ED 207 551 IR 009 662

AUTHOR Mayer, Richard E.
TITLE Contributions of Cognitive Science and Related Research on Learning to the Design of Computer Literacy Curricula. Report No. 81-1. Series in Learning and Cognition.
INSTITUTION California Univ., Santa Barbara. Dept. of Psychology.
SPONS AGENCY National Inst. of Education (DHEW), Washington, D.C.
PUB DATE Dec 80
GRANT NIE-80-G-0118
NOTE 35p.; Shorter version of this paper was presented at the Conference on National Computer Literacy Goals for 1985 (Reston, VA, December 18-20, 1980). For a related document, see IR 009 660.

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS Calculators; *Cognitive Processes; *Computer Science Education; *Learning Processes; Models; *Programing; Psychological Studies; Teaching Methods
IDENTIFIERS BASIC Programing Language; *Computer Literacy; Intuition (Mathematics)

ABSTRACT

A review of the research on techniques for increasing the novice's understanding of computers and computer programming, this paper considers the potential usefulness of five tentative recommendations pertinent to the design of computer literacy curricula: (1) provide the learner with a concrete model of the computer; (2) encourage the learner to actively restate the new technical information in his or her own words; (3) assess the learner's existing intuitions about computer operation and try to build on or modify them; (4) provide the learner with methods for chunking statements into smaller, meaningful parts; and (5) provide the learner with methods for analyzing statements into smaller, meaningful parts. It is concluded that, while results of cognitive research provide qualified support for the first two recommendations, more active research is needed on the other three. A bibliography lists 59 references, and appendices include seven statements used in a BASIC-like instructional booklet, examples of six types of test problems for a BASIC-like language, an example of an elaboration exercise, and data from a study included in the review. (MER)

SERIES IN LEARNING AND COGNITION

Contributions of Cognitive Science and Related Research on Learning to the Design of Computer Literacy Curricula

Richard E. Mayer

Report No. 81-1

Preparation of this report was supported by Grant NIE-80-G-0118 from the National Institute of Education. A shorter version of this paper was presented at the Conference on National Computer Literacy Goals for 1985, Reston, Virginia, December 18-20, 1980.

The goal of this paper is to examine techniques for increasing the novice's understanding of computers and computer programming. In particular, this paper examines the potential usefulness of five recommendations concerning the design of computer literacy curricula, as listed below.

(1) Provide the learner with a concrete model of the computer.

(2) Encourage the learner to actively restate the new technical information in his or her own words.

(3) Assess the learner's existing intuitions about computer operation and try to build on them, or modify them, as needed.

(4) Provide the learner with methods for chunking statements into a larger, single, meaningful unit.

(5) Provide the learner with methods for analyzing statements into smaller, meaningful parts.

For each recommendation, this paper will provide a clear statement of the issue, an example, relevant background, and a brief review of relevant research literature.

As can be seen, each recommendation is concerned with increasing the meaningfulness of learning new computer information by novices. For purposes of the present paper, meaningful learning is viewed as a process in which the learner connects new material with knowledge that already exists in memory (Bransford, 1979). Figure 1 provides a general framework for discussing the conditions of meaningful learning (see Mayer, 1975, 1979a). The figure shows that information enters the human cognitive system from the outside (e.g., through text or lectures, etc.), and must go through the following steps: (1) Reception. First, the learner must pay attention to the incoming information so that it reaches working memory, as indicated by arrow a. (2) Availability. Second, the learner must possess appropriate prerequisite concepts in long term memory to use in assimilating the new information, as indicated by point b. (3) Activation. Finally, the learner must actively use this prerequisite knowledge during learning so that the new material may be connected with it, as indicated by arrow c from long term memory to working memory. Thus, in the course of meaningful learning, the learner must come into contact with the new material (by bringing it into working memory), then must search long term memory for what Ausubel (1968) calls "appropriate anchoring ideas", and then must transfer those ideas to working memory so they can be combined with the new information in working memory. Each recommendation is aimed at insuring one or more of these conditions is met.

The traditional way of evaluating meaningful learning is to test whether learners can transfer what they have learned to new situations. For example, Wetheimer (1959) taught students how to find the area of a parallelogram using a rote method (i.e., memorizing a formula) or a meaningful method (i.e., involving the structure of the figure). Although both groups performed equally well on problems like those given during instruction, Wertheimer claimed that the meaningful learners were able to solve unusual problems requiring creative transfer. Thus, this paper will focus on transfer as a measure of meaningful learning of computer programming.

I. USE CONCRETE MODELS

Statement of the Problem

Novices tend to lack domain specific knowledge (Greeno, 1980; Simon, 1980; Spilich, Vesonder, Chiesi & Voss, 1979). Thus, one technique for improving the novice's understanding of new technical information is to provide them with a domain-specific framework that can be used for assimilating new information--i.e. by allowing for "availability" as indicated by point b in Figure 1. The present section focuses on the effects of concrete models on people's understanding of computers and computer programming.

