Psychological Review
1982, Vol. 89, No. 4, 369-406

0033-295X/82/8904-0369$00.75

Acquisition of Cognitive Skill

John R. Anderson
Carnegie-Mellon University

A framework for skill acquisition is proposed that includes two major stages in the development of a cognitive skill: a declarative stage in which facts about the skill domain are interpreted and a procedural stage in which the domain knowledge is directly embodied in procedures for performing the skill. This general framework has been instantiated in the ACT system in which facts are encoded in a propositional network and procedures are encoded as productions. Knowledge compilation is the process by which the skill transits from the declarative stage to the procedural stage. It consists of the subprocesses of composition, which collapses sequences of productions into single productions, and proceduralization, which embeds factual knowledge into productions. Once proceduralized, further learning processes operate on the skill to make the productions more selective in their range of applications. These processes include generalization, discrimination, and strengthening of productions. Comparisons are made to similar concepts from past learning theories. How these learning mechanisms apply to produce the power law speedup in processing time with practice is discussed.

It requires at least 100 hours of learning and practice to acquire any significant cognitive skill to a reasonable degree of proficiency. For instance, after 100 hours a student learning to program a computer has achieved only a very modest facility in the skill. Learning one's primary language takes tens of thousands of hours. The psychology of human learning has been very thin in ideas about what happens to skills under the impact of this amount of learning—and for obvious reasons. This article presents a theory about the changes in the nature of a skill over such large time scales and about the basic learning processes that are responsible.

This research was sponsored by the Personnel and Training Research Programs, Psychological Sciences Division, Office of Naval Research, under Contract N00014-81-C-0335, Contract Authority Identification Number, NR 157-465, and Grant IST-80-15357 from the National Science Foundation. My ability to put together this theory has depended critically on input from colleagues I have worked with over the past few years—Charles Beasley, Jim Greeno, Paul Kline, Pat Langley, and David Neves. This is not to suggest that any of the above would endorse all of the ideas in this paper. I would like to thank those who have provided me with valuable advice and feedback on the paper: Renee Elio, Jim Greeno, Paul Kline, Jill Larkin, Clayton Lewis, Miriam Schustack, and especially Lynne Reder.

Requests for reprints should be sent to John Anderson, Department of Psychology, Carnegie-Mellon University, Schenley Park, Pittsburgh, Pennsylvania 15213.

Fitts (1964) considered the process of skill acquisition to fall into three stages of development. The first stage, called the *cognitive stage*, involves an initial encoding of the skill into a form sufficient to permit the learner to generate the desired behavior to at least some crude approximation. In this stage it is common to observe verbal mediation in which the learner rehearses information required for the execution of the skill. The second stage, called the *associative stage*, involves the "smoothing out" of the skill performance. Errors in the initial understanding of the skill are gradually detected and eliminated. Concomitantly, there is a dropout of verbal mediation. The third stage, the *autonomous stage*, is one of gradual continued improvement in the performance of the skill. The improvements in this stage often continue indefinitely. Although these general observations about the course of skill development seem true for a wide range of skills, they have defied systematic theoretical analysis.

The theory to be presented in this article is in keeping with these general observations of Fitts (1964) and provides an explanation of the phenomena associated with his three stages. In fact, the three major sections of this article correspond to the three stages. In the first stage the learner receives instruc-

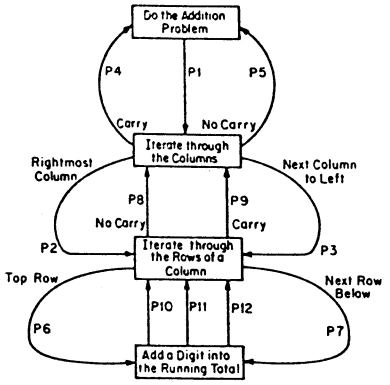


Figure 1. A representation of the flow of control in Table 1 between various goals. (The boxes correspond to goal states, and the arrows to productions that can change these states. Control starts with the top goal.)

tion and information about a skill. The instruction is encoded as a set of facts about the skill. These facts can be used by general interpretive procedures to generate behavior. This initial stage of skill corresponds to Fitts's cognitive stage. In this article it will be referred to as the *declarative stage.* Verbal mediation is frequently observed because the facts have to be rehearsed in working memory to keep them available for the interpretive procedures.

According to the theory to be presented here, Fitts's second stage is really a transition between the declarative stage and a later stage. With practice the knowledge is converted into a procedural form in which it is directly applied without the intercession of other interpretive procedures. The gradual process by which the knowledge is converted from declarative to procedural form is called *knowledge compilation.* Fitts's associative stage corresponds to the period over which knowledge compilation applies.

According to the theory, Fitts's autonomous stage involves further learning that occurs after the knowledge achieves procedural form. In particular, there is further tuning of the knowledge so that it will apply more appropriately, and there is a gradual process of speedup. This will be called the *procedural stage* in this article.

This article presents a detailed theory about the use and development of knowledge in both the declarative and procedural form and about the transition between these two forms. The theory is based on the ACT production system (Anderson, 1976) in which the distinction between procedural and declarative knowledge is fundamental. Procedural knowledge is represented as productions, whereas declarative knowledge is represented as a propositional network. Before describing the theory of skill acquisition, it will be necessary to specify some of the basic operating principles of the ACT production system.

The ACT Production System

The ACT production system consists of a set of productions that can operate on facts in the declarative data base. Each production has the form of a primitive rule that specifies a cognitive contingency, that is, a production specifies when a cognitive act should take place. The production has a condition that specifies the circumstances under which the production can apply and an action that specifies what should be done when the production applies. The sequence of productions that apply in a task correspond to the cognitive steps taken in performing the task. In the actual computer simulations, these production rules have often quite technical syntax, but in this article I will usually give the rules quite English-like renditions. For current purposes application of a production can be thought of as a step of cognition. Much of the ACT performance theory is concerned with specifying how productions are selected to apply, and much of the ACT learning theory is concerned with how these production rules are acquired.

An Example

To explain some of the basic concepts of the ACT production system, it is useful to have an illustrative set of productions that performs some simple task. Such a set of productions for performing addition is given in Table 1. Figure 1 illustrates the flow of control in that production set among goals. It is easiest to understand such a production system by tracing its application to a problem such as adding a column of numbers:

Table 1
A Production System for Performing Addition

P1. IF the goal is to do an addition problem,
THEN the subgoal is to iterate through the columns of the problem.

P2. IF the goal is to iterate through the columns of an addition problem
and the rightmost column has not been processed,
THEN the subgoal is to iterate through the rows of that rightmost column
and set the running total to zero.

P3. IF the goal is to iterate through the columns of an addition problem
and a column has just been processed
and another column is to the left of this column,
THEN the subgoal is to iterate through the rows of this column to the left
and set the running total to the carry.

P4. IF the goal is to iterate through the columns of an addition problem
and the last column has been processed
and there is a carry,
THEN write out the carry
and POP the goal.

P5. IF the goal is to iterate through the columns of an addition problem
and the last column has been processed
and there is no carry,
THEN POP the goal.

P6. IF the goal is to iterate through the rows of a column
and the top row has not been processed,
THEN the subgoal is to add the digit of the top row into the running total.

P7. IF the goal is to iterate through the rows of a column
and a row has just been processed
and another row is below it,
THEN the subgoal is to add the digit of the lower row to the running total.

P8. IF the goal is to iterate through the rows of a column
and the last row has been processed
and the running total is a digit,
THEN write the digit
and delete the carry
and mark the column as processed
and POP the goal.

P9. IF the goal is to iterate through the rows of a column
and the last row has been processed
and the running total is of the form "string + digit,"
THEN write the digit
and set carry to the string
and mark the column as processed
and POP the goal.

P10. IF the goal is to add a digit to a number
and the number is a digit
and a sum is the sum of the two digits,
THEN the result is the sum
and mark the digit as processed
and POP the goal.

P11. IF the goal is to add a digit to a number
and the number is of the form "string + digit"
and a sum is the sum of the two digits
and the sum is less than 10,
THEN the result is "string + sum"
and mark the digit as processed
and POP the goal.

P12. IF the goal is to add a digit to a number
and the number is of the form "string + digit"
and a sum is the sum of the two digits
and the sum is of the form "1 + digit*"
and another number sum* is the sum of 1 plus string,
THEN the result is "sum* + digit*"
and mark the digit as processed
and POP the goal.

614
438
683.

Production P1 is the first to apply and would set as a subgoal to iterate through the columns. Then Production P2 applies and changes the subgoal to adding the digits of the rightmost column. It also sets the running total to 0. Then Production P6 applies to set the new subgoal to adding the top digit of the column (4) to the running total. In terms of Figure 1, this sequence of three productions has moved the system down from the top goal of doing the problem to the bottom goal of performing a basic addition operation. The system has the four goals in Figure 1 stacked with attention focused on the bottom goal.

At this point Production P10 applies, which calculates 4 as the new value of the running total and POPs the goal of adding

the digit to the running total. This amounts to removing this goal from the stack and returning attention to the goal of iterating through the rows of the column. Then P7 applies, which sets the new subgoal of adding 8 into the running total. P10 applies again to change the running total to 12; then P7 applies to create the subgoal of adding 3 into the running total; then P11 calculates the new running total as 15. At this point the system is back at the goal of iterating through the rows and has processed the bottom row of the column. Then Production P9 applies, which writes out the 5 in 15, sets the carry to the 1, and POPs back to the goal of iterating through the columns. At this point the production system has processed one column of the problem.

I will not trace out any further the application of this production set to the problem, but the reader is invited to carry out the hand simulation. Note that Productions P2–P5 form a subroutine for iterating through the columns, Productions P6–P9 an embedded subroutine for processing a column, and Productions P10–P12 an embedded subroutine for adding a digit to the running total. In Figure 1 all the productions corresponding to a subroutine emanate from the same goal box.

Significant Features of the Performance System

A number of features of the production system are important for the learning theory to be presented. The productions themselves are the system's procedural component. For a production to apply, the clauses specified in its condition must be matched against information active in working memory. This information in working memory is part of the system's declarative component. Elsewhere (Anderson, 1976, 1980) I have discussed the network encoding of that declarative knowledge and the process of spreading activation defined on that network.

Goal structure. As noted above, the productions in Table 1 are organized into subroutines, where each subroutine is associated with a goal state that all the productions in the subroutine are trying to achieve. Because the system can have only one goal at any moment in time, productions from only one of these subroutines can apply at any one time. This forces a considerable seriality into the behavior of the system. These goal-seeking productions are hierarchically organized. The idea that hierarchical structure is fundamental to human cognition has been emphasized by Miller, Galanter, and Pribram (1960) and many others. Greeno (1976) used the idea of goal structuring for production systems. Brown and Van Lehn (1980) have recently introduced a similar goal structuring for production systems.

In the original ACT system (Anderson, 1976), there was a scheme for achieving the effect of subroutines by the setting of control variables. There are several important differences between the current scheme and that older one. First, as noted, the current scheme forces a strong degree of seriality into the system. Second, because the goals are not arbitrary nodes but rather meaningful assertions, it is much easier for ACT's learning system to acquire new productions that make reference to goals. Evidence for this last assertion will be given as the various ACT learning mechanisms are discussed.

In achieving a hierarchical subroutine structure by means of a goal–subgoal structure, I am accepting the claim that the hierarchical control of behavior derives from the structure of problem solving. This amounts to making the assertion that problem solving, and the goal structure it produces, is a fundamental category of cognition. This assertion has been advanced by others (e.g., Newell, 1980). Thus, this learning discussion contains a rather strong presupposition about the architecture of cognition. I think the presupposition is too abstract to be defended directly; rather, evidence for it will come from the fruitfulness of the systems that we can build based on the architectural assumption.

Conflict resolution. Every production system requires some rules of conflict resolution, that is, principles for deciding which of those productions that match will be executed. ACT has a set of conflict-resolution principles that can be seen as variants of the 1976 ACT (Anderson, 1976) or the OPS system (Forgy & McDermott, 1977). One powerful principle is refractoriness: that the

same production cannot apply to the same data in working memory twice in the same way. This prevents the same production from repeating over and over again and was implicit in the preceding hand simulation of Table 1.

The two other principles of conflict resolution in ACT are specificity and strength. Neither was illustrated in Table 1 but both are important to understanding the learning discussion. If two productions can apply and the condition of one is more specific than the other, then the more specific production takes precedence. Condition A is more specific than Condition B if the set of situations in which Condition A can match is a proper subset of the set of situations where Condition B can match. The specificity rule allows exceptions to general rules to apply because these exceptions will have more specific conditions. For instance, suppose we had the following pair of productions:

PA. IF the goal is to generate the plural of man,
THEN say "MEN."

PB. IF the goal is to generate the plural of a noun,
THEN say "noun + s."

The condition of Production PA is more specific than the condition of Production PB and so will apply over the general pluralization rule.

Each production has a strength that reflects the frequency with which that production has been successfully applied. I will describe the rules that determine strength accumulation later in this article; here I describe the role of production strength in conflict resolution. Elsewhere (e.g., Anderson, 1976; Anderson, Kline, & Beasley, 1979) I have given a version of this role of strength that assumes discrete time intervals. Here I give a continuous version. Productions are indexed by the constants in their conditions. For instance, the Production PA above would be indexed by *plural* and *man*. If these concepts are active in working memory, the production will be selected for consideration. In this way ACT can focus its attention on just the subset of productions that may be potentially relevant. Only if a production is selected is a test made to see if its condition is satisfied. (For future reference if a production is selected, it is said to be on the APPLYLIST.) A production takes a time T_1 to be selected and another time T_2 to be tested and to apply. The selection time T_1 varies with the production's strength, whereas the application time is a constant over productions. It is further assumed that the time T_1 for the production to be selected will randomly vary from selection to selection. The expected time is a/s where s is the production strength and a is a constant. Although there are no compelling reasons for making any assumption about the distribution, we have assumed that T_1 has an exponential distribution, and this is its form in all our simulations.

A production will actually apply if it is selected and it has completed application before a more specific production is selected. This provides the relationship between strength and specificity in the theory. A more specific production will take precedence over a more general production only if its selection time is less than the selection plus application times of the more general production. Because strength reflects frequency of practice, only exceptions that have some criterion frequency will be able to reliably take precedence over general rules. This corresponds, for instance, to the fact that words with irregular inflections tend to be of relatively high frequency. It is possible for an exception to be of borderline strength so that it sometimes is selected in time to beat out the general rule but sometimes not. This corresponds, for instance, to the stage in language development when an irregular inflection is being used with only partial reliability (R. Brown, 1973).

Variables. Productions contain variable slots that can take on different values in different situations. The use of these variables is often implicit, as in Table 1, but sometimes it is important to acknowledge the variables that are being assumed. As an illustration let us consider a variabilized form of a production from Table 1. If Production P9 from that table were to be written in a way to expose its variable structure, it would have the form below, where the terms prefixed by LV are local variables:

IF the goal is to iterate through the rows of LVcolumn
and LVrow is the last row of LVcolumn

and LVrow has been processed
and the running total is of the form "LVstring + LVdigit,"
THEN write LVdigit
and set carry to LVstring
and mark LVcolumn as processed
and POP the goal.

Local variables can be reassigned to new values each time the production applies. Thus, for instance, the terms *LVcolumn*, *LVrow*, *LVstring*, and *LVdigit* will match to whatever elements lead to a complete match of the condition to working memory. Suppose, for instance, that the following elements were in working memory:

The goal is to iterate through the rows of Column 2.
Row *x* is the last row of Column 2.
Row *x* has been processed.
Running total is of the form 2 + 4.

