches proof skills in geometry. Both tutors work from a curriculum of exercises. Students work their way through each exercise, and the tutor silently works along with them. When the student strays from a correct solution path or asks for help, the tutor advises the student how to proceed with the problem. The tutoring process uses two important mechanisms.

First, the student and the tutor share knowledge about the problem-solving process. The geometry tutor encourages the student to develop and record conjectures about proof steps. Figure 5 shows how the tutorial system records these conjectures. Note

that the student can develop forward-reasoning conjectures—steps that can be inferred from what is known, and backward-reasoning conjectures—steps that lead to the desired conclusion. The student is free to pursue more than one line of reasoning at a time.

Second, the tutor is able to follow the student's problem-solving approach because it has a production-system model of geometry problem solving. Rules in the system for both forward and backward reasoning can be used to produce proofs much like those of highly competent students. In addition, Anderson's tutor has a bug catalog, a set of productions that characterize commonly observed errors.

The geometry tutor examines each conjecture the student proposes. If that conjecture is consistent with a rule in the tutor's competent model, the tutor remains silent. But the tutor takes the opportunity to teach the student if the student appears to be using a buggy rule or if the conjecture cannot be accounted for by any rule. This tutoring may address a particular bug or suggest a more fruitful approach to the problem.

The Future of AI in Education

Instructional applications of AI are still primarily research endeavors, but with working tutors like Anderson's and with the decreasing cost of computer hardware, we can expect workable applications within the near future.

"When we speak of computers that bring some degree of intelligence to instruction, the first question that comes to mind is, 'They aren't going to replace teachers, are they?' The answer, of course, is no."

Computers and teachers. When we speak of computers that bring some degree of intelligence to instruction, the first question that comes to mind is, "They aren't going to replace teachers, are they?" The answer, of course, is no. The responsibilities and roles of teachers in a school classroom extend far beyond the computer applications mentioned above. Artificial intelligence will increase the effectiveness of teaching, it will expand the range of skills that can be taught, and it will bring difficult skills within the grasp of more individuals. But it will not replace the teacher or even measurably reduce the size of teaching staffs. That said, where can we expect AI applications to instruction?

Automated tutoring. In spite of the essential role of teachers, their numbers probably will not increase in the near- to midterm, and it would be foolish to expect every student to have a personal tutor. Still, the effectiveness of one-on-one teaching is well documented (Bloom 1984), and it is essential to learning in some situations. Automated tutors have an obvious application in situations where instruction and practice should be tightly coupled.

For example, many students flounder in laboratory or in-class exercises since teachers normally do not have time to follow each student's attack on each exercise and provide remedial help. Working with automated tutors, students might learn far more in a much shorter period of time than students left to their own devices. There is some evidence (Anderson, Boyle, and Reiser 1985) that the LISP tutor approaches human tutors in efficiency and leaves traditional methods of instruction far behind.

Society may have justifiable concern that computers will amplify social inequities in the educational system, since rich parents and school districts can afford computers and poor ones cannot. The backgrounds of students' parents, however, are probably far more influential in perpetuating social disadvantages. Well-educated parents have the resources to help and encourage their children with just about any school work, but less well-educated parents can provide no such help. But suppose that students in the future would carry home not just exercises but computer tutors to provide the right kinds of help with homework. These portable tutors could give disadvantaged children the support they might not be able to get from their parents.

Testing and diagnosis. One lesson from the story of student modeling is that the grading methods used in many classrooms do little to illuminate the deficiencies underlying poor performance. As an instructor, I would find it far more useful to know that the child's work illustrated in Figure 4 exhibited a "$0 - n = n$" bug than to know that he or she was correct on one of the five problems. But as an instructor, I would also find it personally impossible to analyze even a few children's exams for particular bugs. It seems obvious, therefore, that a program to identify student problems at the cognitive level could be a powerful teaching tool. Each approach to student modeling constitutes a potential diagnostic tool that can take teachers beyond simple right-wrong scoring.

Language arts is another potential area for applying diagnostic tools. Several commercially available programs (Braby and Kincaid 1981–82; Macdonald, Frase, Gingrich, and Kennan 1982)

can analyze and critique the surface features of a piece of writing, such as spelling, awkward constructions, and so forth. At least two research projects (Heidorn, Jensen, Miller, and Byrd 1982; Kieras 1985) are exploring ways of using the techniques of understanding natural language to provide more meaningful analyses of writing. Programs resulting from this research should be able to detect grammatical errors, and perhaps provide stylistic and semantic analyses as well.

Curriculum development. Looking further into the future, we can expect computers to learn how we construct exercises, examples, and other instructional materials. Natural language techniques could be used to actually read draft materials as students would, and thus provide a more precise analysis than the reading-grade measures now used. Also, a program that knew the skills to be taught in a lesson or curriculum might be able to construct exercises and tests for the target material.

Artificial intelligence has the potential to profoundly change educational practice. I have limited my discussion of these changes to a few direct applications of artificial intelligence in education. For a broader view, I recommend two highly readable books: *Learning and Teaching with Computers: Artificial Intelligence in Education* by O'Shea and Self (1983), and *The Cognitive Computer* by Schank with Childers (1984).

