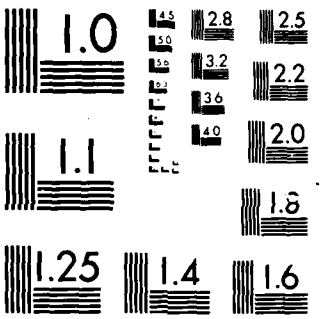
AD-A163 618 EDUCATION AND COMPUTERS: AN AI (ARTIFICIAL INTELLIGENCE) PERSPECTIVE(U) YALE UNIV NEW HAVEN CT DEPT OF COMPUTER SCIENCE R C SCHANK ET AL. OCT 85 1/1

UNCLASSIFIED YALEU/CSD/RR-431 N00014-85-K-0108 F/G 5/10 NL

END

FILMED

DTIC



Education and Computers: An AI Perspective

Roger C. Schank
Stephen Slade

YALEU/CSD/RR #431

October 1985

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Education and Computers: An AI Perspective

Roger C. Schank and Stephen Slade

YALEU/CSD/RR #431

October 1985

This work was done at the Yale Artificial Intelligence Project. The Project is supported by the Advanced Research Projects Agency of the Department of Defense and monitored under the Office of Naval Research under contracts N00014-85-K-0108 and N00014-82-K-0149, and by the Air Force Office of Scientific Research under grant AFO-SR-85-0343.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: YALEU/CSD/RR #431	2. GOVT ACCESSION NO.: AD-A163618
	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Education and Computers: An AI Perspective	5. TYPE OF REPORT & PERIOD COVERED: Research Report
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Roger C. Schank and Stephen Slade	8. CONTRACT OR GRANT NUMBER(s): N00014-85-K-0108, N00014-82-K-0149
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Yale University Department of Computer Science, 10 Hillhouse Avenue, New Haven, Connecticut 06520	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Advanced Research Projects Agency, 1400 Wilson Boulevard, Arlington, VA 22209	12. REPORT DATE: October 1985
	13. NUMBER OF PAGES: 34
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): Office of Naval Research, Information Systems Program, Arlington, VA 22217	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

learning
Computer assisted instruction
Intelligent tutoring systems
Artificial Intelligence

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Our work at the Yale Artificial Intelligence Project has provided key scientific insights into human cognitive behavior, in general, and learning in particular. These scientific developments can be applied to education.

At the heart of thinking is the need to explain the unfamiliar. We regard intellectual curiosity as a major educational resource to be nurtured and promoted. Our computer models of human cognitive behavior indicate that explanation plays a critical role in learning. Unfortunately, the educational

DD FORM 1 JAN 73 1473

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

system has been unable to foster and exploit what should be central in our schools, namely, the love of learning.

Primary and secondary education today face many problems. Children aren't learning the basic skills of reading, writing and math.

The introduction of computers in the classrooms has not provided the predicted panacea, but has instead brought problems of its own, primarily due to inadequate software for the core curriculum. Most of the available educational software is terrible. It is ineffective, costly, unproven, and inappropriate.

We discuss the tremendous potential of the computer in education and offer standards for achieving high-quality educational software for all parts of the curriculum. In particular, compters offer a unique opportunity to cultivate the student's natural desire to explore and explain the world. Keywords:

L

FLD 19

OFFICIAL DISTRIBUTION LIST

Defense Documentation Center — 12 copies
Cameron Station
Alexandria, Virginia 22314

Office of Naval Research — 2 copies
Information Systems Program
Code 437
Arlington, Virginia 22217

Dr. Judith Daly — copies
Advanced Research Projects Agency
Cybernetics Technology Office
1400 Wilson Boulevard
Arlington, Virginia 22209

Office of Naval Research — 1 copy
Branch Office - Boston
495 Summer Street
Boston, Massachusetts 02210

Office of Naval Research — 1 copy
Branch Office - Chicago
536 South Clark Street
Chicago, Illinois 60615

Office of Naval Research — 1 copy
Branch Office - Pasadena
1030 East Green Street
Pasadena, California 91106

Mr. Steven Wong — 1 copy
New York Area Office
715 Broadway - 5th Floor
New York, New York 10003

Naval Research Laboratory — 6 copies
Technical Information Division
Code 2627
Washington, D.C. 20375

Dr. A.L. Slafkosky — 1 copy
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380

Office of Naval Research — 1 copy
Code 455
Arlington, Virginia 22217

Office of Naval Research — 1 copy
Code 458
Arlington, Virginia 22217

Naval Electronics Laboratory Center — 1 copy
Advanced Software Technology Division
Code 5200
San Diego, California 92152

Mr. E.H. Gleissner — 1 copy
Naval Ship Research and Development
Computation and Mathematics Department
Bethesda, Maryland 20084

Captain Grace M. Hopper, USNR — 1 copy
Naval Data Automation Command, Code 00H
Washington Navy Yard
Washington, D.C. 20374

Dr. Robert Engelmore — 2 copies
Advanced Research Project Agency
Information Processing Techniques
1400 Wilson Boulevard
Arlington, Virginia 22209

Professor Omar Wing — 1 copy
Columbia University in the City of New York
Department of Electrical Engineering and
Computer Science
New York, New York 10027

Office of Naval Research — 1 copy
Assistant Chief for Technology
Code 200
Arlington, Virginia 22217

Computer Systems Management, Inc. — 5 copies
1300 Wilson Boulevard, Suite 102
Arlington, Virginia 22209

Ms. Robin Dillard — 1 copy
Naval Ocean Systems Center
C2 Information Processing Branch (Code 8242)
271 Catalina Boulevard
San Diego, California 92152

Dr. William Woods — 1 copy
BBN
50 Moulton Street
Cambridge, MA 02138

Professor Van Dam — 1 copy
Dept. of Computer Science
Brown University
Providence, RI 02912

Professor Eugene Charniak — 1 copy
Dept. of Computer Science
Brown University
Providence, RI 02912

Professor Robert Wilensky — 1 copy
Univ. of California
Elec. Engr. and Computer Science
Berkeley, CA 94707

Professor Allen Newell — 1 copy
Dept. of Computer Science
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Professor David Waltz — 1 copy
Univ. of Ill at Urbana-Champaign
Coordinated Science Lab
Urbana, IL 61801

Professor Patrick Winston — 1 copy
MIT
545 Technology Square
Cambridge, MA 02139

Professor Marvin Minsky — 1 copy
MIT
545 Technology Square
Cambridge, MA 02139

Professor Negroponte — 1 copy
MIT
545 Technology Square
Cambridge, MA 02139

Professor Jerome Feldman — 1 copy
Univ. of Rochester
Dept. of Computer Science
Rochester, NY 14627

Dr Nils Nilsson — 1 copy
Stanford Research Institute
Menlo Park, CA 94025

Dr. Alan Meyrowitz — 1 copy
Office of Naval Research
Code 437
800 N. Quincy Street
Arlington, VA 22217

Dr Edward Shortliffe — 1 copy
Stanford University
MYCIN Project TC-117
Stanford Univ. Medical Center
Stanford, CA 94305

Dr Douglas Lenat — 1 copy
Stanford University
Computer Science Department
Stanford, CA 94305

Dr. M.C. Harrison — 1 copy
Courant Institute Mathematical Science
New York University
New York, NY 10012

Dr Morgan — 1 copy
University of Pennsylvania
Dept. of Computer Science & Info. Sci.
Philadelphia, PA 19104

Mr Fred M. Griffee — 1 copy
Technical Advisor C3 Division
Marine Corps Development
and Education Command
Quantico, VA 22134

Education and Computers:

An AI Perspective

Roger Schank and Stephen Slade

Technical Report

Yale Artificial Intelligence Project
Yale Department of Computer Science
Box 2158
New Haven, Conn. 06520

Report date: 25 October 1985

Abstract

- Our work at the Yale Artificial Intelligence Project has provided key scientific insights into human cognitive behavior, in general, and learning in particular. These scientific developments can be applied to education.

- At the heart of thinking is the need to explain the unfamiliar. We regard intellectual curiosity as a *major educational resource* to be nurtured and promoted. Our computer models of human cognitive behavior indicate that explanation plays a critical role in learning. *Unfortunately, the educational system* has been unable to foster and exploit what should be central in our schools, namely, the love of learning.

- Primary and secondary education today face many problems. Children aren't learning the basic skills of reading, writing, and math.

