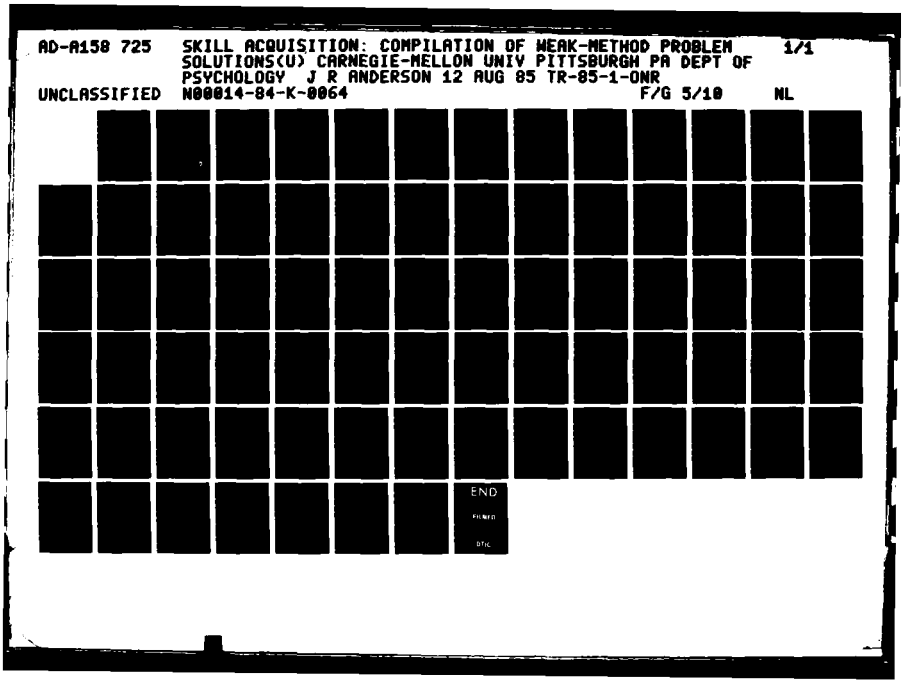
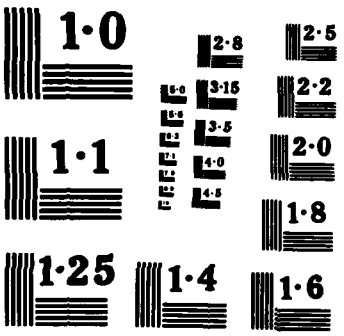

AD-A158 725
SKILL ACQUISITION: COMPILATION OF WEAK-METHOD PROBLEM SOLUTIONS(U) CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF PSYCHOLOGY J R ANDERSON 12 AUG 85 TR-85-1-ONR
1/1
UNCLASSIFIED
N00014-84-K-0064
F/G 5/10
NL
END
FILMED
DTIC



(2)

Skill Acquisition: Compilation of Weak-Method Problem Solutions[1]

John R. Anderson
Carnegie-Mellon University
Pittsburgh, PA 15213

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

ADA 158 725

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) Technical Report No. ONR-85-1	5. MONITORING ORGANIZATION REPORT NUMBER(S) Technical Report No. ONR-85-1
6a. NAME OF PERFORMING ORGANIZATION John R. Anderson 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Psychology Department Carnegie-Mellon University Pittsburgh, PA 15213	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Office of Naval Research 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-84-K-0064
8c. ADDRESS (City, State, and ZIP Code) Personnel and Training Research Programs Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO. NR 667-530

11. TITLE (Include Security Classification)
SKILL ACQUISITION: COMPILATION OF WEAK-METHOD PROBLEM SOLUTIONS (UNCLASSIFIED)

12. PERSONAL AUTHOR(S)
Anderson, John R.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Interim	FROM 8/84 TO 8/85	85/8/12	75

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	artificial intelligence, production system, cognitive science, LISP, skill acquisition,

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

Cognitive skills are encoded by a set of productions, which are organized according to a hierarchical goal structure. People solve problems in new domains by applying weak problem-solving procedures to declarative knowledge they have about this domain. From these initial problem solutions, production rules are compiled which are specific to that domain and that use of the knowledge. Numerous experimental results follow from this conception of skill organization and skill acquisition. These experiments include predictions about transfer among skills, differential improvement on problem types, effects of working memory limitations, and applications to instruction. The theory implies that all variety of skill acquisition, including that typically regarded as inductive, conforms to this characterization.

Keywords: -

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Susan Chipman	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

Abstract

Cognitive skills are encoded by a set of productions, which are organized according to a hierarchical goal structure. People solve problems in new domains by applying weak problem-solving procedures to declarative knowledge they have about this domain. From these initial problem solutions, production rules are compiled which are specific to that domain and that use of the knowledge. Numerous experimental results follow from this conception of skill organization and skill acquisition. These experiments include predictions about transfer among skills, differential improvement on problem types, effects of working memory limitations, and applications to instruction. The theory implies that all variety of skill acquisition, including that typically regarded as inductive, conforms to this characterization.

Research on the acquisition of cognitive skills has received a great deal of recent attention, both within the psychological and the artificial intelligence literature (Anderson, 1981, 1982, 1983; Brown & VanLehn, 1980; Carbonell, 1983; Chi, Glaser, & Farr, in press; Kieras & Bovair, 1985; Laird, Rosenbloom, & Newell, 1984; Langley, 1982; Larkin, McDermott, Simon, and Simon, 1980; Lesgold, 1984; Newell & Rosenbloom,1981; Van Lehn, 1983) . There are a number of factors motivating this surge of attention. First, there is increasing evidence that the structure of cognition changes from domain to domain and that behavior changes qualitatively as experience increases in a domain. This shows up both in comparisons between novices and experts within a domain (Adelson, 1981; Chase & Simon, 1973; Chi et al., in press; Jeffries, Turner, Polson, & Atwood, 1981; Larkin et al., 1980; Lesgold, 1984) and the failure of descriptions of cognition to be maintained across domains (Anderson, 1985, Ch. 9; Cheng, Holyoak, & Nisbett, in press) . For instance, expert problem-solving in physics involves reasoning forward to the goal, whereas problem-solving in programming involves reasoning backward from the goal. In developing a learning theory one is striving for the level of generality that will unify these diverse phenomena. The belief is that learning principles will show how the novice becomes the expert and how the structure of different problem domains is mapped into different behavior. Basically, the goal is to use a learning theory to account for differences in behavior by differences in experience.

A complementary observation to the first is that theories of cognition for a particular domain are not sufficiently constrained (Anderson, 1976, 1983). There are multiple theoretical frameworks for accounting for a particular performance and the performance itself does not allow for an adequate basis for choosing among the accounts. A learning theory places an enormous constraint on these theoretical accounts because the theoretical structure proposed to encode the domain knowledge underlying the performance must be capable of being acquired. This is like the long-standing claim that a learning theory would

provide important constraints on a linguistic theory (for recent efforts see McWhinney, in press; Pinker, 1984; Wexler and Culicover, 1980).

While there may be serious questions of the uniqueness of the theoretical accounts of cognition that have been advanced, there is little question that the accounts have become increasingly more sophisticated and cover much more complex phenomena. This success at creating accounts of various complex domain behaviors has emboldened the effort to strive for a learning account. Now that we have the theoretical machinery to account for complex cognition, we have frameworks in which the learning question can be addressed. This optimism has also been fueled by work, largely in artificial intelligence (Hayes-Roth & McDermott, 1976; Lewis, 1978; Michalski, 1983; Michalski, Carbonell, & Mitchell, 1983; Mitchell, 1982; Vere, 1977) studying mechanisms for knowledge acquisition. The psychological research on skill acquisition (Klahr, Langley, & Neches, somewhere) often takes the form of trying to apply these abstract concepts from AI to the theoretical mechanisms that have evolved to account for the realities of human cognition.

Another motivation for interest in skill acquisition is that it seems many of the fundamental issues in the study of human cognition turn on accounts of how new human competences are acquired. For instance, the faculty position, which claims that language and possibly other higher cognitive functions are based on special principles (Chomsky, 1980; Fodor, 1983), seems to rest on claims that different cognitive competencies are acquired in different manners. For instance, it is claimed that language acquisition depends on using "universals of language" to restrict the induction problem. As another example, when we compare "symbolic" (Newell & Rosenbloom, 1981; VanLehn, 1983) versus "neural" models of cognition (Rumelhart & Zipser, 1985; McClelland, 1985; Ackley, Hinton, & Sejnowski, 1985), we see that one of the few places where they make fundamentally different (as opposed to complementary) claims is with respect to how knowledge is

acquired. Many researchers in various theoretical camps are working on learning because they realize that a prerequisite to advancing their grander claims about the nature of cognition is working out the details about how knowledge is acquired in their framework and substantiating these details.

The ACT* theory of Skill Acquisition

The ACT* theory of skill acquisition (Anderson, 1982, 1983) was developed to meet a set of three constraints. First, the theory had to function within an existing theory of cognitive performance, the ACT* production system model of human cognition-- which had its own considerable independent motivation. Second, the theory had to meet sufficiency conditions, which is to say that it had to be capable of acquiring the full range of cognitive skills that humans acquire under the same circumstances that they acquire it. Third, it had to meet necessity conditions, which is to say that it had to be consistent with what was known about learning from various empirical studies.

At the time of development of the ACT* theory, the existing data base fell seriously short of testing the full range of complex predictions that could be derived from the learning theory. The theory applies to acquisition of elaborate skills over long time courses. For obvious reasons, there is a dearth of such research. Moreover, much of the existing research only reports gross descriptive statistics (e.g., whether "discovery" learning or "guided" learning is better), rather than analyses at the level of the detail addressed by the ACT theory. We had a few detailed protocol studies of skill acquisition over the course of tens of hours, but these studies can be criticized for the small sample size (often 1) and the subjectiveness of the data interpretation.

In response to this we have set out on a course of research looking at the acquisition of complex mathematical and technical skills taught in various sectors of our

society--eg., geometry, programming, and text-editing. The most ambitious line of research is the development of computer-based tutors for large-scale and objective data collection (Anderson, Boyle, Farrell, & Reiser, 1984; Anderson, Boyle, & Reiser, 1985). Most of the results from this research line have yet to come in. On a less ambitious scale, we have performed laboratory analysis of these skills and their acquisition. Most of the research to be reviewed in this paper comes from this second line of research.

This paper has three goals--one is to set forth some general claims about the course of skill acquisition. The second goal is to discuss a series of relatively counterintuitive predictions to be derived from the ACT* theory and to review the state of empirical evidence relevent to these predictions. The third goal is to report some revisions to that theory in response to both the empirical data and further theoretical considerations. However, before attempting any of these it is necessary to review the ACT* theory of skill acquisition and its empirical connection. I will do this with respect to the domain of writing LISP programs, where the theory has had its most extensive application (Anderson, Farrell, and Sauers, 1984) . The examples are deliberately chosen to use only the most basic concepts of LISP so that lack of prior knowledge of LISP will not be a barrier to understanding the points.

Acquisition of LISP programming skills

Cognitive processing in the ACT theory occurs as the result of the firing of productions. Productions are condition-action pairs which specify that if a certain state occurs in working memory, then particular mental (and possibly physical) actions should take place. Below are two "Englishified" versions of productions that are used in our simulation of LISP programming:

P1: IF the goal is to write a solution to a problem
and there is an example of a solution to a similar problem
THEN set a goal to map that template to the current case.

P2: IF the goal is to get the first element of List1
THEN write (CAR List1)

The first production, P1, is one of a number of productions for achieving structural analogy. It looks for a similar problem with a worked-out solution. If such an example problem exists, this production will fire and propose using this example as an analog for solving the current problem. This is a domain-general production and executes, for instance, when we use last-year's income tax forms as models for this year's income tax forms. In the domain of LISP it helps implement the very common strategy of using one LISP program as a model for creating another.

The second production rule, P2, above is one that is specific to LISP and recognizes the applicability of CAR, one of the most basic of LISP functions, which gets the first element of a list. For instance, (CAR '(a b c)) = a. One important question a learning theory has to address is how does a system, starting out with only domain-general productions like the first, acquire domain-specific productions like the second.[2]

To illustrate both how productions like the above are combined to solve a problem and how domain-specific productions are acquired from domain-general productions, it is useful to examine a subject's protocol when first learning to define new functions in LISP. Defining functions is the principle means for creating LISP programs. The subject (BR) had spent approximately five hours before this point practicing using existing functions in LISP and becoming familiar with variables and list structures.

BR had just finished reading the instruction in Winston and Horn (1981) on how to define functions. The only things she made reference to in trying to solve the problem were a template of what a function definition looked like and some examples in the text of what function definitions looked like. The template and an example of one of the functions

she looked at are given below. The function is called F-TO-C, and it converts a temperature in fahrenheit to centigrade:

```
(DEFUN <function name>
   (<parameter 1> <parameter 2> ... <parameter n>)
     <process description>)

(DEFUN F-TO-C (TEMP)
   (QUOTIENT (DIFFERENCE TEMP 32) 1.8))
```

The problem she was given was to write a LISP definition that would create a new function called FIRST which would return the first element of a list. As already noted, there is a function that does this, called CAR, which comes with LISP. Thus, she is creating a redundant function, and this problem is really just an exercise in the syntax of function definitions.

Anderson, Farrell, and Sauers (1984) report a simulation of BR's protocol that perfectly reproduces her major steps. The production set behind this simulation produces the goal structure, in Figure 1 which is useful for explaining BR's behavior. She starts out selecting the template as an analog for building a production. A set of domain-general productions for doing analogy then try to use this template. Subgoals are set to map each of major components of the template. Knowing "DEFUN" is a special LISP function she writes this first and then she writes "FIRST" which is the name of the function.

Insert Figure 1 about here

She has trouble in deciding how to map the structure "(<parameter 1><parameter 2>...<parameter n>)" because she has no idea what parameters are. However, she looks at the concrete example of F-TO-C and sees that there is just the argument to the function and correctly infers she should put a variable name that will be the argument to FIRST, which she chooses to call LIST1.

Then she turns to trying to map <process-description> which she again cannot understand but sees in an example that this is just the LISP code that calculates what the function is supposed to do. and so she writes CAR. which gets the first element of a list. We have seen above a LISP production that generates CAR. This is the only place in the original coding of the function where a LISP-specific production fires. We assume it was acquired from her earlier experience with LISP. To review. her code at this time is:

(DEFUN FIRST (LIST1) (CAR

The major hurdle still remains in this problem. to write the argument to the function. She again looks to the example for guidance about how to code the argument to function within a function definition. She notes that the first argument in an example like F-TO-C is contained in parentheses--(DIFFERENCE TEMP 32). This is because the argument is itself a function call and function calls must be placed in parentheses. She does not need parentheses in defining FIRST where she simply wants to provide the variable LIST1 as the argument. However. she does not recognize the distinction between her situation and the example. She places her argument in parentheses producing a complete definition:

(DEFUN FIRST (LIST1) (CAR (LIST1)))

When the argument to a function is a list. LISP attempts to treat the first thing inside the list as a function definition. Therefore when she tests FIRST she gets the the error message "Undefined function object: LIST1". In the past she had corrected the error by quoting the offending list. So she produces the patch.