The production would match this working memory information with the following variable bindings:

LVcolumn = Column 2.
LVrow = Row *x*.
LV string = 2.
LVdigit = 4.

Local variables assume values within a production for the purposes of matching the condition and executing the action. After application of the production, variables lose their values.

Learning in ACT

This article is concerned with the processes underlying the acquisition of cognitive skill. As is clear from examples like Table 1, there is a closer connection in ACT between productions and skill performance than between declarative knowledge and skill performance. This is because the control over cognition and behavior lies directly in the productions. Facts are used by the productions. So in a real sense facts are instruments of the productions, which are the agents. For instance, we saw that Production P10 used the addition fact that 4 + 8 = 12. Although productions are closer to performance than facts, I will be claiming that when a person initially learns about a skill, he or she learns only facts about the skill and does not directly acquire productions. These facts are used interpretively by general-purpose productions. The first major section of this article, on the declarative stage, will illustrate how general-purpose productions can interpret these facts to generate performance of the skill.

The next major section of the article will discuss the evidence for and nature of the knowledge compilation process that results in the translation from a declarative base for a skill to a procedural base for the skill. (For instance, the production set in Table 1 is a procedural base for the addition skill.) After this section the remainder of the article will discuss two aspects of the continued improvement of a skill after it has achieved a procedural embodiment: one is the tuning of skill so that it is applied more appropriately; the second is the very lawful way in which application of the skill speeds up.

The Declarative Stage: Interpretive Procedures

One of the things that becomes apparent in studying the initial stages of skill acquisition in areas of mathematics like geometry or algebra (e.g., Neves, 1981) is that the instruction seldom if ever directly specifies

1-7 *Proofs in Two-Column Form*

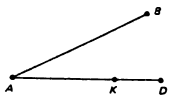
You prove a statement in geometry by using deductive reasoning to show that the statement follows from the hypothesis and other accepted material. Often the assertions made in a proof are listed in one column, and reasons which support the assertions are listed in an adjacent column.

EXAMPLE. A proof in two-column form.

Given: $\overline{AKD}$; $AD = AB$

Prove: $AK + KD = AB$

Proof:



STATEMENTS	REASONS
1. $\overline{AKD}$	1. Given
2. $AK + KD = AD$	2. Definition of between
3. $AD = AB$	3. Given
4. $AK + KD = AB$	4. Transitive property of equality

Some people prefer to support Statement 4, above, with the reason *The Substitution Principle*. Both reasons are correct.

The reasons used in the example are of three types: *Given* (Steps 1 and 3), *Definition* (Step 2), and *Postulate* (Step 4). Just one other kind of reason, *Theorem*, can be used in a mathematical proof. Postulates and theorems from both algebra and geometry can be used.

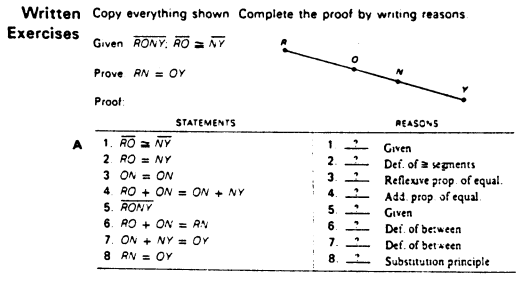
Reasons Used in Proofs

Given (Facts provided for a particular problem)
Definitions
Postulates
Theorems that have already been proved.

Figure 2. The text instruction for a two-column proof. (From *Geometry* by R. C. Jurgensen, A. J. Donnelly, J. E. Maier, and G. R. Rising. Boston: Houghton Mifflin, 1975, p. 25. Copyright 1975 by Houghton Mifflin Co. Reprinted by permission.)

a procedure to be applied. Still, the student is able to emerge from this type of instruction with an ability to generate behavior that reflects knowledge contained in the instruction. Figures 2, 3, and 4 from my work on geometry illustrate this point. Figure 2 is taken from the text of Jurgensen, Donnelly, Maier, & Rising (1975) and represents the total of that text's instruction on two-column proofs. Immediately after studying this, three of our students attempted to give reasons for two-column proof problems. The first such proof problem is the one illustrated in Figure 3. All three of the students were able to deal with this problem with some success. Behavior on these reason-giving problems is rather constant across subjects, at least at a global level. Figure 4 is a representation at the global level of these constancies. Clearly, nowhere in Figure 2 is there a specification of the flow of control that is in Figure 4. However, before reading the instruction of Figure 2, subjects were not capable of the flow of control in Figure 4, and after reading the instruction they were. So somehow the instruction in Figure 2 makes the procedure in Figure 4 possible.

Given that the instructions do not specify flow of control, the learners must call on existing procedures to direct their behavior in this task. These procedures must use the information specified in the instructions to guide their behavior. The instructional information is being used by these procedures in the same way Production P10 in Table 1



Written Exercises Copy everything shown. Complete the proof by writing reasons.

Given $\overline{RONY}$; $\overline{RO} \cong \overline{NY}$

Prove $RN = OY$

Proof:

	STATEMENTS	REASONS
A	1. $\overline{RO} \cong \overline{NY}$	1. ? Given
	2. $RO = NY$	2. ? Def. of ≅ segments
	3. $ON = ON$	3. ? Reflexive prop. of equal.
	4. $RO + ON = ON + NY$	4. ? Add. prop. of equal.
	5. $\overline{RONY}$	5. ? Given
	6. $RO + ON = RN$	6. ? Def. of between
	7. $ON + NY = OY$	7. ? Def. of between
	8. $RN = OY$	8. ? Substitution principle

Figure 3. A reason-giving task that is the first problem that the student encounters requiring use of the knowledge about two-column proofs. (This figure is taken from the instructors' copy of the text and shows the correct reasons. These reasons are not given in the students' text. From *Geometry* by R. C. Jurgensen, A. J. Donnelly, J. E. Maier, and G. R. Rising. Boston: Houghton Mifflin, 1975, p. 26. Copyright 1975 by Houghton Mifflin Co. Reprinted by permission.)

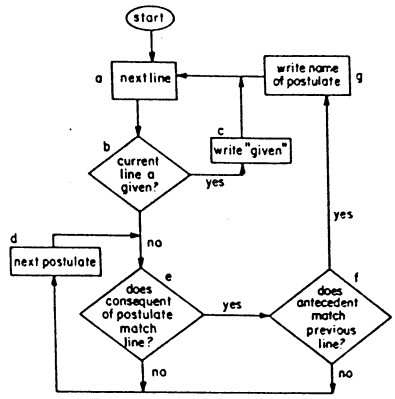


Figure 4. A flowchart showing the general flow of control in a reason-giving task.

used the addition facts to guide its behavior. For this reason, I say that the knowledge about the skill is being used *interpretatively*. The term reflects the fact that the knowledge is data for other procedures in just the way a computer program is data for an interpreter.

The basic claim is that general interpretative procedures with no domain-specific knowledge can be applied to some facts about the domain and produce coherent and domain-appropriate behavior. On first consideration it was not obvious to many people, including myself, how it was possible to model novice behavior in a task like Figure 3 by a purely interpretive system. However, it is absolutely critical to the theory presented here that there be an interpretive system in ACT that accurately describes the behavior of a novice in a new domain. It is essential because the theory claims that knowledge in a new domain always starts out in declarative form and is used interpretively. Therefore, I took it as a critical test case to develop a detailed example of how a student, extracting only declarative information from Figure 2, could generate task-appropriate behavior in Figure 3. The example, described below, assumes some general-purpose working-backwards problem-solving techniques. It captures important aspects of the behavior of my three high school students on this problem.

I want to emphasize that I am not claiming that such general problem-solving procedures are the only way that learners can bridge the gap between instruction and be-

Table 2
Interpretive Productions Evoked in Performing the Reason-Giving Task

P1. IF the goal is to do a list of problems,
THEN set as a subgoal to do the first problem in the list.

P2. IF the goal is to do a list of problems
and a problem has just been finished,
THEN set as a subgoal to do the next problem.

P3. IF the goal is to do a list of problems
and there are no unfinished problems on the list,
THEN POP the goal with success.

P4. IF the goal is to write the name of a relation for an argument,
THEN set as a subgoal to find what the relation is for the argument.

P5. IF the goal is to write the name of a relation for an argument
and a name has been found,
THEN write the name
and POP the goal with success.

P6. IF the goal is to write the name of a relation for an argument
and no name has been found,
THEN POP the goal with failure.

P7. IF the goal is to find a relation
and there is a list of methods for achieving the relation,
THEN set as a subgoal to try the first method.

P8. IF the goal is to find a relation
and there is a list of methods for achieving the relation
and a method has just been unsuccessfully tried,
THEN set as a subgoal to try the next method.

P9. IF the goal is to find a relation
and there is a list of methods for achieving the relation
and a method has been successfully tried,
THEN POP the goal with success.

P10. IF the goal is to find a relation
and there is a list of methods for achieving the relation
and they have all proven unsuccessful,
THEN POP the goal with failure.

P11. IF the goal is to try a method
and that method involves establishing a relationship.
THEN set as a subgoal to establish the relationship.

P12. IF the goal is to try a method
and the subgoal was a success,
THEN POP the goal with success.

P13. IF the goal is to try a method
and the subgoal was a failure,
THEN POP the goal with failure.

P14. IF the goal is to establish that a statement is among a list
and the list contains the statement,
THEN POP the goal with success.

P15. IF the goal is to establish that a statement is among a list
and the list does not contain the statement,
THEN POP the goal with failure.

P16. IF the goal is to establish that a line is implied by a rule in a set
and the set contains a rule of the form "consequent if antecedents"
and the consequent matches the line,
THEN set as a subgoal to determine if the antecedents correspond to established statements
and tag the rule as tried.

P17. IF the goal is to establish that a line is implied by a rule in a set
and the set contains a rule of the form "consequent if antecedents"
and the consequent matches the line
and the antecedents have been established,
THEN POP the goal with success.

P18. IF the goal is to establish that a line is implied by a rule in a set
and there is no untried rule in the set that matches the line,
THEN POP the goal with failure.

P19. IF the goal is to determine if antecedents correspond to established statements
and there is an unestablished antecedent clause
and the clause matches an established statement,
THEN tag the clause as established.

P20. IF the goal is to determine if antecedents correspond to established statements
and there are no unestablished antecedent clauses,
THEN POP the goal with success.

P21. IF the goal is to determine if antecedents correspond to established statements
and there is an unestablished antecedent clause
and it matches no established statement,
THEN POP the goal with failure.

havior. There are probably many different types of interpretive procedures. From my work on geometry, it is clear, for instance, that procedures for making analogies between worked-out examples and new tasks serve as an additional, important means for bridging the gap.

Application of General Problem-Solving Methods

Even though students coming upon the instruction in Figure 2 have no procedures specific to doing two-column proof problems, they have procedures for solving problems in general, for doing mathematicslike exercises, and perhaps even for certain types of deductive reasoning. These general problem-solving procedures can use the instruction such as that in Figure 2 as data for generating task-appropriate behavior when faced with a problem like that in Figure 3. They serve as the procedures for interpreting the task-specific example. Table 2 provides a list of the set of productions that embody the needed problem-solving procedures, and Figure 5 illustrates their flow of control when applied to problems like the one in Figure 2.

I will trace the application of this production set to the first two lines in Figure 5. It is assumed that the student encodes (declaratively) the exercise in Figure 3 as a list of subproblems, where each subproblem is to write a reason for a line of the proof. Production P1 from Table 2 matches to this list encoding of the problems. Therefore, P1 applies first and focuses attention on the first subproblem, that is, it sets as the subgoal to write a reason for $\overline{RO} \cong \overline{NY}$. Next, Production P4 applies. P4's condition, "the goal is to write the name of a relation for an argument," matches the current subgoal "to write the name of the reason for $\overline{RO} \cong \overline{NY}$." P4 creates the subgoal of finding a reason for the line. P4 is quite general and reflects the existence of a prior procedure for writing statements that satisfy a constraint.

The student presumably has encoded the boxed information in Figure 2 as indicating a list of methods for providing a reason for a line. If so, Production P7 applies next and

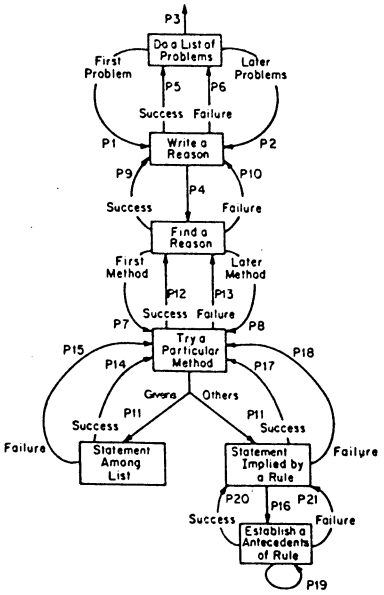


Figure 5. A representation of the flow of control in Table 2 between the various goals. (Control starts with the top goal.)

sets as a subgoal to try givens (the first rule on the reason list) as a justification for the current line. Note this is one point where a fragment of the instruction is used by a general problem-solving procedure (in this case for searching a list of methods) to determine the course of behavior. Two of the students in fact went back and reviewed the methods when they started this problem.

The students I studied had extracted from the instruction in Figure 2 that the "given" reason is used when the line to be justified is among the givens of the problem. Note that this fact is not explicitly stated in the instruction but is strongly implied. Thus, it is assumed that the student has encoded the fact that "the givens method involves establishing that the statement is among the givens." Production P11 will match this fact in its condition and so will set as a subgoal to establish that $\overline{RO} \cong \overline{NY}$ is among the givens of the problem. Production P14 models the successful recognition that $\overline{RO} \cong \overline{NY}$ is among the givens and returns a success from the subgoal. That is, its action "POP the goal with success" tags the goal "to find $\overline{RO} \cong \overline{NY}$ among the givens" with success and sets as the current goal the higher goal of trying

the givens method. Then P12 and P9 POP success back up to the next-to-top-level goal of writing a reason for the line. Then Production P5 applies to write *given* as a reason and POPs back to the top-level goal. In fact, all of the students likewise scanned the given list and had no difficulty with the first line of this proof.

At this point Production P2 applies to set the subgoal of writing a reason for the second line, RO = NY. Then Productions P4, P7, and P11 apply in that order, setting the subgoal of seeing whether RO = NY was among the givens of the problem. Production P15 recognizes this as a failed goal and then Production P13 returns control back to the level of choosing methods to establish a reason. Production P8 selects the definition reason to try next. Thus, the production set first unsuccessfully tries the given method before trying other methods. One of the students explicitly verbalized trying givens and failed. There was no indication in the protocol of our other students that givens was considered first, although it may have been implicitly.

Clearly, the instruction in Figure 2 contains no explanation of how a definition should be applied. However, the assumption of the text is that the student knows that a definition should imply the statement. There were some earlier exercises on conditional and biconditional statements that makes this assumption at least conceivable. All three subjects knew that some inferencelike activity was required, but they had a faulty understanding of the nature of the application of inference to this task. In any case, assuming that the student knows as a fact (in contrast to a procedure) that use of definitions involves inferential reasoning, Production P11 will match in its condition the fact that "definitions involve establishing that the statement is implied by a definition," and P11 will set the subgoal of proving that RO = NY was implied by a definition.