- The introduction of computers in the classrooms has not provided the predicted panacea, but has instead brought problems of its own, primarily due to inadequate software for the core curriculum. Most of the available educational software is terrible. It is ineffective, costly, unproven, and inappropriate.

- We discuss the tremendous potential of the computer in education and offer standards for achieving high-quality educational software for all parts of the curriculum. In particular, computers offer a unique opportunity to cultivate the student's natural desire to explore and explain the world

Table of Contents

1. Learning to Think

The Master said, 'He who learns but does not think is lost.' He who thinks but does not learn is in great danger.
Confucius, *The Analects of Confucius* (Book II, No. 15)

Imagine that you are a child of eleven who lives in a rather run down section of a big city. You are riding your bicycle slowly along streets that are not very crowded. With you is a friend and you are talking. You pass a man who is riding a bicycle too. As you pass by, he catches a small part of what you are saying to your friend, and he assumes that it in some way *relates to him*. *He begins to threaten* you and you realize that he is not joking, he is very angry and possibly just a bit crazy as well. He begins to chase after you.

What do you do?

Let's put this question another way. Assume that you have not experienced anything like this before. (a rather large assumption if you are riding through a neighborhood in which you live, so let's assume that you are from the suburbs and are visiting your city cousin). How can you figure out what to do? That is, how can you come up with a new idea, a creative solution, to your problem?

It seems obvious that whatever solution a child can come up with here, it is unlikely that he will have *learned that solution in school*. The reason why a child is unlikely to have learned how to deal with such a situation in school is not because schools don't teach *street smarts*. The reason is that schools don't teach children how to think. Nevertheless, many children do know how to think. Who has been teaching them then?

The answer is simple enough. For the most part, they have been teaching themselves.

CREATIVITY

At this point, we are not concerned about teaching or the school system. We are concerned with thinking, learning, and perhaps most important, creativity. Not creativity in the artistic sense. Our *inner-city child* above may never be an artist, but there are many possible creative solutions that he may be able to come up with to help him in his dilemma. And, if he comes out of it successfully, he may become rather pleased with his planning ability. Perhaps he will strike out in another domain. Perhaps not. The point here is this: we are all endowed with the ability to be creative. Maintaining that ability can be fairly tricky however.

Let's get back to our frightened little boy.

What are his alternatives? He can try to outrace the angry man, but as he is a boy it is unlikely that he can do that. He could stand and fight, but that seems like a bad idea. It seems obvious that seeking protection would be a good idea, but where? Some questions come to mind for the boy to answer. Is there a friend or relative around who could protect him? How about the police? Let's assume that the answers here are *no*. What other alternatives are there? How do you get someone who doesn't know you to protect you? One answer is to find people who are likely to consider themselves to be tough. Especially helpful, would be a group of such people who might be obligated to help him simply because of group social pressure.

Now, our boy can consider where he might find such people. One place is a neighborhood bar. Three things are wrong with this suggestion. One is that it is daytime and the bar might not be especially full. The second is that in getting off the bike, he would have to slow down and this might cause him to get caught before he got into the bar. The third is that, the man might just take his bike and forget about him, and this might not be such a good exchange from the point of view of this boy.

So, what other places are available? This particular boy happens to know that there is a park nearby where a baseball game is almost always going on. He peddles quickly to the pitcher's mound of the ball field. The men who are playing are ready to kill him for interrupting the game when they see the man who is chasing the boy. They immediately threaten the man with their bats and the boy spends the rest of the day watching baseball.

READY-MADE ANSWERS

Was this boy being creative? That can be a suprisingly easy question to answer. We tend to view creativity as something mystical. We believe that not everybody has the mystical quality in the same amounts, and that it is very difficult to judge exactly who has exactly how much of this mystical quality at any given time. But this is really the wrong way to look at the issue.

Actually creativity is very simply defined: Creativity is the ability to *think up a solution that is new to you, to a problem that is new to you, by yourself.*

Would we consider this boy to have been creative in his solution to his problem if he was regularly chased in this way? Or, to put this another way, if he were chased again tomorrow, and he did exactly the same thing would we remark on his creativity? Certainly not. We do not consider a solution to be creative for a person to be creative, if what is being done is simply something the person has done before.

Suppose this boy had a big brother who had been in similar fixes, which he told his little brother about. We would not consider our boy to be creative if all he did was apply what others had told

Now let's consider another example. Let's assume that this same boy is in school and he has been asked to find the length of the hypotenuse of a right triangle whose two other sides are 3 and 4 respectively. He has been daydreaming while this subject was being taught and has never learned how to do this. On the other hand, two other children in his class have been paying attention. One gets the answer immediately. The second makes some calculations and then raises his hand to indicate that he too has the answer. The third, our boy, draws the triangle and is ready to make a guess. Let's consider their answers.

The first boy knew the answer was 5 because he has been paying attention to what has been going on in class and he knows that there is such a thing as a 3,4,5 right triangle, and therefore this must be it. Thus, he immediately knows the answer. Our second boy knew the formula, namely $a^2 + b^2 = c^2$. He thus found a^2 (9) and b^2 (16), added them together to get 25, and took the square root of 25 to get 5. Our boy took out a pencil and carefully drew the triangle. He estimated the length of the hypotenuse to be slightly bigger than the longest side, and guessing that the answer was a whole number, reasoned that the answer was probably 5.

So which of these boys is *right*? Or to put it another way, there are two questions we could ask: *Which of these boys will do best in school?* and *Which of these boys will do best in life?*

When a child does well in school, we consider him to be *smart*. When a child does well on the streets, when he gets himself out of complex situations, we consider him to have *street smarts*. The ability to apply the ready-made answers you have learned to the questions at hand is not to be denigrated, but it is not creativity. Those who learn only the formulas don't create anything new by doing so.

WHO IS THINKING?

The problem is how to think, and how to teach a child to think for himself. We will approach this problem from the rather unusual perspective of Artificial Intelligence (AI). Although AI is a rather arcane branch of Computer Science, it is not necessary to understand anything about either AI or computers to understand our premise. In fact, what AI has to contribute to this discussion is simply one question, namely the question that every AI researcher asks himself every day: *How would I get a machine to do that?*

Let's consider our two childhood problems again in the light of this question. Suppose we wanted a computer to answer the two questions, *How do you get out of situation where someone much bigger than you is chasing you?* and *In a right triangle with sides 3 and 4, what is the length of the hypotenuse?*

For a computer, the second question is trivial in comparison to the first one. There are basically two issues involved in teaching a computer to answer the second question. The first is teaching the computer to understand the English used in expressing the second question. This is difficult but not really all that

complicated because the English in that example is fairly straightforward. In fact, as long ago as twenty years ago, programs were written that could understand algebra word problems far more complicated than this one.

The second issue is the formula for answering the second question. Putting that formula into a computer program is so trivial that anyone with the slightest bit of programming knowledge could do it in a matter of minutes. Thus equipped, our computer could answer problems about right triangles (no more and no less) endlessly.

So, what does this tell us about our three boys above? It says that while the boy who had learned the formula was best equipped to handle the problem, and all future problems of the same type, he simply did not have to *think* at all. He was thinking only in the trivial sense of that term that we also ascribe to today's computers. (The question of whether computers could ever think in a more human way is not relevant here. For a discussion of that question, see [Schank 84]. The short answer to the question is yes, in principle, but applying formulas is certainly not an example of thinking.) If all we want our children to do is apply ready-made formulas or spout correct answers, we will end up with very unimaginative young people. We are not recommending against the learning and application of formulas where applicable. What we are concerned about right here is analyzing and evaluating the thought processes that each of these boys used.

In a fundamentally important way each of these boys has made an unconscious decision about how creative he intends to be. The reason for this is that the schools have elected, again unconsciously, to reward the least creative type of reasoning, namely, applying a ready-made formula or rule. We shall now explore the issue of reasoning types in greater depth.

REASONING TYPES

Each of these boys is applying a different method of reasoning which we shall call in turn:

REASONING BY REMINDING
REASONING BY RULE APPLICATION
REASONING BY ASKING

When the first boy remembers that there is a 3,4,5 triangle he is taking a short cut to the right answer. He need not know the formula to get the right answer in this case and thus he may well fail to get the answer in more complicated examples. On the other hand, he may realize that the 3,4,5 relationship expresses what is captured by the formula, but in a different way. In any case, he was able to solve this problem, because of what we might call *a good memory*. He recalled the right answer, or to put it another way, he was reminded of the right answer.