(DEFUN FIRST (LIST1) (CAR '(LIST1)))[3]

When she tests this function out on a list like (A B C) she got the answer LIST1 rather than A because LISP now returns the first item of the literal list (LIST1). Eventually

after some work she finally produces the correct code which is:

(DEFUN FIRST (LIST1) (CAR LIST1))

We have observed 38 subjects solve this problem in the LISP tutor (Anderson & Reiser, 1985) and a number of further subjects in informal experiments. BR's solution is typical of novice problem-solving in many ways. The two places where she has difficulty, specifying the function parameters and specifying the argument to CAR, are the two major points of difficulty for our LISP tutor subjects. Her error made in specifying the argument to CAR is made by over half of the tutor subjects. An interesting informal observation that we have made is that people with no background at all in LISP (i.e., not working with the tutor), given information about what CAR does, given the function definition template, and given the F-TO-C example, tend also to solve the problem in the same way as BR and tend to make the same first error. Thus, it seems that much of the problem-solving is controlled by analogy and not by detailed understanding of LISP. The success of our simulation also suggests that this problem-solving by analogy can be well modelled by a production system with hierarchical goal structure.

Knowledge Compilation

Our simulation of BR after producing the solution to this problem created two new productions which summarized much of the solution process. It does this by a knowledge compilation process (described in Anderson, 1982, 1983) that collapses sequences of productions into single productions that have the same effect as the sequence. Typically, as in this case, it converts problem-solving by domain-general productions into domain-specific productions. The two productions acquired by ACT are given below:

C1: IF the goal is to write a function of one variable
THEN write (DEFUN function (variable)
and set as a subgoal to code the relation calculated
by this function
and then write).

C2: IF the goal is to code an argument
and that argument corresponds to a variable of the function
THEN write the variable name.

Figure 1 indicates the set of productions that were collapsed to produce each of the productions above. The first production summarizes the analogy process that went into figuring out the syntax of a function definition, and we now have a production which directly produces that syntax without reference to the analog. The second production summarizes the search that went into finding the correct argument to the function and directly produces that argument.

We gave our simulation another LISP problem to solve, armed with these two additional productions. This was to write a function called SECOND which retrieved the second element of a list. Although SECOND is a more complex function definition, both the simulation and our subject produced very much more rapid and successful solutions to this problem. The speed up in the simulation was due to the fact that fewer productions were involved thanks to the compiled productions.

One feature of this knowledge compilation process is that it predicts a marked improvement as we go from a first to a second problem of the same kind. Elsewhere (Anderson, 1982), we have commented upon this marked speed-up in the domain of geometry proof generation. We have found the same thing in studies with the LISP tutor where students first code FIRST and then SECOND. We find that errors in the function definition syntax (compiled into C1 above) drops from a median of 2 in FIRST to a median of 0 errors in SECOND, and the time to type in the function template type drops from a median of 237 seconds to a median of 96 seconds. Success at specifying the variable argument (compiled into C2 above) changes from a median of 3 errors to a median of 0 errors; time to enter this argument successfully drops from a median of 96 seconds to a

median of 26 seconds. By any measure these are extremely impressive one-trial learning statistics.

Important Features of the ACT theory

This simulation of BR illustrates a number of important features of the ACT* theory that will serve as the basis for the predictions and empirical tests to be reported in the next section of the paper and the theoretical analyses offered in the last section of the paper. These features are:

1. Productions as the Units of Procedural Knowledge. The major presupposition of the entire ACT theory and certainly key to the theory of skill acquisition is the idea that productions form the units of knowledge. Productions define the steps in which a problem is solved and are the units in which knowledge is acquired.

2. Hierarchical Goal Structure. The ACT production system specifies a hierarchical goal structure that organizes the problem solving. Such a hierarchical goal structure is illustrated in Figure 1. This goal structure is an additional control construct that was not found in many of the original production system models (Anderson, 1976; Newell, 1973) but now is becoming quite popular (e.g., Brown & VanLehn, 1980; Laird, et al., 1984). It has proven impossible to develop satisfactory cognitive models that do not have some overall sense of direction in their behavior. As can be seen with respect to this example, the hierarchical goal structure closely reflects the hierarchical structure of the problem. More important than their rule in controlling behavior, goals are important to structuring the learning by knowledge compilation. They serve to indicate which parts of the problem solution belong together and can be compiled into a new production.

3. Initial Use of Weak Methods. This simulation nicely illustrates the critical role that analogy to examples plays in getting initial performance off the ground. There is a serious

question about how, before a student has any productions specific to performing a task, the student can do the task. We see in this example that the student can in fact mimic the structure of a previous solution. However, this simulation also shows that, in contradiction to frequent characterizations of imitation as mindless (e.g., Fodor, Bever, & Garrett, 1974), the analogy mechanisms that implement this process of imitation can be quite sophisticated. For other discussions fo use of analogy in problem-solving see Kling (1971), Carbonell (1983), and Winston (1979).

While analogy is a very frequent method for getting started in problem-solving, we do not think that it is the only way. It is an instance of what are referred to as weak problem-solving methods (Newell, 1969) , which have as their characteristic that they are general and can apply in any domain. They are called weak because they do not take advantage of domain characteristics. Other examples of weak problem-solving methods include means-ends analysis and pure forward search.

4. Knowledge Compilation. All knowledge in the ACT theory starts out in declarative form and must be converted to procedural (production) form. This declarative knowledge can be encodings of examples of instructions, of general properties of objects, etc. The weak problem-solving methods are the ones that can apply to the knowledge while it is in declarative form and interpret its implications for performance. The actual form of the declarative knowledge determines the weak method adopted. For instance, in the simulation of BR analogy was used because the declarative knowledge was almost exclusively knowledge about the template, example programs, and knowledge about the symbols used therein. In a geometry simulation discussed in Anderson (1982) the declarative knowledge largely came in the form of rules (e.g., "A reason for a statement can either be that it is given, or that it can be derived by means of a definition, postulate, or theorem"). This tended to evoke a working backwards problem-solving method.

When ACT solves a problem it produces a hierarchical problem-solution generated by productions. Knowledge compilation is the process that creates efficient domain-specific productions from this trace of the problem-solving episode. The goal structure is critical to the knowledge compilation process in that it indicates which steps of the original solution belong together. The Appendix to this paper contains a more technical specification of knowledge compilation and the role of goals in this process.

Outlined above is a complete theory of how new skills are acquired: Knowledge comes in declarative form, is used by weak methods to generate problem-solutions, and the knowledge compilation process forms new productions. The key step is the knowledge compilation process, which produces the domain-specific skill.

While the four points above are the key features to understand the origin of new skills, they do not address the question of how well the knowledge underlying the skill will be translated into performance. There are two factors determining the success of execution of productions. These might be viewed as performance factors that modulate the manifestation of learning except, as we will see, these factors can also be improved with learning.

5. Strength. The strength of a production determines how rapidly it applies, and production rules accumulate strength as they successfully apply. While accumulation of strength is a very simple learning mechanism, there is good reason to believe that strengthening is often what determines the rate of skill development in the limit. The ACT strengthening mechanism predicts the typical power-function shape of speed-up in skill performance (Anderson, 1982).

6. Working Memory Limitations. In the ACT theory there are two reasons why errors are made: The productions are wrong, or the information in working memory on which they

operate is wrong. However, the important observation in terms of performance limitations is that perfect production sets can display errors due to working memory failures. In fact, the only way "slips" can occur in the ACT theory is because critical information is lost from working memory and consequently the wrong production fires or the right production fires but produces the wrong result. Just as learning impacts on production strength, it seems learning impacts on working memory errors. Working memory capacity for a domain can increase (Chase & Ericsson, 1982) , reducing the number of such errors with expertise.

The six features reviewed above lead to a number of interesting consequences when we begin to analyze how they interact within the framework of the ACT theory. The remainder of this paper is devoted to exploring these consequences.

Transfer

The commitment to productions as the units of procedural knowledge has some interesting empirical consequences. In particular it leads to some strong predictions about the nature of transfer between skills. Specifically, the prediction is that there will be positive transfer between skills to the extent that the two skills involve the same productions. This is a variation of Thorndike's (1903) identical elements theory of transfer. Thorndike argued that there would be transfer between two skills to the extent that they involved the same content. Thorndike was a little vague on exactly what was meant by content, but he has been interpreted to mean something like stimulus-response pairs (Orata, 1928). The ACT theory offers a more abstract concept, the production, to replace the more concrete concept, the stimulus-response pair. In ACT the abstraction comes in two ways. First, the productions can be variablized and hence not refer to specific elements. Second, there is a hierarchical goal structure controlling behavior, and many of the productions are concerned with generating this goal structure rather than specific behavior.

Unfortunately, there is a serious problem in putting the prediction about production-based transfer to test because it depends on agreement as to what the productions are that underlie two tasks. There is always the danger of fashioning the production system models to fit the observed degree of transfer. Therefore, it is important to select tasks where there is already independent evidence as to the appropriate production system analysis. Singley and Anderson (in press) chose text editing because there already existed production-system models or production-system-like models of this task (Card, Moran, and Newell, 1983; Kieras & Polson, in press). While these are not ACT production systems, they serve to nearly completely constrain how one would produce ACT production system models for these tasks.

Using the Card, Moran, and Newell model as a guideline, the first production in performing a text edit is:

E0: IF the goal is to edit a manuscript
THEN set as subgoals
1. To characterize the next edit to perform
2. To perform the edit

The first subgoal sets up the process of scanning for the next thing to be changed and encoding that change. The assumption is that this process should be common to all editors. The actual performance of the edit can vary from editor to editor. For instance, in the line editor ED, the following productions would fire among others to replace one word by another:

E1: IF the goal is to perform an edit
THEN set as subgoals
1. To locate the line
2. To type the edit

E2: IF the goal is to locate the target line
and the current line is not the target line
THEN increment the line

E3: IF the goal is to locate the target line
and the current line is the target line
THEN POP success

E4: IF the goal is to type an edit

THEN set as subgoals
1. To choose the arguments
2. To type the command

E5: IF the goal is to type the command
THEN set as subgoals
1. To type the command name
2. To type the arguments

E6: IF the goal is to type the command name for substitution
THEN type S

Figure 2 illustrates the goal structure of these productions. The triangles in Figure 2 indicate goals that would be achieved by productions other than those not specified above.

Insert Figure 2 about here

We studied transfer to two other editors--EDT and EMACS. EDT has different command names than ED and, in the subset of EDT that we used, subjects had to achieve the goal of locating a line by specifying a search string, rather than incrementing the line number. Otherwise, the goal structure is identical for ED and EDT. EMACS, on the other hand, is identical only in the top-level production and the process of characterizing what needs to be done. It is a screen editor rather than a line editor and so has very different means of making edits. Figure 2 illustrates where the two editors are similar and where they are different. From this analysis we made the following two predictions about transfer among editors based on the production overlap:

1. There should be large positive transfer between ED and EDT. The failure of transfer should be localized mainly to the components associated with locating lines where the two differ substantially in terms of productions. This prediction of high positive transfer holds despite the fact that the actual stream of characters typed is quite different in the two editors. What is important is that the two editors have nearly identical goal structures. So it violates Thorndike's surface interpretation of the identical elements principle.

2. There should be little positive transfer from either ED or EDT to EMACS despite the fact that they are both text editors. Moreover, what positive transfer there is should be largely confined to the process of characterizing the edit rather than executing it.

The experiment had five experimental groups. One practiced EMACS for six days. This will be referred to as the one-editor group. A second group practiced ED for four days and then learned and practiced EMACS for the last two. A third group practiced EDT for the first four days and then transfered to EMACS for the last two days. Groups 2 and 3 above will be referred to as the two-editor groups. A fourth group practiced ED for two days, EDT for two more, and EMACS for the last two. A fifth group practiced EDT for two days, ED for two more, and EMACS for the last two. These groups will be called the three-editor groups. Note that all groups worked in EMACS for the last two days of the experiment.

The results are presented in Figure 3 in terms of time to perform an edit. The first, uninteresting, thing to note is that EMACS is a good deal more efficient than the line editors--that is, the one-editor group is much faster at the beginning when they are using EMACS than the other groups who are using a line editor. The second, significant, observation is that there is virtually no difference between the two and three editor groups. In particular, on Day 3 when the three editor subjects are transfering to a new editor they are almost as fast as the two editor subjects who are using that editor for the third day. Thus, we have almost total positive transfer as predicted. Moreover, when we examined the microstructure of the times, we found that the only place the three-editor subjects were at a significant disadvantage to the two-editor subjects was in line location, as predicted.

Insert Figure 3 about here

In contrast, when any of the line-editor groups transfers to EMACS on Day 5, they

are at a considerable disadvantage to subjects who have been practicing EMACS all along. They are faster than the EMACS-only subjects were on Day 1. We would attribute this advantage to subjects' ability to characterize the edits they have to perform. Consistent with this theory, we found nearly as much transfer to EMACS in a control group who spent four days simply retyping an edited text. Presumably, this group was also learning how to interpret edits marked on the page. Thus, we see that if we take a production system analysis quite seriously, we are able to apply an identical element metric to predict transfer. Singley and Anderson (in preparation) should be consulted for a very fine-grained analysis of the data from this experiment in terms of a production system analysis. There is reason to believe that production system analyses may prove to be generally useful in predicting transfer. Quite independently, Kieras & Bovair (1985) and Polson & Kieras (1985) have found similar success using a production system analysis to predict transfer among a different set of skills.

Negative transfer among cognitive skills?

There is a peculiar consequence of the identical elements model which is that it does not directly predict any negative transfer among skills in the sense of one procedure running less effectively because another has been learned. The worst possible case is that there is zero overlap among the productions that underlie a skill in which case there will be no transfer, positive or negative. It is possible to get negative transfer defined in terms of overall measure like total time or success if a procedure optimal in one situation is transferred to another domain where it is not optimal, as in the Einstellung phenomenon (Luchins & Luchins, 1959). However, in our analyses of the Einstellung effect we would assume perfect transfer of the production set. It is a case of perfect transfer of productions leading to very non-optimal performance, not a case of the productions firing more slowly or in a more errorful way.

Interference or negative transfer is quite common with declarative knowledge, but it can be quite a bit more difficult to get in the domain of skills (Anderson, 1985, Ch. 9), suggesting a basic difference between procedural and declarative knowledge. Singley and I decided to create in the text editor domain what would be a classic interference design in the verbal learning domain. We created an editor called Perverse EMACS, which was just like EMACS except that the assignment of keys to function was essentially permuted. So, for instance, in EMACS control-D erases a letter, escape-D a word, and control-N goes down a line while in Perverse EMACS control-D went down a line, while escape-R erased a letter. If we assume that the functionalities are the stimuli and the keys the responses we have an A-B,A-Br interference paradigm in paired-associate terms (Postman, 1971).

We compared two groups of subjects. One spent six days with EMACS while the other spent their first two days with EMACS, then the next two days with Perverse EMACS, and the final two days with EMACS again. Figure 4 shows the results of this experiment. Throughout the experiment, including on the first two days where both groups of subjects are learning EMACS, the Perverse EMACS transfer subjects are slightly worse. However, the only day they are significantly worse is on the third day when they transfer to Perverse EMACS. That difference largely disappears by the fourth day when they are still working with Perverse EMACS. Compared to Day 1 on EMACS, there is large postive transfer on Day 3 to Perverse EMACS, reflecting the production overlap. The difference between the two groups on Day 3 reflects the cost of learning the specific rules of Perverse EMACS. When the transfer subjects went back to EMACS on Day 5 they picked up on the same point on the learning curve as had the subjects who had stayed with EMACS all through. (While they are slower than the pure EMACS subjects in Days 5 and 6, they are no more slow than they were on Day 2 when they last used EMACS.) This is because they had been practicing in Perverse EMACS largely the same productions as they would be using in

regular EMACS.

Insert Figure 4 about here

In conclusion, this research on text editors supports the claim that production systems should be viewed seriously as analyses of procedural knowledge. We see that transfer can be predicted by measuring production overlap (for actual detailed measures see Singley & Anderson, in preparation). We also see that procedural knowledge behaves differently than declarative knowledge in that it does not seem to be subject to principles of interference.