At this point I have to leave these students behind momentarily and describe the ideal student. The textbook assumes that the student already has a functioning procedure for finding a rule that implies a statement by means of a set of established rules. Productions P16–P21 constitute such a procedure. None of my students had this procedure in its entirety. These productions work as a general inference-testing procedure and apply equally well to postulates and theorems as well as definitions. Production P16 selects a conditional rule that matches the current line (the exact details of the match are not unpacked in Table 2). It is assumed that a biconditional definition is encoded as two implications each of the form "consequent if antecedent." The definition relevant to the current Line 2 is that two line segments are congruent if and only if they are of equal measure, which is encoded as

$$\underline{XZ} = \underline{UV} \text{ if } \overline{XZ} \cong \overline{UV}$$
$$\text{and } \overline{XZ} \cong \overline{UV} \text{ if } XZ = UV.$$

The first implication is the one that is selected, and the subgoal is set to establish the antecedent $\overline{XZ} \cong \overline{UV}$ (or $\overline{RO} \cong \overline{NY}$, in the current instantiation). The production set P19–P21 describes a procedure for matching zero or more clauses in the antecedent of a rule. In this case P19 finds a match to the one condition, $\overline{XZ} \cong \overline{UV}$, with $\overline{RO} \cong \overline{NY}$ in the first line. Then P20 POPs with success followed by successful POPping of P17, then P12, and then P9, which returns the system to the goal of writing out a reason for the line.

Significant features of the example. I will not further trace the application of the production set to the example. I would like to identify, however, the essential aspects of how this production set allows the student to bridge the gap between instruction and the problem demands. Figure 5 illustrates the flow of control with each box being a level in the goal structure and serving as a subroutine. Although it is not transparent, the subgoal organization in Figure 5 results in the same flow of control as the flowchart organization of Figure 4. However, as the production rendition of Figure 5 establishes, the flow of control in Figure 5 is not something fixed ahead of time but rather emerges in response to the instruction and the problem statement.

The top level goal in Figure 5 of iterating through a list of problems is provided by the problem statement and, given the problem statement, it is unpacked into a set of

subgoals to write statements indicating the reasons for each line. This top level procedure reflects a general strategy the student has for decomposing problems into linearly ordered subproblems. Then another prior routine sets as subgoals to find the reasons. At this point the instruction about the list of acceptable relationships is called into play (through yet another prior problem-solving procedure) and is used to set a series of subgoals to try out the various possible reasons for a statement. So the unpacking of subgoals in Figure 5 from "do a list of problems" to "find a reason" is in response to the problem statement; the further unpacking into the methods of givens, postulates, definitions, and theorems is in response to the instruction. The instruction is the source of information identifying that the method of givens involves searching the given list, and the other methods involve application of inferential reasoning. The ability to search a list for a match is assumed by the text, reasonably enough, as a prior procedure on the part of the student. The ability to apply inferential reasoning is also assumed as a prior procedure, but in this case the assumption is mistaken.

In summary, then, we see in Figure 5 a set of separate problem-solving procedures that are joined together in a novel combination by the declarative information in the problem statement and instruction. In this sense the student's general problem-solving procedures are interpreting the problem statement and instruction. Note that the problem statement and the instruction are brought into play by being matched as data in the conditions of the productions of Table 2.

Student understanding of implication. All three students that were studied had serious misunderstandings about how one determines whether a statement is implied by a rule, and some time was spent correcting each student's misconceptions. However, their misunderstandings did not become clear on Line 2. Their faulty understanding was sufficient for that line but the problems showed up on later lines. Thus, rather than a correct subroutine for inference application like the one embodied by Productions P16–P21, these students had subroutines that only sometimes produced the correct answer.

Two of the students thought that it was sufficient to determine that the consequent of the rule matched the to-be-justified statement and did not bother to test the antecedent. For them the subroutine call (subgoal setting) of Production P16 did not exist. One student argued, for instance, that Line 4 could be justified by the substitution principle of equality because that principle gave an equality as its consequent.

The third student had more exotic misunderstandings. This is also best illustrated in his efforts to justify Line 4 (RO + ON = ON + NY) in Figure 3. The student thought the transitive property of equality was the right justification for Line 4. The transitive property of equality is stated as "a = b, b = c, implies a = c." The student physically drew out the following correspondence between the antecedents of this postulate and the to-be-justified statement:

$$\begin{array}{ccccccc} a & = & b, & & b & = & c \\ | & | & | & & | & | & | \\ RO & + & ON & = & ON & + & NY \end{array}.$$

That is, he found that he could put the variables of the antecedent in order with the terms of the statement. He noted that he also needed to match to a = c in the consequent of the transitive postulate but noted that a previous line had RO = NY, which given the earlier variable matches satisfied his need.

This student had at least two misunderstandings. First, he seemed unable to appreciate the tight constraints on pattern matching (e.g., one cannot match "=" against "+"). Second, he failed to appreciate that the consequent of the postulate should be matched to the statement, and the antecedent to earlier statements. Either he had it the other way around or, more likely, he did not think it mattered which way it was used. However, given the instruction he had to date, this is not surprising because none of this was specified.

All students required remedial instruction. Thus, these errors created the opportunity for new learning. Although I have not analyzed this in detail, I believe that remedial instruction amounted to providing ad-

ditional declarative information. This information could be used by other general procedures to provide interpretive behavior in place of the compiled procedures that Table 2 is assuming in Productions P16–P21. This is a simple form of debugging: When the instruction assumes precompiled procedures that do not exist, remedial instruction can correct the situation by providing the data for interpretive procedures.

The Need for an Initial Declarative Encoding

This section has been concerned with showing how students can generate behavior in a new domain when they do not have specific procedures for acting in that domain. Their knowledge of the domain is declarative and is interpreted by general procedures. One can argue that it is adaptive for a learning system to start out this way. New productions have to be integrated with the general flow of control in the system. Clearly, we are not in possession of an adequate understanding of our flow of control to form such productions directly. One of the reasons why instruction is often so inadequate is that the teacher likewise has a poor conception of the flow of control in the student. Attempts to directly encode new procedures, as in the Instructible Production System (Rychener & Newell, 1978; Rychener, Note 1), have run into trouble because of this problem of integrating new elements into a complex existing flow of control.

As an example of the problem with creating new procedures out of whole cloth, consider the use of the definition of congruence by the production set in Table 2 to provide a reason for the second line in Figure 3. One could build a production that would directly recognize the application of the definition of this situation rather than go through the interpretive rigmarole of Figure 5 (Table 2). This production would have the form

IF the goal is to give a reason for $\underline{XY} = UV$
and a previous line has $XY \cong UV$,
THEN POP with success,
and the reason is definition of segment congruence.

However, it is very implausible that the subject could know that this knowledge was needed in this procedural form before he or she stumbled on its use to solve Line 2 in Figure 3. Thus, ACT should not be expected to encode its knowledge into procedures until it has seen examples of how the knowledge is to be used.

Although new productions have to be created sometimes, forming new productions is potentially dangerous. Because productions have direct control over behavior, there is the ever-present danger that a new production may wreak great havoc in a system. Anyone who incrementally augments computer programs will be aware of this problem. A single erroneous statement can destroy the behavior of a previously fine program. In computer programming the cost is slight—one simply has to edit out the bugs the new procedure brought in. For an evolving creature the cost of such a failure might well be death. In the next section I will describe a highly conservative and adaptive way of entering new procedures.

As the examples reviewed in this section illustrate, declarative knowledge can have impact on behavior, but that impact is filtered through an interpretive system that is well oiled in achieving the goals of the system. This does not guarantee that new learning will not result in disaster, but it does significantly lower the probability. If a new piece of knowledge proves to be faulty, it can be tagged as such and so disregarded. It is much more difficult to correct a faulty procedure.

As a gross example suppose I told a gullible child, "If you want something, then you can assume it has happened." Translated into a production it would take on the following form:

IF the goal is to achieve X,
THEN POP with X achieved.

This would lead to a perhaps blissful but deluded child who never bothered to try to achieve anything because he or she believed it was already achieved. As a useful cognitive system he or she would come to an immediate halt. However, even if the child were gullible enough to encode this in declarative form at face value and perhaps even act upon it, he or she would quickly identify it as a

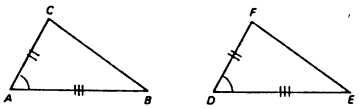
lie (by contradiction procedures he or she has), tag it as such, and so prevent it from having further impact on behavior and continue on a normal life of goal achievement. New information should enter in declarative form because one can encode information declaratively without committing control to it and because one can be circumspect about the behavioral implications of declarative knowledge.

In earlier publications (e.g., Anderson et al., 1979, 1980) we proposed a designation process that allowed productions to be directly created. Basically, we could build an arbitrary knowledge structure in working memory and, by the action of a single designating production, convert that knowledge structure into a production. This was rightfully criticized (e.g., Norman, 1980) as too powerful computationally to be human. It meant, for instance, that one could directly commit to memory production rules for applying a novel procedure. For instance, a subject given a target set in the Sternberg (1969) task could designate specific productions to recognize each member of the set and so avoid any effect of set size. We were always aware of such problems with designation. For instance, in my discussion of induction in the 1976 ACT book (Anderson, 1976, Section 12.3), I was stubbornly avoiding such a process. However, a few years ago there seemed no way to construct a learning theory without such a mechanism. Now, thanks to the development of ideas about knowledge compilation, the designation mechanism is no longer necessary.

Knowledge Compilation

Interpreting knowledge in declarative form has the advantage of flexibility, but it also has serious costs in terms of time and working memory space. The process is slow because interpretation requires retrievals of declarative information from long-term memory and because the individual production steps of an interpreter are small in order to achieve generality. (For instance, the steps of problem refinement in Table 2 and Figure 5 were painfully small.) The interpretive productions require that the declarative information be represented in working memory and this can place a heavy burden on working memory capacity. Many of the subjects' errors and much of their slowness seem attributable to working memory errors. Students can be seen to repeat themselves over and over again as they lose critical intermediate results and have to recompute them. This section of the paper is devoted to describing the compilation processes by which the system goes from this interpretive application of declarative knowledge to procedures (productions) that directly apply the knowledge. By building up procedures to perform specific tasks like reason giving in geometry, a great deal of efficiency is achieved both in terms of time and working memory demands.

POSTULATE 14 (SAS POSTULATE) If two sides and the included angle of one triangle are congruent to the corresponding parts of another triangle, the triangles are congruent.



According to Postulate 14:

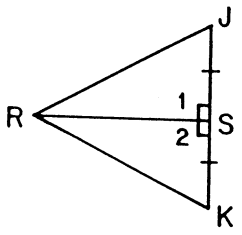
If $\overline{AB} \cong \overline{DE}$, $\overline{AC} \cong \overline{DF}$, and $\angle A \cong \angle D$,
then $\triangle ABC \cong \triangle DEF$.

Figure 6. Statement in the text of the side-angle-side (SAS) postulate. (From *Geometry* by R. C. Jurgensen, A. J. Donnelly, J. E. Maier, and G. R. Rising. Boston: Houghton Mifflin, 1975, p. 122. Copyright 1975 by Houghton Mifflin Co. Reprinted by permission.)

The Phenomenon of Compilation

One of the processes in geometry that I have focused on is how students match postulates against problem statements. Consider the side-angle-side (SAS) postulate whose presentation in the text is given in Figure 6. I followed a student through the exercises in the text that followed the section that contained this postulate and the side-side-side (SSS) postulate. The first problem that required use of SAS is illustrated in Figure 7. The following is the portion of his protocol where he actually called up this postulate and managed to put it in correspondence to the problem:

If you looked at the side-angle-side postulate [long pause] well RK and RJ could almost be [long pause] what the missing [long pause] the missing side. I think



Given: ∠1 and ∠2 are right angles
JS = KS
Prove: ΔRSJ = ΔRSK

Figure 7. The first proof-generation problem that a student encounters that requires application of the side-angle-side postulate.

somehow the side-angle-side postulate works its way into here [long pause]. Let's see what it says: "two sides and the included angle." What would I have to have to have two sides. JS and KS are one of them. Then you could go back to RS = RS. So that would bring up the side-angle-side postulate [long pause]. But where would ∠1 and ∠2 are right angles fit in [long pause] wait I see how they work [long pause] JS is congruent to KS [long pause] and with Angle 1 and Angle 2 are right angles that's a little problem [long pause]. OK, what does it say—check it one more time: "If two sides and the included angle of one triangle are congruent to the corresponding parts." So I have got to find the two sides and the included angle. With the included angle you get Angle 1 and Angle 2. I suppose [long pause] they are both right angles, which means they are congruent to each other. My first side is JS is to KS. And the next one is RS to RS. So these are the two sides. Yes, I think it is the side-angle-side postulate.

After reaching this point there was still a long process by which the student actually went through writing out the proof, but this is the relevant portion in terms of assessing what goes into recognizing the relevance of SAS.

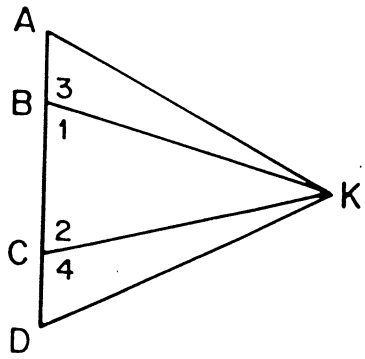
After a series of four more problems (two were solved by SAS and two by SSS), I came to the student's application of the SAS postulate for the problem illustrated in Figure 8. The method recognition portion of the protocol follows:

Right off the top of my head I am going to take a guess at what I am supposed to do: ∠DCK ≅ ∠ABK. There is only one of two and the side-angle-side postulate is what they are getting to.

A number of things seem striking about the contrast between these two protocols. One is that there has been a clear speedup in the application of the postulate. A second is that there is no verbal rehearsal of the statement of the postulate in the second case. I take this as evidence that the student is no longer calling a declarative representation of the problem into working memory. Note also in the first protocol that there are a number of failures of working memory—points where the student recomputed forgotten information. The third feature of difference is that in the first protocol there is a clear piecemeal application of the postulate by which the student is separately identifying every element of the postulate. This is absent in the second protocol. It gives the appearance of the postulate being matched in a single step. These three features—speedup, dropout of verbal rehearsal, and elimination of piecemeal application—are among the features that I want to associate with the processes of knowledge compilation.

The Mechanisms of Compilation

The knowledge compilation processes in ACT can be divided into two subprocesses. One, which is called *composition*, takes sequences of productions that follow each other in solving a particular problem and



Given: ∠1 ≅ ∠2
$\overline{AB} \cong \overline{DC}$
$\overline{BK} \cong \overline{CK}$
Prove : ΔABK ≅ ΔDCK

Figure 8. The fourth proof-generation problem that a student encounters that requires application of the side-angle-side postulate.

collapses them into a single production that has the effect of the sequence. The idea of composition was first developed by Lewis (1978). This produces considerable speedup by creating new operators that embody the sequences of steps that are used in a particular problem domain. The second process, *proceduralization*, builds versions of the productions that no longer require the domain-specific declarative information to be retrieved into working memory. Rather, the essential products of these retrieval operations are built into the new productions.