This is a rather simple example of REASONING BY REMINDING. A more complicated example is illustrated by the following: Suppose you were asked which candidate for President was most likely to make peace, the *sabre-rattling hawk* or the *let's withdraw the troops liberal?* You might respond that you were reminded of Richard Nixon. You might argue that he was virulantly anti-communist but yet was the first to go to China. You might reason that maybe candidates tend to do the opposite of what they say. This might remind you that Lyndon Johnson was the *peace candidate* during the early stages of the Viet Nam War. This is an example of REASONING BY REMINDING.

The second boy was REASONING BY RULE APPLICATION. Simply, he learned a rule and applied it. When the rule is iron-clad, like this one is, that is often the best method of solving the problem. (Although even here that method took a little longer than the first.)

There are a great many rules like the Pythagorean Theorem in daily life. They do not exist solely in mathematics. These rules are called proverbs or cliches, depending on their presumed profundity. So, when a person takes vitamins every day he may be living (consciously or unconsciously) by the rule: *an ounce of prevention is worth a pound of cure*. Or when a child walks away from people who are taunting him, he may yell back *sticks and stones may break my bones but names will never harm me*. Much of our lives are governed by "rules" such as these, and it is neither right nor wrong to do so. At times these rules form perfectly sound advice. The problem for the creative reasoner is being able to reason when there are no rules available or when the available rules are for some reason inappropriate.

The third boy was REASONING BY ASKING. In other words, he really didn't know where to start and he was figuring it out by asking himself a set of questions to guide his reasoning This is of course, what he was doing in the bicycle example as well. Now the question is: what are we doing when we are *just figuring it out*?

THE VALUE OF FIGURING IT OUT

How do we get a computer *to figure it out?* We ask this question here not because the issue of getting computers to think is intrinsically important, but rather because by reference to that question we can begin to see the complexity of the processes involved. What would it mean to give a computer the rules for figuring something out? What would those rules look like? What kind of background knowledge would a computer have to have so that it could figure things out?

Most important, of course, is how people figure things out. What does our boy have to know in order to reason his way out of trouble or in order to figure the answer to a question in mathematics? In our research, we deal with the specifics of how people reason. We also deal with the question of how we can learn to reason better. As part of the answer to this second question, we must examine why it is better, in the long run, to figure something out than simply to recall the answer.

Let's put this another way. Is our boy better off because he failed to learn the formula? Will he have a better chance of being a creative individual in later life if he learns the formula, memorizes exemplars, or just brings his basic reasoning powers to bear whenever he is confronted with a problem?

Our system of education stresses the learning of the formula. If you do not believe that this is so, ask yourself if you learned the formula noted above (known as the Pythagorean Theorem). Most everyone who completed an academic high school education learned this theorem. Now ask why you learned it? What value did it have in your later life? Did you use it again (except when helping your children with their math homework)?

It seems obvious that the reason to learn a formula is the same reason that we have for writing a computer program to follow a formula. That is, to the extent that we ourselves embody tight repetitive procedures, to the extent that we need to use a formula again and again, that is the level of importance of learning that formula. But, of course, in the computer age, even that need has vanished. There can be little justification for learning any formula as long as there exist machines that can slavishly do these computations for us.

There is an exception to this of course. Because we have hand calculators available to us is no reason for failing to learn to add. Why? Because, we need to understand first principles in order to reason for ourselves. In order to REASON BY ASKING we must have at our disposal the basic questions to ask. In other words, we must know how to add to understand multiplication, we must understand multiplication in order to understand division, and so on. But, a formula is not a basic tool of asking. It introduces no new concepts, but merely applies some old ones.

Schools have been teaching formulas for a long time now. In the computer age, teaching children to do what computers can do makes little sense. Rather, we must teach children to do what computers cannot do. And, if in some years computers equal the achievements of these children, if creative thought becomes formulaic in nature, then at least the equation of people with computers will be on a higher plane than it is now.

THERE ARE NO RIGHT ANSWERS

How then do we teach creative reasoning? The first thing we have got to do is to get over the idea that there are right answers to questions. Our boy on his bicycle didn't need the right answer, he needed *an* answer. There may have been other better answers, but the one he selected worked and that's what mattered most at the time.

Further, in the geometry class, the boy who did the best was again our boy, because he was the one who tried to figure it out. You can be sure, however, that few teachers would see it that way. In school we

expect answers to questions. We want facts. We ask who discovered America and want the name Columbus, not some hedging about Vikings, or comments about American Indians. We are a fact-oriented society. Schools, as they are presently constituted in the United States have one primary purpose educationally, and that is the effective performance of their students on standardized tests.

There clearly are other functions of schools, such as, keeping kids off the street, providing a mechanism for social assimilation, teaching children a common set of facts about the world, and introducing students to the need to get along within a bureaucracy (that is, school itself). Underlying all of these functions should be ability to reason and think. This clear and unifying focus is usually lost in the attempt to score well on tests.

Children are taught from the very early grades that there *are* right answers and that they will be rewarded for getting as many of them as possible. And what of our creative bicyclist? He may well fail in school because, although he can figure things out for himself, he will get the wrong answer a great deal. He may do poorly on standardized tests and feel very frustrated in school.

What should we do? We should learn about what it means to think. Schools cannot teach thinking if they don't know what it means or how to do it. None of us have been taught to think (at least not in school), so very few people really can think very creatively.

2. Teach Questions, not Answers

To give a glimpse of the thrust of our work, we shall discuss some ideas about thinking and creativity.

Probably the most significant thing one can say about thinking is that it is inspired by questions. In many cases, as in the bicycle example above, these questions can come from the real needs of everyday life. So, if one is frequently asked questions, or to put it another way, if one frequently comes upon situations that are out of the ordinary that pose problems, one will get lots of practice in thinking.

The school situation could be an acceptable format for the asking of questions, but as long as formulaic answers satisfy the questions, then the questions are not of any use. Further, questions where there is one and only one right answer are of no use in stimulating thinking. To think, one must learn to justify one's answers in front of a severe and respected critic. This can be your spouse, a teacher, or even your child, but it must be someone who is willing to argue with you. It is necessary to be able to justify your answer. If you cannot, you have not answered the question.

We propose that learning to think in a creative, stimulating fashion requires learning to be inquisitive. Specifically,

1. You must be asked questions, either by yourself, others, or by situations you encounter.

2. These questions must be out of the ordinary. If you have been asked this question before, such that answering it requires no more than mentally looking up the answer, the question won't help you think.

3. Someone whom you respect must evaluate your answer.

Student learning -- Intrinsic and Extrinsic Motivation

In education circles today, there is much discussion of the *student learning* problem. Simply put, how do you get students to learn subjects that they don't care about? A typical example of the student learning problem is high school physics. Very few high school students seem to be excited by the traditional introductory physics curriculum.

As a matter of fact, this is not a problem that seems to occur in our experience with either graduate students, on the one hand, or young children (ages 2 through 8), on the other. Why? At these opposite ends of the educational spectrum, there seems to be little in common, except the most important factor: *they want to learn*. They enjoy thinking about new concepts. Young children and Ph.D. candidates are both excited by ideas. They want to know stuff. They love asking questions. They find learning *intrinsically* satisifying.

The problem-solving nature of physics, combined with the accepted difficulty in teaching the subject,

has lead several researchers in cognitive science and artificial intelligence to look at the underlying cognitive processes involved in solving physics problems [Larkin et al. 80, diSessa 82, Larkin 83, Heller and Reif 84, Anzai and Yokoyama 84]. These researchers have analyzed subjects' behavior in detail and created computer models of the behavior. While this research is useful and interesting, it doesn't address the basic question of how to *motivate* students to learn physics in the first place. Moreover, it doesn't touch on the need to stimulate *creative* thinking.

What currently motivates most of our high school students? Grades. Scores on standardized tests. Class rank. Getting into the college of your first choice. Students not headed for college may simply want to graduate, to get a diploma. Thus, the grade often becomes the end in itself. The refrain of *Is this going to be on the exam?* usually means: *Do I have to pay attention to what you're saying?* So what? Shouldn't a high school student still want to learn physics to make an A? (or just to pass?) Maybe, but these are all *extrinsic* motivations. Psychological research has demonstrated that students tend to learn better when motivated *intrinsically*, than when given extrinsic movitation [Lepper and Malone 85, Malone and Lepper 85].