Effects of Knowledge Compilation

There are a number of predictions that follow from the knowledge compilation process, which is the basic production learning mechanism in the ACT* theory. This process collapses sequences of productions into single operators. If one repeatedly presents problems that require the same combination of productions, the subject should compose productions to reflect that combination and strengthen that production. This implies that frequently co-occurring combinations of operations should gain an advantage over less frequently co-occurring combinations of the same operations.

Table 1 illustrates part of an Experiment by McKendree and Anderson (in press) that was designed to test this prediction. We were looking at subjects' ability to evaluate various combinations of LISP expressions. For instance, subjects had to evaluate (CAR (CDR ((A B)(C D)(E F)))). CDR returns all but the first element of a list: so (CDR ((A B)(C D)(E F))) is ((C D)(E F)). The function CAR will return the first element of this or (C D) which is the correct answer to the original problem. We did the experiment to see if subjects would learn operators to evaluate combinations of functions such as this CAR-CDR combination.

Insert Table 1 about here

We had subjects study various combinations of functions with differential frequency. Table 1 gives how frequently subjects would see combinations of each function each day. (It is the actual function combination that was seen with the given frequency: we had subjects apply that function combination to new arguments each time.) Some function combinations were seen three times as frequently as were other function combinations. As indicated in Table 1 we counterbalanced over two groups which combinations were the high and which were the low frequency combinations. While the frequency of combination varies between the two groups, the actual frequency of individual functions does not. Thus, if we see an advantage for a combination, we will know that the advantage is in fact for that combination not the individual functions.

The knowledge compilation process predicts a speed advantage in evaluating high-frequency combinations because of two factors. First, the compiled productions for the high-frequency combinations are likely to be learned sooner. Second, once learned they will acquire strength more rapidly because of their greater frequency.

The experiment involved presenting students with these function combinations, combinations of other pairs of LISP functions that similarly varied in frequency, problems that required that subjects just evaluate single LISP functions, and problems that required that subjects evaluate triples of functions. Subjects were given four days of practice in which the frequencies of the pairs were maintained as in Table 1. We measured subjects' time to type solutions into the computer for these problems. Averaged over the four days subjects took 6.55 seconds to evaluate the high frequency combinations and 8.61 seconds for the low frequency combinations. This difference is predicted by the assumptions that subjects are compiling productions to reflect various function combinations and are strengthening these compiled productions according to the frequency of their applicability.

This also shows that the development of skill is highly specific to its use. That is, even though the subject may be good at evaluating the combinations car-cdr and cdr-car, the subject may be poor at evaluating the combinations car-car and cdr-cdr, despite the fact that these evaluations involve the same primitive knowledge as the first two.

Use-Specificity of Knowledge

Declarative knowledge is flexible and not committed to how it will be used. However, often knowledge compilation will derive productions from that declarative knowledge which can only be used in certain ways. Often the production sets underlying different uses of the same knowledge can be quite different. For instance, productions for language comprehension would be quite different from productions for language generation.

There is no reason to predict transfer between different uses of the same knowledge. Neves and Anderson (1981), for instance, compared subjects' ability to generate a proof in a logic-like system with their ability to give the reasons that justified the steps of a worked out proof. We were interested in this comparison because even though these two tasks make use of the same logical postulates, there is no overlap in the production system that implements one skill versus another. Generating a proof involves search for a path between givens of the problem and the statement to be proven, while reason-giving involves a search through the rule space looking for a rule whose consequence matches the statement to be justified. Neves and I found evidence that 10 days of practice at reason giving had little positive transfer to to proof generation. However, that evidence was based on a very small sample of subjects and informal evaluation of their proof generation skills.

Mckendree and Anderson looked at this issue more thoroughly in their study of LISP evaluation skills. To review, that experiment was concerned with having subjects evaluate the results of LISP functions applied to arguments. A typical problem was

(LIST (CAR '(A B)) '(B C)) = ?

where the correct answer is (A (B C)). In addition to massive practice on these evaluation problems, on the first and fourth day we gave subjects a few generation problems that were isomorphic to these evaluation problems. For the example above, the isomorphic generation problem would require subjects would be asked to write a LISP expression that will operate on (A B) and (B C) and produce (A (B C)), for which the expression above would be the correct answer.

There is again no overlap between productions that solve the evaluation problems and productions that solve the generation problem, although again these productions are based on the same abstract declarative knowledge. However, that declarative knowledge has different production form when it has been compiled for a particular use. For instance, consider the abstract fact that the LISP function LIST inserts its argument in a list. Used in evaluation form this is encoded by the production:

IF the goal is to evaluate (LIST X Y)
and A is the value of X
and B is the value of Y
THEN (A B) is the value

Used in generation form this is encoded by the production (actually taken from our LISP tutor):

IF the goal is to code (A B)
THEN use the function LIST and set as subgoals
1. To code A
2. To code B

This is basically the same knowledge in different form, but the ACT claim is that the difference in form is significant, and there will not be transfer between the two forms. Most people find this prediction quite counter-intuitive. Standard educational practice (at least at the college level) seems to believe that we should get students to understand general principles and how to apply them in one domain. The belief is that if they have this knowledge, they will know how to apply the principles in any domain.

The McKendree and Anderson experiment allowed us to address the issue of whether massive practice on evaluation would generalize to generation. Averaging over single, pair, and triple function evaluations, subjects evaluation time dropped from 18.2 sec on Day 1 to 10.0 seconds on Day 4. Subjects' accuracy increased from 58% correct to 85% correct over the four days that they were practicing evaluation. In contrast, they only changed from 71% correct in the first generation test at the end of the first day to 74% correct in the second generation test at the end of the fourth day. (We don't have reaction time measures for the generation task because subjects had to work out the solutions by hand rather than typing them into a computer.) Thus, it seems that there is little transfer from evaluation to generation--two different skills that use the same knowledge. This reinforces the idea that production representation captures significant features of our procedural knowledge and that differences between production forms are psychologically real.

I think in combination with the earlier mentioned results on transfer, a rather startling result is emerging. We can get total transfer across tasks if the same knowledge is used in the same way (e.g., transfer between text editors ED and EDT). On the other hand, we get no transfer at all if the same knowledge is used in different ways (e.g., transfer between evaluation and generation).

Effects of Proceduralization

Part of the compilation process is the elimination of the need to hold declarative information in working memory for interpretation; rather, that information is built into the proceduralized production. There are a number of predictions about qualitative changes in skill execution which follow from proceduralization. Many of these are discussed in Anderson (1982). These changes include disappearance of verbalization of the declarative knowledge and decrease in errors due to working memory forgetting.

Part of the Singley and Anderson experiment was a very specific test of the qualitative changes implied by proceduralization. Before proceduralization there should be declarative interference between performing a skill and remembering similar declarative knowledge, but this content-based interference should disappear when the knowledge is proceduralized. In a dual-task experiment Singley and Anderson required subjects to memorize facts being presented to them in the auditory modality and to perform text editing operations involving the visual modality. We varied the content of the information they were being asked to remember to either be the description of a hypothetical text editor or to be unrelated to text editing. The hypothetical text editing information started out poorly but improved with practice. Memory for the unrelated information did not start out so poorly but did not improve. This is consistent with the view that subjects were initially suffering declarative interference with the text editor information from the execution of the text editor skill, which disappeared when the skill became proceduralized with practice.

Working Memory Limitations

As we noted earlier, there is reason to believe that in domains such as writing computer programs, the major source of errors will be working-memory failures. The student gets good feedback on the facts about the programming language, and so we would expect misconceptions to be rapidly stamped out. On the other hand, the student is being asked to coordinate a great deal of knowledge in writing a program, and one would expect working memory capacity to be frequently overwhelmed.

Anderson & Jeffries (in press) looked at the errors made by over a hundred students in introductory LISP classes. We found that over 30% of all answers were errors. The results we obtained were consistent with the working memory failure analysis. Increasing the complexity of one part of the problem increased errors in another part, suggesting that the capacity requirements to represent one part overflowed and impacted on the representation

of another part. Further, errors were non-systematic: that is, subjects did not repeat errors as one would expect if there were some systematic misconception.

Interestingly, the best predictor of individual subject differences in errors on problems that involved one LISP concept was number of errors on other problems that involved different concepts. This in turn was correlated with amount of programming experience and experience with LISP. It seems that subjects differ in their general proneness to these working memory errors in a domain and that experience in the domain increases working memory capacity. Jeffries, Turner, Polson, and Atwood (1981) found that memory capacity was a major difference separating beginning programmers from experienced programmers. Chase and Ericsson (1982) showed that experience in a domain can increase capacity for that domain. Their analysis implied that what was happening is that storage of new information in long-term memory became so reliable that long-term became an effective extension of short-term memory.

Consequences of Working Memory Failures

Loss of declarative information from working memory can cause good procedures to behave badly. This can happen in a number of ways. Most simply, if one looses information that is needed in the answer, the answer will be missing this information. Interestingly, Anderson and Jeffries found that the most common error in calculating LISP answers is to drop parentheses. This error would occur if a parentheses were lost from the representation of the problem they were manipulating in working memory.

This was more than twice as frequent as adding parentheses. A slightly more profound type of working memory error occurs when students loose a goal they are trying to achieve. Katz and Anderson (in preparation) found that a major error in syntactically correct programs was the ommission of parts of code that correspond to goals. Often subjects

mention these goals in protocols taken while writing the programs, but they just never get around to achieving them.

A third kind of error that can occur because of working memory loss is that subjects will loose a feature that discriminates between two productions and so fire the wrong production. A common example of this is in geometry where students will apply a postulate involving congruence of segments to information about equality of measure of segments. Students when queried can explain why they are wrong but there is just a momentary intrusion of the postulate. An example in the context of LISP occurs is the confusion of similar LISP functions in programs. Again subjects can explain their mistake when queried, but it seems that the production generating the wrong function intruded. Anderson and Jeffries have shown that the frequency of these intrusions can be increased by increasing the concurrent memory load on subjects. This is consistent with the view that discriminating information is being crowded out of working memory.

Norman (1981) has provided a classification of what he calls action slips, which appear to be identical in reference to our working memory errors. However, he interprets these with respect to an activation-based schema system rather than a production system. With respect to cognitive-level errors such as occur in LISP programming, there has hardly been adequate research to separate the two views. However, evidence that such errors increase with working memory load is certainly consistent with the working memory plus production system hypothesis.

Immediate Feedback

The importance of immediate feedback is an interesting consequence of the interaction between compilation and working memory limitations. For knowledge compilation to work correctly with delayed feedback, it is necessary to preserve information about the

decision point until information is obtained about the right decision. Then an operator can be compiled, attaching the correct action to the critical information at the decision point. Working memory failures will cause information about the decision point to be lost and consequently incorrect operators will be formed. Such failures will increase with the delay of the feedback. Thus, the commonly recommended practice of having subjects discover their errors may have a negative impact on learning rate.

Lewis & Anderson (1985) performed an experiment to see whether it was better to have subjects discover errors or to inform them immediately that an error had been made. We had subjects learn to solve dungeon-quest-like games. Figure 5 shows a typical situation that a subject might face. A room is described with certain features and the subject can try certain operators (like waving the wand, slaying the roomkeeper) which will move the subject on to other rooms. A subject might move to a room which leads to a deadend. In the immediate feedback condition subjects were immediately told they had entered such a room while in the delay condition they had to discover by further exploration that they had reached a deadend.

Insert Figure 5 about here

Lewis and Anderson found that subjects performed considerably better, in terms of number of correct moves, given immediate feedback. Other researchers (R. C. Anderson, Kulhavy, & Andre, 1972) have found that the type of feedback is critical to determining its effectiveness. It is important that the feedback not give away the correct answer but only signal to the learner that there is an error state. This is because the student must go through the process of calculating the correct answer, rather than copying it, if an effective operator is going to be compiled. If subjects copy they will compile a procedure for copying which will only make them more efficient at copying answers and not at producing

them.

The issue of immediate feedback versus learning by discovery is quite controversial, and I do not mean to imply that immediate feedback is the perfect condition for all types of learning. For instance, Lewis and Anderson found, reasonably enough, that subjects allowed to learn by discovery were better able to recognize error states. Similarly, if one wants to learn a skill such as debugging computer programs, one might let students make errors in their programs so that there is something to debug. However, the theoretical point is that immediate feedback on an operator is important to learning the operator. Both in the error detection and the debugging examples, skill at successfully generating solutions is being sacrificed so that circumstances can be created to facilitate learning skills of dealing with errors.

Further Implications for Instruction

This view of skill acquisition has some important implications for instruction. One straightforward observation is that in a system that can only learn skills by doing them, the importance of formal instruction diminishes and the importance of practice increases. There is already reason to doubt the value of elaborate textual instruction for communicating declarative knowledge (Reder and Anderson, 1980). Carroll (1985) has shown that standard text-editor manuals can be made more effective if they are greatly reduced in length to focus on just the information necessary to perform the skill. His analysis of this phenomena is that a shorter text gives the student more opportunity to focus on practising the skill. Reder, Charney, and Morgan (in press) have shown that the only elaborations that are effective in a manual on personal computers are those that provide examples of how to use the personal computer.

Much of technical instruction is not focused on telling students how to solve problems

in the technical domain but explaining why solutions work in the domain. This distinction might seem subtle, but it is important. A good example of this is recursive functions in the language LISP. Most textbooks focus on explaining to students how recursive programs are evaluated in LISP to produce their results rather than how one thinks up a recursive function to solve a programming problem. The question naturally arises whether being told how recursive functions are evaluated or how recursive programs are created is more effective for the purpose of writing recursive programs.

There are analyses in AI (Rich & Shrobe, 1978; Soloway & Woolf, 1980) about how to write recursive programs. Pirolli and Anderson (1984) have developed an ACT production system model of recursive programming based on these analyses. We performed an experiment in which we either provided students with the standard instruction about how recursion is evaluated or with instruction about how to structure recursive code, based on our production system model. The standard instruction was modelled on existing textbooks. The how-to instruction basically described the productions in our model.

We had students try to write a set of four simple recursive programs. Students given the how-to information took 57 minutes whereas students given the how-it-works information took 85 minutes. Of course, how-to and how-it-works information need not be mutually exclusive as they were in this experiment, but this experiment makes the point that instruction for a skill is most effective when it directly provides information needed in a production system model of that skill.

Instruction in the problem-solving context

While the Pirolli and Anderson experiment confirmed that how-to instruction was better than conventional instruction, there is reason to question how effective it really was. Even the best designed instruction is given out of the context of the actual problem-solving

situation and is often difficult for students to integrate properly into their problem-solving efforts. There are memory problems in retrieving what has been read in one context when it is needed in another. Secondly, it is very easy to misunderstand abstract instruction. For instance, many high school students misunderstand the following statement of the side-angle-side postulate:

> "If two sides and the included angle of one triangle are congruent to the corresponding parts of another triangle, the triangles are congruent."

They interpret "included" to mean included within the triangle rather than included between the two sides. This occurs despite the fact that the statement of the postulate is accompanied by an appropriate diagram. While this particular misunderstanding is so common one might imagine placing remedial instruction right in the text one cannot write into the textbook remedial instruction for all possible confusions. This is one of the major advantages of private tutors--they can diagnose the misconceptions and provide the appropriate instruction in context.

We (Anderson & Reiser, 1985) have performed a number of studies of our LISP tutor in which we contrast one group solving problems on their own with another group solving problems with the LISP tutor. We think the important aspect of the tutor is that it remediates students' confusions by providing instruction in the context of these confusions. One study compared two groups of students who read instruction on recursion much like the instruction used in the original by Pirolli and Anderson study. However, one group had the LISP tutor which forced them to follow this advice and corrected any misconceptions they had about what that advice meant. The other group was left to try to use the instruction as they would--as in the original Pirolli and Anderson study. The tutor group took an average of 5.76 hrs to go through the problems and got 7.6 points on the recursion section of a paper-and-pencil final exam. The no-tutor group took 9.01 hrs and got an average of 4.8 points.