Example

For example, in our own work on teaching a simple BASIC-like language to novices, we presented a model of the computer such as shown in Figure 2. The model provides concrete analogies for four functional units of the computer: (1) input is represented as a ticket window in which data is lined up waiting to be processed and is placed in the finished pile after being processed, (2) output is represented as a message note pad with one message written per line, (3) memory is represented as an erasable scoreboard in which there is natural destructive read-in but non-destructive read-out, and (4) executive control is represented as a recipe or shopping list with a pointer arrow to indicate the line being processed. This model may be presented to the learner either as a diagram, or as an actual board containing these useable parts.

Background

There is ample evidence that concrete models are widely used in mathematics instruction. For example, early work by Brownell & Moser (1949) indicated that children who learned subtraction algorithms with the aid of "bundles of sticks" were better able to transfer to new problems than children who were given the rules for subtraction in abstract form with plenty of "hands on" experience in executing the procedures. More recently, the important role of "manipulatives" such as coins, sticks, blocks, etc., has been documented by Weaver & Suydam (1972) and by Resnick & Ford (1980).

There is also some evidence that concrete models may enhance comprehension of text. For example, students' recall of an ambiguous passage was enhanced when a title or diagram or introductory sentence was given prior to reading but not when given after reading (Bransford & Johnson, 1972; Dooling & Lachman, 1971; Dooling & Mullet, 1973). Similarly, Ausubel (1963, 1960, 1968) has provided some evidence that expository learning may be enhanced by using an "advance organizer"--a short, expository introduction, presented prior to the text, containing no specific content from the text and providing a general framework for subsuming the information in the text. More recent reviews of the advance organizer literature reveal that advance organizers tend to have their strongest effects in situations where learners are unlikely to already possess useful prerequisite concepts--namely, for technical or unfamiliar material, for low ability subjects, and when the test involves transfer to new situations (Mayer, 1979a, 1979b).

Royer and his colleagues (Royer & Cable, 1975, 1976) have demonstrated that concrete models may serve as effective advance organizers in learning new scientific information. For example, the concrete analogy of electrical conduction as a chain of falling dominoes, influenced subsequent learning. Similarly, White & Mayer (1980) analyzed the concrete models used by physics textbooks. For example, Ohm's Law is described in terms of water flowing in pipes, a boy pushing a heavy load up an inclined street, or electron flow in a circuit. Recent results by Cook & Mayer (1980) show that when concrete analogies are embedded in a technical text, novices tend to perform best on recalling these familiar models and tend to recognize the information related to the models.

DuBoulay and his colleagues (DuBoulay & O'Shea, 1976; DuBoulay, O'Shea & Monk, 1980) have distinguished between two approaches to learning computer programming. In the <u>black box</u> approach, the operations of the computer are hidden to the learner so that the learner has no idea of what goes on inside the computer. In the <u>glass box</u> approach, the user is able to understand the changes that occur inside the computer for each statement. Although the description need not, indeed should not, be on a machine language level, DuBoulay et. al. (1980) suggest two properties for making the hidden operations of a computer language more clear to the novice: simplicity--there should be a "small number of parts that interact in ways that can be easily understood", and visibility--novices should be able to "view selected parts and processes of this notational machine in action". DuBoulay et. al. have implemented these suggestions in an instructional course in LOGO, since each statement is related to a concrete model called "the LOGO machine". However, there is yet no empirical test concerning the effects of the LOGO machine on learning.

Research of Concrete Models

Transfer. In order to provide some information concerning the role of models on learning computer programming, a series of studies was conducted (Mayer, 1975). In the studies, subjects were either given a concrete model of the computer (such as shown in Figure 1) or not; then, all subjects read a 10-page manual describing seven BASIC-like statements (see Table 1). Following reading, subjects took a test that consisted of six types of problems (as shown in Table 2). For generate problems, the subject had to write a program; for interpret problems, the subject had to describe what the program would do.

The proportion correct response by type of problem is given in the top of Table 3. As can be seen, the control gropu performs well on problems that are very much like the material in the instructional text, e.g., generate-statement and generate-nonloop. However, on problems that require moderate amounts of transfer--e.g., generate-loop and the shorter interpret problems; the model group excels. This difference in the pattern of performance suggests that models enhance transfer performance but not simple retention of presented material. Apparently, the model provided an assimilative context in which novices could relate new technical information in the booklet to a familiar analogy. This assimilative process resulted in a broader learning outcome that supported moderate transfer.

Locus of the effect. One problem with the above study is that the model subjects received more information than controls. Therefore, another series of studies was conducted (Mayer, 1976a) in which all subjects read the same BASIC-like manual, but some subjects were given a concrete model of the computer before reading while others were given the same model after reading the manual.

The proportion correct response by type of problem for the two groups is shown in the bottom of Table 3. As in the previous study, the before group excels on creative transfer to new situations but the after group excels on simple retention of the presented material. Thus, as predicted by assimilation theory, the model serves as an assimilative context for learning only if it is available to the learner at the time of learning.