What goes on in a physics course? The student learns lots of formulas, and rules to recognize when to apply which formula. If there is a laboratory associated with the course, the student gets an opportunity to witness that the formulas are pretty accurate, except of course if the student makes a mistake. Then the formula still must be correct, and the student knows that his experimental technique is poor (and so is his grade, most likely). What does the student learn? He learns to REASON BY RULE APPLICATION. As we have argued above, this approach to thinking is basic, but *limited*. *What the student does not* learn is to be creative. The problem solving paradigm obviates creativity. Furthermore, this rule application approach to teaching science has a fundamental premise which is unfounded and even damaging, namely, that scientists know what the right answer is. This assumption of correctness and precision in physics and other sciences is very misleading and one that is readily questioned by actual practitioners. Still, science educators cling to the crutch of the *right answer*.

An alternative can be found in a literature or philosophy course. In these fields, the notion of a right answer has largely been discarded. What is the right interpretation of *Hamlet*? What is freedom? What can be known? Philosophers often delight in arguing the same question from two opposing points of view (or three or four or five points of view). The archetype of this style of reasoning is the great philosopher Socrates. Socrates didn't give his students answers. He gave them questions. In doing so, he taught them how to ask questions themselves, and how to think.

Discussion-based teaching has become widely accepted in English, history and other humanities. In these fields, there is much less concern with the student learning problem, and we submit that this is not

merely a coincidence. The interactive nature of discussions provides the student with greater involvement in the material and, most importantly, with intrinsic motivation. The problem-solving paradigm typical of physics is more remote.

Imagine then, a Socratic dialog about physics. The teacher would not be there to write a proof on the blackboard demonstrating the veracity of some formula. The instructor would raise questions about physical phenomema, and stimulate the students to ask questions of their own. The students shouldn't be handed the accepted wisdom without first understanding why it is important in the first place. To the student, the question should be *Why does the world behave in this way?*. It is much better for the subject matter to stimulate the students to investigate and ask more questions, than to provide merely extrinsic goals that prompt the student to ask *Is this going to be on the exam?*

Great, but how do you teach students to ask questions? How can students be trained to generate hypotheses? We offer two answers. First, look at results from AI in making computers generate hypotheses. Second, use interactive computer programs to allow the students to test hypotheses.

Creative Computers

For the past 11 years, the Yale Artificial Intelligence Project has produced pioneering research in developing computer models of human cognitive behavior. A wide variety of theories and models have been explored.

- programs that analyse the conceptual content of natural language texts [Riesbeck and Schank 76, Gershman 79, Granger 80, Riesbeck and Martin 85]
- programs for writing stories on their own [Meehan 76]
- programs for reading newspaper headlines and stories [Schank and Abelson 77, Cullingford 78, Wilensky 78, DeJong 79]
- programs that translate texts from one language to another [Carbonell et al. 78, Lytinen and Schank 82, Lytinen 84]
- programs that answer questions given in normal English [Lehnert 78, Dyer 82]
- programs that can learn language [Selfridge 80]
- computer models of human memory organization [Schank 79, Lehnert 79, Kolodner 80]
- models of learning [Lebowitz 80, Schank 80, Schank 81, Schank 82]

Our work in AI has always focussed on the basic question: how does the human mind work? In each of the cases cited above, we have tried to isolate certain aspects of human cognition and model them with computer programs.

One of the most fundamental insights of this work has been the recognition of the central role that past experience plays when faced with understanding a new situation. That is, people are reminded of past events when solving a new problem. The new situation most likely will not be exactly like the prior situation, but there might be enough overlap to provide useful information.

We referred to this type of thinking earlier as REASONING BY REMINDING. We have explored this type of creative reasoning in AI research. An example from an AI program may serve to illustrate the importance of reminding, asking questions, and explanation.

2.1 CHEF - An example of Reasoning by Reminding and Asking

CHEF [Hammond 84] is a computer program which generates original plans, (which take the form of recipes), in the domain of cooking, by modifying existing plans. It demonstrates how episodic knowledge can be used to guide planning and avoid past failures.

When presented with a problem, how to prepare a certain dish, the program is reminded of previous related recipes. It modifies the most similar previous recipe to fit the new requirements, and then tries out the new recipe. (It does this through a simulation involving rules which specify the physical effects of each step of the cooking process.) The results are then examined to see if they match the intended dish. If the program recognizes a *failure*, it then tries to analyze and explain the failure through a process of reasoning by asking questions. Finally, the program modifies the recipe in light of its explanation to correct the failure.

In the following example, the program has been asked by the user to make a souffle with strawberries. We present actual output from the program with annotation.

```
Searching for plan that satisfies -
   Include strawberry in the dish.
   Make a souffle.

Found recipe -> REC4 VANILLA-SOUFFLE

Recipe exactly satisfies goals ->
    Make a souffle.

Recipe must be altered to match ->
    Include strawberry in the dish.

Building new name for copy of VANILLA-SOUFFLE based on its goals.

Calling recipe STRAWBERRY-SOUFFLE
```

So the program is reminded of a previous related recipe for Vanilla Souffle. It then copies and modifies

the recipe to add strawberries. The computer simulates the execution of the recipe to see if its goals are satisfied. It discovers that most of the goals are met: the batter is baked, the dish tastes like berries, the dish is sweet, etc. However, there are some important goals that failed.

```
Checking for negative features ->

Unfortunately: The batter is now a bad texture.
In that: The batter is now flat.

Recipe -> STRAWBERRY-SOUFFLE has failed goals.

The goal: The batter is now risen.
is not satisfied.

Changing name of recipe STRAWBERRY-SOUFFLE

to BAD-STRAWBERRY-SOUFFLE
```

The program now tries to *learn* from this failure by asking questions and explaining why the failure happened.

```
ASKING THE QUESTION: 'What is the failure?'
ANSWER-> The failure is: It is not the case that: The batter is now risen.

ASKING THE QUESTION: 'What is the preferred state?'
ANSWER-> The preferred state is: The batter is now risen.

ASKING THE QUESTION: 'What was the plan to achieve the preferred state?'
ANSWER-> The plan was: Bake the batter for twenty five minutes.

ASKING THE QUESTION: 'What were the conditions that led to the failure?'
ANSWER-> The condition was: There is an imbalance between the whipped stuff
 and the thin liquid.

          ************ noting balance failure ************

Only one aspect of the imbalance:
 There is an imbalance between the whipped stuff and the thin liquid.
is unexpected.

The state:
 There is whipped stuff in the bowl from the total equaling 60 teaspoons.
normally participates in the goal:
 The batter is now risen.

Only the other aspect of the imbalance:
 There is thin liquid in the bowl from the strawberry equaling 2.4 teaspoons
is an unexpected condition.
```

The program continues its analysis and eventually decides to modify the recipe to reduce the imbalance between the whipped egg whites and the liquid from the strawberry mixture. The program uses a

Thematic Organization Packets (TOPs [Schank 82]) to make the high level association necessary to alter the plan. It has five strategies which it considers.

```
Found TOP TOP3 -> SIDE-EFFECT:DISABLED-CONDITION:BALANCE
TOP -> SIDE-EFFECT:DISABLED-CONDITION:BALANCE has 5
       strategies associated with it:

       USE-DIFFERENT-PLAN:PLAN1
       USE-DIFFERENT-PLAN:PLAN2
       ADJUNCT-PLAN
       RECOVER
       ADJUST-BALANCE

Applying TOP -> SIDE-EFFECT:DISABLED-CONDITION:BALANCE
to failure It is not the case that: The batter is now risen.
in recipe BAD-STRAWBERRY-SOUFFLE

Asking questions needed for evaluating strategy: USE-DIFFERENT-PLAN:PLAN1

ASKING ->
Is there an alternative to
 Pulp the strawberry.
that will enable
 The dish now tastes like berries.
which does not cause
 There is thin liquid in the bowl from the strawberry equaling 2.4 teaspoons

Found plan: Instead of doing step: Pulp the strawberry
 do: Using the strawberry preserves.
```

The program continues to evaluate the other four strategies, compares the results, and finally decides to apply the ADJUST-BALANCE strategy, which in this case means to add more egg whites.

```
Changing name of recipe BAD-STRAWBERRY-SOUFFLE
to STRAWBERRY-SOUFFLE

Implementing plan -> Increase the amount of egg white used.
Suggested by strategy ADJUST-BALANCE

New recipe is -> STRAWBERRY-SOUFFLE

                STRAWBERRY-SOUFFLE

          Two teaspoons of vanilla
          A half cup of flour
          A quarter cup of sugar
          A quarter teaspoon of salt
          A half cup of milk
          Two cups of milk
          One piece of vanilla bean
          A quarter cup of butter
          Five egg yolks
          Six egg whites
```

One cup of strawberry

Mix the flour with the sugar and salt.
Mix the milk with the mixture of sugar, salt and flour.
Boil the milk and vanilla bean for less than a half minute
Remove the vanilla bean from the milk.
Mix the mixture of milk, sugar, salt and flour with the milk.
Simmer the mixture of milk, sugar, salt and flour for five minutes.
Whip the egg yolk.
Add the butter and mixture of egg yolk to the mixture of milk, sugar, salt and flour.
Cool the mixture of egg yolk, milk, sugar, salt, flour and butter.
Whip the egg white.
Add the vanilla and mixture of egg white to the mixture of egg yolk, milk, sugar, salt, flour and butter.
Pulp the strawberry.
Mix the strawberry with the spices, egg, milk, salt, flour and butter.
Pour the mixture of egg, spices, strawberry, salt, milk, flour and butter into a nine inch baking-dish.
Bake the batter for twenty five minutes.