In our view much of the advantage of intelligent computer-based tutors is their ability to facilitate the conversion of abstract declarative instruction into procedures by providing that instruction appropriately in context. Of course, a prerequisite to doing this is to be able to correctly interpret the students behavior in terms of a cognitive model. Much of our actual efforts in intelligent tutoring have gone into developing such a cognitive model and developing techniques for diagnosing student behavior.

Inductive Learning

We have been discussing research that is relevant to the ACT* learning mechanisms of strengthening and compilation. In the original ACT* theory there were two other learning processes which were responsible for formation of new productions. These learning processes actually changed what the system did rather than simply made more effective existing paths of behavior. These were the inductive learning mechanisms of generalization and discrimination. In contrast to our success in relating strengthening and compilation to empirical data, we have had a notable lack of success in our tests of generalization and discrimination.

The mechanisms of generalization and discrimination can be nicely illustrated with respect to language acquisition. Suppose a child has compiled the following two productions from experience with verb forms:

IF the goal is to generate the present tense of KICK
THEN say KICK + S

IF the goal is to generate the present tense of HUG
THEN say HUG + S

The generalization mechanism would try to extract a more general rule that would cover these cases and others:

IF the goal is to generate the present tense of X
THEN say X + S

where X is a variable.

Discrimination deals with the fact that such rules may be overly general and need to be restricted. For instance, the rule above generates the same form independent of whether the subject of the sentence is singular and plural. Thus, it will generate errors. By considering different features in the successful and unsuccessful situations the discrimination mechanisms would generate the following two productions:

IF the goal is to generate the present tense of X
and the subject of the sentence is singular
THEN say X + S

IF the goal is to generate to present tense of X
and the subject of the sentence is plural
THEN say X

These discrimination and generalization mechanisms are very much like similar knowledge acquisition mechanisms which have been proposed in the artificial intelligence literature (e.g., Hayes-Roth &. McDermott. 1976; Vere, 1977). In particular they are called syntactic methods in that they only look at the form of the rule and the form of the contexts in which it succeeds or fails. There is no attempt to use any semantic knowledge about the context to influence the rules that are formed. A consequence of this feature in the ACT theory is that generalization and discrimination are regarded as automatic processes, not subject to strategic influences and not open to conscious inspection. The reports of unconscious learning by Reber (1976) seem consistent with this view. He finds that subjects can learn from examples whether strings are consistent with the rules of finite state grammars without ever consciously formulating the rules of these grammars.

There are now a number of reasons (Anderson, in press) for questioning whether the ACT theory is correct in its position that inductive learning is automatic. First, there is evidence that the generalizations people form from experience are subject to strategic control (Elio & Anderson, 1984; Kline, 1983). In a prototype formation experiment, Elio and

Anderson (1983) showed that subjects could adopt either *memorization* or *hypothesis formation* strategies, and the two strategies led them to *differential success*, depending on what the instances were. Second, Lewis and Anderson found that subjects were able to restrict the application of a problem-solving operator (i.e., discriminate a production) only if they could consciously formulate the discrimination rule. Indeed, there is now reason to believe that even in the Reber unconscious learning situation subjects have conscious access to low level rules that help them classify the examples (Dulany, Carlson, and Dewey, 1984). They do not form hypotheses about finite state grammars but do notice regularities in the example sentences (e.g., grammatical strings have two X's in second and third position).

Another interesting problem with the ACT generalization mechanism is that subjects often appear to emerge with generalizations from a single example (Elio & Anderson, 1983; Kieras & Bovair, 1985), while the syntactic methods of ACT* are intended to extract the *common features* of a number of examples. It is interesting to note that the coding of the LISP function FIRST at the beginning of this paper was based on extraction of generalizations by analogy from the single F-TO-C function. By compiling that analogy process, the student simulation emerged with general production rules for the syntax of function definitions and for specifying a function argument within a function-definition context. It is hard to see how one could extract a generalization from a single example without some semantic understanding of why the example worked. Otherwise, it is not possible to know which features are critical and which features can be ignored (generalized over).

This is an example of a more general point, which is the leaning of our current thinking (Anderson, in press) about induction. The actual process of forming a generalization or a discrimination can be modelled by a set of problem-solving productions. The example above is a case where the problem-solving method is analogy but other weak problem-solving methods can apply also. The inductive processes of *generalization* and *discrimination*

are things that can be implemented by a set of problem-solving productions. Also, because productions are sensitive to the current contents of working memory, induction can be influenced by semantic and strategic factors, as it apparently is. Since the information used by the productions has to be in working memory then people should have conscious access to the information they are using for induction, which they apparently do. Knowledge compilation can convert these inductive problem-solving episodes into productions that generalize beyond the current example.

Thus, we see that there really is no logical need to propose separate learning processes of discrimination and generalization. Besides this argument of parsimony the empirical evidence does not seem consistent with such unconscious learning processes. On the other hand the empirical evidence does seem consistent with the unconscious strengthening and compilation processes. We have never observed subjects to report the changing strengths of their procedures nor compilation of productions.

Conclusions

The ACT theory contains within it the outline of an answer to the epistemological question: "How does structured cognition emerge?" The answer is that we approach a new domain with general problem-solving skills such as analogy, trial and error search, or means-ends analysis. Our declarative knowledge system has the capacity to store in relatively unanalyzed form our experiences in any domain, including instruction if it is available, models of correct behavior, successes and failures of our attempts, etc. It is the basic characteristic of the declarative system that it does not require that one understand how the knowledge will be used before it is stored. This means that we can easily get relevent knowledge into our system but that considerable effort may have to be expended when it comes time to convert this knowledge to behavior.

The knowledge learned about a domain feeds the weak but general problem-solving procedures which try to generate successful behavior in that domain. Knowledge compilation produces new production rules that summarize the outcome of the efforts and strengthening enhances those production rules that repeatedly prove useful. The knowledge-compilation process critically depends on the goal structures generated in the problem solving. These goal structures indicate how to fold the sequence of events into new productions.

Most of our research has been looking at adults or near-adults learning very novel skills like LISP programming. However, the learning theory is intended to generalize downward to child development. Young children are the universal novices--everything they are being asked to learn is a novel skill. In looking at something like LISP programming we are just finding the best approximation to the child's situation in the experimentally more tractable adult. The claim would be that children bring weak problem solving methods to new domains and like adults eventually compile domain-specific procedures. Klahr (1985) has found evidence for early use of weak methods like means-ends analysis and hillclimbing.

The theory of knowledge proposed here is not a extreme empiricist theory in that it requires that the learner bring some prior knowledge to the learning situation. On the other hand, the amount and kind of knowledge being brought to bear is quite different than assumed in many nativist theories. In fact, the real knowledge comes in the form of the weak but general problem-solving procedures that the learner applies to initial problems in these domains. These procedures enable the initial performance and goal structures that impose an organization on the behavior which allows the knowledge compilation process to successfully operate. Given this organization, knowledge compilation is just a process of learning by contiguity applied to production systems. The goal structures provide the belongingness that Thorndike (1935) saw as a necessary complement to learning by contiguity.

If this view of the development of knowledge is correct, the relevant question becomes the nature and origin of the weak methods. Currently, they are modelled in ACT as production sets. Laird and Newell (1983) in their analysis of weak methods similarly model them. In ACT they are assumed as givens, whereas Newell and Laird treat them as the product of encoding the domain. Laird and Newell propose that people start out with a single "universal weak method" that a person starts out with that serves to interpret knowledge, encoded about a domain. This universal weak method is encoded as a set of productions. When the person encodes new information about a specific domain, this information is encoded by additional productions. (The Laird and Newell theory does not have declarative, non-production encodings.) The domain specific productions in the context of the universal weak method can lead ot other of the weak methods.

Of course, much learning is not in completely novel domains. If the learner knows something relevent then the course of knowledge acquisition can be altered quite fundamentally. We saw how productions can transfer wholesale from one domain to another. Even if this is not possible one can use the structure of a solution in one domain as an analog for the structure of the solution in another domain. Also one can use one's knowledge to interpret the declarative information and so store a highly sophisticated interpretation of the knowledge that is presented. We have been comparing students learning LISP as a first programming language versus students learning LISP after Pascal. The Pascal students learn LISP much more effectively and one of the reasons is that they appreciate much better the semantics of various programming concepts such as what a variable is. Weak problem-solving methods like analogy can be much more effective if they can more richly represent the knowledge. Indeed Pirolli and Anderson (1985) looked at the acquisition of recursive programming skill and found that almost all students develop new programs by analogy to example recursive programs but that their success is determined by

how well they understand why these examples work.

Appendix: Knowledge Compilation: Technical Discussion

The description below describes knowledge compilation as it is implemented in the GRAPES production system (Sauers & Farrell, 1982), which is intended to embody certain aspects of the ACT* theory. Knowledge compilation consists of two subcomponents, proceduralization and composition. Proceduralization eliminates reference to certain declarative facts by building into productions the effect of that reference. Composition collapses numbers of productions into one. These will be discussed separately.

Proceduralization

Proceduralization requires a separation between goal information and context information in the condition of a production. Consider the following production:

G1 IF the goal is to create a structure
and there is a operation that creates such a structure
and it requires a set of substeps
THEN set as subgoals to perform those steps

This is a classic working-backwards operator, an instance of a general, weak, problem-solving method. This production might apply if our goal was to insert an element into a list and we knew that there was a LISP function CONS which achieved this--i.e., (CONS 'A '(B C)) = (A B C). In this production, the first line of the condition describes the current goal and the subsequent context lines identify relevant information in declarative memory. Proceduralization eliminates the context lines but gets their effect by building a more specific goal description:

D1 IF the goal is insert an element into a list
THEN write CONS and set as subgoals to
1. Code the element
2. Code the list

The transition from the first production to the second is an example of the transition from domain-general to domain-specific productions.

To understand in detail how domain-general productions apply and how proceduralization occurs, one needs to be more precise about the encoding of the production, the goal, and the knowledge about CONS. With respect to the production we have to identify its variable components. Below is a production more like its GRAPES implementation where terms prefixed by "=" denote variables:

G1' IF the goal is to achieve =relation on =arg1 and =arg2
and =operation achieves =relation on =arg1* and =arg2*
and requires =list as substeps
THEN make =list subgoals

The goal is "to achieve insertion of arg1 into arg2," and our knowledge about CONS is represented:

CONS achieves insertion of argument1 into argument2, and it involves the substeps of writing "(," writing CONS, coding argument1, coding argument2, and writing ")."

The production G1' applies to the situation with the following binding of variables:

=relation	:	insertion
=operation	:	CONS
=arg1	:	arg1
=arg2	:	arg2
=arg1*	:	argument1
=arg2*	:	argument2
=list	:	writing "(", writing CONS, coding argument1, coding argument2, and writing ")"

Thus the production would execute and set the subgoals of doing the steps in =list.

The actual execution of the production required that the definition of CONS be held active in working memory and be matched by the production. This can be eliminated by proceduralization which builds a new production that contains the relevant aspects of the definition within it. This is achieved by replacing the variables in the old production by what they matched to in the definition of CONS. The proceduralized production that would be built in this case is:

D1' IF the goal is to achieve insertion of =arg1 into =arg2

THEN make subgoals to
1. write (
2. write CONS
3. code =arg1
4. code =arg2
5. write)

or, as we generally write such productions for simplicity:

D1″ IF the goal is to insert =arg1 into =arg2
THEN write CONS and set as subgoals to
1. code =arg1
2. code =arg2

In general, proceduralization operates by eliminating reference to the declarative knowledge that permitted the problem solution by the weak-method productions and building that knowledge into the description of the problem solution.

It would be useful to have a detailed analysis of proceduralization of solution by analogy since this figured so prominently in the discussion of the paper. Usually, analogy is implemented in ACT by the operation of a sequence of productions (e.g., see the Appendix to Ch. 5 in Anderson, 1983), but the example here will compact the whole analogy process into a single production.

G2 IF the goal is to achieve =relation1 on =arg3
and (=function =arg4) achieves =relation1 on =arg4.
THEN write (=function =arg3)

Suppose the subject wanted to get the first element of the list (a b c) and saw the example from LISP that (CAR '(b r)) = b. Then this production would code (CAR '(a b c)) by analogy. Proceduralization can operate by eliminating reference to the declarative source of knowledge--in this case, the declarative source of knowledge is the example that (CAR '(b r)) = b. It would produce the new production:

D2 IF the goal is to achieve the first-element of =arg3
THEN write CAR and set as a subgoal to
1. code =arg3

Composition

Composition (Lewis, 1978) is the process of collapsing multiple productions into single productions. Whenever a sequence of productions apply in ACT and achieve a goal, a single production can be formed which will achieve the effect of the set. Note that the existance of goals is absolutely critical in defining what sequences of productions to compose.

While many times composition applies to sequences of more than two productions, its effect on longer sequences is just the composition of its effect on shorter sequences. Thus, if S1.S2 is a sequence of productions to be composed and C is the composition operator, C(S1.S2) = C(C(S1),C(S2)). So all we have to do is specify the pairwise compositions.

Let "IF C1 THEN A1", and "IF C2 THEN A2" be a pair of productions to be composed where C1 and C2 are conditions and A1 and A2 are actions. Then their composition is "IF C1 & (C2 - A1) THEN (A1 - G(C2)) & A2." C2-A1 denotes the conditions of the second production not satisfied by structures created in the action of the first. All the conditional tests in C1 and (C1 - A2) must be present from the beginning if the pair of productions are to fire. A1 - G(C2) denotes the actions of the first production minus the goals created by the first production that were satisfied by the second.

As an example, consider a situation where we want to insert the first element of one list into a second list. The first production above would fire and write CONS setting subgoals to code the two arguments to CONS. Then the second production above would fire to code CAR and set a subgoal to code the argument to CAR. Composing these two together would produce the production:

C3 IF the goal is to insert the first element of =arg3 into =arg2
THEN code CONS and then code CAR and set as subgoals to

1. code =arg3
2. code =arg2

It is just productions of this form that we speculate subjects were forming in the McKendree and Anderson experiment described in the main part of this paper.

Composition and Proceduralization

Much of the power of knowledge compilation in the actual simulations comes from the combined action of proceduralization and composition together. The composed productions C1 and C2 discussed with respect to Figure 1 earlier have such a history. A number of analogical productions had to be proceduralized and composed to form each production. Thus, in real learning situations we simultaneously drop out the reference to declarative knowledge and collapse many (often more than two) productions into a single productions. This can produce the enormous qualitative changes we see in problem solving with just a single trial.

References

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, *9*, 147-169.

Adelson, B. (1981). Problem solving and the development of abstract categories in programming languages. *Memory and Cognition*, *9*, 422-423.

Anderson, J.R. (1976). *Language, Memory, and Thought*. Hillsdale, NJ: Erlbaum.

Anderson, J. R. (1981). *Cognitive Skills and their Acquisition*. Hillsdale, NJ: Erlbaum Associates.

Anderson, J.R. (1982). Acquisition of Cognitive Skill. *Psychological Review*, *89*, 369-406.

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

Anderson, J.R. (1985). *Cognitive Psychology and Its Implications, Second Edition*. New York: W. H. Freeman & Company.

Anderson, J. R. (In press). Knowledge compilation: The general learning mechanism. In R. Michalski, J. Carbonnell, & T. Mitchell (Eds.) Machine Learning II. Palo Alto, CA: Tioga Press.

Anderson, J.R. and Jeffries, R. (In press). Novice LISP Errors: Undetected Losses of Information from Working Memory. Human-Computer Interaction.