The technical details about how knowledge compilation is implemented are given in Neves and Anderson (1981). Here I will simply present the basic ideas and then focus on assessing some of the relevant literature. I will explain the basic processes with respect to an interesting example of telephone numbers. It has been noted (Anderson, 1976) that people develop special procedures for dialing frequently used telephone numbers. Sometimes declarative access to the number is lost and the only access one has to the number is through the procedure for dialing it.

Consider the following two productions that might serve to dial a telephone number:

P1. IF the goal is to dial LVtelephone-number
and LVdigit1 is the first digit of LVtelephone-number,
THEN dial LVdigit1.

P2. IF the goal is to dial LVtelephone-number
and LVdigit1 has just been dialed
and LVdigit2 is after LVdigit1 in LVtelephone-number,
THEN dial LVdigit2.

Composition creates "macroproductions," which do the operation of a pair of productions that occurred in sequence. Applied to the sequence of Production P1 above followed by P2, composition would create

P1&P2. IF the goal is to dial LVtelephone-number
and LVdigit1 is the first digit of LVtelephone-number
and LVdigit2 is after LVdigit1,
THEN dial LVdigit1 and then LVdigit2.

Compositions like this reduce the number of production applications to perform the task.

A composed production like P1&P2 still requires that the information (in this case, the phone number) be held in working memory. This information must be retrieved from long-term memory and matched to the second and third clauses in P1&P2. Proceduralization eliminates clauses in the condition of a production that require information to be retrieved from long-term memory and held in working memory. In the above production, P1&P2, the second and third condition clauses would be eliminated. The local variables that would have been bound in matching these clauses are replaced by the values they are bound to in the special case. So suppose this production is repeatedly applied in the dialing of Mary's telephone number, which is 432-2815. The local variables in P1&P2 would be bound as follows:

LVtelephone-number = Mary's number.
LVdigit1 = 4.
LVdigit2 = 3.

Producing the substitution of these values for the variables and eliminating the second and third condition clauses we get

P1&P2*. IF the goal is to dial Mary's telephone number,
THEN dial 4 and then 3.

By continued composition and proceduralization, a production can be built that dials the full number.

P*. IF the goal is to dial Mary's number,
THEN dial 432-2815.

It should be emphasized that forming this production does not imply the necessary loss of the declarative representation of the knowledge. The few instances reported of loss of declarative access to a telephone number probably reflect cases where the declarative knowledge ceases to be used and is simply forgotten.

An important consequence of proceduralization is that it reduces the load on working memory in that the long-term information need no longer be held in working memory. This makes it more likely that the system can simultaneously perform a second task that does make working memory demands. Of course, this is achieved by creating a procedure that is knowledge specific. The original Productions P1 and P2 could dial any telephone number. P* can only dial Mary's number.

It should be clear from this example how knowledge compilation produces the three phenomena noted in the protocols at the introduction of this section. The composition of multiple steps into one produces the speedup and leads to unitary rather than piecemeal application. The dropout of verbal rehearsal is a result of the fact that proceduralization eliminates the need to hold long-term memory information in working memory.

An important issue concerns the limits on the composition process. How many small productions can be combined to form a large one? The limit comes from the capacity of working memory. All the information in the production's condition must be active in working memory for the production to apply. If a composed production is created with a condition too large to be matched by working memory, that production will never apply and so will not be able to enter into further compositions. It should be noted that proceduralization serves to reduce the demands made by a production on working memory. Hence, proceduralized versions of productions may be able to enter into more compositions than nonproceduralized forms. However, even when proceduralized, the conditions of productions will grow with further compositions and hence there will be a limit on the amount of composition. I will discuss later the potential for practice to increase the capacity of working memory and so permit productions, which had been too large to match, to match and be composed.

Remarks About the Composition Mechanism

In the above discussion and in the computer implementation, the assumption has been that a pair of productions will be composed if they follow each other. This means that on repeated applications of the same problem, the number of productions should be halved each time. More generally, however, one might assume that the number of productions in each application is reduced to a proportion a of the previous application that involved composition. If $a > 1/2$, this might reflect that compositions are formed with probability less than 1. If $a < 1/2$, this might reflect the fact that composition involved more than a pair of productions. Thus, after n compositions the expected number of productions would be Na^n, where N was the initial number. As will be argued later, the rate of composition (n) may not be linear in number of applications of the production set to problems.

There is the opportunity for spurious pairs of productions to accidently follow each other and so be composed together. If spurious pairs of productions were allowed to be composed together, there would not be disastrous consequences but it would be quite wasteful. Also, spurious productions might intervene between the application of productions that really belong together. So, for instance, suppose the following three productions had happened to apply in sequence:

P1. IF the subgoal is to add in a digit,
THEN set as a subgoal to add the digit and the running total.

P2. IF I hear footsteps in the aisle,
THEN the teacher is coming my way.

P3. IF the goal is to add two digits
and a sum is the sum of the two digits,
THEN the result is the sum
and POP.

This sequence of productions might apply, for instance, as a child is performing arithmetic exercises in a class. The first and third are set to process subgoals in solving the problem. The first sets up the subgoal that is met by the third. The second production is not related to the other two and is merely an inference production that interprets sounds of the teacher approaching. It just happens to intervene between the other two. Composition as described would produce the following pairs:

P1&P2. IF the subgoal is to add in a digit
and I hear footsteps in the aisle,
THEN set as a subgoal to add the digit and the running total
and the teacher is coming my way.

P2&P3. IF I hear footsteps in the aisle
and the goal is to add two digits
and a sum is the sum of the two digits,
THEN the teacher is coming my way
and the result is the sum
and POP.

These productions are harmless but basically useless. They have also prevented formation

of the following useful composition:

P1&P3. IF the subgoal is to add in a digit
and the sum is the sum of the digit
and the running total,
THEN the result is the sum.

Therefore, it seems reasonable to advance a sophistication over the composition mechanism proposed in Neves and Anderson (1981). In this new scheme productions are composed only if they are linked by goal setting (as in the case of P1&P3), and productions that are linked by goal setting will be composed even if there are intervening productions that make no goal reference (as in the case of P2). This is an example where the learning mechanisms can profitably exploit the goal structuring of production systems.

Phenomena Explained by Knowledge Compilation

In addition to the three qualitative features of skill development (speedup, unitary application, elimination of verbal rehearsal), knowledge compilation can help explain some of the more provocative results in the experimental literature. In this section, I would like to focus on three such results: disappearance of set size effects in the Sternberg (1969) paradigm, disappearance of set size and display size effects in the scan task, (Shiffrin & Schneider, 1977), and the Einstellung effect (Luchins, 1942) in problem solving.

Before we get into the specifics of ACT's explanation of these three phenomena, it should be acknowledged that there already exist other explanations of these phenomena in the literature. However, ACT's explanation relies on mechanisms that were not fashioned to address these phenomena. Rather, they were fashioned by Neves and Anderson (1981) to address various phenomena associated with the speedup of postulate application in geometry. Thus, the fact that the compilation mechanisms extend to these other phenomena is an important demonstration of the generality of the theory.

The Sternberg paradigm. In the Sternberg paradigm (e.g., Sternberg, 1969) subjects are asked to indicate if a probe comes from a small set of items. The classic result is that decision time increases with set size. It has been shown that effects of size of memory set can diminish with repeated practice (Briggs & Blaha, 1969). A sufficient condition for this to occur is that the same memory set be used repeatedly. The following are two productions that a subject might use for performing the scan task at the beginning of the experiment:

PA. IF the goal is to recognize LVprobe
and LVprobe is an LVtype
and the memory set contains an LVitem of LVtype,
THEN say "yes"
and POP the goal.

PB. IF the goal is to recognize LVprobe
and LVprobe is an LVtype
and the memory set does not contain an LVitem of LVtype,
THEN say "no"
and POP the goal.

In the above, LVprobe and LVitem will match tokens of letters and LVtype will match a particular letter type (e.g., the letter *A*). This production set is basically the same as the production system for the Sternberg task given in Anderson (1976) except that it is in a somewhat more readable form that will expose the essential character of the processing. These productions require that the contents of the memory set be held active in working memory. As discussed in Anderson (1976), the more items required to be held active in working memory, the lower the activation of each and the slower the recognition judgment, which produces the typical set-size effect.

Repeated practice of those productions on the same memory set will produce an eventual elimination of set-size effects. Consider, to be concrete, a situation where these productions are repeatedly applied on the same list—say a list consisting of A, J, and N with foils coming from a list of L, B, and K—then through proceduralization we would get the following productions from PA:

P1. IF the goal is to recognize LVprobe
and LVprobe is an A,
THEN say "yes"
and POP the goal.

P2. IF the goal is to recognize LVprobe
and LVprobe is a J,
THEN say "yes"
and POP the goal.

P3. IF the goal is to recognize LVprobe
and LVprobe is an N,
THEN say "yes"
and POP the goal.

The preceding are productions for recognizing the positive set. Specific productions would also be produced by proceduralization from PB to reject the foils:

P4. IF the goal is to recognize LVprobe
and LVprobe is an L,
THEN say "no"
and POP the goal.

P5. IF the goal is to recognize LVprobe
and LVprobe is a B,
THEN say "no"
and POP the goal.

P6. IF the goal is to recognize LVprobe
and LV probe is a K,
THEN say "no"
and POP the goal.

It is interesting to note here that Shiffrin and Dumais (1981) report that the automatization effect they observe in such tasks is as much due to subjects' ability to reject specific foils as it is their ability to accept specific targets. These productions no longer require the memory set to be held in working memory and will apply in a time independent of memory set size. However, there still may be some effect of set size in the subject's behavior. These productions do not replace PA and PB; rather, they coexist and it is possible for a classification to proceed by the original PA and PB. Thus, we have two parallel bases for classification racing, with the judgment being determined by the fastest one. This will produce a set-size effect that will diminish as P1–P6 become strengthened.

The scan task. Shiffrin and Schneider (1977) report an experiment in which they gave subjects a set of items to remember. Then subjects were shown a series of displays in rapid succession, each display containing a set of items. The subjects' task was to decide if any of the displays contained an item in the memory set. When Shiffrin and Schneider kept both the members of the study set and the distractors constant, they found considerable improvement with practice in subjects' performance on the task. They interpreted their results as indicating diminishing effects of both memory set size and the number of alternatives in the display.

Again ACT's knowledge compilation mechanisms can be shown to predict the result. Consider what a production set might be like that scanned an array to see if any member of the array matched a memory set item:

PC*. IF the goal is to see if LVarray contains a memory item
and LVprobe is in POSITION*,
THEN the subgoal is to recognize LVprobe.

PD. IF the goal is to recognize LVprobe
and LVprobe is an LVtype
and the memory set contains LVitem of LVtype,
THEN tag the goal as successful
and POP the goal.

PE. IF the goal is to recognize LVprobe
and LVprobe is an LVtype
and the memory set does not contain an LVitem of LVtype,
THEN tag the goal as failed
and POP the goal.

PF. IF the goal is to see if LVarray contains a memory item
and there is a successful subgoal,
THEN say "yes"
and POP the goal.

PG. IF the goal is to see if LVarray contains a memory item,
THEN say "no"
and POP the goal.

Production PC* is a schema for a set of productions such that each one would recognize an item in a particular position. An example might be

IF the goal is to see if LVarray contains a memory item
and LVprobe is in the upper right corner,
THEN set as a subgoal to recognize LVprobe.

PD and PE are similar to PA and PB given earlier—they check whether each position focused by PC* contains a match. PF will apply if one of the probes lead to a successful match, and the default production PG will apply if none of the positions leads to success. The behavior of this production set is one in which individual versions of PC* apply serially, focusing attention on individual positions. PD and PE are responsible for the judgment of individual probes. This continues until a positive probe is hit and PF applies or until there are no more probe positions and PG applies. (PG will only be selected when there are no more positions

because specificity will prefer PC* and PF over it.) Because of the need to keep the memory set active, an effect of set size is expected. The serial examination of positions produces an effect of display size. These two factors should be multiplicative, which is what Schneider and Shiffrin (1977) report.

Consider what will happen with knowledge compilation. Composing a PC* production with PD and with PF and proceduralizing, we will get positive productions of the form

P7. IF the goal is to see if LVarray contains a memory item
and the upper right-hand position contains an LVprobe
and the LVprobe is an A,
THEN say "yes"
and POP the goal.

The negative production would be formed by composing together a sequence of PC* productions paired with PE and a final application of PG. All the subgoal and POPping would be composed out. The strict composition of this sequence would be productions like

P8. IF the goal is to see if LVarray contains a memory item
and the upper left-hand position contains an LVprobe1
and LVprobe1 is a K
and the upper right-hand position contains an LVprobe2
and LVprobe2 is a B
and the lower left-hand position contains an LVprobe3
and LVprobe3 is an L
and the lower right-hand position contains an LVprobe4
and LVprobe is a K,
THEN say "no"
and POP the goal,

where a separate such production would have to be formed for each possible foil combination. These productions would not be affected by set size or probe size. This is consistent with the Schneider and Shiffrin (1977) findings.

The Einstellung phenomenon. Another phenomenon that is predictable from knowledge compilation processes is the Einstellung effect (Luchins, 1942; Luchins & Luchins, 1959) in problem solving. One of the examples used by Luchins to demonstrate this phenomena is illustrated in Figure 9 (a).

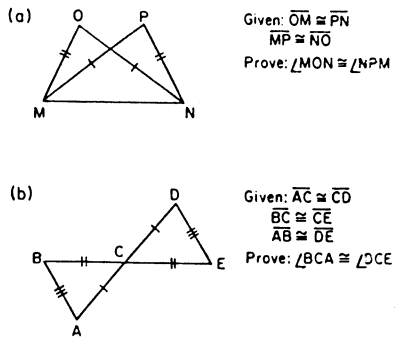


Figure 9. After solving a series of problems like (a), students are more likely to choose the nonoptimal solution for (b).

Luchins presented his subjects with a sequence of geometry problems like the one in Figure 9 (a). For each problem in the sequence, the student had to prove that two triangles were congruent in order to prove that two angles were congruent. Then subjects were given a problem like the one in Figure 9 (b). Subjects proved this by means of congruent triangles even though it has a much simpler proof by means of vertical angles. Subjects not given the initial experience with problems like the one in Figure 9 (a) show a much greater tendency to use the vertical angle proof. Their experimental experience caused subjects to solve the problem in a nonoptimal way.

Lewis (1978) has examined the Einstellung effect and its relation to the composition process. He defines as perfect composites compositions that do not change the behavior of the system but just speed it up. Such compositions cannot produce Einstellung effects, of course. However, he notes that there are a number of natural ways to produce nonperfect composites that produce Einstellung effects. The ACT theory provides an example of such a nonperfect composition process. Composites are nonperfect in ACT because of its conflict-resolution principles.

Productions P1 through P4 provide a model of part of the initial state of the student's production system.

P1. IF the goal is to prove $\angle XYZ \cong \angle UVW$
and the points are ordered X, Y, and W on a line
and the points are ordered Z, Y, and U on a line,

THEN this can be achieved by vertical angles and POP the goal.

P2. IF the goal is to prove $\angle XYZ \cong \angle UVW$,
THEN set as subgoals:
1. To find a triangle that contains $\angle XYZ$,
2. To find a triangle that contains $\angle UVW$,
3. To prove the two triangles congruent, and
4. To use corresponding parts of congruent triangles.

P3. IF the goal is to find a figure that has a relation to an object
and Figure X has the relation to the object,
THEN the result is Figure X
and POP the goal.