If this plan is successful, the following should be true:

The batter is now baked.
The batter is now risen.
The dish now tastes like berries.
The dish now tastes sweet.
The dish now tastes like vanilla.

Once all new interactions have been validated, the new recipe is indexed in the data base in terms of those interactions. When the program is then given the task of creating a raspberry souffle, it is reminded of the dish it just created and produces a new recipe which has no failures. The program has learned by asking questions and being creative.

The CHEF program illustrates explicitly what we mean by REASONING BY REMINDING. The program was presented with a situation for which it had no exact previous match. It was *reminded* of a similar previous case, and used that as the basis for solving the new problem. In the course of adapting that previous case to fit the new situation, the program encountered several additional problems, but was able to respond to those on the basis of previous cases as well. This is an example of creative reasoning. When it encountered failures, the program was able to REASON THROUGH ASKING. That is, it explained the failures through a question-based reasoning chain. Both of these r oning mechanisms contribute to the creativity of the program.

2.2 Computers as Teachers of Reasoning through Asking

Earlier, we discussed the need for teachers to stimulate students to ask questions. We feel that in many ways, computers are well suited for this role. Computer programs, as demonstrated by CHEF, can capture knowledge in restricted domains and reason about that knowledge. Computers can relate prior experiences to new situations.

What we ask of our computer teacher is to stimulate the student to ask questions. The basic cycle for the interaction would be:

1. Computer poses difficult question --- for which there may be no right answer.
2. Student generates a hypothesis.
3. Computer responds with a counterexample from its data base of remindings.

Then, the student must revise the hypothesis, and the cycle continues. Note that a central part of this process is *the student fails to get the right answer*. The computer is continually trying to point out holes in the student's answers. We maintain that a computer can get away with this, but a teacher in a classroom can't. The reason is simple, but compelling: the computer is not judgmental. Children recognize that there is no social stigma attached to being corrected by a computer (especially if everyone is treated that way). However, children are very sensitive to the attitudes of teachers and other students. No child wants to be continually singled out as unable to answer a question. The computer has greater latitude.

Computers offer students a great and important luxury: the opportunity to fail. As we have written before [Schank 81], failure plays a critical role in the basic cognitive mechanism of learning. In describing the cognitive process involved in understanding a new situation, we suggested the following procedure for understanding social situations:

1. Utilize the appropriate high-level knowledge structure to process input. (e.g., scripts, plans, etc. [Schank and Abelson 77])
2. When an expectation generated by those structures fails, attempt to explain the failure.
3. To explain the failure of another person to act according to your predictions, attempt to figure out his or her beliefs. This includes:
   a. Assessing the implicit beliefs that you expect him or her to hold in that situation.
   b. Producing an alternative belief that the actor might hold by combining your assumptions with the actor's behavior.
4. Use the Alternative Belief as an index to memory to find other memories previously classified with the Alternative Belief.

5. Use the other features of the episode as additional indices with the range of behavior delimited by the Alternative Belief to find an actual memory to be used for generalization and modification of predictions.

In other words, people rely on past experience to understand new situations. Previous episodes provide predictions which can be applied to new cases. Sometimes a prediction does not work, that is, the world does not always behave the way you expect it to behave. It is these *expectation failures* which provide an opportunity for learning. These failures should stimulate the person to *explain* what went wrong. These explanations then be used to locate other previous experiences that may be related. The explanation and associated remindings then get incorporated into the knowledge-base of prediction, and help to prevent the person from repeating the failure.

We believe this process of FAIL-EXPLAIN-REMIND to be basic to learning. Our educational system should capitalize on this underlying cognitive mechanism by, in effect, providing opportunities for the child to make mistakes and fail, without the stigma normally associated with negative reinforcement. Computers provide a one-on-one teaching environment which is free from the intimidating effects of public scrutiny.

3. More Problems: Computers in Schools

The following story appeared on the front page of the Sunday *New York Times*, December 9, 1984:

School's Use Of Computers Disappointing

> After investing heavily in microcomputers, public schools in the New York metropolitan area are finding that they are still far from achieving the academic revolution expected from the new technology ...

Computers have arrived in classrooms across the country amid very high expectations. In the past 20 years, we have witnessed the dramatic and exciting developments in computer hardware, resulting in the wide availability of powerful machines at a small fraction of the cost of the decades before. These new computers were going to change the way children were taught, and start a revolution in learning.

The observer of the current state of computers in education will realize that most of the learning from computers that has occurred is by the teachers and administrators who have learned that computers are not living up to their early promise. Computers have not transformed the schools into a technological forum of learning. The schools are still having a hard time teaching the bread and butter subjects of reading, writing, and arithmetic. It turns out that the only subject that absolutely requires the use of a computer is learning about the computer itself --- hardly a surprising result.

3.0.1 The Problems of Computers and Education

The substance of the problems of computers in the schools is summarized accurately in the article from *The New York Times* cited above.

- Computers are used largely to teach computer literacy, and have neglected the principle areas of the school curriculum. Little is being done to use computers to enhance the regular curriculum.
- The biggest problem is the lack of adequate software for other subjects. The available software does not fit well into the existing curriculum.
- Schools lack means of identifying good software and training teachers to use it.
- Most computerized instruction is routine drill and practice (electronic workbook, see below) which is proving not to be effective.
- School software does not fully utilize the power of the machines, but instead mimics other (less expensive) media, such as books or overhead projectors.

In spite of these discouraging developments, we are optimistic about the future possibilities of computers in the schools. Today, almost all primary and secondary schools use computers in the classroom. Aside from the disappointment due to the lack of adequate software, there have been positive experiences

Teachers have realized that when appropriate software is available, computer exercises can be used to reward advanced students or to deal more patiently with slow learners.

The problems of our schools today are tremendous and tragically underrated [Schank 84, Ragosta 83]:

- Children are not learning the basic skills.
- Children are bored in school.
- Schools often treat children as a mass instead of as individuals.
- Children don't get enough personal attention from their teachers.
- Many teachers get bored and frustrated and stop caring.

These are major problems facing society. It is trite, but true to view our children as society's major natural resource. Children must be nourished, encouraged, and educated. The problems we find in the schools are then not just the schools' problems, but society's.

Computers can provide a solution, at least in part, to many of these problems.

- Computers can be programmed to teach far more thoroughly and interactively than textbooks. Not only can the computer program ask questions, but the child can ask the computer questions in return. The child can get prompt and meaningful feedback.
- Computers can treat children individually. A child can have his own computer teacher who keeps track of his progress. A good computer program can monitor the mistakes a child has made and focus on those particular problems.
- Computers can be used by almost any child, no matter how hyperactive or lazy or disturbed. Children have demonstrated an amazing affinity for computers -- given the proper software. [Lepper 85, Lepper and Malone 85, Malone and Lepper 85]
- Computers can be excessively patient instructors. They don't get bored or frustrated with students or with teaching. They need not punish or ridicule a student to make him feel inadequate -- though current poorly designed educational software does just that.
- Computers are fun -- or at least they can be.

One could probably make those same arguments on behalf of some exceptionally gifted teacher. Most of us have been lucky enough to have been exposed to such an inspirational and dedicated instructor at least once in our lives. Would that all teachers could be like that. The problem is that such teachers are quite rare and cannot be replicated very easily.