Recall. The above studies used transfer tests as measures of what is learned under different instructional treatments. In a follow-up study (Mayer & Bromage, 1980), subjects read the manual and were given model either before or after the reading as in the previous study. However, as a test, subjects were asked to recall all they could about certain portions of the manual.

In order to score the protocols, the information in the manual was broken down into idea units. Each idea unit expressed one major idea or action. There were three kinds of idea units in the manual: (1) conceptual idea units related to the internal operation of the computer, (2) technical idea units gave examples of code, and (3) format idea units gave grammer rules. Table 4 gives examples of each type of idea unit.

Table 5 shows the average number of idea units recalled from each category by the two groups. As can be seen, the before group recalls more conceptual information while the after group excels on recall of technical and format information. This pattern is consistent with the idea good retention requires recall of specific code, but good transfer requires understanding of conceptua. ideas. Also, the before group included more intrusions about the model and other sections of the manual, suggesting they integrated the information more broadly.

Different language. Although the above results are consistent and were obtained in a long series of studies, their generality is limited by the fact that just one type of language was used. Thus, a follow-up study was conducted (Mayer, 1980a) in which subjects learned a file management language (Gould & Ascher, 1974) either with or without a concrete model. Table 6 lists the eight statements that were described in the instructional manual. Figure 3 shows the concrete model that was used: long-term memory is represented as a file cabinet; the sorting function is represented as an in-basket, out-basket and save basket; temporary memory is represented as an erasable scoreboard; executive control is represented as a list and pointer arrow; output is represented as a message pad. After instruction, all subjects took a transfer test including simple retention-like problems (sort-1)and problems that required putting all of the learned commands together in a novel way (compute-1 and 2). (See Table 7.)

Table 8 gives the proportion correct response by type of problem for the two treatment groups. As in the previous studies, the control group performs best on simple problems like those in the manual, but the model group excels on longer problems that require creative integration. Thus, previous results and conclusion seem to generalize to this new domain.

Ability. The pattern of result described above tended to be strongest for low ability subjects (Mayer, 1975) where ability is defined in terms of Mathematics SAT scores. Apparently, high ability learners already possessed their own useful "models" for thinking about how a computer works, but low ability students would be more likely to lack useful prerequisite knowledge.

Text organization. The pattern of results described above also tended to be strongest when material was poorly organized (Mayer, 1978). Apparently, the model is more useful when material is poorly structured because it helps the reader to hold the information together.

Evaluation. These results provide clear and consistent evidence that a concrete analogical model can have a strong effect on the encoding of new technical information in novices. These results provide empirical support to the claims of DuBoulay & O'Shea (1976, 1978) that allowing novices to "see the works" allows them to encode information in a more coherent and useful way. When appropriate models are used, the learner seems to be able to assimilate each new statement to his or her image of the computer system. Thus, one straightforward implication is: if your goal is to produce learners who will not need to use the language creatively, then no model is needed; if your goal is to produce learners who will be able to come up with creative solutions to novel (for them) problems, then a concrete model early in learning is quite useful. More research is needed in order to determine the specific effects of concrete models on what is learned, and to determine the characteristics of a useful model.

2. ENCOURAGE LEARNERS TO "PUT IT IN THEIR OWN WORDS"

Statement of the Problem

A second technique for increasing the meaningfulness of technical information is elaboration--encouraging the learner to explain the information in his or her own words and to verbally relate the material to other concepts or ideas. Elaboration techniques may influence meaningful learning because they encourage the activation of existing knowledge that is relevant for comprehending the presented material--i.e. elaboration may affect the activation process as indicated by the arrow c in Figure 1.

Example

For example, in our own research, we have taught subjects a simple file management language as described in the previous section (see Tables 6 and 7). In order to encourage subjects to elaborate on the material, we presented questions after each page of the instructional booklet. Table 9 gives examples of "model elaboration questions" which ask the learner to relate the material to a familiar context, and "comparative elaboration questions" which ask the learner to relate one part of the material to another.

Background

There is some evidence that asking subjects to put ideas into their own words during learning can enhance the breadth of learning. For example, Gagne & Smith (1962) found that subjects who were required to give a verbal rationalization for each move as they learned to solve a new problem resulted in longer learning time but better transfer performance than non-verbalizers. Results by Seidel & Hunter (1970) suggest that verbalization per se may not significantly enhance computer programming performance.

More recently, Wittrock (1974) has proposed the "generative hypothesis"--

i.e. learning occurs when the learner actively generates associations between what is presented and what he or she already has in memory. For example, when school children were asked to generate a one-sentence summary for each paragraph in a prose passage, Wittrock (1974) found that recall was nearly double that of a control group. Apparently, when students are encouraged to actively put information in their own words, they are able to better connect new information to existing knowledge.