P4. IF the goal is to prove $\Delta XYZ \cong \Delta UVW$
and $\overline{XY} \cong \overline{UV}$
and $\overline{YZ} \cong \overline{VW}$
and $\overline{ZX} \cong \overline{WU}$,
THEN this can be achieved by SSS
and POP the goal.

Production P1 is responsible for immediately recognizing the applicability of the vertical angle postulate. Productions P2–P4 are part of the production set that is responsible for proof through the route of corresponding parts of congruent triangles. Production P2 decomposes the main goal into the subgoals of finding the containing triangles, of proving they are congruent, and then of using the corresponding-parts principle. P3 finds the containing triangles, and P4 encodes one production that would recognize triangle congruence. This production set, applied to a problem like that in Figure 9 (b), would lead to a solution by vertical angles. This is because Production P1, for vertical angles, is more specific in its condition than is Production P2, which starts off the corresponding-angles proof. As explained earlier, ACT's conflict resolution prefers specific productions.

Consider, however, what would happen after Productions P2–P4 had been exercised on a number of problems and composition had taken place. Production P2&P3&P3&P4 represents a composition of the sequence P2, then P3, then P3, and then P4. Its condition is not less specific than that of P1 and, in fact, contains more clauses. However, because these clauses are not a superset of P1's clauses, it is not the case that either production is technically more specific than the other. They are both in potential conflict and, because both change the goal state, application of one will block the application of the other. In this case strength serves as the basis for resolving the conflict. Production P2&P3&P3&P4, because of its recent practice, may be stronger and therefore might be selected.

P2&P3&P3&P4. IF the goal is to prove $\angle XYZ \cong \angle UVW$
and $\angle XYZ$ is part of ΔXYZ
and $\angle UVW$ is part of ΔUVW
and $\overline{XY} \cong \overline{UV}$
and $\overline{YZ} \cong \overline{VW}$
and $\overline{ZX} \cong \overline{WU}$,
THEN set $\Delta XYZ \cong \Delta UVW$
and set as a subgoal to use corresponding parts of congruent triangles.

This example illustrates how practice can change the specificity ordering of productions through composition and how it can directly change the strength. These two factors—change of specificity and change of strength—can cause ACT's conflict-resolution mechanism to change the behavior of the system, producing Einstellung effects. Under this analysis it can be seen that the Einstellung effect is an aberrant phenomenon reflecting what is basically an adaptive adjustment on the system's part. Through strength and composition ACT is unitizing and favoring sequences of problem-solving behaviors that have been recently successful. It is a good bet that such sequences will prove useful again. It is to the credit of the cleverness of Luchins's design (1942, Luchins & Luchins, 1959) that it exposed the potential cost of these usually beneficial adaptations.

It has been suggested that one could produce the Einstellung effect by simply strengthening particular productions. So one might suppose that Production P2 is strengthened over P1. The problem with this explanation is that subjects can be shown to have a preference for a particular sequence of productions not merely single productions in isolation. Thus, in the water jug problems, described by Luchins (1942), subjects will fixate on a specific sequence of operators implementing a subtraction method and will not notice other simpler subtraction methods. The composition mechanism explains how the subject encodes this operator sequence.

It is interesting to compare the time scale for producing Einstellung effects with the time scale for producing the automatization effects in the Sternberg paradigm and the scan paradigm. Strong Einstellung effects can be produced after a half-dozen trials, whereas the automatization results require hundreds of trials. This suggests that composition that underlies the Einstellung effect can proceed more rapidly than the proceduralization that underlies the automatization effects. Proceduralization is really more responsible for creating domain-specific procedures than is composition. Composition creates productions that encode the sequence of general productions for a task, but the composed productions are still general. In contrast, by replacing variables with domain constants, proceduralization creates productions that are committed to a particular task. Apparently, the learning system is reluctant to create this degree of specialization unless there is ample evidence that the task will be repeated frequently.

The Adaptive Value of Knowledge Compilation

In the previous section on initial encoding, it was argued that it is dangerous for a system to directly create productions to embody knowledge. For this reason and for a number of others, it was argued that knowledge should first be encoded declaratively and then interpreted. This declarative knowledge could affect behavior, but only indirectly, via the intercession of existing procedures for correctly interpreting that knowledge. We have in the process of composition and proceduralization a means of converting declarative facts into production form.

It is important to note that productions created from compilation really do not change the behavior of the system, except in terms of possible reorderings of specificity relations as noted in our discussion of the Einstellung effect. Thus, knowledge compiled in this way has much of the same safeguards built into it that interpretative application of the knowledge does. The safety in interpretative applications is that a particular piece of knowledge does not impact on behavior until it has undergone the scrutiny of all the system's procedures (which can, for instance, detect contradiction of facts or of goals). Because compilation only operates on successful sequences of productions that pass this scrutiny, it tends to produce only production embodiments of knowledge that pass that scrutiny. This is the advantage of learning from doing. Another advantage with interpretive application is that the use of the knowledge is forced to be consistent with existing conventions for passing control among goals. By compiling from actual use of this knowledge, the compiled productions are guaranteed to be likewise consistent with the system's goal structure.

We can understand why human compilation is gradual (in contrast to computer compilation) and occurs as a result of practice if we consider the difference between the human situation and the typical computer situation. For one thing the human does not know what is going to be procedural in an instruction until he or she tries to use the knowledge in the instruction. In contrast, the computer has built in the difference between program and data. Another reason for gradual compilation is to provide some protection against the errors that enter into a compiled procedure because of the omission of conditional tests. For instance, if the system is interpreting a series of steps that include pulling a lever, it can first reflect on the lever-pulling step to see if it involves any unwanted consequences in the current situation. These tests will be in the form of productions checking for error conditions. (These error-checking productions can be made more specific so that they would take precedence over the normal course of action.) When that procedure is totally compiled, the lever pulling will be part of a prepackaged sequence of actions with many conditional tests eliminated (see the discussion of the Einstellung effect). If the procedure transits gradually between the interpretive and compiled stages, it is possible to detect the erroneous compiling out of a test at a stage where the behavior is still being partially monitored interpretively and can be corrected. It is interesting to note here the folk wisdom that most errors in acquisition of a skill, like airplane flying, occur neither with the novices

nor with experts. Rather, they occur at intermediate stages of learning. This is presumably where the conversion from procedural to declarative is occurring and the point where unmonitored mistakes might slip into the performance. So by making compilation gradual, one does not eliminate the possibility of error, but one does reduce the probability.

Procedural Learning: Tuning

Much learning goes on after a skill has been compiled into a task-specific procedure, and this learning cannot just be attributed to further speedup due to more composition. One type of learning involves an improvement in the choice of method by which the task is performed. All tasks can be characterized as having a search associated with them, although in some cases the search is trivial. By search I mean that there are alternate paths of steps by which the problem can be tackled, and the subject must choose among them. Some of these paths lead to no solution and some lead to more complex solutions than necessary. A clear implication of much of the novice–expert research (e.g., Larkin, McDermott, Simon, & Simon, 1980) is that with high levels of expertise in a task domain, the problem solver becomes much more judicious in his choice of paths and may fundamentally alter his method of search. In terms of the traditional learning terminology, the issue is similar to, though by no means identical to, the issue of trial and error versus insight in problem solving. A novice's search of a problem space is largely a matter of trial-and-error exploration. With experience the search becomes more selective and more likely to lead to rapid success. I refer to the learning underlying this selectivity as *tuning*. My use of the term is quite close to that of Rumelhart and Norman (1978).

In 1977 we proposed a set of three learning mechanisms that still serves as the basis for much of our work on the tuning of search (Anderson, Kline, & Beasley, Note 2). There was a generalization process by which production rules become broader in their range of applicability, a discrimination process by which the rules become narrower, and a strengthening process by which better rules are strengthened and poorer rules weakened. These ideas have nonaccidental relations to concepts in the traditional learning literature, but as we will see they have been somewhat modified to be computationally more adequate. One can think of production rules as implementing a search, where individual rules correspond to individual operators for expanding the search space. Generalization and discrimination serve to produce a "metasearch" over the production rules, looking for the right features to constrain the application of these productions. Strength serves as an evaluation for the various constraints produced by the other two processes.

In this section I will illustrate how these three central learning constructs operate in the ACT system with examples from language acquisition. It is a major claim of the theory that these learning mechanisms will apply equally well to domains as diverse as language processing and geometry proof generation. Elsewhere we have discussed how these processes apply to schema abstraction or prototype formation (Anderson et al., 1979) and to proof generation (Anderson, Greeno, Kline, & Neves, 1981). That they apply in such diverse circumstances is important evidence for the generality of these mechanisms.

In Anderson (1981a) I focused on the issue of language acquisition per se and that article should be consulted for a fuller discussion of these issues. Language acquisition is being used here to illustrate the basic learning mechanisms. The examples here will concern how production rules are acquired to generate the correct syntactic structures. Unlike some of the other domains (e.g., proof generation), productions for language generation seldom result in one path for generation being tried and then a second path being tried after the first fails. Rather, in the language generation case, if the wrong generation path is followed, a faulty syntactic structure will be generated. Thus, errors of choice or search in language generation result in incorrect generations. In proof generation they would more likely result in longer times to reach a solution.

Generalization

The ability to perform successfully in novel situations is the hallmark of human cognition. For example, productivity has often been identified as the most important feature of natural languages, where this refers to the speaker's ability to generate and comprehend utterances never before encountered. Traditional learning theories have been criticized because of their inability to account for this productivity (e.g., McNeill, 1968), and it was one of our goals in designing ACT to avoid this sort of criticism.

An example. ACT's generalization algorithm looks for similarities between a pair of productions and creates a new production rule that captures what these individual production rules have in common. Consider the following pair of rules for language generation that might arise as the consequence of compiling productions to encode specific instances of phrases:

P1. IF the goal is to indicate that a coat belongs to me,
THEN say "My coat."

P2. IF the goal is to indicate that a ball belongs to me,
THEN say "My ball."

From these two production rules, ACT can form the following generalization:

P3. IF the goal is to indicate that LVobject belongs to me,
THEN say "My LVobject,"

in which the variable LVobject has replaced the particular object. The rule now formed is productive in the sense that it will fill in the LVobject slot with any object. Of course, it is just this productivity in child speech that has been commented on, at least since Braine (1963). It is important to note that the general production does not replace the original two and that the original two will continue to apply in their special circumstances.

The basic function of the ACT generalization process is to extract from different special productions what they have in common. These common aspects are embodied in a production that will apply in new situations where original special productions do not apply. Thus, the claim for the ACT generalization mechanism is that transfer is facilitated if the same components are taught in two procedures so generalization can occur. So, for instance, transfer to a new text editor will be more facilitated if one has studied two other text editors than if one has studied only one.

Another example. The example above does not illustrate the full complexity at issue in forming generalizations. The following is a fuller illustration of the complexity:

P4. IF the goal is to indicate the relation in (LVobject1 chase LVobject2)
and LVobject1 is dog
and LVobject1 is singular
and LVobject2 is cat
and LVobject2 is plural,
THEN say "CHASES."

P5. IF the goal is to indicate the relation in (LVobject3 scratch LVobject4)
and LVobject3 is cat
and LVobject3 is singular
and LVobject4 is dog
and LVobject4 is plural,
THEN say "SCRATCHES."

P6. IF the goal is to indicate the relation in (LVobject1 LVrelation LVobject2)
and LVobject1 is singular
and LVobject2 is plural,
THEN say "LVrelation + s."

P6 is the generalization that would be formed from P4 and P5. It illustrates that clauses can be deleted in a generalization as well as variables introduced (in this case LVrelation). In this example, the generalization has been made that the verb inflection does not depend on the category of the subject or of the object and does not depend on the verb. This generalization remains overly specific in that the rule still tests whether the object is plural—this is something the two examples have in common. Further generalization would be required to delete this unnecessary test. On the other hand, the generalized rule does not test for present tense and so is overly general. This is because this information was not represented in the original productions. The discrimination process (to be described later) can bring in this missing information.

The technical work defining generalization in ACT is given in Anderson et al. (1980) and similar definitions are to be found in Hayes-Roth and McDermott (1976)

and Vere (1978). I will skip these technical definitions here for brevity's sake. The basic generalization process is clear without them.

Comparisons to earlier conceptions of generalization. The process of production generalization clearly has similarities to the process of stimulus generalization in earlier learning theories (for a review see Heinemann & Chase, 1975), but there are also clear differences. Past theories frequently proposed that a response conditioned to one stimulus would generalize to stimuli similar on various dimensions. So, for instance, a bar press conditioned to one tone would tend to be evoked by other tones of similar pitch and loudness. An important feature of this earlier conception is that generalization was an automatic outcome of a single learned connection and did not require any further learning. Learning in these theories was all a matter of discrimination—restricting the range of the learned response. In contrast, in the ACT theory generalization is an outcome of comparing two or more learned rules and extracting what they have in common. Thus, it requires additional learning over and above the learning of the initial rules, and it depends critically on the relation between the rules learned. There is evidence (Elio & Anderson, 1981) for ACT's stronger assumption that generalization depends on the interitem similarity among the learning experiences as well as the similarity of the test situation to the learning experiences.

Another clear difference between generalization as presented here and many earlier generalization theories is that the current generalization proposed is structural and involves clause deletion and variable creation rather than the creation of ranges on continuous dimensions. We have focused on structural generalizations because of the symbolic domains that have been our concern. However, these generalization mechanisms can be extended to apply to generalization over intervals on continuous dimensions (Brown, 1977; Larson & Michalski, 1977). ACT's generalization ideas are much closer to what happens in stimulus-sampling theory (Burke & Estes, 1957; Estes, 1950), where responses conditioned to one set of stimulus elements can generalize to overlapping sets. This is the same as the notion in ACT of generalization on the basis of clause overlap. However, there is nothing in stimulus-sampling theory that corresponds to ACT's generalization by replacing constants in clauses with variables. This is because stimulus-sampling theory does not have the representational construct of propositions with arguments.

Discrimination

Just as it is necessary to generalize overly specific procedures, so it is necessary to restrict the range of application of overly general procedures. It is possible for productions to become overly general either because of the generalization process or because the critical information was not attended to in the first place. It is for this reason that the discrimination process plays a critical role in the ACT theory. This discrimination process tries to restrict the range of application of productions to just the appropriate circumstances. The discrimination process requires that ACT have examples both of correct and incorrect application of the production. The discrimination algorithm remembers and compares the values of the variables in the correct and incorrect applications. It randomly chooses a variable for discrimination from among those that have different values in the two applications. Having selected a variable, it looks for some attribute that the variable has in only one of the situations. A test is added to the condition of the production for the presence of this attribute.

An example. Suppose ACT starts out with the following production:

P1. IF the goal is to indicate the relation in (LVsubject LVrelation LVobject),
THEN say "LVrelation + s."

This rule for generating the present tense singular of a verb is, of course, overly general in the above form. For instance, this rule would apply when the sentence subject was plural, generating "LVrelation + s," when what is wanted is "LVrelation." By comparing circumstances where the above rule applied correctly with the current incorrect situation, ACT could notice that the variable LVsubject was bound to different values and

that the value in the correct situation had singular number but the value in the incorrect situation had plural number. ACT can formulate a rule for the current situation that recommends the correct action:

P2. IF the goal is to indicate the relation in
(LVsubject LVrelation LVobject)
and LVsubject is plural,
THEN say "LVrelation."