It may well be that exceptional educational software will be very rare, but fortunately, it can be duplicated in mass quantities and made available to every school in the country. Once we build a stable

of star computer software, we can begin to realize the dream of providing outstanding instruction to every child.

Furthermore, while we may bemoan the fact that the star teacher is a rarity, we should also recognize that soon *any* teacher may be scarce. We are basing this prediction not on a hope that computers will replace humans, but rather on the plain fact that fewer and fewer people choose to go into education as a career. School superintendents and principals are quite aware of the problems of attracting and retaining good teachers. They realize that they will have to turn to the technology to keep up with the demand. In effect, education will become less labor intensive and more capital intensive.

3.0.2 BASIC is not basic

What has happened with computers in the schools? We see that they are being used largely to teach something called *computer literacy*. In practice, this usually means teaching computer programming, usually in BASIC, which is the *lingua franca* of microcomputers. There are three reasons why the schools are using computers to teach computer programming:

1. To provide the student with a job skill.
2. To train the student to think logically and develop reasoning skills.
3. There is nothing else suitable for which to use the computer.

Students taught to program in BASIC by a teacher who only recently learned a smattering of BASIC are not getting much of a leg up in the job market -- especially if the computer training is at the expense of the regular subjects in school. A child will not get any job without an adequate ability to read, write, and do math.

Thinking algorithmically is certainly a valuable skill, and one that is salutary for an educated person to possess. However, even though we computer scientists might like to think otherwise, learning to program is not a necessary or sufficient precursor to reasoning ability. There are many other ways to train the mind. Learning to program computers is no magic nostrum.

We believe that using computers to teach computer literacy to every student is a wasteful enterprise. The only legitimate reason for using computers in the schools to teach children about computers is the lack of any adequate software for teaching the truly *basic* skills: reading, writing, and arithmetic. These are skills that form the solid foundation of literacy, both computational and otherwise, in a civilized society. It is criminal to neglect these subjects in our schools under the false hope that computer literacy will prove more useful to our children. We know that people need to know these core subjects, and we know that they are not being taught adequately in all schools. We should also realize that there is no social imperative for children to learn how to program.

Our task is to find better ways of educating our children in the basic skills of civilization. At the same time, we should take aim at another oft neglected goal of a good education: we should want our children to leave school with a true love of learning that can sustain them throughout the rest of their lives. The love of learning is exciting to watch in a young child, and often it seems to be driven away as the child progresses through school. A person who continues through life with an open and inquisitive mind is both rare and wonderful. We should hope that our schools will achieve not only the basic goals of education, but also extend the vision and reach of our children beyond their years in school. Computers may not be the answer to nurturing a love of learning, but at least they should not stand in the way.

We recognize that schools should not be singled out for criticism regarding children lacking a sense of intrinsic reward in learning. This low regard for the love of learning is a societal attitude. We believe though that the schools are a good place to start to try to turn this attitude around. Furthermore, the evidence suggests that an approach to teaching which focuses on intrinsic motivation will achieve greater success.

3.1 Hardware in the Schools

So far, we have discussed the problems of the schools and of computers in the schools. We saw that most of the problems associated with computers in schools were due to the lack of suitable software. Before we discuss the software requirements and standards for education, we should briefly examine what computer hardware is available in the schools.

Computers have long been available in colleges and universities, where they have been used for many scientific and administrative applications. Colleges often spent a million dollars or more in the 60's and 70's to set up computer centers.

Needless to say, few, if any, primary and secondary schools can afford to spend a million dollars for a computer. Furthermore, they do not have the motivation of large scientific and administrative problems to solve. There are some high schools that have bought or borrowed time on someone else's large machine to teach programming, but it wasn't until the advent of the inexpensive microcomputers that many schools started to get on the computer bandwagon. As history has shown, most of those machines were purchased to teach about computers themselves.

The dust has settled somewhat in the hardware arena. We can now look around and see what has transpired, and we can well predict what will be available in schools in the coming years. For now, we can assume that a machine with the power and capabilities of an Apple II or an IBM PC will be the norm. We feel this to be a realistic view. If the schools had more money and could afford a DEC VAX or other supermini computer, then the possibilities would be much greater.

Thus, the machines that are available are not extremely powerful by the standards of today's computers. However, computers of the class of an Apple II or IBM PC have several important capabilities. For the software standards described below, we shall assume a microcomputer with the following abilities:

- at least 64,000 characters of memory
- at least one disk drive
- text display
- color graphics
- music generation (at least one voice)

This is a minimum set of requirements. One may dispute the need for color graphics, but the benefits outweigh the small marginal costs. Furthermore, the costs of high-quality color displays are dropping as the demand for this feature increases. We suggest that this ability can be considered a standard for most educational software for the near future.

Some of the items that we do *not* require of an educational microcomputer would include a joystick, mouse, graphics tablet, ligat pen, voice synthesizer, modem, and a printer for each machine. Cursor keys can provide adequate and accurate positioning; joysticks are more useful for arcade video games which require quick reflexes instead of cognitive skills.

Economical voice synthesizers are still hard to understand and not yet feasible for the one area where they would be extremely useful: programs teaching children to read. A computer that could talk to the child in an intelligible voice would be of great benefit. Older children can better understand today's voice synthesizers, but those same children should also be able to read instructions printed by the computer. An area where voice synthesizers can be especially helpful at present is in computer programs for the blind.

Modems and data communication will someday be an important part of computers in the school, but not until the schools have some well-established central facility for record keeping or curriculum distribution. For the present, we can expect teachers and students to be shuffling floppy disks back and forth.

Printers are very useful, and every school should have several. However, there is no need at present for every machine to have one. A ratio of 1 printer for every 5 or 10 machines should prove adequate for the near future.

Over the next few years, more money will be made available for computers in the schools, both from school boards and from computer manufacturers. It is most likely that the schools will spend the money in three main areas: more equipment (compatible with existing hardware), more software (especially in areas not presently covered), and more training for the teachers in the integration of computers into the school day.

As more money is spent in the near future, we can view the expanding role of computers in the classroom in specific terms: how many hours per week the student will have access to the computer. Today the school that exposes the student to the computer 30 minutes a week will in the next 5 years have moved to 30 minutes a day --- a five-fold increase. Again, this can only happen if there is adequate software to support the commitment of resources.

We view this catalog of hardware capabilites as descriptive, not prescriptive. We recognize that it would be far better for each student to have access to a $50,000 workstation, than to share an Apple II with 30 classmates. Thus, the software is clearly constrained by the hardware realities.

3.2 Software for education: the good, the bad, and the ugly

We now turn to software, bad and good. We first look at the bad. In discussing the problems with existing educational software, we present several categories of software that sometimes overlap:

- computer literacy
- drill and practice
- electronic books
- the educational arcade
- adventure software

As should be clear at this point, there are a lot of computers in the schools today that are not being well-utilized. There are several different problems that we have discussed. The first is that the schools are devoting excessive resources to teaching computer literacy.

Let us imagine a person who buys a new car, takes it home, and then spends several months doing nothing except dismantling and reassembling the automobile. Clearly this person will have learned a lot about auto mechanics and car repair, but he won't have taken advantage of the car to do what it does best: taking people from one place to another. Computer literacy follows a similar path: the schools have bought all these machines and now they aren't using them to help with the problems that existed in the schools before, namely, teaching reading, writing, math, and other subjects.

Now assume that our car owner has decided that he is going to try out the car, instead of merely examining it. He then sits in the driver's seat; puts the key in the ignition; turns on the radio; and spends the rest of the afternoon sitting in his driveway listening to music. The poor man could have bought a much better stereo for a lot less, but he wanted to *use* his car and didn't know how to drive.

Much educational software suffers from this syndrome of underutilization. There are programs which are little more than electronic workbooks in which questions appear on the screen and the child types in an answer. A correct answer causes the machine to go to the next lesson. An incorrect response takes the child back over the previous material.

This type of software is termed *drill and practice* and teachers who have seen children use it recognize it as *dull in practice*. These programs typically make little use of the graphics capabilities of the machines. They are also lacking in what should be viewed as the intrinsic capability of educational software: interactivity. Children who have been exposed to video games are aware of the many appealing features of microcomputers and recognize that simple drill and practice software is deficient. It doesn't hold their attention. They get bored. This is not progress. The child loses interest in the subject, and the reason now is not a $5 workbook, but a $50 computer program, running on a $2,500 computer. It's like buying a Lincoln Continental to play the radio.