Anderson, J. R. & Reiser, B. J. (April 1985). The LISP Tutor. *Byte*, *10*, 159-175.

Anderson, J. R., Boyle, C. F., & Reiser, B. J. (April 1985). Intelligent tutoring systems. *Science*, *228*, 456-462.

Anderson, J., Boyle C.F., Farrell, R. & Reiser, B. (1984). Cognitive Principles in the Design of Computer Tutors. In *Sixth Annual Conference of the Cognitive Science Program*. Carnegie-Mellon University.

Anderson, J.R., Farrell, R., & Sauers, R. (1984). Learning to program in LISP. *Cognitive Science*, *8*, 87-129.

Anderson, R. C., Kulhavy, R. W., & Andre, T. (1972). Conditions under which feedback facilitates learning for programmed lessons. *Journal of Education Psychology, 63*, 186-188.

Brown, J.S. & Van Lehn, K. (1980). Repair theory: A generative theory of bugs in procedural skills. *Cognitive Science, 4*, 379-426.

Carbonell, J. G. (1983). Learning by analogy: Formulating and generalizing plans from past experience. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), *Machine Learning*. Palo Alto, CA: Tioga.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Erlbaum.

Carroll, J. (March 1985). *Designing minimalist training materials* Research Report 46643). IBM Watson Research Center, Computer Science Department.

Chase, W. G., & Ericsson, K. A. (1982). Skill and working memory. In G. H. Bower (Ed.), *The psychology of learning and motivation, Vol. 16*. New York: Academic Press.

Chase, W.G. & Simon, H.A. (1973). The mind's eye in chess. In W.G. Chase (Ed.), *Visual Information Processing*. New York: Academic Press.

Chi, M.T.H., Glaser, R., & Farr, M. (Eds.). (In press). The Nature of Expertise. Hillsdale, NJ: Erlbaum.

Chomsky, N. (1980). Rules and representations. *Behavioral and Brain Sciences, 3*, 1-61.

Dulany, D. E., Carlson, R. A., & Dewey, G. I. (1984). A case of syntactical learning and judgment: How conscious and how abstract? *Journal of Experimental Psychology: General, 113*, 541-555.

Elio, R. & Anderson, J.R. (1983). Effects of category generalizations and instance similarity on schema abstraction *Journal of Experimental Psychology: Human Learning and Memory, 7*, 397-417.

Elio, R. & Anderson, J. R. (1984). The effects of information order and learning mode on

schema abstraction. *Memory and Cognition, 12,* 20-30.

Fodor, J.A. (1983). *The Modularity of Mind.* Cambridge, MA: MIT/Bradford Books.

Fodor, J.A., Bever, T.G. & Garrrett, M.F. (1974). *The Psychology of Language.* New York: McGraw-Hill.

Hayes-Roth, F. & McDermott, J. (1976). Learning structured patterns from examples. In *Proceedings of the Third International Joint Conference on Pattern Recognition.* International Joint Conference on Pattern Recognition.

Jeffries, R., Turner, A. A., Polson, P. G., & Atwood, M. E. (1981). The processes involved in designing software. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition.* Hillsdale, NJ: Erlbaum.

Katz, I. R. & Anderson, J. R. (In preparation). An exploratory study of novice programmers' bugs and debugging behavior.

Kieras, D. E. & Bovair, S. (1985). *The acquisition of procedures from text: A production-system analysis of transfer of training* Technical Report 16). University of Michigan.

Kieras, D. E. & Polson, P. G. (in press). An approach to the formal analysis of user complexity. International Journal of Man-Machine Studies.

Klahr, D. (in press). Solving problems with ambiguous subgoal ordering: pre-schoolers' performance. *Child Development,* Vol. 56.

Klahr, D., Langley, P., & Neches, R. (in press). *Self-modifying production system: Models of learning and development.* Cambridge, MA: Bradford Books/MIT Press.

Kline, P. J. (1983). Computing the similarity of structured objects by means of a heuristic search for correspondences. Ph.D. dissertation, University of Michigan.

Kling, R. E. (1971). A paradigm for reasoning by analogy. *Artificial Intelligence 2.* 147-178

Laird, J. E. & Newell, A. (1983). Universal weak method: Summary of results. In *Proceedings of the Eight IJCAI.* IJCAI.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1984). Towards chunking as a general

learning mechanism. In *Proceedings of AAAI84.* AAAI.

Langley, P. (1982). Language acquisition through error recovery. *Cognition and Brain Theory, 5,* 211-255.

Larkin, J.H., McDermott, J., Simon, D.P., & Simon, H.A. (1980). Models of competence in solving physics problems. *Cognitive Science. 4.* 317-345.

Lesgold, A. M. (1984). Acquiring expertise. In J. R. Anderson and S. M. Kosslyn (Eds.). *Tutorials in Learning and Memory.* San Francisco, CA: Freeman.

Lewis, C. W. (1978). Production system models of practice effects. Unpublished doctoral dissertation, University of Michigan.

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

Luchins, A. S., & Luchins, E. H. (1959). *Rigidity of Behavior: A Variational Approach to the Effect of Einstellung.* Eugene, Oregon: University of Oregon Books.

MacWhinney, B. (in press). Mechanism of Language Acquisition. Hillsdale, NJ: Erlbaum.

McClelland, J. L. (1985). Putting knowledge in its place: A scheme for programming parallel processing structures on the fly. *Cognitive Science. 9.* 113-146.

McKendree, J., & Anderson, J.R. (In press). Frequency and practice effects on the composition of knowledge in LISP evaluation. To appear in J.M. Carroll [Ed.] Cognitive Aspects of Human-Computer Interaction.

Michalski, R. S. (1983). Theory and methodology of inductive learning. In R. Michalski, J. G. Carbonnell, & T. M. Mitchell (Eds.). *Machine Learning.* Palo Alto, CA: Tioga Press.

Michalski, J. G., Carbonell, J. G., & Mitchell, T. M. (1983). *Machine Learning.* Palo Alto, CA: Tioga Press.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence. 18.* 203-226.

Neves, D. M., & Anderson, J. R. (1981). Knowledge compilation: Mechanisms for the

automatization of cognitive skills. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition*.. Hillsdale, NJ: Erlbaum.

Newell, A. (1969). Heuristic programming: Ill-structured problems. In J. Aronofsky (Ed.). *Progress in Operations Research, III.* New York: Wiley.

Newell, A. (1973). Production systems: Models of control structures. In W. G. Chase (Ed.), *Visual information processing*.. New York: Academic Press.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition*.. Hillsdale, NJ: Erlbaum.

Norman, D. A. (1981). Categorization of action slips. *Psychological Review, 88.* 1-15.

Orata, P. T. (1928). *The Theory of Identical Elements.* Columbus, OH: Ohio State University Press.

Pinker, S. (1984). *Language Learnability and Language Development.* Cambridge, MA: Harvard Press.

Pirolli, P. L. & Anderson, J. R. (1984). The role of mental models in learning to program. Paper presented at the Twenty-fifth Annual Meeting of the Psychonomic Society, San Antonia, TX.

Pirolli, P. L. & Anderson, J. R. (1985). The role of learning from examples in the acquisition of recursive programming skill. *Canadian Journal of Psychology.* 39. 240-272.

Polson, P. G., & Kieras, D. E. (1985). A quantitative model of learning and performance of text editing knowledge. In L. Bormann & B. Curtis (Eds.). *Human Factors in Computing Systems: CHI 1985 Conference Proceedings.* New York: CHI 1985.

Postman, L. (1971). Transfer, interference, and forgetting. In L. W. King & L. A. Riggs (Eds.). *Experimental Psychology.* New York: Holt, Rinehart & Winston.

Reber, A. S. (1976). Implicit learning of synthetic languages: The role of instructional set. *Journal of Experimental Psychology: Human Learning and Memory. 2.* 88-94.

Reder, L. M., & Anderson, J. R. (1980). A comparison of texts and their summaries: Memorial consequences. *Journal of Verbal Learning and Verbal Behavior, 19*, 121-134.

Reder, L. M., Charney, D. H., & Morgan, K. I. (In press). The role of elaborations in learning a skill from an instructional text. Memory and Cognition.

Rich, C. & Shrobe, H. (1978). Initial report of a LISP programmers' apprentice. *IEEE Trans. Soft. Eng., SE-4:6*, 456-466.

Rumelhart, D. E. & Zipser, D. (1985). Feature discovery by competitive learning. *Cognitive Science. 9*, 75-112.

Sauers, R. & Farrell, R. (1982). *GRAPES User's Manual* Technical Report ONR-82-3). Carnegie-Mellon University.

Singley, K. & Anderson, J.R. (in press). The transfer of text-editing skill. Journal of Man-Machine Studies.

Singley, K. & Anderson, J. R. (in preparation). A key-stroke analysis of learning and transfer in text-editing.

Soloway, E. L. & Woolf, B. (1980). *From problems to programs via plans: The content and structure of knowledge for intorductory LISP programming* Technical Report Coins 80-19). University of Massachussetts.

Thorndike, E. L. (1903). *Educational Psychology.* New York: Lemke & Buechner

Thorndike, E. L. (1935). *The psychology of wants, interests, and attitudes.* New York: Appleton-Century-Crofts.

Van Lehn, K. (1983). *Felicity conditions for human skill acquisition: Validating an AI-based theory* Technical Report CIS-21). Xerox Parc, Palo Alto.

Vere, S. A. (1977). Induction of relational productions in the presence of background information. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence.* Boston, MA: IJCAI.

Wexler, K. & Culicover, P. (1980). *Formal Principles of Language Acquisition.* Cambridge,

MA: MIT Press.

Winston, P. H. (1979). Learning and reasoning by analogy. *Communications of the ACM*. 23, 689-703.

Winston, P. H. & Horn, B. K. P. (1981). *LISP*. Reading, MA: Addison-Wesley.

Figure Captions

Figure 1 — A representation of the goal structure in the subject BR's solution to the problem of writing the function FIRST. The boxes represent goals, and the arrows indicate that a production has decomposed the goal above into the subgoals below. Checks indicate successful goals, and X's indicate failed goals. The dotted lines indicate parts of the goal three combined in composition.

Figure 2 — A representation of the goal structure set up by the production for text-editing in ED. Labelled are goals and goal structures in common with EDT or EMACS. Goal structures not in common with either are labelled 0000.

Figure 3 — Transfer among text editors.

Figure 4 — Transfer between EMACS and Perverse EMACS.

Figure 5 — An example of the problem description used in the Lewis and Anderson (1985) experiment.

Table 1

Sample material from McKendree and Anderson (in press)

ITEM	GROUP 1 FREQUENCY	GROUP 2 FREQUENCY
(CAR (CAR '((A B) (C D) (E F)))) =	12	4
(CAR (CDR '((A B) (C D) (E F)))) =	4	12
(CDR (CAR '((A B) (C D) (E F)))) =	4	12
(CDR (CDR '((A B) (C D) (E F)))) =	12	4

Notes

[1]This research is supported by Contract No. N00014-84-K-0064 from the Office of Naval Research and Grant No. IST-83-18629 from the National Science Foundation. I would like to thank Al Corbett, David Klahr, Allen Newell, and Lynne Reder for their comments on this manuscript.

[2]The assumption, which I will elaborate later, is that the domain-general productions are innate.

[3]The single quote in front of (LIST1) causes LISP to treat this as literally a list with the element LIST1.

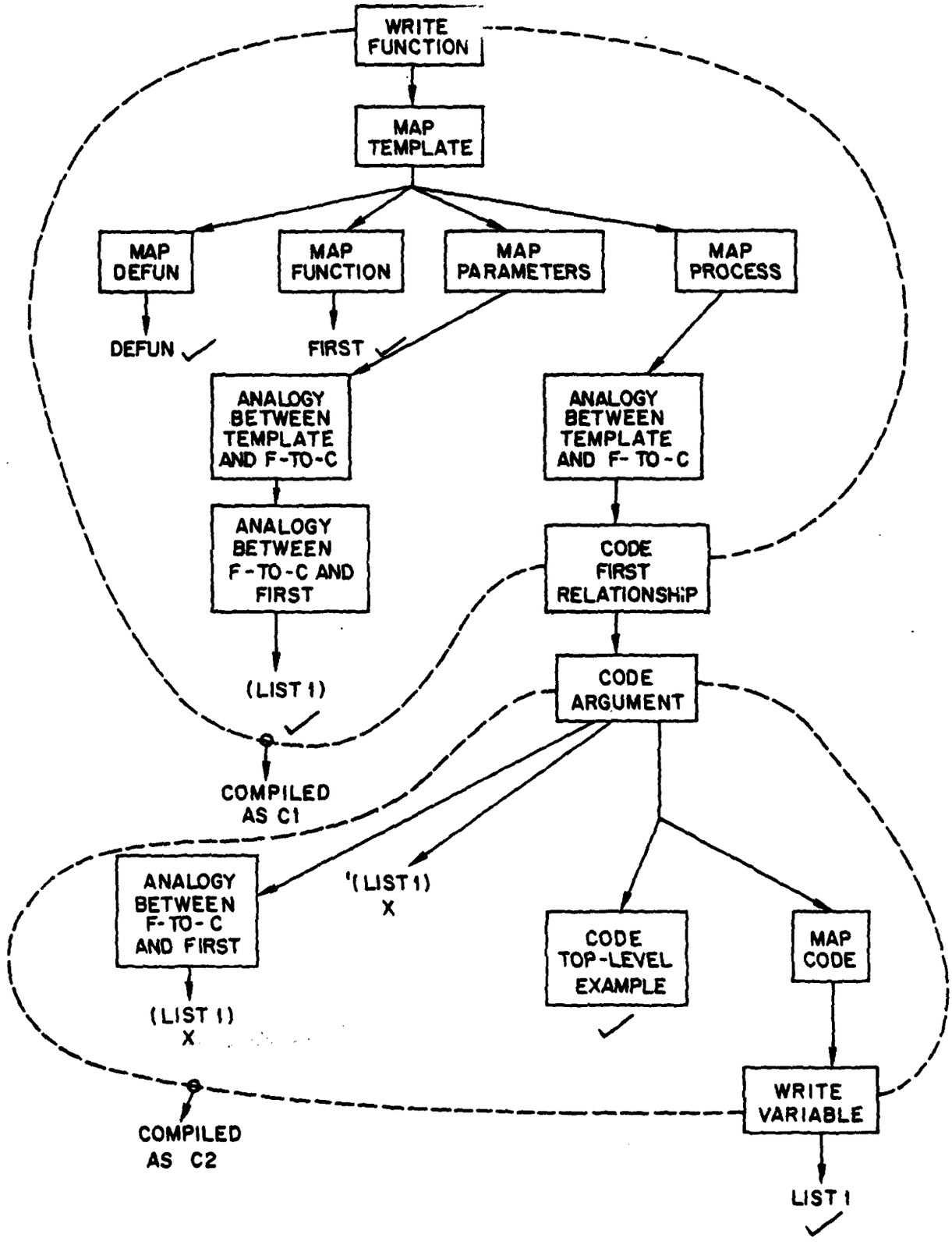
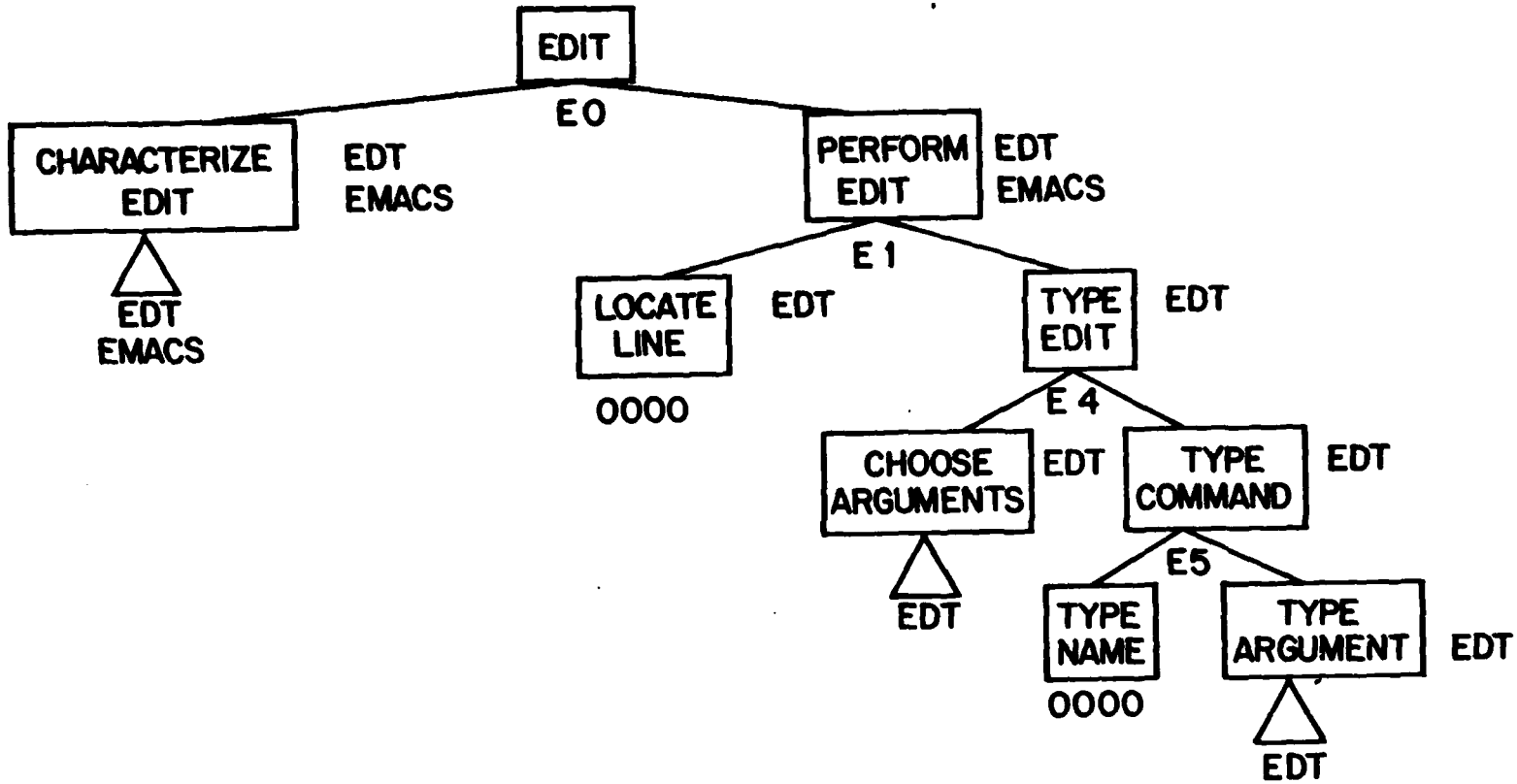