Elaboration techniques have long been used to enhance learning of paired associates. For example, when students are asked to actively form images or sentences involving word pairs, paired associate recall is greatly enhanced (Bower, 1972; Paivio, 1969). More recently, elaboration techniques have been used in school curricula (see Dansereau, 1978; Weinstein, 1978). Several researchers have argued that students should be given explicit training in "learning strategies"--i.e. how to actively process new material (see O'Neill, 1978).

Transfer. In a typical study, (Mayer, 1980a) subjects read the instructional booklet covering a simple file management language, with some subjects having an elaboration page after each page in the booklet (model elaboration) and others not (control). A second study followed the same procedure but there was a comparative elaboration page after each page for half the subjects.

On a subsequent transfer test, using problems described in Table 7, the control groups performed well on simple retention-like problems but the elaboration groups (both model and comparative) perform better on problems requiring creative transfer. Table 10 shows the proportion correct by type of problem for four treatment groups. Thus, there is evidence that requiring the learners to put technical information in their own words through relating the material to a familiar situation or through making comparisons, results in broader learning.

Recall. In order to assess the generality of these findings, the studies were replicated using recall as the test (Mayer, 1980a). For scoring, the manual was divided into idea units. Some idea units described how the computer operated (conceptual idea units) and others emphasized the grammar and technical aspects of each statement (technical idea units). Table 11 shows the average number of idea units recalled by type for model elaboration, comparison elaboration, and control groups. As can be seen, the control group tends to recall equal amounts of both types of information, but the elaboration groups tend to emphasize recall of conceptual as compared to techni[illegible]l information. This attern is consistent with the idea that conceptual emphasis is likely to support transfer performance.

Notetaking. In order to provide further generality, an additional series of studies was conducted (Peper & Mayer, 1978) using a different language (a BASIC-like language) and a different elaboration activity (note-taking). Subjects watched a 20 minute videotape lecture describing seven BASIC-like statements similar to the manual described earlier. Some subjects were asked to take notes by putting the basic information in their own words. Others simply viewed the lecture without taking notes. As a test some subjects were given transfer problems and some were asked to recall portions of the lesson.

As in previous studies, there was a pattern in which note-taking improved performance on far transfer problems but not on simple retention problems. Similarly, there was a pattern in which note-takers performed better on recall of conceptual information but not technical information. These patters were observed for subjects scoring low in Mathematics SAT, but not for high ability subjects. Presumably high ability learners already possess strategies for putting new information into their own words.

Evaluation. Unfortunately, there is no fool-proof way to design elaboration activities. However, it is important to keep in mind that the goal of elaboration is to help the learner be able to describe the key concepts in his or her own words, using existing knowledge. Emphasis on format or grammatical details, and emphasis on errorless verbatim recall of statements will not produce the desired effects. The learner should be able to describe the effects of each statement in his or her own words.

3. ASSESS AND BUILD ON LEARNERS' INTUITIONS

Statement of the Problem

Learners come to the learning situation with certain existing expectations and intuitions about how to interact with computers. For example, since students have experience with conversations in English, they are likely to try to view computer conversations in the same way (Miller & Thomas, 1977; Sackman, 1970). Similarly, since most users are familiar with calculators, they may view interactions with computers in the same way (Mayer & Bayman, 1980; Young, 1980).

Example

For most users, calculators represent the first exposure to interacting with a computational machine. Thus, intuitions that are established may be important for later learning of computer programming languages. For example, consider the keystrokes:

7 + =

If subjects have a conception of incrementing internal registers, they might suppose that this sequence would result in 14 being displayed. However, less sophisticated intuitions might predict that the display would show 7 or 0.

Background

There is a growing interest in using words and logical structures that are similar to everyday English. For example, Ledgard, Whiteside, Singer & Seymour (1980) found that text editing systems that use "natural language" are easier to learn than those that use "computerese" for commands. Similarly, Shneiderman (1980) reports that meaningful or mnemonic variable names may affect programming performance. Finally, there is evidence that branching structures used in BASIC are not as intuitive or as easy to learn as other branching structures (Green, 1977; Mayer, 1976b; Sime, Green & Guest, 1977; Sime, Arblaster & Green, 1977).

More recently, Young (1980) has developed "mental models" of calculators--i.e. representations of the internal components that a learner needs to understand. Scandura, Lowerre & Veneski (1976) have interviewed children who learned to use calculators through "hands on experience". Many develop bizzarre intuitions even though they can use the calculator to solve routine problems. Thus, in order to build on the learners intuitions, and modify them as needed, one must assess what those intuitions are. In other words, the instructor should have techniques for determining the learner's "mental model".

Analysis of Users' Intuitions of Calculator Operations

A series of studies was conducted (Mayer & Bayman, 1980) in order to determine the intuitions that novice and expert users have concerning how pocket calculators operate. The novices were college students with no experience with computers or computer programming, while the experts were intermediate level computer science students. Each subject was given a 4-page questionnaire with 88 problems. Each problem listed a series of key presses and asked the student to predict what number would be in the display, assuming a standard four-function calculator was being used.