ACT can also form a modification of the previous rule for the past situation:

P3. IF the goal is to indicate the relation in
(LVsubject LVrelation LVobject)
and LVsubject is singular,
THEN say "LVrelation + s."

The first discrimination, P2, is called an *action discrimination* because it involves learning a new action, whereas the second discrimination, P3, is called a *condition discrimination* because it involves restricting the condition for the old action. Because of specificity ordering, the action discrimination will block misapplication of the overly general P1. The condition discrimination, P3, is an attempt to reformulate P1 to make it more restrictive. It is important to note that these discriminations do not replace the original production; rather, they coexist with it.

ACT does not always form both action and condition discriminations. ACT can only form an action discrimination when feedback is obtained about the correct action for the situation. If ACT only receives feedback that the old action is incorrect, it can only form a condition discrimination. However, ACT will only form a condition discrimination if the old rule (i.e., P1 in the above example) has achieved a level of strength to indicate that it has some history of success. The reason for this restriction on condition discriminations is that a rule can be formulated that is simply wrong and we do not want to have it perserverate by a process of endlessly proposing new discriminations.

Note that Productions P2 and P3 are improvements over P1 but are still not sufficiently refined. The discrimination algorithm can apply to these, however, comparing where they applied successfully and unsuccessfully. If discriminations of these productions were formed on the basis of tense and if both response and condition discriminations were formed, we would have the following set of productions:

P4. IF the goal is to indicate the relation in
(LVsubject LVrelation LVobject)
and LVsubject is plural
and LVrelation has past tense,
THEN say "LVrelation + ed."

P5. IF the goal is to indicate the relation in
(LVsubject LVrelation LVobject)
and LVsubject is plural
and LVrelation has present tense,
THEN say "LVrelation."

P6. IF the goal is to indicate the relation in
(LVsubject LVrelation LVobject)
and LVsubject is singular
and LVrelation has past tense,
THEN say "LVrelation + ed."

P7. IF the goal is to indicate the relation in
(LVsubject LVrelation LVobject)
and LVsubject is singular
and LVrelation has present tense,
THEN say "LVrelation + s."

A more thorough consideration of how these mechanisms would apply to acquisition of the verb auxiliary system of English is given in Anderson (1981a). The current example is only an illustration of the basic discrimination mechanism.

Recall that the feature selected for discrimination is determined by comparing the variable bindings in the successful and unsuccessful production applications. A variable on which they differ is selected, and features are selected to restrict the bindings. It is possible for this discrimination mechanism to choose the wrong variables or wrong features on which to discriminate. So, for instance, it may turn out that LVobject has a different number in two circumstances, and the system may set out to produce a discrimination on that basis, rather than discriminating on the correct variable, LVsubject. In the case of condition discriminations, such mistakes have no negative impact on the behavior of the system. The discriminated production produces the same behavior as the original in the restricted situation, so it cannot lead to worse behavior. (Recall that the original production still exists to produce the same behavior in other situations.) If an incorrect action discrimination is produced, it may block by specificity the correct application of the original production

in other situations. However, even here the system can recover by producing the correct discrimination and then giving the correct discrimination a specificity or strength advantage over the incorrect discrimination.

The current discrimination mechanism also attempts to speed up the process of finding useful discriminations by its method of selecting propositions from the data base. Though a random process is used to guarantee that any appropriate propositions in the data will eventually be found, this random choice is biased in certain ways to increase the likelihood of a correct discrimination. The discrimination mechanism chooses propositions with probabilities that vary with their activation levels. The greater the amount of activation that has spread to a proposition, the more likely it is that the proposition will be relevant to the current situation.

The previous example illustrated a critical prerequisite for discrimination to work. The system must receive feedback indicating that a particular production has misapplied, and in the case of an action discrimination, it must receive feedback as to what the correct action should have been.

In principle, a production application could be characterized as being in one of three states: known to be incorrect, known to be correct, or correctness unknown. However, the mechanisms we have implemented for ACT do not use the distinction between the second and third states. If a production applies and there is no comment on its success, it is treated as if it were a successful application. So the real issue is how ACT identifies that a production application is in error. A production is considered to be in error if it puts into working memory a fact that is later tagged as incorrect. There are two basic ways for this error tagging to occur: one is through external feedback and the other is through internal computation. In the external-feedback situation, learners may be directly told that their behavior is in error or they may infer this by comparing their behavior to an external referent (e.g., the behavior of a model or a textbook answer). In the internal-computation case, the learner must identify that a fact is contradictory, that a goal has failed, or that there is some other failure to meet internal norms. As discussed in Anderson (1981b), the goal structure of ACT productions is very helpful in correctly identifying successful and failed productions.

Strengthening

The generalization and discrimination mechanisms are the inductive components of the learning system in that they are trying to extract from examples of success and failure the features that characterize when a particular production rule is applicable. The generalization and discrimination processes produce multiple variants on the conditions controlling the same action. It is important to realize that at any time the system is entertaining as its hypothesis a set of different productions with different conditions to control the action—not just a single production (condition-action rule). There are advantages to be gained in expressive power by means of multiple productions for the same action, differing in their conditions. Because the features in a production condition are treated conjunctively but separate productions are treated disjunctively, one can express the condition for an action as a disjunction of conjunctions of conditions. Many real-world categories have the need for this rather powerful expressive logic. Also, because of specificity ordering, productions can enter into more complex logical relations, as we noted.

However, because they are inductive processes, sometimes generalization and discrimination will err and produce incorrect productions. There are possibilities for overgeneralizations and useless discriminations. The phenomenon of overgeneralization is well documented in the language acquisition literature, occurring both in the learning of syntactic rules and in the learning of natural language concepts. The phenomena of pseudodiscriminations are less well documented in language because a pseudodiscrimination does not lead to incorrect behavior, just unnecessarily restrictive behavior. However, there are some documented cases in the careful analyses of language development (e.g., Maratsos & Chalkley, 1981). One reason that a strength mechanism is needed is because of these inductive failures. It is also the case that the system may simply create

productions that are incorrect—either because of misinformation or because of mistakes in its computations. ACT uses its strength mechanism to eliminate wrong productions, whatever their source.

The strength of a production affects the probability that it will be placed on the APPLYLIST and is also used in resolving ties among competing productions of equal specificity on the APPLYLIST. These factors were discussed earlier with respect to the full set of conflict-resolution principles in ACT (see section on conflict resolution). ACT has a number of ways of adjusting the strength of a production in order to improve performance. Productions have a strength of .1 when first created. Each time it applies, a production's strength increases by an additive factor of .025. However, when a production applies and receives negative feedback, its strength is reduced by a multiplicative factor of .25. Because a multiplicative adjustment produces a greater change in strength than does an additive adjustment, this "punishment" has much more impact than a reinforcement does.

Although these two mechanisms are sufficient to adjust the behavior of any fixed set of productions, additional strengthening mechanisms are required to integrate new productions into the behavior of the system. Because these new productions are introduced with low strength, they would seem to be victims of a vicious cycle: They cannot apply unless they are strong, and they are not strong unless they have applied. What is required to break out of this cycle is a means of strengthening productions that does not rely on their actual application. This is achieved by taking all of the strength adjustments made to a production that applies and making these adjustments to all of its generalizations as well. Because a general production will be strengthened every time any one of its possibly numerous specializations applies, new generalizations can amass enough strength to extend the range of situations in which ACT performs successfully. Also, because a general production applies more widely, a successful general production will come to gather more strength than its specific variants.

For purposes of strengthening, re-creation of a production that is already in the system, whether by proceduralization, composition, generalization, or discrimination, is treated as equivalent to a successful application. That is, the re-created production receives a .025 strength increment and so do all of its generalizations.

The exact strengthening values encoded into the ACT system are somewhat arbitrary. The general relations among the values are certainly important, but the exact relations probably are not. If all the strength values were multiplied by some scaling factor, one would get the same performance from the system. They were selected to give satisfactory performance in a set of language-learning examples described by Anderson et al. (1980).

Comparison to Other Discrimination Theories

As in the case of generalization, ACT's mechanisms for discrimination have clear similarities to earlier ideas about discrimination. ACT's discrimination mechanisms, like its generalization mechanisms, focus on structural relations, whereas traditional efforts were more focused on continuous dimensions. Brown (1977) has sketched out ways for extending ACT-like discrimination mechanisms to continuous dimensions, although we have not developed them in ACT. Also, it is the case that ACT discrimination mechanisms are really specified for an operant-conditioning paradigm (in that the action of productions are evaluated according to whether they achieve desired behavior and goals) and do not really address the classical-conditioning paradigm in which a good deal of research has been done on discrimination. However, despite these major differences in character, a number of interesting connections can be drawn between ACT and the older conceptions of discrimination. In making these comparisons I will be drawing on strengthening and other conflict-resolution principles in ACT as well as the discrimination mechanism.

Shift experiments. One of the supposedly critical issues in choosing between the discontinuity and continuity theories of discrimination was the shift experiment (Spence, 1940). In that paradigm, subjects who were still responding at a chance level

with respect to some discrimination (e.g., white–black) had the reinforcement contingencies changed so that a different response was appropriate. According to the discontinuity theory, the subject's chance performance indicated control by an incorrect hypothesis, so the shift should not hurt, but according to the continuity theory, the subject could still be building up "habit strength" for the correct response and a shift would hurt. Continuity theory tended to be supported on this issue for infrahuman subjects (e.g., see Kendler & Kendler, 1975). ACT is like the discontinuity theory in that its various productions represent alternative hypotheses about how to solve a problem; however, its predictions are in accord with the continuity theory because it can be accruing strength for a hypothesis before the production is strong enough to apply and produce behavior. Of course, ACT's discrimination mechanisms cannot account for the shift data with adults (e.g., Trabasso & Bower, 1968), but we have argued elsewhere (Anderson et al., 1979) that such data should be ascribed to a conscious hypothesis-testing process that produces declarative learning rather than an automatic procedural learning process.

Stimulus generalization and eventual discrimination. As noted earlier, the clauses in a production condition are like the elements of stimulus-sampling theory. A problem for stimulus-sampling theory (see Medin, 1976, for a recent discussion) is how to accommodate both the fact of stimulus generalization and the fact of eventual perfect discrimination. The fact of stimulus generalization can easily be explained in stimulus-sampling theory by assuming that two stimulus conditions overlap in their elements. However, if so, the problem becomes how perfect discrimination behavior can be achieved when the common elements can be associated to the wrong response.

In the ACT theory one can think of the original productions for behavior as basically testing for the null set of elements:

P1. IF the goal is X,
THEN do Y.

With discrimination, elements can be brought in to discriminate between successful and unsuccessful situations, for example,

P2. IF the goal is X
and B is present,
THEN do Y.

P3. IF the goal is X
and B is present
and C is present,
THEN do Z.

P4. IF the goal is X
and D is present,
THEN do Y,

and so forth. This is like the conditioning of features to responses in stimulus-sampling theory.

If some features occur sometimes in situations for Response Y and sometimes in situations for Response Z, discrimination can cause them to become parts of productions recommending one of the actions. For instance, suppose B is such a feature that really does not discriminate between the actions. Suppose B is present in the current situation where response Z is executed, but the system receives feedback indicating that Y is correct. Further, suppose B was not present (or not attended) in the past prior situation where response Z had proved successful. Production P2 would be formed as an action discrimination. The B test is useless because B is just as likely to occur in a Z situation. This corresponds to the conditioning of common elements. However, in ACT the strengthening, discrimination, and specificity processes can eventually repress productions that are responding just to common elements. For instance, further discriminative features can be added (as in P3) that will serve to block out the incorrect application of P2. Also, it is possible to simply weaken P2 and add a new production like P4, which perhaps contains the correct discrimination.

Patterning effects. The ACT discrimination theory also explains how subjects can learn to give a response in the presence of Stimuli A and B together but neither A nor B alone. This simply requires that two discriminative clauses be added to the production, one for A and one for B. Responding to such configural cues was a problem for some of the earlier discrimination theories (see Rudy & Wagner, 1975, for a review). The power of the ACT theory over these

early theories is that productions can respond to patterns of elements rather than to each element separately.

ACT also explains the fact that in the presence of correlated stimuli, one stimulus may partially or completely overshadow a second one (see MacKintosh, 1975, for a review). Thus, if both A and B are trained as a correlated pair to response R, one may find that A has less ability to evoke R alone than if it were the only cue associated with R. Sometimes if B is much more salient, A may have no control over R at all. In ACT the discrimination mechanism will choose among the available features (A, B, and other irrelevant stimuli) with probabilities reflecting their salience. Thus, it is possible that a satisfactory discrimination involving B will be found and that this production will be strengthened to where it is dominating behavior and producing satisfactory results so that the A discrimination will never be made. It is also possible that even after a production is formed with the B discrimination, it is too weak to apply, an error occurs, and an A discrimination occurs. In that case both A and B might develop as alternate and equally strong bases for responding. Thus, the ACT theory does not predict that overshadowing will always occur but allows it to occur and predicts it to be related to the differential salience of the competing stimuli.

Evidence for the ACT Tuning Mechanisms

We have spent and are spending some considerable effort in gathering evidence relevant to evaluation of the tuning mechanisms described here. One issue concerns sufficiency: Are the learning mechanisms described here adequate to produce intelligent, adaptive, and stable performance as an end product? Anderson et al. (1981) describe our efforts to establish sufficiency for the domain of geometry theorem proving. We were able to show adaptive behavior on the small scale, but size limitations on our simulation program prevented us from assessing what the eventual behavior of the program would be if it were to work through a course of study like that of a high school student.

Because of various technical optimizations, I was able to assess this issue of sufficiency more adequately in the case of language acquisition (Anderson, 1981a). Although not able to achieve anything so grand as a system with total competence in a language, I was able to show that the learning mechanisms did converge to produce correct syntactic behavior on various subsets of a number of different languages.

It is difficult to assess the psychological accuracy of the programs in these areas. Because of the scale of the phenomena, there are no careful empirical characterizations of the variety with which an experimental psychologist likes to work. For the same reason it is not possible to get reliable statistics about the performance of the simulation programs. In addition, the simulations require rather major simplifying assumptions. So, to check the empirical accuracy of the tuning mechanisms, we have looked at behavior of the program on various schema-abstraction or prototype-formation tasks (e.g., Franks & Bransford, 1971; Hayes-Roth & Hayes-Roth, 1977; and Medin & Schaffer, 1978). These are relatively tractible empirical phenomena on which it is possible to do careful analytic experiments. In Anderson et al. (1979), we were able to show our mechanisms capable of simulating many of the established phenomena. In Elio and Anderson (1981), we were able to confirm predictions from ACT's generalization mechanism that served to discriminate it from other theories.