WRITE FUNCTION
MAP TEMPLATE
MAP DEFUN
MAP FUNCTION
MAP PARAMETERS
MAP PROCESS
DEFUN ✓
FIRST ✓
ANALOGY BETWEEN TEMPLATE AND F-TO-C
ANALOGY BETWEEN TEMPLATE AND F-TO-C
ANALOGY BETWEEN F-TO-C AND FIRST
CODE FIRST RELATIONSHIP
(LIST 1) ✓
CODE ARGUMENT
COMPILED AS C1
ANALOGY BETWEEN F-TO-C AND FIRST
'(LIST 1) X
CODE TOP-LEVEL EXAMPLE ✓
MAP CODE
(LIST 1) X
WRITE VARIABLE
COMPILED AS C2
LIST 1 ✓

Figure 1



0000 = different from EDT

Figure 2

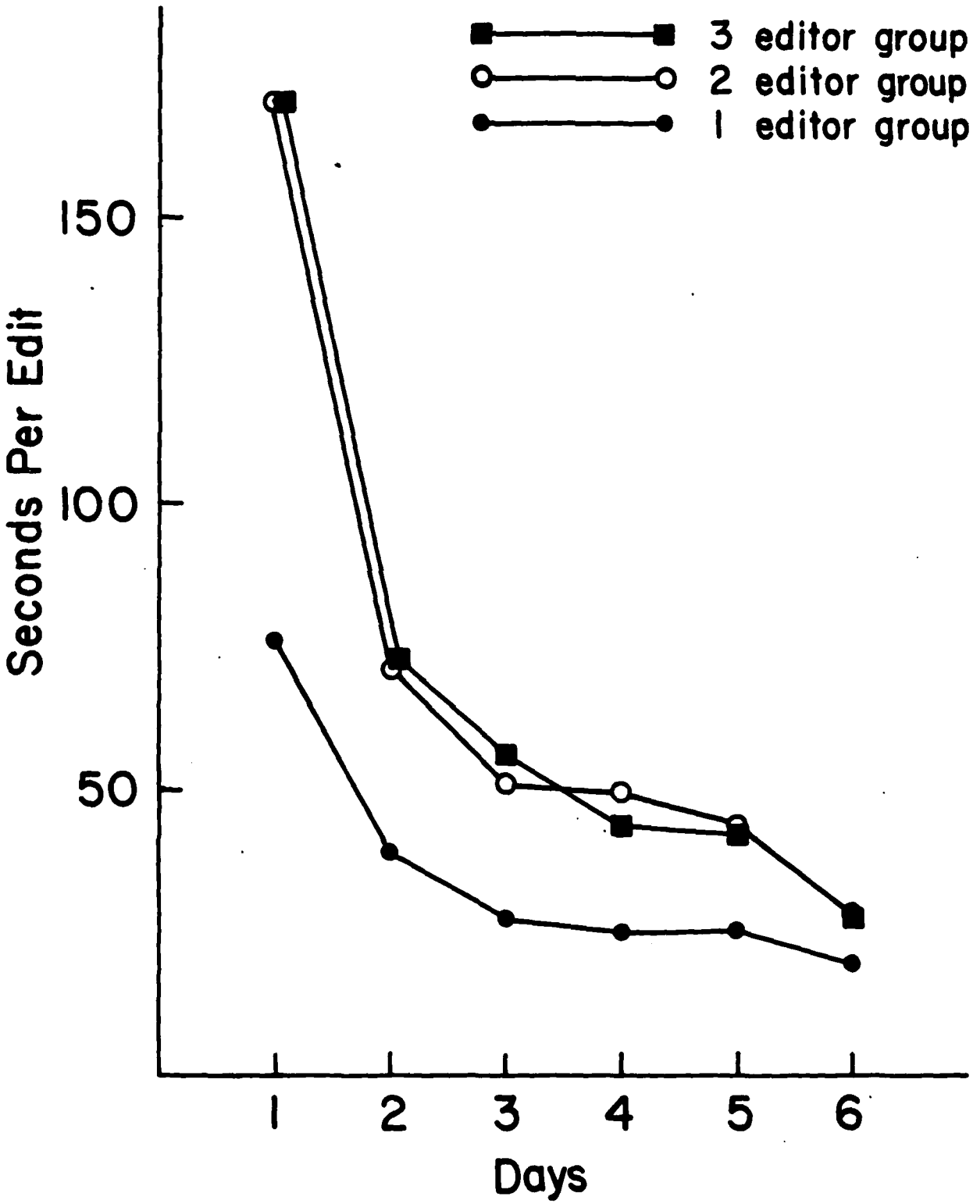

3 editor group
2 editor group
1 editor group
150
100
50
Seconds Per Edit
1
2
3
4
5
6
Days

Figure 3

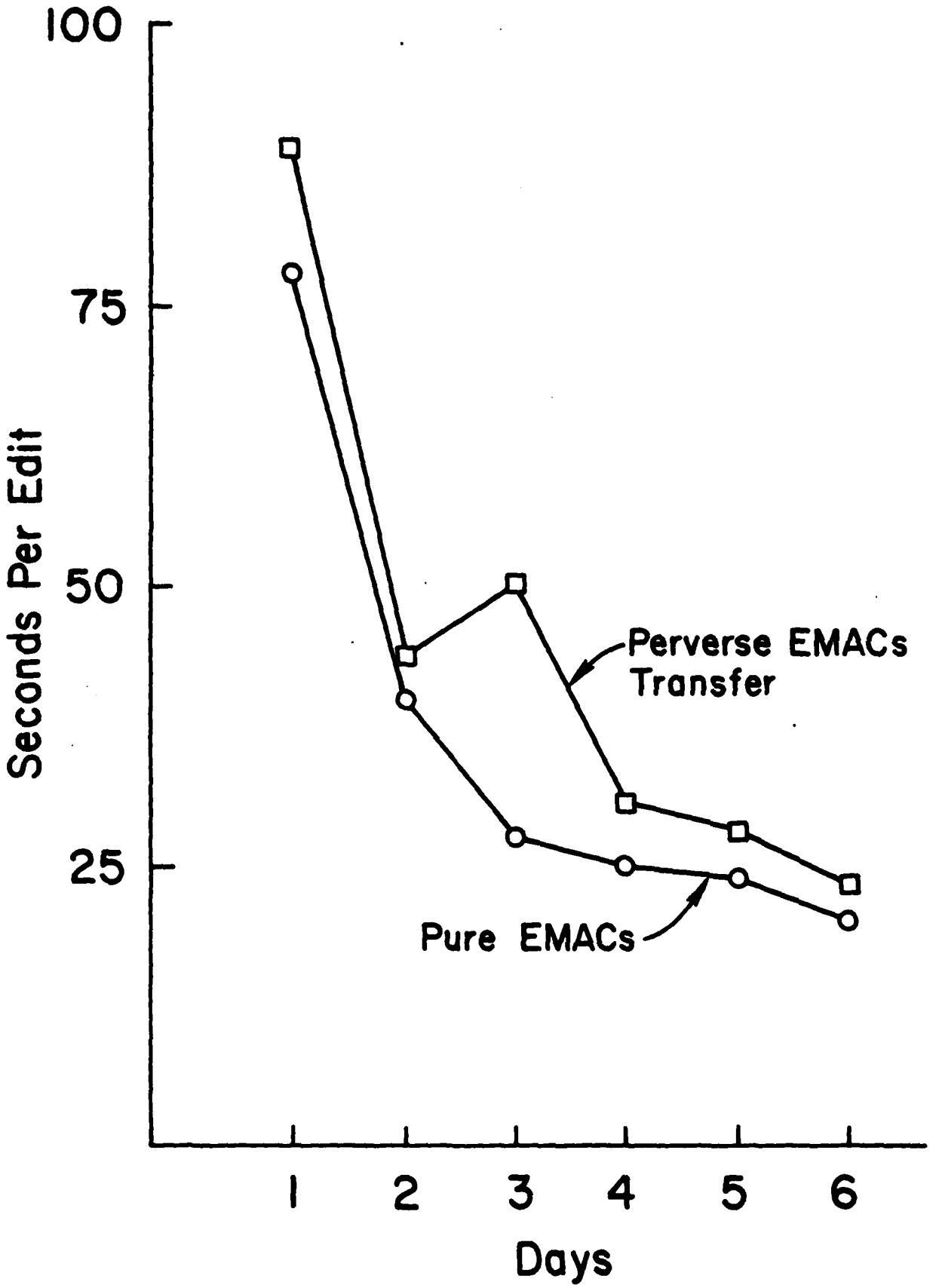

100
75
50
25
Seconds Per Edit
Perverse EMACs
Transfer
Pure EMACs
1
2
3
4
5
6
Days

Figure 4

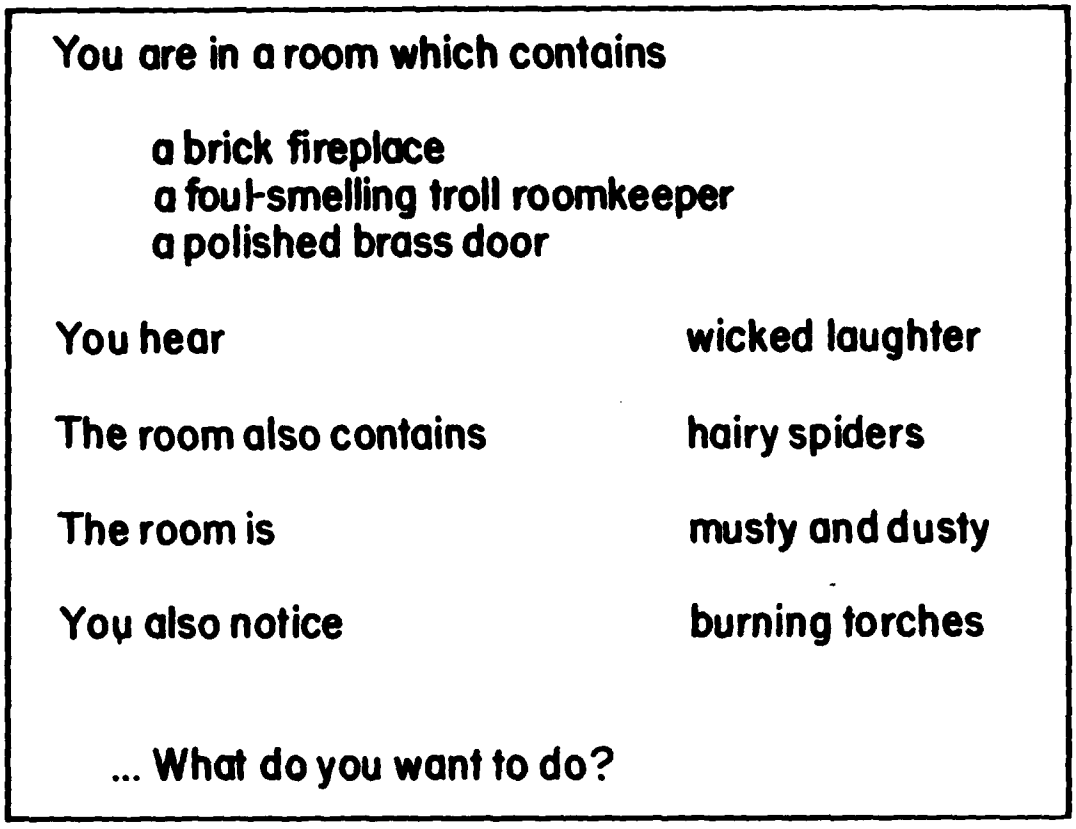

You are in a room which contains
a brick fireplace
a foul-smelling troll roomkeeper
a polished brass door
You hear
wicked laughter
The room also contains
hairy spiders
The room is
musty and dusty
You also notice
burning torches
... What do you want to do?