The subjects differed greatly with respect to when they thought an expression should be evaluated. For example, consider the problems,

2 + 3

2 + 3 +

2 + 3 + 7

2 + 3 + 7 =

Some subjects behaved as if an expression was evaluated only when an equals was pressed; thus, the answers were 3, 3, 7, 12. Others behaved as if an expression was evaluated as soon as an operator key was pressed, yielding answers of 3, 5, 7, 12. Finally, some subjects behaved as if an expression was evaluated as soon as a number was pressed, giving answers of 5, 5, 12, 12. Results indicated significant differences between experts and novices, with most experts opting for second approach while novices were fairly split among all three approaches. There also were important differences concerning how to evaluate a chain of arithmetic such as, 2 + 3 x 7 =, and how to handle non-standard sequences such as 2 + = + =.

<u>Evaluation</u>. We are just beginning to develop techniques for describing users' intuitions, e.g., users' mental models of computational machines. However, as techniques become available, teachers may use them to diagnose whether students have acquired useful intuitions, and to remediate where needed.

4. <u>PROVIDE TRAINING IN CHUNKING</u>

<u>Statement of the Problem</u>

One technique for making storage of information easier is to form meaningful chunks of schemas (Bransford, 1979). Within the context of computer programming, this means that learners should develop the ability to view a cluster of statements as a single unit that accomplishes some namable goal.

Example

For example, Atwood & Ramsey (1978) suggest that experienced programmers encode a segment such as,

```
SUM = 0
DO 1 I = 1, N
     SUM = SUM + (I)
1 CONTINUE
```

as "CALCULATE THE SUM OF ARRAY X".

Background

There is some evidence that experts and novices in a particular domain differ with respect to how they organize information in memory, with experts using more efficient chunking techniques (Larkin, McDermott, Simon & Simon, 1980). In recent reviews of research on how to teach people to become better problem solvers, Greeno (1980) and Simon (1980) conclude that good problem solving performance requires that the user has large amounts of domain specific knowledge organized into chunks. For example, Simon (1980) estimates that a person needs 50,000 chunks of domain specific knowledge (e.g., such as the example given above) to become an expert.

In a classic study, Chase & Simon (1973) asked subjects to view briefly presented chess board configurations and then try to reconstruct them. Chess masters performed better than less experienced players in reconstructing positions from actual games, but the advantage was lost when random board positions were presented. In an analogous study reported by Shneiderman (1980), experienced and inexperienced programmers were given programs to study. The experts remembered more than the novices when actual programs were presented but not for random lines of code. These findings suggest that experts have a large repertoire of many meaningful chunks, i.e. ways of grouping many lines of code into a single meaningful unit. More recently, Mayer (1979c, 1980b) has suggested that highly used chunks, such as looping structures, should be explicitly taught and labeled as part of instruction. For example, frequent looping structures in BASIC include "repeating a READ", "waiting for a data number", "waiting for a counter", and "branching down".

5. PROVIDE TRAINING IN ANALYSIS OF STATEMENTS

Statement of the Problem

What does it mean to "understand" a statement" In many psycholinguistic theories, comprehension involves relating a statement to its underlying case grammar (see Kintsch, 1974).

Example

In a previous paper (Mayer, 1979c), I have suggested a possible case grammar for BASIC. Each statement may be described as a list of transactions. A transaction consists of an action applied to some object at some location in the computer. For example, the statement, LET X = 5, consists of six transactions:

1. Find the number indicated on the right of the equals.
2. Find the number in the memory space indicated on the left of the equals.
3. Erase the number in that memory space.
4. Write the new number in that space.
5. Go on to the next statement.
6. Do what it says.

Background

An implication of the "transaction" approach is that the same statement names may actually refer to several different types of transactions. For example, we have shown that a counter set LET such as LET X = 5 is different from an arithmetic LET such as LET X = 10/2 (Mayer, 1979c, 1980c). Explicit naming and describing of different types of statements with the same keyword may become a useful part of computer literacy curricula. More recently this approach has been successfully applied to the analysis of commands in "calculator language" (Mayer & Bayman, 1980) and text editor languages (Card, Moran & Newell, 1980).

CONCLUSION

This paper has provided five tentative recommendations, listed in the introduction, for increasing the meaningfulness of computer concepts for novices. Reviews of cognitive research indicate that there is qualified support for the first two recommendations, and that active research is needed concerning the latter three recommendations.

Note

I wish to thank Bob Seidel and Ron Anderson for their useful comments on an earlier version of this manuscript. A more detailed version of this paper is available as a technical report from the author. Much of the work cited in this paper was supported by grant SED77-19875 from the National Science Foundation and grant NIE-G80-01[illegible] from the National Institute of Education.

Requests for reprints should be sent to: Richard E. Mayer, Department of Psychology, University of California, Santa Barbara, CA 93106.

References

Atwood, M. E. & Ramsey, H. R. Cognitive structure in the comprehension and memory of computer programs: An investigation of computer programming debugging. (ARI Technical Report TR-78-A210. Englewood, Colorado: Science Applications, Inc., August, 1978.