Aside from their limited effectiveness, drill and practice programs suffer from another major drawback: they are easy for anyone to develop. Thus, there is a large supply of simpleminded, inadequate educational software produced with very little thought. Easy as it is to write drill and practice programs, some programmers have seen fit to make it even easier through what are called *authoring systems*. These allow a user to create a program without even minimal programming ability. The result is a program with some educational content, but lacking the most basic sophistication and awareness of the full capabilities of the machine. It is comparable to a children's textbook without any supporting illustrations, maps, charts, figures, tables, graphs, or pictures. We would question why the author failed to take advantage of the other effective modes of communication provided by the printed page. We wouldn't take such a book seriously. Neither should we accept the products of these authoring systems. We should demand first class products for our children. They can tell the difference.

One variation on the drill and practice programs is the electronic book. Here a textbook, often with illustrations, is converted to a computer disk. The child reads a page from the computer screen and then presses a key to view the next page. Sometimes the screen will display a picture taken from the book to depict a certain concept. However, the bulk of the program relies on reading text from the screen. After a while, it becomes apparent that this type of program is no better than the book itself. In fact, it is in many ways worse. With a book, the child has more freedom to jump around from one part to another

and to proceed at his own pace. These programs are reminiscent of early motion pictures which were filmed stage plays, rather than tightly edited cinematic productions. Those early movies did not take advantage of the dynamic editing possibilities inherent in movies and missing on the stage. Similarly, much educational software is still tied to the idea of the book, and not to the dynamic and interactive capabilities of computers.

We should be energetic in exploiting the full power of computers for education. We should not be satisfied with the mediocre software currently available. It was not very long ago that television was touted as a great medium for cultural advancement. Television provided an opportunity to bring high-quality drama and art to the masses. The great awakening never happened though. Mediocrity triumphed. The high hopes for uplifting society were never realized. Still, this does not mean that computers for education will suffer the same fate. It is within our power to shape the future.

Television has of course been used for education. Programs such as *Sesame Street* and *Electric Company* are widely embraced. However, computers can be more stimulating to a child than television, which is after all, a very passive experience. Children enjoy working with computers because the child is the one who can control the action. He makes things happen. He is involved. With television, the child is a spectator. With computers, the child is a player. That difference is key. As a player, the child has something at stake and has to be thinking.

However, it is not sufficient for the child to be a player. There must be some cogent educational content to the program. The educational arcade games fall prey to this problem. Arcade games or video games rely on quick reflexes and reaction time responses. They don't develop cognitive skills, but motor skills. Many educational programs have been based on arcade games. These programs link some cognitive task such as spelling or multiplication to a game strategy. Often the link is very loose, such that the game is merely the reward for successful completion of the cognitive task. We believe that this approach may answer some of the criticisms leveled at drill and practice, but is seriously deficient in other respects. The primary task requires motor skills, not cognitive ability. Furthermore, there is no connection between the task, such as addition, and the goal, such as shooting an alien. There is no context of adding numbers as a useful thing to do in real life.

One final type of program that is gaining in popularity and often portrayed as being educational is the adventure/mystery programs. These are interactive novels in which the reader issues commands to the computer to tell the protagonist what action to take at regular intervals in the narrative. These programs are fine when viewed as entertainment software, however, they do not readily fit into any school curriculum. Furthermore, their claim of educational content is usually mitigated by the fact that the competent reader must already have achieved a level of sophistication beyond that of their claimed

instructional level. (A program to teach reading, for example, should not require that the child already know how to read.)

Bad software will always be with us. What is lacking in educational software these days is a suitable set of standards for distinguishing between the good and the bad. We shall now propose initial standards for good educational software.

4. Computers in Schools: The Solution

4.1 Standards for educational software

We have leveled many criticisms at existing educational software. We feel that there is a tremendous need for good software in the schools and that most of what is currently available is inadequate, for the variety of reasons listed above. What should good educational software look like then? We shall now attempt to answer that question. We present the following set of principles of educational software design.

1. The program should achieve a defined educational objective.
2. The child should learn through discovery.
3. The program should be intrinsicly interesting and fun.
4. The program should use concepts already familiar to the child.
5. The tasks and rewards should be appropriate to the target age and background, and to the concepts being taught.
6. The program should make and require reasonable responses.
7. The program should be easy to use.
8. The program should not break.
9. There should be both rewards and remediation.
10. The program should allow mixed-initiative interaction.
11. The program should be part of integrated series.
12. The program should be open-ended.

4.1.1 Defined Educational Objectives

The educational content of a program should not merely be a side-effect or after thought. This is clearly the case with numerous arcade or entertainment programs. It is easy to see that these programs were not designed primarily for education. Their educational content was not an *a priori* concern of the programmer, but an *a posteriori* idea of the marketer.

4.1.2 Learning through discovery

Computers make it possible to learn through simulations. The user can make decisions that change the state of an imaginary world. In this environment, the child can experiment with impunity. The child is then more involved in the action and has to think about his decisions and their consequences. This approach can be applied to the entire school curriculum -- not simply for math problems or spelling drills. A history program should allow the student to simulate political decisions and view events occuring in a causal sequence. A geography program should let the child explore a region and discover its traditions, economy, and so on. A student might manipulate the economy of a country at a micro or macro level, and discover the underlying explanations for consumer decisions and national fiscal policies. A computer chemistry lab should allow the student not only to perform experiments for qualitative and quantitative analysis, but also manipulate molecular models of the results. A biology program could perform simulated genetics experiments instantly without waiting the few days required for real fruit fly results.

4.1.3 Motivation, interest, and entertainment

The program should be so enjoyable that the child wants to use it for its own sake, perhaps never realizing that it's educational. By involving the child as much as possible, the program requires his attention. The program should be so much fun that the child is highly motivated to learn. Here it is important to stress the range of rewards available with graphics, music and interactive choices. We want the child to enjoy learning. As we have discussed earlier, intrinsic motivation is more effective than extrinsic [Lepper 85, Lepper and Malone 85, Malone and Lepper 85].

4.1.4 Familiarity breeds expertise

Children have considerable knowledge about the world and this knowledge should be brought to bear in learning new material. The programs should always try to present the subject in a realistic and familiar context to provide a suitable grounding for the child. The more concepts that the child can bring to bear increases the facility the child will have in learning the new concept. This is especially important in teaching abstract concepts such as algebra or molecular chemistry. Here, the simulation and discovery approach can be best applied. The child is then directly involved in the subject through a concrete context.

4.1.5 Tailor the program tasks and rewards to the user

The tasks, illustrations, and rewards in the program must be suited to the age and cognitive skill level of the intended user. Programs designed for preschoolers should require the simplest of tasks, and provide appropriate rewards. Programs designed for more mature children can require more complex manual or cognitive tasks. Furthermore, the rewards should be matched to the subject matter. For example, money rewards make sense in economic or business contexts.

4.1.6 Don't violate expectations

There must be a reason for everything that happens, and a reason for every action required of the child. The dramatic action of the program must not be arbitrary at any point in the program. The child must not be distracted from the educational task by trying to figure out why the program behaved in some strange way or trying to second guess just exactly what the program is expecting the child to reply. The child's attention should not be diverted from the basic involvement with the program. It is important that the universe of actions embodied in the program be consistent.

4.1.7 Make it easy to use

This would seem axiomatic for any piece of software, but it is especially important when you expect the program to be used by children. Instructions should be easy to understand, which means that the user interface should be very simple and consistent throughout the program. For example, one common way to request input is through a highlighted menu for which the space bar cycles through each item, and the return key selects the current item. The appropriate use of icons is also beneficial.

4.1.8 Make it hard to break

This is really an extension of the previous principle: a program that is easy to break is perforce not easy to use. Again, it is important to realize that the programs are to be played by children who may try all kinds of strange things when using the program. The program should be able to withstand a whole gamut of inappropriate responses without crashing.

4.1.9 Encourage success and expect failure

The program should always be responsive to the child's progress. When the child makes an incorrect response, the program should instruct the child as well as correct the child. If the child has demonstrated competence, he should not be required to recapitulate previous material. The positive response following a correct answer should always be more rewarding than the negative response for a wrong answer. It is important that the child not be encouraged to choose the incorrect answer simply to get a more stimulating reward.