Artificial intelligence will probably continue to develop along three important lines. First, AI has the potential for making subject-matter expertise and teaching expertise more portable. Once represented in a computer, these two kinds of expertise can be combined more flexibly and delivered more effectively to more students. Second, like other computer applications, AI will automate tasks (like finding mind bugs) that are infeasible or impossible by manual methods. Third, and perhaps most important, every new AI application to instruction requires detailed investigation of teaching and learning. The results of these investigations alone are of immense value for what they teach us about the educational enterprise. □

References

Anderson, J. R., F. C. Boyle, and B. J. Reiser. "Intelligent Tutoring Systems." *Science* 228: (1985) 456-462.

Anderson, J. R., R. Farrell, and R. Sauers. "Learning to Program in LISP." *Cognitive Science* 8 (1984): 87–129.

Bloom, B. S. "The Search for Methods of Group Instruction as Effective as One-to-One Tutoring." *Educational Leadership* 41 (May 1984): 4–17.

Braby, R., and J. P. Kincaid. "Computer-Aided Authoring and Editing." *Journal of Educational Technology Systems* 10, 2 (1981–82): 109–124.

Brown, J. S., and R. R. Burton. "Diagnostic Models for Procedural Bugs in Basic Mathematical Skills." *Cognitive Science* 2 (1978): 155–192.

Brown, J. S., R. R. Burton, and J. de Kleer. "Pedagogical, Natural Language and Knowledge Engineering Techniques in SOPHIE I, II, and III." In *Intelligent Tutoring Systems*, edited by D. Sleeman and J. S. Brown. London: Academic Press, 1982.

Brown, J. S., and K. VanLehn. "Repair Theory: A Generative Theory of Bugs in Procedural Skills." *Cognitive Science* 4 (1980): 379–426.

Burton, R. R. "Diagnosing Bugs in a Simple Procedural Skill." In *Intelligent Tutoring Systems*, edited by D. Sleeman and J. S. Brown, pp. 157–183. London: Academic Press, 1982.

Carbonell, J. R. "AI in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction." *IEEE Transactions on Man-Machine Systems* 11 (1970): 190–202.

Carr, B., and I. P. Goldstein. *Overlays: A Theory of Modeling for Computer-Assisted Instruction* (AI Memo No. 406). Cambridge: Massachusetts Institute of Technology, 1977.

Clancey, W. J. "Tutoring Rules for Guiding a Case Method Dialogue." In *Intelligent Tutoring Systems*, edited by D. Sleeman and J. S. Brown, pp. 201–226. London: Academic Press, 1984.

Clancey, W. J. "Methodology for Building an Intelligent Tutoring System." In *Method and Tactics in Cognitive Science*, edited by W. Kintsch, J. R. Miller, and P. G. Polson, pp. 51–83. Hillsdale, N.J.: Erlbaum, 1984.

Cullingford, R. E. *Script Application: Computer Understanding of Newspaper Stories*. (Research Report 116). New Haven, Conn.: Yale University, Computer Science Dept., 1978.

Harris, L. R. "Natural Language Front Ends." In *The AI Business: The Commercial Uses of Artificial Intelligence*, edited by P. H. Winston and K. A. Prendergast, pp. 149–161. Cambridge, Mass.: MIT Press, 1984.

Heidorn, G. E., K. Jensen, L. A. Miller, and R. J. Byrd. "The EPISTLE Text Critiquing System." *IBM Systems Journal* 21 (1982): 305–326.

Johnson, W. L., and E. Soloway. "PROUST." *BYTE* 10, 4 (1985): 179–190.

Kay, A., and A. Goldberg. "Personal Dynamic Media." *Computer* 10 (1977): 31–41.

Kieras, D. E. "Potential for Advanced Computerized Aids for Comprehensible Writing of Technical Documents" (Report No. TR-85/ONR-17). Ann Arbor: University of Michigan, College of Engineering, 1985.

Kosslyn, S. M. *Image and Mind*. Cambridge: Harvard University Press, 1980.

Macdonald, N. H., L. T. Frase, P. S. Ginrich, and S. A. Kennan. "The Writer's Workbench: Computer Aids for Text Analysis." *IEEE Transactions on Communications* 30 (1982): 105–110.

O'Shea, T., and J. Self. *Learning and Teaching with Computers: Artificial Intelligence in Education*. Englewood Cliffs, N.J.: Prentice-Hall, 1983.

Papert, S. *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books, 1980.

Schank, R. S., with P. G. Childers. *The Cognitive Computer*. Reading, Mass: Addison-Wesley, 1984.

Shortliffe, T. *Computer-Based Medical Consultations: MYCIN*. New York: American Elsevier, 1976.

Simon, D. P., and H. A. Simon. "Individual Differences in Solving Physics Problems." In *Children's Thinking: What Develops?* edited by R. S. Siegler, pp. 325–348. Hillsdale, N.J.: Erlbaum, 1978.

Sleeman, D. "Assessing Aspects of Competence in Basic Algebra." In *Intelligent Tutoring Systems*, edited by D. Sleeman and J. S. Brown, pp. 185–200. London: Academic Press, 1982.

Winograd, T. "Computer Software for Working with Language." *Scientific American* 251, 3 (1984): 130–145.

Young, R. M., and T. O'Shea. "Errors in Children's Subtraction." *Cognitive Science* 5 (1982): 153–177.

Henry M. Halff is chief scientist, Halff Resources, Inc., 4918 33rd Rd., N. Arlington, VA 22207.