Figure 5

DISTRIBUTION LIST FOR Carnegie-Mellon University/Anderson (NR 667-530)

Personnel Analysis Division,
AF/MPXA
5C360, The Pentagon
Washington, DC 20330

Air Force Human Resources Lab
AFHRL/MPD
Brooks AFB, TX 78235

AFOSR,
Life Sciences Directorate
Bolling Air Force Base
Washington, DC 20332

Dr. Robert Ahlers
Code N711
Human Factors Laboratory
NAVTRAEQUIPCEN
Orlando, FL 32813

Dr. Ed Aiken
Navy Personnel R&D Center
San Diego, CA 92152

Dr. William E. Alley
AFHRL/MOT
Brooks AFB, TX 78235

Dr. Earl A. Alluisi
HQ, AFHRL (AFSC)
Brooks AFB, TX 78235

Dr. John R. Anderson
Department of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. Nancy S. Anderson
Department of Psychology
University of Maryland
College Park, MD 20742

Dr. Steve Andriole
Perceptronics, Inc.
21111 Erwin Street
Woodland Hills, CA 91367-3713

Dr. John Annett
University of Warwick
Department of Psychology
Coventry CV4 7AJ
ENGLAND

Dr. Phipps Arabie
University of Illinois
Department of Psychology
603 E. Daniel St.
Champaign, IL 61820

Technical Director, ARI
5001 Eisenhower Avenue
Alexandria, VA 22333

Special Assistant for Projects,
OASN(M&RA)
5D800, The Pentagon
Washington, DC 20350

Dr. Michael Atwood
ITT - Programming
1000 Oronoque Lane
Stratford, CT 06497

Dr. Patricia Baggett
University of Colorado
Department of Psychology
Box 345
Boulder, CO 80309

Dr. Meryl S. Baker
Navy Personnel R&D Center
San Diego, CA 92152

Mr. J. Barber
HQS, Department of the Army
DAPE-ZBR
Washington, DC 20310

Capt. J. Jean Belanger
Training Development Division
Canadian Forces Training System
CFTSHQ, CFB Trenton
Astra, Ontario, K0K
CANADA

CDR Robert J. Biersner, USN
Naval Biodynamics Laboratory
P. O. Box 29407
New Orleans, LA 70189

Dr. Menucha Birenbaum
School of Education
Tel Aviv University
Tel Aviv, Ramat Aviv 69978
ISRAEL

DISTRIBUTION LIST FOR Carnegie-Mellon University/Anderson (NR 667-530)

Dr. Werner P. Birke
Personalstammamt der Bundeswehr
Kolner Strasse 262
D-5000 Koeln 90
FEDERAL REPUBLIC OF GERMANY

Dr. Gautam Biswas
Department of Computer Science
University of South Carolina
Columbia, SC 29208

Dr. John Black
Yale University
Box 11A, Yale Station
New Haven, CT 06520

Arthur S. Blaiwes
Code N711
Naval Training Equipment Center
Orlando, FL 32813

Dr. Robert Blanchard
Navy Personnel R&D Center
San Diego, CA 92152

Cdt. Arnold Bohrer
Sectie Psychologisch Onderzoek
Rekruterings-En Selectiecentrum
Kwartier Koningen Astrid
Bruijnstraat
1120 Brussels, BELGIUM

Dr. Jeff Bonar
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Nick Bond
Office of Naval Research
Liaison Office, Far East
APO San Francisco, CA 96503

Dr. Gordon H. Bower
Department of Psychology
Stanford University
Stanford, CA 94306

Dr. Richard Braby
NTEC Code 10
Orlando, FL 32751

Dr. Robert Breaux
Code N-095R
NAVTRAEQUIPCEN
Orlando, FL 32813

Dr. Ann Brown
Center for the Study of Reading
University of Illinois
51 Gerty Drive
Champaign, IL 61280

Dr. John S. Brown
XEROX Palo Alto Research
Center
3333 Coyote Road
Palo Alto, CA 94304

Dr. Bruce Buchanan
Computer Science Department
Stanford University
Stanford, CA 94305

Dr. Patricia A. Butler
NIE Mail Stop 1806
1200 19th St., NW
Washington, DC 20208

Dr. Tom Cafferty
Dept. of Psychology
University of South Carolina
Columbia, SC 29208

Dr. Robert Calfee
School of Education
Stanford University
Stanford, CA 94305

Dr. Jaime Carbonell
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

Mr. James W. Carey
Commandant (G-PTE)
U.S. Coast Guard
2100 Second Street, S.W.
Washington, DC 20593

DISTRIBUTION LIST FOR Carnegie-Mellon University/Anderson (NR 667-530)

Dr. Susan Carey
Harvard Graduate School of
Education
337 Gutman Library
Appian Way
Cambridge, MA 02138

Dr. Pat Carpenter
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

Dr. Robert Carroll
NAVOP 01B7
Washington, DC 20370

Dr. Fred Chang
Navy Personnel R&D Center
Code 51
San Diego, CA 92152

Dr. Davida Charney
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Eugene Charniak
Brown University
Computer Science Department
Providence, RI 02912

Dr. Michelene Chi
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Mr. Raymond E. Christal
AFHRL/MOE
Brooks AFB, TX 78235

Dr. William Clancey
Computer Science Department
Stanford University
Stanford, CA 94306

Scientific Advisor
to the DCNO (MPT)
Center for Naval Analysis
2000 North Beauregard Street
Alexandria, VA 22311

Chief of Naval Education
and Training
Liaison Office
Air Force Human Resource Laboratory
Operations Training Division
Williams AFB, AZ 85224

Assistant Chief of Staff
for Research, Development,
Test, and Evaluation
Naval Education and
Training Command (N-5)
NAS Pensacola, FL 32508

Dr. Allan M. Collins
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

Dr. Stanley Collyer
Office of Naval Technology
800 N. Quincy Street
Arlington, VA 22217

Dr. Lynn A. Cooper
Learning R&D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Dr. Meredith P. Crawford
American Psychological Association
Office of Educational Affairs
1200 17th Street, N.W.
Washington, DC 20036

Dr. Hans Crombag
University of Leyden
Education Research Center
Boerhaavelaan 2
2334 EN Leyden
The NETHERLANDS

Dr. Lee Cronbach
16 Laburnum Road
Atherton, CA 94205

Dr. Kenneth B. Cross
Anacapa Sciences, Inc.
P.O. Drawer Q
Santa Barbara, CA 93102

DISTRIBUTION LIST FOR Carnegie-Mellon University/Anderson (NR 667-530)

Dr. Mary Cross
Department of Education
Adult Literacy Initiative
Room 4145
400 Maryland Avenue, SW
Washington, DC 20202

CDR Mike Curran
Office of Naval Research
800 N. Quincy St.
Code 270
Arlington, VA 22217-5000

Bryan Dallman
AFHRL/LRT
Lowry AFB, CO 80230

Mr. Robert Denton
AFMPC/MPCYPR
Randolph AFB, TX 78150

Mr. Paul DiRenzo
Commandant of the Marine Corps
Code LBC-4
Washington, DC 20380

Dr. R. K. Dismukes
Associate Director for Life Sciences
AFOSR
Bolling AFB
Washington, DC 20032-6448

Defense Technical
Information Center
Cameron Station, Bldg 5
Alexandria, VA 22314
Attn: TC
(12 Copies)

Dr. Thomas M. Duffy
Communications Design Center
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Barbara Eason
Military Educator's
Resource Network
InterAmerica Research Associates
1555 Wilson Blvd
Arlington, VA 22209

Edward E. Eddowes
CNATRA N301
Naval Air Station
Corpus Christi, TX 78419

Dr. John Ellis
Navy Personnel R&D Center
San Diego, CA 92252

Dr. Richard Elster
Deputy Assistant Secretary
of the Navy (Manpower)
Washington, DC 20350

Dr. Susan Embretson
University of Kansas
Psychology Department
Lawrence, KS 66045

Dr. Randy Engle
Department of Psychology
University of South Carolina
Columbia, SC 29208

Lt. Col Rich Entlich
HQ, Department of the Army
OCSA(DACS-DPM)
Washington, DC 20310

Dr. William Epstein
University of Wisconsin
W. J. Brogden Psychology Bldg.
1202 W. Johnson Street
Madison, WI 53706

ERIC Facility-Acquisitions
4833 Rugby Avenue
Bethesda, MD 20014

Dr. K. Anders Ericsson
University of Colorado
Department of Psychology
Boulder, CO 80309

Edward Esty
Department of Education, OERI
MS 40
1200 19th St., NW
Washington, DC 20208

DISTRIBUTION LIST FOR Carnegie-Mellon University/Anderson (NR 667-530)

Dr. Beatrice J. Farr
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Marshall J. Farr
2520 North Vernon Street
Arlington, VA 22207

Dr. Pat Federico
Code 511
NPRDC
San Diego, CA 92152

Dr. Paul Feltovich
Southern Illinois University
School of Medicine
Medical Education Department
P.O. Box 3926
Springfield, IL 62708

Mr. Wallace Feurzeig
Educational Technology
Bolt Beranek & Newman
10 Moulton St.
Cambridge, MA 02238

Dr. Craig I. Fields
ARPA
1400 Wilson Blvd.
Arlington, VA 22209

Dr. Gerhard Fischer
University of Colorado
Department of Computer Science
Boulder, CO 80309

Dr. Linda Flower
Carnegie-Mellon University
Department of English
Pittsburgh, PA 15213

Dr. Kenneth D. Forbus
University of Illinois
Department of Computer Science
1304 West Springfield Avenue
Urbana, IL 61801

Dr. Carl H. Frederiksen
McGill University
3700 McTavish Street
Montreal, Quebec H3A 1Y2
CANADA

Dr. John R. Frederiksen
Bolt Beranek & Newman
50 Moulton Street
Cambridge, MA 02138

Dr. Norman Frederiksen
Educational Testing Service
Princeton, NJ 08541

Dr. Alfred R. Fregly
AFOSR/NL
Bolling AFB, DC 20332

Dr. Bob Frey
Commandant (G-P-1/2)
USCG HQ
Washington, DC 20593

Dr. Alinda Friedman
Department of Psychology
University of Alberta
Edmonton, Alberta
CANADA T6G 2E9

Dr. R. Edward Geiselman
Department of Psychology
University of California
Los Angeles, CA 90024

Dr. Michael Genesereth
Stanford University
Computer Science Department
Stanford, CA 94305

Dr. Dedre Gentner
University of Illinois
Department of Psychology
603 E. Daniel St.
Champaign, IL 61820

Dr. Robert Glaser
Learning Research
& Development Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Arthur M. Glenberg
University of Wisconsin
W. J. Brogden Psychology Bldg.
1202 W. Johnson Street
Madison, WI 53706

DISTRIBUTION LIST FOR Carnegie-Mellon University/Anderson (NR 667-530)

Dr. Marvin D. Glock
13 Stone Hall
Cornell University
Ithaca, NY 14853

Dr. Sam Glucksberg
Princeton University
Department of Psychology
Green Hall
Princeton, NJ 08540

Dr. Joseph Goguen
Computer Science Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

Dr. Sherrie Gott
AFHRL/MODJ
Brooks AFB, TX 78235

Jordan Grafman, Ph.D.
Department of Clinical
Investigation
Walter Reed Army Medical Center
6825 Georgia Ave., N. W.
Washington, DC 20307-5001

Dr. Richard H. Granger
Department of Computer Science
University of California, Irvine
Irvine, CA 92717

Dr. Wayne Gray
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. James G. Greeno
University of California
Berkeley, CA 94720

H. William Greenup
Education Advisor (E031)
Education Center, MCDEC
Quantico, VA 22134

Ms. Glenda Greenwald
Human Intelligence Newsletter
P. O. Box 1163
Birmingham, MI 48012

Dipl. Pad. Michael W. Habon
Universitat Dusseldorf
Erziehungswissenschaftliches
Universitatsstr. 1
D-4000 Dusseldorf 1
WEST GERMANY

Dr. Henry M. Halff
Halff Resources, Inc.
4918 33rd Road, North
Arlington, VA 22207

Dr. Nancy F. Halff
Halff Resources, Inc.
4918 33rd Road, North
Arlington, VA 22207

Dr. Ronald K. Hambleton
Laboratory of Psychometric and
Evaluative Research
University of Massachusetts
Amherst, MA 01003

Dr. Cheryl Hamel
NTEC
Orlando, FL 32813

Dr. Ray Hannapel
Scientific and Engineering
Personnel and Education
National Science Foundation
Washington, DC 20550

Stevan Harnad
Editor, The Behavioral and
Brain Sciences
20 Nassau Street, Suite 240
Princeton, NJ 08540

Mr. William Hartung
PEAM Product Manager
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Wayne Harvey
SRI International
333 Ravenswood Ave.
Room B-S324
Menlo Park, CA 94025

DISTRIBUTION LIST FOR Carnegie-Mellon University/Anderson (NR 667-530)

Dr. Reid Hastie
Northwestern University
Department of Psychology
Evanston, IL 60201

Prof. John R. Hayes
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Barbara Hayes-Roth
Department of Computer Science
Stanford University
Stanford, CA 95305

Dr. Frederick Hayes-Roth
Teknowledge
525 University Ave.
Palo Alto, CA 94301

Dr. Joan I. Heller
Graduate Group in Science and
Mathematics Education
c/o School of Education
University of California
Berkeley, CA 94720

Dr. Jim Hollan
Code 51
Navy Personnel R & D Center
San Diego, CA 92152

Dr. Melissa Holland
Army Research Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Keith Holyoak
University of Michigan
Human Performance Center
330 Packard Road
Ann Arbor, MI 48109

Prof. Lutz F. Hornke
Universitat Dusseldorf
Erziehungswissenschaftliches
Universitatsstr. 1
Dusseldorf 1
WEST GERMANY

Mr. Dick Hoshaw
NAVOP-135
Arlington Annex
Room 2834
Washington, DC 20350

Dr. Steven Hunka
Department of Education
University of Alberta
Edmonton, Alberta
CANADA

Dr. Ed Hutchins
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Dillon Inouye
WICAT Education Institute
Provo, UT 84057

Dr. Zachary Jacobson
Bureau of Management Consulting
365 Laurier Avenue West
Ottawa, Ontario K1A 0S5
CANADA

Dr. Claude Janvier
Directeur, CIRADE
Universite' du Quebec a Montreal
Montreal, Quebec H3C 3P8
CANADA

Margaret Jerome
c/o Dr. Peter Chandler
83, The Drive
Hove
Sussex
UNITED KINGDOM

Dr. Joseph E. Johnson
Assistant Dean for
Graduate Studies
College of Science and Mathematics
University of South Carolina
Columbia, SC 29208

Col. Dominique Jouslin de Noray
Etat-Major de l'Armee de Terre
Centre de Relations Humaines
3 Avenue Octave Greard
75007 Paris
FRANCE

DISTRIBUTION LIST FOR Carnegie-Mellon University/Anderson (NR 667-530)

Dr. Marcel Just
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Milton S. Katz
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Norman J. Kerr
Chief of Naval Education
and Training
Code 00A2
Naval Air Station
Pensacola, FL 32508

Dr. Dennis Kibler
University of California
Department of Information
and Computer Science
Irvine, CA 92717

Dr. David Kieras
University of Michigan
Technical Communication
College of Engineering
1223 E. Engineering Building
Ann Arbor, MI 48109

Dr. Peter Kincaid
Training Analysis
& Evaluation Group
Department of the Navy
Orlando, FL 32813

Dr. Walter Kintsch
Department of Psychology
University of Colorado
Campus Box 345
Boulder, CO 80302

Dr. David Klahr
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Mazie Knerr
Program Manager
Training Research Division
HumRRO
1100 S. Washington
Alexandria, VA 22314

Dr. Janet L. Kolodner
Georgia Institute of Technology
School of Information
& Computer Science
Atlanta, GA 30332

Dr. Kenneth Kotovsky
Department of Psychology
Community College of
Allegheny County
800 Allegheny Avenue
Pittsburgh, PA 15233

Dr. David H. Krantz
2 Washington Square Village
Apt. # 15J
New York, NY 10012

Dr. Benjamin Kuipers
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139

Dr. Patrick Kyllonen
AFHRL/MOE
Brooks AFB, TX 78235

Dr. David R. Lambert
Naval Ocean Systems Center
Code 441T
271 Catalina Boulevard
San Diego, CA 92152

Dr. Pat Langley
University of California
Department of Information
and Computer Science
Irvine, CA 92717