4.1.10 Allow mixed-initiative interaction

Most computerized instruction is one-sided: the computer asks all the questions. This is especially true of drill-and-practice programs, but is also characteristic of other kinds of software. The opposite extreme, where the user asks all the questions, is more common in interactive help systems or information retrieval programs. A teaching situation is best conducted as a combination of the two -- both the student and the teacher initiate questions, and respond to questions posed by the other. Computer programs which have this ability are referred to as mixed-initiative systems, and educational software should strive for this type of productive feedback and interaction.

4.1.11 Integrate within a series

One of the advantages of computer-based instruction is that computers can present course material in discrete, digestible quantities. Each program contains a specific lesson. We have already discussed the importance of building on previous knowledge and experience. It stands to reason that part of that experience should be the previous computer-based lessons themselves. Thus, even though each program may stand on its own as it presents a new concept, the program can benefit from being part of an integrated series in which the lesson design can take advantage of the content of the previous programs in the series.

This approach is especially important in designing a computerized curriculum. For example, basic reading and arithmetic skills for just one primary grade would require several dozen programs in an integrated series. If one selected the same number of programs for the same subjects at random, there would be lots of overlap and many gaps. Furthermore, the disparity in approaches and techniques would present an additional obstacle for the child.

4.1.12 Make it open-ended

It is not possible to step twice into the same river. [Heraclitus] Knowledge is a river, and educational software should reflect the fact that there is always something new and different to discover. The program should embody a diversity of possible approaches and outcomes. Each time the child uses the program, it should provide a fresh instantiation. The more open-ended the program, the greater the opportunity there will be for the child to explore and learn.

4.2 Summary

Children are not being taught to think, but to get grades. Computers have been put in school not to teach basic skills, but to teach about computers.

The way to teach students to think is to stimulate them to ask questions and to explain the unfamiliar. Computers can be used very effectively to prod students in a non-judgemental way. Children find computers intrinsically stimulating.

One final anecdote is illustrative.

A high school acquired a personal computer and a simple computer program for analysing the nutritional content of meals. The school decided that a wonderful way to introduce the computer to the students would be to put the machine in the cafeteria so that the students could instantly find out the nutritional value of their lunches.

After the machine was installed, it would regularly attract large crowds of students. However, teachers discovered that the students would exit the nutrition program and start writing programs of their own.

The teachers then rewrote the nutrition program to remove the exit condition. Soon, the crowds of students returned though, having discovered that flipping the power switch off and on would effectively restart the machine and allow them to bypass the nutrition program.

The teachers responded by disabling the power switch. Still, the students found that the same effect could be achieved by unplugging the computer's power cord and then plugging it back in.

Finally, the teachers secured the power cord to the wall, and were confident that they had sealed off all alternatives for the unruly students. The teachers were wrong.

At the back of the computer was a set of metal connectors. If someone rubbed a coin across the connectors, the computer would short out, and then start up again allowing a new program to be run.

This story may provide some hope. It appears that students are resilient enough to survive clumsy and inappropriate attempts to introduce computers in schools.

5. References

[Anzai and Yokoyama 84]
Anzai, Y., and Yokoyama, T.
Internal models in physics problem solving.
Cognition and Instruction 1(4):397-450, 1984.

[Carbonell et al. 78]
Carbonell, J.G., Cullingford, R.E., and Gershman, A.V.
Knowledge-based Machine Translation.
Technical Report 146, Yale University Department of Computer Science, 1978.

[Cullingford 78] Cullingford, R.
Script Application: Computer Understanding of Newspaper Stories.
PhD thesis, Yale University, 1978.
Research Report #116.

[DeJong 79] DeJong, G.
Skimming Stories in Real Time: An Experiment in Integrated Understanding.
PhD thesis, Yale University, May, 1979.

[diSessa 82] diSessa, A.A.
Unlearning aristotelian physics.
Cognitive Science 6:37-75, 1982.

[Dyer 82] Dyer, M.
IN-DEPTH UNDERSTANDING: A Computer Model of Integrated Processing For Narrative Comprehension.
Technical Report 219, Yale University Department of Computer Science, May, 1982.

[Gershman 79] Gershman, A.
Knowledge-based Parsing.
PhD thesis, Yale University, 1979.

[Granger 80] Granger, R.H.
Adaptive Understanding.
PhD thesis, Yale University, January, 1980.

[Hammond 84] Hammond, K.
Indexing and Causality: The organization of plans and strategies in memory.
Technical Report 351, Yale University Department of Computer Science, December, 1984.

[Heller and Reif 84]
Heller, J.I. and Reif, F.
Prescribing effective human problem-solving processes: problem description in physics.
Cognition and Instruction 1(2):177-216, 1984.

[Kolodner 80] Kolodner, J.L.
Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model.
PhD thesis, Yale University, November, 1980.

[Larkin 83] Larkin, J.H.
The role of problem representation in physics.
In Gentner and Stevens (editors), *Cognitive Science: Mental Models*, chapter 5, pages 75-98. Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.

[Larkin et al. 80] Larkin, J.H., McDermott, J., Simon, D.P., and Simon, H.A.
Models of Competence in Solving Physics Problems.
Cognitive Science 4(4):317-348, October-December, 1980.

[Lebowitz 80] Lebowitz, M.
Generalization and Memory in an Integrated Understanding System.
PhD thesis, Yale University, October, 1980.

[Lehnert 78] Lehnert, W.G.
The Process of Question Answering.
Lawrence Erlbaum Associates, Hillsdale, NJ, 1978.

[Lehnert 79] Lehnert, W.G.
Text Processing Effects and Recall Memory.
Technical Report 157, Yale University Department of Computer Science, May, 1979.

[Lepper 85] Lepper, M.R.
Microcomputers in Education: Motivational and Social Issues.
American Psychologist 40(1):1-18, 1985.

[Lepper and Malone 85]
Lepper, M.R. and Malone, T.W.
Intrinsic Motivation and Instructional Effectiveness in Computer-Based Education.
In Snow, R.E. and Farr, M.J. (editors), *Aptitude, learning, and instruction: III. Conative and affective process analysis*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1985.
In press.

[Lytinen 84] Lytinen, S.
The Organization of Knowledge In a Multi-lingual, Integrated Parser.
PhD thesis, Yale University, 1984.
Research Report #340.

[Lytinen and Schank 82]
Lytinen, S.L., and Schank, R.C.
Representation and Translation.
Technical Report 234, Yale University Department of Computer Science, 1982.

[Malone and Lepper 85]
Malone, T.W. and Lepper, M.R.
Making Learning Fun: A Taxonomy of Intrinsic Motivations for Learning.
In Snow, R.E. and Farr, M.J. (editors), *Aptitude, learning, and instruction: III. Conative and affective process analysis*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1985.
In press.

[Meehan 76] Meehan, J
The Metanovel: Writing Stories by Computer.
PhD thesis, Yale University, 1976

[Ragosta 83] Ragosta, M.
Computer-assisted instruction and compensatory education: a longitudinal analysis
Machine-Mediated Learning 1(1):97-127, 1983.

[Riesbeck and Martin 85]
Riesbeck, C.K. and Martin, C.E.
Direct Memory Access Parsing.
Technical Report 354, Yale University Department of Computer Science, January, 1985.

[Riesbeck and Schank 76]
Riesbeck, C., and Schank, R.C.
Comprehension by Computer: Expectation-based Analysis of Sentences in Context.
Technical Report 78, Yale University Department of Computer Science, October, 1976.

[Schank 79] Schank, R.C.
Reminding and Memory Organization: An Introduction to MOPs.
Technical Report 170, Yale University Department of Computer Science, 1979.

[Schank 80] Schank, R.C.
Language and Memory.
Cognitive Science 4(3):243-284, 1980.

[Schank 81] Schank, R.C.
Failure-Driven Memory.
Cognition and Brain Theory 4(1):41-60, 1981.

[Schank 82] Schank, R.C.
Dynamic memory: A theory of learning in computers and people.
Cambridge University Press, 1982.

[Schank 84] Schank, R.C.
The Cognitive Computer: On Language, Learning, and Artificial Intelligence.
Addison-Wesley, Reading, Mass., 1984.

[Schank and Abelson 77]
Schank, R.C. and Abelson, R.
Scripts, Plans, Goals and Understanding.
Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

[Selfridge 80] Selfridge, M.
A Process Model of Language Acquisition.
PhD thesis, Yale University, January, 1980.

[Wilensky 78] Wilensky, R.
Understanding Goal-Based Stories.
PhD thesis, Yale University, 1978.
Research Report #140.

END

FILMED

3-86

DTIC