Ausubel, D. P. The use of advance organizers in the learning and retention of meaningful verbal material. Journal of Educational Psychology, 1960, 51, 267-272.

Ausubel, D. P. The psychology of meaningful verbal learning. New York: Gruene and Stratton, 1963.

Ausubel, D. P. Educational psychology: A cognitive view. New York: Holt Rinehart & Winston, 1968.

Bower, G. H. Mental imagery and associative learning. In L. Gregg (Ed.), Cognition in Learning and Memory. New York: Wiley, 1972.

Bransford, J. D. Human cognition. Monterey, CA: Wadsworth, 1979.

Bransford, J. D. & Johnson, M. K. Contextual prerequisites for understanding: Some investigations of comprehension and recall. Journal of Verbal Learning and Verbal Behavior, 1972, 11, 717-726.

Brownell, W. A. & Moser, H. E. Meaningful vs. mechanical learning: A study in grade III subtraction. In Duke University research studies in education, No. 8. Durham, N. C.: Duke University Press, 1949, 1-207.

Card, S. K. Moran, T. P., & Newell, A. Computer textediting: An information processing analysis of a routine cognitive skill. Cognitive Psychology, 1980, 12, 32-74.

Chase, W. G. & Simon, H. A. Perception in chess. Cognitive Psychology, 1973, 4, 55-81.

Cook. L. & Mayer, R. E. Effects of shadowing on prose comprehension and problem solving. Memory and Cognition, 1980, 8, in press.

Dansereau, D. The development of a learning strategies curriculum. In H. F. O'Neil (Ed.), Learning strategies. New York: Academic Press, 1978.

Dooling, D. J. & Lachman, R. Effects of comprehension on the retention of prose. Journal of Experimental Psychology, 1971, 88, 216-222.

Dooling, D. J. & Mullet, R. L. Locus of thematic effects on retention of prose. Journal of Experimental Psychology, 1973, 97, 404-406.

du Boulay, B. & O'Shea, T. How to work the LOGO machine. Edinburgh: Department of Artifical Intelligence, Paper No. 4, 1976,

du Boulay, B., O'Shea, T. & Monk, J. The black box inside the glass box: Presenting computering concepts to novices. Edinburgh: Department of Artificial Intelligence, Paper No. 133, 1980.

Gagne, R. M. & Smith, E. C. A study of the effects of verbalization on problem solving. Journal of Experimental Psychology, 1962, 63, 12-18.

Gould, J. D. & Ascher, R. M. Query by non-programmers. Paper presented at American Psychological Association, 1974.

Green, T. R. G. Conditional program statements and their comprehensibility to professional programmers. Journal of Occupational Psychology, 1977, 50, 93-109.

Green, T. R. G. & Arblaster, A. T. As you'd like it: Contributjons to easier computing. Shefdield, U. K.: MRC Social & Applied Psychology Unit, Memo No. 373, 1980.

Greeno, J. G. Trends in the theory of knowledge for problem solving. In D. T. Tuma & F. Reif (Eds.), Problem solving and education: Issues in teaching and research. Hillsdale, NJ: Erlbaum, 1980.

Holtzman, T. G. & Glaser, R. Developing computer literacy in children: Some observations and suggestions. Educational Technology, 1977, 17, No. 8, 5-11.

Kintsch, W. The representation of meaning in memory. Hillsdale, NJ: Erlbaum, 1974.

Larkin, J., McDermott, J., Simon, D. & Simon, H. A. Expert and novice performance in solving physics problems. Science, 1980, 208, 1335-1342.

Ledgard, H., Whiteside, J., Singer, A. & Seymour, W. The natural language of interactive systems. Communications of the ACM, 1980, 23, in press.

Mayer, R. E. Different problem-solving competencies established in learning computer programming with and without meaningful models. Journal of Educational Psychology, 1975, 67, 725-734.

Mayer, R. E. Some conditions of meaningful learning for computer programming: Advance organizers and subject control of frame order. Journal of Educational Psychology, 1976, 143-150. (a)

Mayer, R. E. Comprehension as affected by the structure of problem representation. Memory & Cognition, 1976, 4, 249-255. (b)

Mayer, R. E. Different rule systems for counting behavior acquired in meaningful and rote contexts of learning. Journal of Educational Psychology, 1977, 69, 537-546.

Mayer, R. E. Advance organizers that compensate for the organization of text. Journal of Educational Psychology, 1978, 70, 880-886.

Mayer, R. E. Can advance organizers influence meaningul learning? Review of Educational Research, 1979, 49, 371-383. (a)

Mayer, R. E. Twenty years of research on advance organizers: Assimilation theory is ~~still the~~ best predictor of results. Instructional Science, 1979, 8, 133-167. (b)

Mayer, R. E A psychology of learning BASIC. Communications of the ACM, 1979, 22, 589-594. (c)

Mayer, R. E. Elaboration techniques for technical text: An experimental test of the learning strategy hypothesis. Journal of Educational Psychology, 1980, 72, in press. (a)

Mayer, R. E. Ten statement spiral BASIC: From calculator to computer. Encino, CA: Glenco Publishing Co., 1980. (b)

Mayer, R. E. & Bayman, P. An information processing analysis of users' knowledge of electron calculators. Technical Report No. 80-1. University of California, Santa Barbara: Department of Psychology, 1980.