M. Diane Langston
Communications Design Center
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

DISTRIBUTION LIST FOR Carnegie-Mellon University/Anderson (NR 667-530)

Dr. Kathleen LaPiana
Naval Health Sciences
Education and Training Command
Naval Medical Command,
National Capital Region
Bethesda, MD 20814-5022

Dr. Jill Larkin
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

Dr. Robert Lawler
Information Sciences, FRL
GTE Laboratories, Inc.
40 Sylvan Road
Waltham, MA 02254

Dr. Alan M. Lesgold
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Alan Leshner
Deputy Division Director
Behavioral and Neural Sciences
National Science Foundation
1800 G Street
Washington, DC 20550

Dr. Jim Levin
University of California
Laboratory for Comparative
Human Cognition
D003A
La Jolla, CA 92093

Dr. Clayton Lewis
University of Colorado
Department of Computer Science
Campus Box 430
Boulder, CO 80309

Science and Technology Division
Library of Congress
Washington, DC 20540

Dr. Charlotte Linde
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

Dr. Marcia C. Linn
Lawrence Hall of Science
University of California
Berkeley, CA 94720

Dr. William L. Maloy
Chief of Naval Education
and Training
Naval Air Station
Pensacola, FL 32508

Dr. Sandra P. Marshall
Department of Psychology
University of California
Santa Barbara, CA 93106

Dr. Manton M. Matthews
Department of Computer Science
University of South Carolina
Columbia, SC 29208

Dr. Richard E. Mayer
Department of Psychology
University of California
Santa Barbara, CA 93106

Dr. James McBride
Psychological Corporation
c/o Harcourt, Brace,
Javanovich Inc.
1250 West 6th Street
San Diego, CA 92101

Dr. Kathleen McKeown
Columbia University
Department of Computer Science
New York, NY 10027

Dr. James McMichael
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Barbara Means
Human Resources
Research Organization
1100 South Washington
Alexandria, VA 22314

Dr. Arthur Melmed
U. S. Department of Education
724 Brown
Washington, DC 20208

DISTRIBUTION LIST FOR Carnegie-Mellon University/Anderson (NR 667-530)

Dr. Al Meyrowitz
Office of Naval Research
Code 433
800 N. Quincy
Arlington, VA 22217-5000

Dr. Ryszard S. Michalski
University of Illinois
Department of Computer Science
1304 West Springfield Avenue
Urbana, IL 61801

Dr. George A. Miller
Department of Psychology
Green Hall
Princeton University
Princeton, NJ 08540

Dr. Lance A. Miller
IBM Thomas J. Watson
Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Dr. Mark Miller
Computer Thought Corporation
1721 West Plano Parkway
Plano, TX 75075

Dr. Andrew R. Molnar
Scientific and Engineering
Personnel and Education
National Science Foundation
Washington, DC 20550

Dr. William Montague
NPRDC Code 13
San Diego, CA 92152

Dr. Tom Moran
Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304

Dr. Allen Munro
Behavioral Technology
Laboratories - USC
1845 S. Elena Ave., 4th Floor
Redondo Beach, CA 90277

Director,
Decision Support
Systems Division, NMPC
Naval Military Personnel Command
N-164
Washington, DC 20370

Director,
Distribution Department, NMPC
N-4
Washington, DC 20370

Director,
Overseas Duty Support
Program, NMPC
N-62
Washington, DC 20370

Head, HRM Operations Branch,
NMPC
N-62F
Washington, DC 20370

Director,
Recreational Services
Division, NMPC
N-65
Washington, DC 20370

Assistant for Evaluation,
Analysis, and MIS, NMPC
N-6C
Washington, DC 20370

Spec. Asst. for Research, Experi-
mental & Academic Programs,
NTTC (Code 016)
NAS Memphis (75)
Millington, TN 38054

Director,
Research & Analysis Div.,
NAVCRUITCOM Code 22
4015 Wilson Blvd.
Arlington, VA 22203

Program Manager for Manpower,
Personnel, and Training,
NAVMAT 0722
Arlington, VA 22217-5000

DISTRIBUTION LIST FOR Carnegie-Mellon University/Anderson (NR 667-530)

Assistant for Long Range
Requirements,
CNO Executive Panel
NAVOP 00K
2000 North Beauregard Street
Alexandria, VA 22311

Assistant for Planning MANTRAPERS
NAVOP 01B6
Washington, DC 20370

Assistant for MPT Research,
Development and Studies
NAVOP 01B7
Washington, DC 20370

Head, Military Compensation
Policy Branch
NAVOP 134
Washington, DC 20370

Head,
Workforce Information Section,
NAVOP 140F
Washington, DC 20370

Head,
Family Support Program Branch,
NAVOP 156
1300 Wilson Blvd., Room 828
Arlington, VA 22209

Head, Economic Analysis Branch,
NAVOP 162
Washington, DC 20370

Head -- Manpower, Personnel,
Training, & Reserve Team,
NAVOP 914D
5A578, The Pentagon
Washington, DC 20350

Assistant for Personnel
Logistics Planning,
NAVOP 987H
5D772, The Pentagon
Washington, DC 20350

Leadership Management Education
and Training Project Officer,
Naval Medical Command
Code 05C
Washington, DC 20372

Technical Director,
Navy Health Research Ctr.
P.O. Box 85122
San Diego, CA 92138

Dr. Richard E. Nisbett
University of Michigan
Institute for Social Research
Room 5261
Ann Arbor, MI 48109

Dr. Donald A. Norman
Institute for Cognitive Science
University of California
La Jolla, CA 92093

Director, Training Laboratory,
NPRDC (Code 05)
San Diego, CA 92152

Director, Manpower and Personnel
Laboratory,
NPRDC (Code 06)
San Diego, CA 92152

Director, Human Factors
& Organizational Systems Lab,
NPRDC (Code 07)
San Diego, CA 92152

Fleet Support Office,
NPRDC (Code 301)
San Diego, CA 92152

Library, NPRDC
Code P201L
San Diego, CA 92152

Commanding Officer,
Naval Research Laboratory
Code 2627
Washington, DC 20390

Dr. Harry F. O'Neil, Jr.
Training Research Lab
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

DISTRIBUTION LIST FOR Carnegie-Mellon University/Anderson (NR 667-530)

Dr. Stellan Ohlsson
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Director, Technology Programs,
Office of Naval Research
Code 200
800 North Quincy Street
Arlington, VA 22217-5000

Director, Research Programs,
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5000

Office of Naval Research,
Code 433
800 N. Quincy Street
Arlington, VA 22217-5000

Office of Naval Research,
Code 442
800 N. Quincy St.
Arlington, VA 22217-5000

Office of Naval Research,
Code 442EP
800 N. Quincy Street
Arlington, VA 22217-5000

Group Psychology Program,
ONR Code 442GP
800 N. Quincy St.
Arlington, VA 22217-5000

Office of Naval Research,
Code 442PT
800 N. Quincy Street
Arlington, VA 22217-5000
(6 Copies)

Special Assistant for Marine
Corps Matters,
ONR Code 100M
800 N. Quincy St.
Arlington, VA 22217-5000

Dr. Judith Orasanu
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Prof. Seymour Papert
20C-109
Massachusetts Institute
of Technology
Cambridge, MA 02139

Lt. Col. (Dr.) David Payne
AFHRL
Brooks AFB, TX 78235

Dr. Douglas Pearse
DCIEM
Box 2000
Downsview, Ontario
CANADA

Dr. Robert Penn
NPRDC
San Diego, CA 92152

Dr. Nancy Pennington
University of Chicago
Graduate School of Business
1101 E. 58th St.
Chicago, IL 60637

Military Assistant for Training and
Personnel Technology,
OUSD (R & E)
Room 3D129, The Pentagon
Washington, DC 20301

Dr. Ray Perez
ARI (PERI-II)
5001 Eisenhower Avenue
Alexandria, VA 2233

Dr. David N. Perkins
Educational Technology Center
337 Gutman Library
Appian Way
Cambridge, MA 02138

Department of Computer Science,
Naval Postgraduate School
Monterey, CA 93940

Dr. Tjeerd Plomp
Twente University of Technology
Department of Education
P.O. Box 217
7500 AE ENSCHEDE
THE NETHERLANDS

DISTRIBUTION LIST FOR Carnegie-Mellon University/Anderson (NR 667-530)

Dr. Martha Polson
Department of Psychology
Campus Box 346
University of Colorado
Boulder, CO 80309

Dr. Peter Polson
University of Colorado
Department of Psychology
Boulder, CO 80309

Dr. Steven E. Poltrock
MCC
9430 Research Blvd.
Echelon Bldg #1
Austin, TX 78759-6509

Dr. Harry E. Pople
University of Pittsburgh
Decision Systems Laboratory
1360 Scaife Hall
Pittsburgh, PA 15261

Dr. Joseph Psotka
ATTN: PERI-1C
Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333

Dr. Lynne Reder
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. James A. Reggia
University of Maryland
School of Medicine
Department of Neurology
22 South Greene Street
Baltimore, MD 21201

Dr. Fred Reif
Physics Department
University of California
Berkeley, CA 94720

Dr. Lauren Resnick
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Dr. Mary S. Riley
Program in Cognitive Science
Center for Human Information
Processing
University of California
La Jolla, CA 92093

William Rizzo
Code 712 NAVTRAEQUIPCEN
Orlando, FL 32813

Dr. Andrew M. Rose
American Institutes
for Research
1055 Thomas Jefferson St., NW
Washington, DC 20007

Dr. William B. Rouse
Georgia Institute of Technology
School of Industrial & Systems
Engineering
Atlanta, GA 30332

Ms. Riitta Ruotsalainen
General Headquarters
Training Section
Military Psychology Office
PL 919
SF-00101 Helsinki 10, FINLAND

Dr. Michael J. Samet
Perceptronics, Inc
6271 Variel Avenue
Woodland Hills, CA 91364

Dr. Robert Sasmor
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Roger Schank
Yale University
Computer Science Department
P.O. Box 2158
New Haven, CT 06520

Mrs. Birgitte Schneidelbach
Forsvarets Center for Lederskab
Christianshavns Voldgade 8
1424 Kobenhavn K
DENMARK

DISTRIBUTION LIST FOR Carnegie-Mellon University/Anderson (NR 667-530)

Dr. Walter Schneider
University of Illinois
Psychology Department
603 E. Daniel
Champaign, IL 61820

Dr. Alan H. Schoenfeld
University of California
Department of Education
Berkeley, CA 94720

Dr. Janet Schofield
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Marc Sebrechts
Department of Psychology
Wesleyan University
Middletown, CT 06475

Dr. Judith Segal
Room 819F
NIE
1200 19th Street N.W.
Washington, DC 20208

Dr. Robert J. Seidel
US Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333

Dr. Ramsay W. Selden
NIE
Mail Stop 1241
1200 19th St., NW
Washington, DC 20208

Dr. W. Steve Sellman
OASD(MRA&L)
2B269 The Pentagon
Washington, DC 20301

Dr. Sylvia A. S. Shafto
National Institute of Education
1200 19th Street
Mail Stop 1806
Washington, DC 20208

Dr. Ted Shortliffe
Computer Science Department
Stanford University
Stanford, CA 94305

Dr. Lee Shulman
Stanford University
1040 Cathcart Way
Stanford, CA 94305

Dr. Randall Shumaker
Naval Research Laboratory
Code 7510
4555 Overlook Avenue, S.W.
Washington, DC 20375-5000

Dr. Miriam Shustack
Code 51
Navy Personnel R & D Center
San Diego, CA 92152

Dr. Robert S. Siegler
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Herbert A. Simon
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Zita M Simutis
Instructional Technology
Systems Area
ARI
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. H. Wallace Sinaiko
Manpower Research
and Advisory Services
Smithsonian Institution
801 North Pitt Street
Alexandria, VA 22314

Dr. Derek Sleeman
Stanford University
School of Education
Stanford, CA 94305

Dr. Edward E. Smith
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

DISTRIBUTION LIST FOR Carnegie-Mellon University/Anderson (NR 667-530)

Dr. Alfred F. Smode
Senior Scientist
Code 7B
Naval Training Equipment Center
Orlando, FL 32813

Dr. Richard Snow
Liaison Scientist
Office of Naval Research
Branch Office, London
Box 39
FPO New York, NY 09510

Dr. Elliot Soloway
Yale University
Computer Science Department
P.O. Box 2158
New Haven, CT 06520

Dr. Richard Sorensen
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Kathryn T. Spoehr
Brown University
Department of Psychology
Providence, RI 02912

Dr. Robert Sternberg
Department of Psychology
Yale University
Box 11A, Yale Station
New Haven, CT 06520

Dr. Albert Stevens
Bolt Beranek & Newman, Inc.
10 Moulton St.
Cambridge, MA 02238

Dr. Paul J. Sticha
Senior Staff Scientist
Training Research Division
HumRRO
1100 S. Washington
Alexandria, VA 22314

Dr. Thomas Sticht
Navy Personnel R&D Center
San Diego, CA 92152

Cdr Michael Suman, PD 303
Naval Training Equipment Center
Code N51, Comptroller
Orlando, FL 32813

Dr. Hariharan Swaminathan
Laboratory of Psychometric and
Evaluation Research
School of Education
University of Massachusetts
Amherst, MA 01003

Dr. John Tangney
AFOSR/NL
Bolling AFB, DC 20332

Dr. Kikumi Tatsuoka
CERL
252 Engineering Research
Laboratory
Urbana, IL 61801

Dr. Maurice Tatsuoka
220 Education Bldg
1310 S. Sixth St.
Champaign, IL 61820

Dr. Martin M. Taylor
DCIEM
Box 2000
Downsview, Ontario
CANADA

Dr. Perry W. Thorndyke
FMC Corporation
Central Engineering Labs
1185 Coleman Avenue, Box 580
Santa Clara, CA 95052

Dr. Douglas Towne
Behavioral Technology Labs
1845 S. Elena Ave.
Redondo Beach, CA 90277

Dr. James Tweeddale
Technical Director
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Paul Twohig
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

DISTRIBUTION LIST FOR Carnegie-Mellon University/Anderson (NR 667-530)

Headquarters, U. S. Marine Corps
Code MPI-20
Washington, DC 20380

Dr. Kurt Van Lehn
Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304

Dr. Beth Warren
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

Roger Weissinger-Baylon
Department of Administrative
Sciences
Naval Postgraduate School
Monterey, CA 93940

Dr. Keith T. Wescourt
FMC Corporation
Central Engineering Labs
1185 Coleman Ave., Box 580
Santa Clara, CA 95052

Dr. Douglas Wetzel
Code 12
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Barbara White
Bolt Beranek & Newman, Inc.
10 Moulton Street
Cambridge, MA 02238

Dr. Mike Williams
IntelliGenetics
124 University Avenue
Palo Alto, CA 94301

Dr. Robert A. Wisher
U.S. Army Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Martin F. Wiskoff
Navy Personnel R & D Center
San Diego, CA 92152

Mr. John H. Wolfe
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Wallace Wulfeck, III
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Joe Yasatuke
AFHRL/LRT
Lowry AFB, CO 80230

Dr. Masoud Yazdani
Dept. of Computer Science
University of Exeter
Exeter EX4 4QL
Devon, ENGLAND

Mr. Carl York
System Development Foundation
181 Lytton Avenue
Suite 210
Palo Alto, CA 94301

Dr. Joseph L. Young
Memory & Cognitive
Processes
National Science Foundation
Washington, DC 20550

Dr. Steven Zornetzer
Office of Naval Research
Code 440
800 N. Quincy St.
Arlington, VA 22217-5000

Dr. Michael J. Zyda
Naval Postgraduate School
Code 52CK
Monterey, CA 93943

END

FILMED

10-85

DTIC