Procedural Learning: The Power Law

One aspect of skill acquisition is distinguished both by its ubiquity and by its surface contradiction to ACT's multiple stage, multiple mechanism view of skill development. This is the log-linear or power law for practice: A plot of the logarithm of time to perform a task against the logarithm of amount of practice approximates a straight line. It has been widely discussed with respect to human performance (Fitts & Posner, 1967; Welford, 1968) and has been the subject of a number of recent theoretical analyses (Newell & Rosenbloom, 1981; Lewis, Note 3). It is found in phenomena as

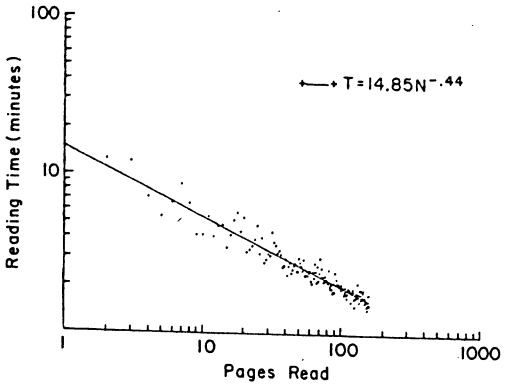


Figure 10. The effect of practice on the speed with which subjects can read inverted text. (From Kolers, 1975.)

diverse as motor skills (Snoddy, 1926), pattern recognition (Neisser, Novich, & Lazar, 1963), problem solving (Neves & Anderson, 1981), memory retrieval (Anderson, in press), and, suspiciously, in machine building by industrial plants (an example of institutional learning rather than human learning; Hirsch, 1952). Figure 10 illustrates one example: the effect of practice on the speed with which inverted text can be read (Kolers, 1975). This ubiquitous phenomenon would seem to contradict the ACT theory of skill acquisition because at first it seems that a theory that proposes changing mechanisms of skill acquisition would not predict the apparent uniformity of the speedup. Also, it is not immediately clear why ACT would predict a power function rather than, say, an exponential function. Because of the ubiquity of the power law, it is important to show that the ACT learning theory is consistent with this phenomenon.

The general form of the equation relating time (T) to perform a task to amount of practice (P) is

$$T = X + AP^{-b}, \qquad (1)$$

where X is the asymptotic speed, X + A is the speed on Trial 1, and b is the slope of the function on a log-log plot—where time is plotted as log(T − X). The value of b is almost always in the interval 0 to 1. The asymptotic X is usually very small relative to X + A, and the rate of approach to asymptote is slow in a power function. This means that it is often possible to get very good fits in plots like Figure 10, assuming a zero asymptote. However, careful analysis of data with enough practice does indicate evidence for nonzero asymptotes.

These facts about skill speedup have appeared contradictory to ACT-like learning mechanisms because ACT mechanisms would seem to imply speedup that is faster than one would obtain with a power law. Composition, as developed earlier, collapsed pairs of productions into single productions. It was noted (p. 384) that composition seems to predict a speedup on the order of Ba^P—which is to say an exponential function of practice, P (a is less than 1). An exponential law, as noted by Newell and Rosenbloom (1981), is in some sense the natural prediction about speedup. It implies that with each practice trial the subject can improve a constant fraction (a) of his current time (or has a constant probability of improving by a constant amount). When we look at ACT's tuning mechanisms of discrimination and generalization, it is harder to make general claims about the speedup they will produce because their speedup will depend on the characteristics of the problem space. However, it is at least plausible to propose that each discrimination or generalization has a constant expected factor of improvement. Composition, generalization, and discrimination improve performance by reducing the expected number of productions applied in performing a task. I will refer to improvement due to reduction in number of productions as *algorithmic improvement.*

In contrast to algorithmic improvement, strengthening reduces the time for individual productions of the procedure to apply. I will show that the strengthening process in ACT does result in a power law. However, even if strengthening obeys a power law, it is not immediately obvious why the total processing, which is a product of both algorithmic improvement and strengthening, should obey a power law. Nonetheless, I will set forth a set of assumptions under which this is just what is predicted by ACT and in so doing will resolve the problem.

Strengthening

Although complex processes like editing or proof generation appear to obey a power

law, it is also the case that simple processes like simple choice reaction time (Mowbray & Rhoades, 1959) or memory retrieval (Anderson, in press) may do the same. In these cases the speedup cannot be modeled as an algorithmic improvement in number of production steps. There cannot be more than a small number of productions (e.g., 10) applying in the less than 500 msec required for these tasks. A process of reducing that number would not produce the continuous improvements observed. Moreover, subjects may well start out with optimal or near optimal procedures in terms of minimum number of productions, so there often is little or no room for algorithmic improvement. Here we have to assume that the speedup observed is due to a basic increase in the rate of production application, as would be produced by ACT's strengthening process.

Recall from the earlier discussions (see the section on conflict resolution) that time to apply a production is $c + (a/s)$ where s is the production strength, c reflects processes in production application, and a is the time for a unit-strength production to be selected. Strength increases by one unit (a unit is arbitrarily .025 in our theory) with each trial of practice. Therefore, we can simply replace s in the expression above by P, the number of trials of practice. Then, the form of the practice function for production execution in ACT takes the form

$$T = c + aP^{-1}, \tag{2}$$

which is a hyperbolic function, one form of the power law. This assumes that on the first measured trial ($P = 1$), the production already has 1 unit of strength from an earlier encoding opportunity. The time for N such productions to apply would be

$$T = cN + aNP^{-1} \tag{3}$$

$$\text{or } T = C + AP^{-1}, \tag{4}$$

where $C = cN$ and $A = aN$. This is a power law where the exponent is equal to 1 and the asymptote is C. The problem is that unless peculiar assumptions are made about prior practice (see Newell & Rosenbloom, 1981), the exponent obtained is typically much less than 1 (usually in the range .1 to .6).

However, the smaller exponents are to be predicted when one takes into account that there is forgetting or loss of strength from prior practice. Thus, a better form of Equation 4 would be

$$T = C + [A/\sum_{i=0}^{P-1} s(i, P)], \tag{5}$$

where $s(i, P)$ denotes the strength remaining from the ith strengthening when the Pth trial comes about. In the above $s(0, P)$ denotes the strength on Trial P of the initial encoding trial. To understand the behavior of this function we have to understand the behavior of the critical sum:

$$S = \sum_{i=0}^{P-1} s(i, P). \tag{6}$$

Wickelgren (1976) has shown that the strength of the memory trace decays as a power law. Assuming that time is linear in number of practice trials we have

$$s(i, P) = D(P - i)^{-d}, \tag{7}$$

where D is the initial strength and $d < 1$. Combining Equations 6 and 7 we get

$$S = \sum_{i=1}^{P} Di^{-d}. \tag{8}$$

This function is bounded below and above as follows:

$$\frac{D}{1-d}[(P+1)^{1-d} - 1)] < S < \frac{D}{1-d}(P^{1-d} - d); \tag{9}$$

S can be approximated by the average of these upper and lower bounds and because the difference between $(P + 1)^{1-d}$ and P^{1-d} becomes increasingly small with large P we may write

$$S \approx \frac{D}{1-d}(P^{1-d} - X),$$

where $X = (1 + d)/2$. This factor X will become increasingly insignificant as P gets large. So, the important observation is that to a close approximation, total strength will grow as a power law. Substituting back into Equation 5 we get

$$T = C(P) + A'P^{-g}, \quad (10)$$

where $A' = A(1 - d)/D$; $g = 1 - d$; and $C(P)$ obeys the equation

$$C(P) = C + \frac{A(1 - d)X}{DP^{1-d}(P^{1-d} - X)}, \quad (11)$$

which converges to C as P gets large. So for large P, Equation 10 will give a good approximation to a power law. Equation 10 has also proven to provide a good approximation for small P in my hand-calculated examples. Thus, the ACT model predicts that time for a fixed sequence of productions should decrease as a power law with the exponent deviating from 1 (and a hyperbolic function) to the degree that there is forgetting. The basic prediction of a power function is confirmed in simple tasks; the further prediction relating the exponent to forgetting is a difficult issue requiring further research. However, it is known that forgetting does reduce the effect of practice (e.g., Kolers, 1975). Given that forgetting must be an important factor in the long-term development of a skill, the ACT analysis of the power law is at a distinct advantage over other analyses that do not accommodate forgetting effects.

Algorithmic Improvement

There is an interesting relation between this power law for simple tasks, based just on strength accumulation, and the power law for complex tasks where there is also the potential for reduction in number of production steps. We noted in the case of composition that a limit on this process is that all the information to be matched by the composed production must be active in working memory. Because the size of production conditions (despite the optimization produced by proceduralization) tends to increase exponentially with compositions, the requirements on working memory for the next composition tend to increase exponentially with the number of compositions. It is also the case that as successful discriminations and generalizations proceed, there will be an increase in the amount of information that needs to be held in working memory so that another useful feature can be identified. In this case it is not possible to make precise statements concerning the factor of increase, but it is not unreasonable to suppose that this increase is also exponential with number of improvements. It is then implied that the following relation should define the size (W) of working memory needed for the ith algorithmic improvement:

$$W = GH^i, \quad (12)$$

where G and H are the parameters of the exponential function.

The ACT theory predicts that a power law should describe the amount of activation of a knowledge structure as a function of practice (in the concepts or links that define that structure). By the same analysis as the one just given for production strength, ACT predicts that the strength of memory structures should increase as a power function of practice. The strength of a memory structure directly determines the amount of activation it will receive. Thus, we have the following equation describing total memory activation (A) as a function of practice:

$$A = QP^r, \quad (13)$$

where Q and r are the parameters of the power function. (Note that P is raised to a positive exponent, r, less than one.) This equation is more than just theoretical speculation; work in our laboratory on effects of practice on memory retrieval has supported this relation. A small proportion of this work is discussed in Anderson (in press).

There is a strong relationship in the ACT theory between the working memory requirements described by Equation 12 and the total activation described by Equation 13. For an amount, W, of information to be available in working memory, the information must reach a threshold level of activation L, which means that the total amount of activation of the information structure will be described by

$$A = WL. \quad (14)$$

Equations 12, 13, and 14 may be combined to derive a relation between the number of improvements (i) and amount of practice:

$$i = r\frac{\log(P)}{\log(H)} + \frac{\log(Q)}{\log(H)} - \frac{\log(L)}{\log(H)} - \frac{\log(G)}{\log(H)} \quad (15)$$

or, more simply,

$$i = a + b \log (P). \quad (16)$$

Thus, because of working memory limitations, the rate of algorithmic improvement is a logarithmic rather than a linear function of practice. Continuing with the assumption that the number of steps (N) should be reduced by a constant fraction, f, with each improvement, we get

$$N = N_0 f^i \quad (17)$$

or

$$N = N_0' P^{-f'}, \quad (18)$$

where

$$f' = -b \log (f) \quad \text{and} \quad N_0' = N_0 f^a. \quad (19)$$

Thus, the number of productions to be applied should decrease as a power function of practice. Equation 18 assumes that in the limit, 0 steps are required to perform the task, but there must be some minimum N* that is the optimal procedure. Clearly, N* has at least the value 1. Exactly, how to introduce this minimum into Equation 18 will depend on one's analysis of the improvements. If we simply add it, we get the standard power function for improvement to an asymptote:

$$N = N^* + N_0' P^{-f'}. \quad (20)$$

Let us review the analysis of the power law to date. We started with the observation that, assuming that the rate of algorithmic improvement is linear with practice and that each improvement has a proportional decrease in number of productions, an exponential practice function is predicted, not a power practice function. We noted that the mechanisms of strength accumulation predict that individual productions should speed up as a power function. Similar strength dynamics governing the growth of working memory size imply that the rate of algorithmic improvement is actually logarithmic and therefore the decrease in number of productions would be a power function.

It should be noted that the relation between working memory capacity and improvements in the production algorithm corresponds to a common subject experience on complex tasks. Initially, subjects report feeling swamped, trying just to keep up with the task, and have no sense of the overall organization of the task. With practice, subjects report beginning to perceive the structure of the task and claim to be able to see how to make improvements. It is certainly the case that we observe subjects to be better able to maintain current state and goal and better able to retrieve past goals and states of the task. Thus, it seems that their working memory for the problem improves with practice and subjects claim that being able to apprehend at once a substantial portion of the problem is what is critical to making improvements.

Algorithmic Improvement and Strengthening Combined

The total time to perform a task is determined by the number of productions and the time per production. Therefore, the simplest prediction about total time (TT) would be to combine multiplicatively Equation 10, describing time per production, and Equation 20, describing number of productions:

$$TT = (N^* + N_0' P^{-f'})(C' + A' P^{-g}). \quad (21)$$

Because of the asymptotic components, N* and C', the above will not be a pure power law but it will look like a power function to a good approximation (as good an approximation as is typically observed empirically). If N* and C' were 0, then we would have a pure power law of the form

$$TT = N_0' A' P^{-(f'+g)}, \quad (22)$$

which has a zero asymptote. Because the initial time is so large relative to final time, most data are fit very well assuming a zero asymptote. This is the form of the equation we will use for further discussion.

One complication ignored in the foregoing discussion is that algorithmic improvements in the number of productions typically mean creation of new productions. According to the theory new productions start off with low strength. Thus, productions at later points in the experiment will not have been practiced since the beginning of the experiment and will have lower strength than assumed in Equations 21 and 22. Another complication on top of this is that a completely new

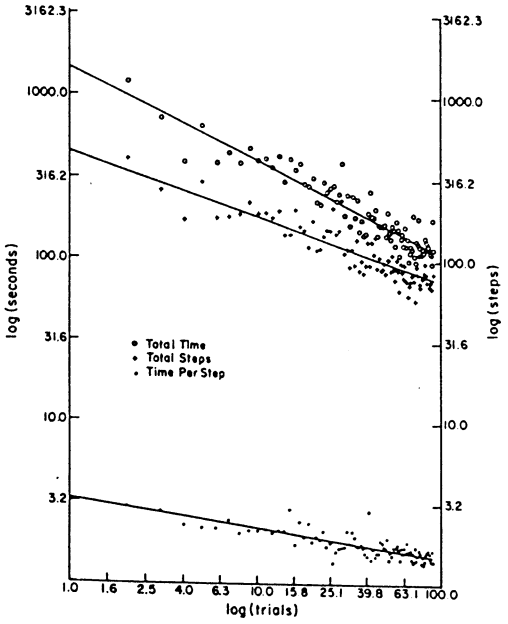


Figure 11. The effect of practice on a reason-giving task. (Plotted separately are the effects on number of steps, time per step, and total time. From Neves and Anderson, 1981.)

set of productions will not be instituted with each improvement; only a subset will change. Suppose that at any time the productions in use were introduced an average of j improvements ago. This means (by Equation 15) that after the ith improvement the average production has been practiced from Trial KL^{i-j} to Trial KL^{i} and therefore has had $KL^{i}(1 - L^{-j})$ trials of practice, where $K = H^{-a/r}$ and $L = H^{1/r}$ from Equations 15 and 16. Thus, the number of trials of practice (P^*) for a production is expected to be a constant fraction of the total number of trials (P) on the task:

$$P^* = qP, \tag{23}$$

where $q = (1 - L^{j})$. This implies that the correct form of Equation 20 is

$$TT = N'_0 A' q^{-g} P^{-(f'+g)}. \tag{24}$$

Thus, this argument does not at all affect the expectation of a power function.

An Experimental Test

The basic prediction of this analysis is that both number of productions and time per production should decrease as a power function of practice. As a result, total time will decrease as a power function. Neves and Anderson (1981) have tested this prediction in an experiment that studied subjects' ability to give reasons for the lines of an abstract logic proof. This reason-giving task is modeled after a frequent kind of exercise found in high school geometry texts (see Figure 3). However, we wanted to use the task with college students and wanted to see the effects of practice, starting from the beginning. Therefore, we invented a novel artificial proof system. Each proof consisted of 10 lines. Each line could be justified as a given or derived from earlier lines by application of one of nine postulates. Subjects could only see the current line of the proof and had to request of a computer to display particular prior lines, givens, or postulates. The method of requesting this information was very easy, and so we hoped to be able to trace, by subjects' request behavior, the steps of the algorithm they were following. The relation between requests and production application is almost certainly one to many, but we believe that we can use these requests as an index of the number of productions that are applying. The basic assumption is that the ratio of productions to requests will not change over time. This assumption certainly could be challenged, but I think it is plausible and is strongly supported by the orderliness of the results. Under this assumption if we plot number of requests as a function of practice, we are looking at the reduction in the number of productions or algorithmic improvement. If we plot time per request, we are looking at the improvement in the speed of individual productions.