Mayer, R. E. & Bromage, B. Different recall protocols for technical text due to sequencing of advance organizers. Journal of Educational Psychology, 1980, 72, in press.

Mayer, R. E., Larkin, J. H. & Kadane, J. Analysis of the skill of solving equations. Paper presented at the Psychonomic Society, 1980.

Miller, L. A. & Thomas, J. C. Behavioral issues in the use of interactive systems. International Journal of Man-Machine Studies, 1977, 9, 509-536.

O'Neil, H. F. Learning strategies. New York: Academic Press, 1978.

Paivio, A. Mental imagery in associative learning and memory. Psychological Review, 1969, 76, 241-263.

Peper, R. J. & Mayer, R. E. Note taking as a generative activity. Journal of Educational Psychology, 1978, 70, 514-522.

Resnick, L. B. & Ford, S. The psychology of mathematics learning. Hillsdale, NJ: Erlbaum, 1980.

Royer, J. M. & Cable, G. W. Facilitated learning in connected discourse. Journal of Educational Psychology, 1975, 67, 116-123.

Royer, J. M. & Cable, G. W. Illustrations, analogies, and facilitative transfer in prose learning. Journal of Educational Psychology, 1976, 68, 205-209.

Sackman, H. Experimental analysis of man-computer problem-solving. Human Factors, 1970, 12, 187-201.

Scandura, A. M., Lowerre, G. F., Veneski, J. & Scandura, J. M. Using electronic calculators with elementary school children. Educational Technology, 1976, 16, No. 8, 14-18.

Seidel, R. J. & Hunter, H. G. The application of theoretical factors in teaching problem-solving by programmed instruction. International Review of Applied Psychology, 1970, 19, 41-81.

Shneiderman, B. Software psychology: Human factors in computer and information systems. New York: Winthrop, 1980.

Sime, M. E., Arblaster, A. A. & Green, T. R. G. Reducing programming errors in nested conditionals by prescribing a writing procedure. International Journal of Man-Machine Studies, 1979, 9, 119-126.

Sime, M. E. Green, T. R. B., & Guest, D. J. Scope marking in computer conditionals: A psychological evaluation. International Journal of Man-Machine Studies, 1977, 9, 107-118.

Simon, H. A. Problem solving and education. In D. T. Tuma & F. Reif (Eds.), Problem solving and education: Issues in teaching and research. Hillsdale, NJ: Erlbaum, 1980.

Spilich, G. J., Vesonder, G. T., Chiesi, H. L., & Voss, J. F. Text processing of domain-related information for individuals with high and low domain knowledge. Journal of Verbal Learning and Verbal Behavior, 1979, 18, 275-290.

Weaver, F., & Suydam, M. Meaningful instruction in mathematics education. Mathematics Education Reports, 1972.

Weinstein, C. Elaboration skills as a learning strategy. In H. F. O'Neil (Ed.), Learning strategies. New York: Academic Press, 1978.

Wertheimer, M. Productive thinking. New York: Harper & Row, 1978.

White, R. T. & Mayer, R. E. Understanding intellectual skills. Instruction Science, 1980, 9, 101-127.

Wittrock, M. C. Learning as a generative process. Educational Psychologist, 1974, 11, 87-95.

Young, R. M. Surrogates and mappings: Two kinds of conceptual models for pocket calculators. Paper presented at conference on Mental Models, La Jolla, California, 1980.

Table 1

Seven Statements Used in BASIC-like Instructional Booklet

Name	Example
READ	P1 READ (A1)
WRITE	P2 WRITE (A1)
EQUALS	P3 A1 = 88
CALCULATE	P4 A1 = A1 + 12
GOTO	P6 GO TO P1
IF	P5 IF (A1 = 100) GO TO P9
STOP	P9 STOP

Table 2

Examples of Six Types of Test Problems for a BASIC-like Language

Generation-Statement	Interpretation-Statement
Given a number in memory space A5, write a statement to change that number to zero.	A5 = 0

Generation-Nonloop	Interpretation-Nonloop
Given a card with a number on it is input, write a program to print out its square.	P1 READ (A1) P2 A1 = A1 * A1 P3 WRITE (A1) P4 STOP

Generation-Looping	Interpretation-Looping
Given a pile of data cards is input, write a program to print out each number and stop when it gets to card with 88 on it.	P1 READ (A1) P2 IF(A1 = 88) GO TO P5 P3 WRITE (A1) P4 GO TO P1 P5 STOP

Table 3

Proportion Correct on Transfer Test by Type of Problem for Model vs. Control Groups, and Before vs After Groups

	Generation			Interpretation		
	Statement	Nonloop	Looping	Statement	Nonloop	Looping
Model vs. Control						
Model	.63	.37	.30	.62	.62	.09
Control	.67	.52	.12	.42	.32	.12
Before vs. After						
Before	.57	.50	.20	.47	.63	.17
After	.77	.63	.13	.27	.40	.17

Note. For model vs. control, $n = 20$ per group; interaction between treatment and problem type, $p < .05$.

For before vs. after, $n = 20$ per group; interaction between treatment and problem type, $p < .05$.

Table 5

Average Number of Recalled Idea Units for the Before and After Groups

	Idea Units			Intrusions		
	Technical	Format	Conceptual	Inappropriate	Appropriate	Model
Before	5.0	1.9	6.6	1.5	1.3	3.1
After	6.0	2.9	4.9	2.5	.8	.5

Note. N = 30 per group; interaction between treatment and type of score, $p < .05$.

Table 8

Proportion Correct on Transfer Test for Model and Control Groups--

File Management Language

	Type of Test Problem				
	Sort-1	Sort-2	Count	Computer-1	Compute-2
Model	.66	.66	.63	.58	.45
Control	.63	.44	.43	.33	.22

Note. N = 20 per group; treatment x problem type interaction, p < .07.

Table 9

Example of the Elaboration Exercise in the Programming Text

Model Elaboration

Consider the following situation. An office clerk has an in-basket, a save basket, a discard basket, and a sorting area on the desk. The in-basket is full of records. Each one can be examined individually in the sorting area of the desk and then placed in either the save or discard basket. Describe the FOR statement in terms of what operations the clerk would perform using the in-basket, discard basket, save basket, and sorting area.

Comparative Elaboration

How is the FOR command like the FROM command?

How is the FOR command different than the FROM command?

Table 10

Proportion Correct on Transfer Test by Type of Problem for Model Elaboration vs. Control Groups and Comparative Elaboration vs. Control Groups

	Type of Test Problem				
Model vs. Control	Sort-1	Sort-2	Count	Compute-1	Computer-2
Model Elaboration	.65	.58	.64	.64	.45
Control	.66	.64	.41	.38	.27
Comparative Elaboration					
Comparative Elaboration	.90	.90	1.00	.75	.55
Control	.90	.90	.65	.65	.25

Note. For model elaboration vs. control, n = 20 per group; treatment x problem type interaction, p < .05. For comparative elaboration vs. control, n = 13 per group; treatment x problem type interaction, p < .05. Data is for interpretation problems only, for comparative and control groups.

Table 11

Average Number of Recalled Idea Units for Model Elaboration, Comparative Elaboration and Control Groups

	Type of Idea Units	
	Technical	Conceptual
Model Elaboration	5.3	13.9
Comparative Elaboration	9.4	14.1
Control	7.5	7.5

Note. N = 20 per group; treatment x type interaction, $p < .05$ for low ability subjects.

TECHNICAL REPORT SERIES IN LEARNING AND COGNITION

Report No.	Authors and Title
78-1	Peper, R. J. and Mayer, R. E. Note-taking as a Generative Activity. (Journal of Educational Psychology, 1978, 70, 514-522.)
78-2	Mayer, R. E. & Bromage, B. Different Recall Protocols for Technical Text due to Advance Organizers. (Journal of Educational Psychology, 1980, 72, 209-225.)
79-1	Mayer, R. E. Twenty Years of Research on Advance Organizers. (Instructional Science, 1979, 8, 133-167.)
79-2	Mayer, R. E. Analysis of a Simple Computer Programming Language: Transactions, Prestatements and Chunks. (Communications of the ACM, 1979, 22, 589-593.)
79-3	Mayer, R. E. Elaboration techniques for Technical Text: An Experimental Test of the Learning Strategy Hypothesis. (Journal of Educational Psychology, 1980, 72, in press.)
80-1	Mayer, R. E. Cognitive Psychology and Mathematical Problem Solving. (Proceedings of the 4th International Congress on Mathematical Education, 1980.)
80-2	Mayer, R. E. Different Solution Procedures for Algebra Word and Equation Problems.
80-3	Mayer, R. E. Schemas for Algebra Story Problems.
80-4	Mayer, R. E. & Bayman, P. Analysis of Users' Intuitions About the Operation of Electronic Calculators.
80-5	Mayer, R. E. Recall of Algebra Story Problems.
80-6	Bromage, B. K. & Mayer, R. E. Aspects of the Structure of Memory for Technical Text that Affect Problem Solving Performance.
80-7	Klatzky, R. L. and Martin, G L. Familiarity and Priming Effects in Picture Processing.
81-1	Mayer, R. E. Contributions of Cognitive Science and Related Research on Learning to the Design of Computer Literacy Curricula.
81-2	Mayer, R. E. Psychology of Computing Programming for Novices.
81-3	Mayer, R. E. Structural Analysis of Science Prose: Can We Increase Problem Solving Performance?
81-4	Mayer, R. E. What Have We Learned About Increasing the Meaningfulness of Science Prose?