Figure 11 presents the analysis of these data averaged over three subjects (individual subjects show the same pattern). Subjects took about 25 minutes to do the first problem. After 90 problems they were often taking under 2 minutes to do the proofs. This reflects the impact of approximately 10 hours of practice. As can be seen from Figure 11, both number of steps (information requests) and time per step (interval between requests) go down as power functions of practice. Hence, total time also obeys a power function. The exponent for the num-

ber of steps is −.346 (ranging from −.315 to −.373 for individual subjects), whereas the exponent for the time per step is −.198 (ranging from −.144 to −.226).

The Neves and Anderson (1981) experiment does provide evidence that underlying a power law in complex tasks are power laws both in number of steps applied and in time per step. I have shown how a power law in strength accumulation may underlie both of these phenomena. Although it is true that algorithmic improvement would tend to produce exponential speedup, the underlying strength dynamics determine working memory capacity and produce a power function in algorithmic improvement. It is tempting to think of these strength dynamics as describing a process at the neural level of the system. Therefore, it is interesting to note Eccles's (1972) review of the evidence that individual neurons increase with practice and decrease with disuse in their rate of transmitter release and pickup across synapses.

Summary

We have now reviewed the basic progression of skill acquisition according to the ACT learning theory. It starts out as the interpretive application of declarative knowledge; this becomes compiled into a procedural form, and this procedural form undergoes a process of continual refinement of conditions and raw increase in speed. In a sense this is a stage analysis of human learning. Much as other stage analyses of human behavior, this stage analysis of ACT is being offered as an approximation to characterize a rather complex system of interactions. Any interesting behavior is produced by a set of elementary components, and different components can be at different stages. For instance, part of a task can be performed interpretively, whereas another part is performed as compiled.

The claim is that the configuration of learning mechanisms described is involved in the full range of skill acquisition from language acquisition to problem solving to schema abstraction. Another strong claim is that the basic control architecture across these situations is hierarchical, goal structured, and basically organized for problem solving. This echoes the claim made elsewhere (Newell, 1980) that problem solving is the basic mode of cognition. The claim is that the mechanisms of skill acquisition basically function within the mold provided by the basic problem-solving character of skills. As skills evolve they become more tuned and compiled, and the original search of the problem space may drop out as a significant aspect. I have presented a variety of theoretical analyses and experimental analyses that provide positive evidence for this broad view of skill acquisition. Clearly, many more analyses and experimental tests can be done. However, the available evidence at least conveys a modest degree of credibility to the theory presented.

In conclusion, I would like to point out that the learning theory proposed here has achieved a unique accomplishment. Unlike past learning theories it has cogently addressed the issue of how symbolic or cognitive skills are acquired. (Indeed, I have been so focused on this, I have ignored some of the phenomena that traditional learning addressed, such as classical conditioning.) The inadequacies of past learning theories to account for symbolic behavior have been a major source of criticism. On the other hand, unlike many of the current cognitive theories, ACT not only provides an analysis of the performance of a cognitive skill but also an analysis of its acquisition. Many researchers (e.g., Estes, 1975; Langley & Simon, 1981; Rumelhart & Norman, 1978) have lamented how the strides in task analysis within cognitive psychology have not been accompanied by strides in development of learning theory.

If I were to select the conceptual developments most essential to this theory of the acquisition of cognitive skills, I would point to two. First, there is the clear separation made in ACT between declarative knowledge (propositional network of facts) and procedural knowledge (production system). The declarative system has the capacity to represent abstract facts. The production system through its use of variables can process the propositional character of this data base. Also, productions through their reference to goal structures have the capacity to shift

attention and control in a symbolic way. These basic symbolic capacities are essential to the success of the learning mechanisms. Knowledge is integrated into the system by first being encoded declaratively and then being interpreted. We argued that the successful integration of knowledge into behavior requires that it first go through such an interpretive stage. The various learning mechanisms all are structured around variable use and reference to goal structures. Moreover, the learning processes impact on the course of the symbolic processing, making it both faster and more judicious in choice. In ACT we see how learning and symbolic processing could be synergetic. These two aspects of cognition surely are synergetic in man, and this fact commends the theory for consideration at least as much as any specific issue that was considered.

Second is the ACT production system architecture itself. Productions are relatively simple and well-defined objects, and this is essential if one is to produce general learning mechanisms. The general learning mechanisms must be constituted so that they will correctly operate on the full range of structures (productions) that they might encounter. It is possible to construct such learning mechanisms for ACT productions; it would not be possible if the procedural formalism were as diverse and unconstrained as are LISP functions. ACT productions have the virtue of stimulus–response bonds with respect to their simplicity but also have considerable computational power. A problem with many production system formalisms with respect to learning is that it is hard for the learning mechanism to appreciate the function of the production in the overall flow of control. This is why the use of goal structures is such a significant augmentation to the ACT architecture. By inspecting the goal structure in which a production application participates, it is possible to understand the role of the production. This is essential to a system that learns by doing.

Reference Notes

1. Rychener, M. D. *Approaches to knowledge acquisition: The instructable production system project.* Unpublished manuscript, Carnegie-Mellon University, 1981.
2. Anderson, J. R., Kline, P. J., & Beasley, C. M. *A theory of the acquisition of cognitive skills* (ONR Tech. Rep. 77-1). New Haven, Conn.: Yale University, Department of Psychology, 1977.
3. Lewis, C. H. *Speed and practice.* Unpublished manuscript, 1979. (Available from C. H. Lewis, IBM Research Center, Yorktown Heights, New York 10598.)

References

Anderson, J. R. *Language, memory, and thought.* Hillsdale, N.J.: Erlbaum, 1976.

Anderson, J. R. *Cognitive psychology and its implications.* San Francisco, Calif.: Freeman, 1980.

Anderson, J. R. A theory of language acquisition based on general learning mechanisms. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, 1981, 97–103. (a)

Anderson, J. R. Tuning of search of the problem space for geometry proofs. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, 1981, 165–170. (b)

Anderson, J. R., Retrieval of information from long-term memory. *Science*, in press.

Anderson, J. R., Greeno, J. G., Kline, P. J., & Neves, D. M. Acquisition of problem-solving skill. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition.* Hillsdale, N.J.: Erlbaum, 1981.

Anderson, J. R., Kline, P. J., & Beasley, C. M. A general learning theory and its application to schema abstraction. In G. H. Bower (Ed.), *The psychology of learning and motivation* (Vol. 13). New York: Academic Press, 1979, 277–318.

Anderson, J. R., Kline, P. J., & Beasley, C. M. Complex learning processes. In R. E. Snow, P. A. Federico, & W. E. Montague (Eds.), *Aptitude, learning, and instruction* (Vol. 2). Hillsdale, N.J.: Erlbaum, 1980.

Braine, M. D. S. On learning grammatical order of words. *Psychological Review*, 1963, *70*, 323–348.

Briggs, G. E., & Blaha, J. Memory retrieval and central comparison times in information processing. *Journal of Experimental Psychology*, 1969, *79*, 395–402.

Brown, D. J. H. Concept learning by feature value interval abstraction. *Proceedings of the Workshop on Pattern-Directed Inference Systems*, 1977, 55–60.

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

Brown, R. *A first language.* Cambridge, Mass.: Harvard University Press, 1973.

Burke, C. J., & Estes, W. K. A component model for stimulus variables in discrimination learning. *Psychometrika*, 1957, *22*, 133–145.

Eccles, J. C. Possible synaptic mechanisms subserving learning. In A. G. Karyman & J. C. Eccles (Eds.), *Brain and human behavior.* New York: Springer-Verlag, 1972.

Elio, R., & Anderson, J. R. Effects of category generalizations and instance similarity on schema abstraction. *Journal of Experimental Psychology: Human Learning and Memory*, 1981, *7*, 397–417.

Estes, W. K. Toward a statistical theory of learning. *Psychological Review*, 1950, *57*, 94–107.

Estes, W. K. The state of the field: General problems and issues of theory and metatheory. In W. K. Estes (Ed.), *Handbook of learning and cognitive processes* (Vol. 1). Hillsdale, N.J.: Erlbaum, 1975.

Fitts, P. M. Perceptual-motor skill learning. In A. W. Melton (Ed.), *Categories of human learning.* New York: Academic Press, 1964.

Fitts, P. M., & Posner, M. I. *Human performance.* Monterey, Calif.: Brooks/Cole, 1967.

Forgy, C., & McDermott, J. OPS, a domain-independent production system. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 1977, 933-939.

Franks, J. J., & Bransford, J. D. Abstraction of visual patterns. *Journal of Experimental Psychology*, 1971, *90*, 65-74.

Greeno, J. G. Indefinite goals in well-structured problems. *Psychological Review*, 1976, *83*, 479-491.

Hayes-Roth, B., & Hayes-Roth, F. Concept learning and the recognition and classification of exemplars. *Journal of Verbal Learning and Verbal Behavior*, 1977, *16*, 321-338.

Hayes-Roth, F., & McDermott, J. Learning structured patterns from examples. *Proceedings of the Third International Joint Conference on Pattern Recognition*, 1976, 419-423.

Heinemann, E. C., & Chase, S. Stimulus generalization. In W. K. Estes (Ed.), *Handbook of learning and cognitive processes* (Vol. 2). Hillsdale, N.J.: Erlbaum, 1975.

Hirsch, W. Z. Manufacturing progress functions. *Review of Economics and Statistics*, 1952, *34*, 143-155.

Jurgensen, R. C., Donnelly, A. J., Maier, J. E., & Rising, G. R. *Geometry.* Boston, Mass.: Houghton Mifflin, 1975.

Kendler, H. H., & Kendler, T. S. From discrimination learning to cognitive development: A neobehavioristic odyssey. In W. K. Estes (Ed.), *Handbook of learning and cognitive processes* (Vol. 1). Hillsdale, N.J.: Erlbaum, 1975.

Kolers, P. A. Memorial consequences of automatized encoding. *Journal of Experimental Psychology: Human Learning and Memory*, 1975, *1*, 689-701.

Langley, P., & Simon, H. A. The central role of learning in cognition. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition.* Hillsdale, N.J.: Erlbaum, 1981.

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

Larson, J., & Michalski, R. S. Inductive inference of VL decision rules. *Proceedings of the Workshop on Pattern-Directed Inference Systems*, 1977, 38-44.

Lewis, C. H. *Production system models of practice effects.* Unpublished doctoral dissertation, University of Michigan, 1978.

Luchins, A. S. Mechanization in problem solving. *Psychological Monographs*, 1942, *54*(6, Whole No. 248).

Luchins, A. S., & Luchins, E. H. *Rigidity of behavior; a variational approach to the effect of Einstellung.* Eugene: University of Oregon Books, 1959.

MacKintosh, N. J. From classical conditioning to discrimination learning. In W. K. Estes (Ed.), *Handbook of learning and cognitive processes* (Vol. 1). Hillsdale, N.J.: Erlbaum, 1975.

Maratsos, M. P., & Chalkley, M. A. The internal language of children's syntax: The ontogenesis and representation of syntactic categories. In K. Nelson (Ed.), *Children's language* (Vol. I). New York: Gardner Press, 1981.

McNeill, D. On theories of language acquisition. In T. R. Dixon & D. L. Horton (Eds.), *Verbal behavior and general behavior theory.* Englewood Cliffs, N.J.: Prentice-Hall, 1968.

Medin, D. L. Theories of discrimination learning and learning set. In W. K. Estes (Ed.), *Handbook of learning and cognitive processes* (Vol. 3). Hillsdale, N.J.: Erlbaum, 1976.

Medin, D. L., & Schaffer, M. M. A context theory of classification learning. *Psychological Review*, 1978, *85*, 207-238.

Miller, G. A., Galanter, E., & Pribram, K. H. *Plans and the structure of behavior.* New York: Holt, Rinehart & Winston, 1960.

Mowbray, G. H., & Rhoades, M. V. On the reduction of choice reaction times with practice. *Quarterly Journal of Experimental Psychology*, 1959, *11*, 16-23.

Neisser, U., Novick, R., & Lazar, R. Searching for ten targets simultaneously. *Perceptual and Motor Skills*, 1963, *17*, 955-961.

Neves, D. M. *Learning procedures from examples.* Unpublished doctoral dissertation, Carnegie-Mellon University, 1981.

Neves, D. M., & Anderson, J. R. Knowledge compilation: Mechanisms for the automatization of cognitive skills. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition.* Hillsdale, N.J.: Erlbaum, 1981.

Newell, A. Reasoning, problem-solving, and decision processes: The problem space as a fundamental category. In R. Nickerson (Ed.), *Attention and performance VIII.* Hillsdale, N.J.: Erlbaum, 1980.

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

Norman, D. A. Discussion: Teaching, learning, and the representation of knowledge. In R. E. Snow, P. A. Federico, & W. E. Montague (Eds.), *Aptitude, learning, and instruction* (Vol. 2). Hillsdale, N.J.: Erlbaum, 1980.

Rudy, J. W., & Wagner, A. R. Stimulus selection in associative learning. In W. K. Estes (Ed.), *Handbook of learning and cognitive processes* (Vol. 2). Hillsdale, N.J.: Erlbaum, 1975.

Rumelhart, D. E., & Norman, D. A. Accretion, tuning, and restructuring: Three modes of learning. In J. W. Cotton & R. Klatzsky (Eds.), *Semantic factors in cognition.* Hillsdale, N.J.: Erlbaum, 1978.

Rychener, M. D., & Newell, A. An instructible production system: Basic design issues. In D. A. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems.* New York: Academic Press, 1978.

Schneider, W., & Shiffrin, R. M. Controlled and automatic human information processing: I. Detection, search, and attention. *Psychological Review*, 1977, *84*, 1-66.

Shiffrin, R. M., & Dumais, S. T. The development of automatism. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition.* Hillsdale, N.J.: Erlbaum, 1981.

Shiffrin, R. M., & Schneider, W. Controlled and automatic human information processing: II. Perceptual learning, automatic attending, and a general theory. *Psychological Review*, 1977, *84*, 127–190.

Snoddy, G. S. Learning and stability. *Journal of Applied Psychology*, 1926, *10*, 1–36.

Spence, K. W. Continuous versus noncontinuous interpretations of discrimination learning. *Psychological Review*, 1940, *47*, 271–288.

Sternberg, S. Memory scanning: Mental processes revealed by reaction time experiments. *American Scientist*, 1969, *57*, 421–457.

Trabasso, T. R., & Bower, G. H. *Attention in learning.* New York: Wiley, 1968.

Vere, S. A. Inductive learning of relational productions. In D. A. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems.* New York: Academic Press, 1978.

Welford, A. T. *Fundamentals of skill.* London: Methuen, 1968.

Wickelgren, W. A. Memory storage dynamics. In W. K. Estes (Ed.), *Handbook of learning and cognitive processes* (Vol. 4). Hillsdale, N.J.: Erlbaum, 1976.

Received July 16, 1981
Revision received December 18, 1981 ■