ACT

A Simple Theory of Complex Cognition

John R. Anderson
Carnegie Mellon University

In the Adaptive Character of Thought (ACT–R) theory, complex cognition arises from an interaction of procedural and declarative knowledge. Procedural knowledge is represented in units called production rules, *and declarative knowledge is represented in units called* chunks. *The individual units are created by simple encodings of objects in the environment (chunks) or simple encodings of transformations in the environment (production rules). A great many such knowledge units underlie human cognition. From this large database, the appropriate units are selected for a particular context by activation processes that are tuned to the statistical structure of the environment. According to the ACT-R theory, the power of human cognition depends on the amount of knowledge encoded and the effective deployment of the encoded knowledge.*

The designation of our species as *homo sapiens* reflects the fact that there is something special about human cognition—that it achieves a kind of intelligence not even approximated in other species. One can point to marks of that intelligence in many domains. Much of my research has been in the area of mathematics and computer programming, fields in which the capacity to come up with abstract solutions to problems is one ability that is frequently cited with almost mystical awe. A good example of this is the ability to write recursive programs.

Consider writing a function to calculate the factorial of a number. The factorial of a number can be described to someone as the result you get when you multiply all the positive integers up to that number. For instance,

$$\text{factorial}(5) = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

In addition (it might appear by arbitrary convention), the factorial of zero is defined to be 1. In writing a recursive program to calculate the factorial for any number N, one defines factorial in terms of itself. Below is what such a program might look like:

$$\begin{aligned} \text{factorial}(N) &= 1 && \text{if } N = 0 \\ &= \text{factorial}(N-1) \times N && \text{if } N > 0. \end{aligned}$$

The first part of the specification, factorial(0) = 1, is just stating part of the definition of factorial. But the second recursive specification seems mysterious to many and appears all the more mysterious that anyone can go from the concrete illustration to such an abstract statement. It certainly seems like the kind of cognitive act that we are unlikely to see from any other species.

We have studied extensively how people write recursive programs (e.g., Anderson, Farrell, & Sauers, 1984; Pirolli & Anderson, 1985). To test our understanding of the process, we have developed computer simulations that are themselves capable of writing recursive programs in the same way humans do. Underlying this skill are about 500 knowledge units called *production rules.* For instance, one of these production rules for programming recursion, which might apply in the midst of the problem solving, is

IF the goal is to identify the recursive relationship in a function with a number argument
THEN set as subgoals to
1. Find the value of the function for some N
2. Find the value of the function for $N-1$
3. Try to identify the relationship between the two answers.

Thus, in the case above, this might lead to finding that factorial(5) = 120 (Step 1), factorial(4) = 24 (Step 2), and that factorial (N) = factorial ($N-1$) × N (Step 3).

We (e.g., Anderson, Boyle, Corbett, & Lewis, 1990; Anderson, Corbett, Koedinger, & Pelletier, 1995; Anderson & Reiser, 1985) have created computer-based instructional systems, called *intelligent tutors,* for teaching cognitive skills based on this kind of production-rule analysis. By basing instruction on such rules, we have been able to increase students' rate of learning by a factor of 3. Moreover, within our tutors we have been able to

Editor's note. Articles based on APA award addresses are given special consideration in the *American Psychologist*'s editorial selection process.

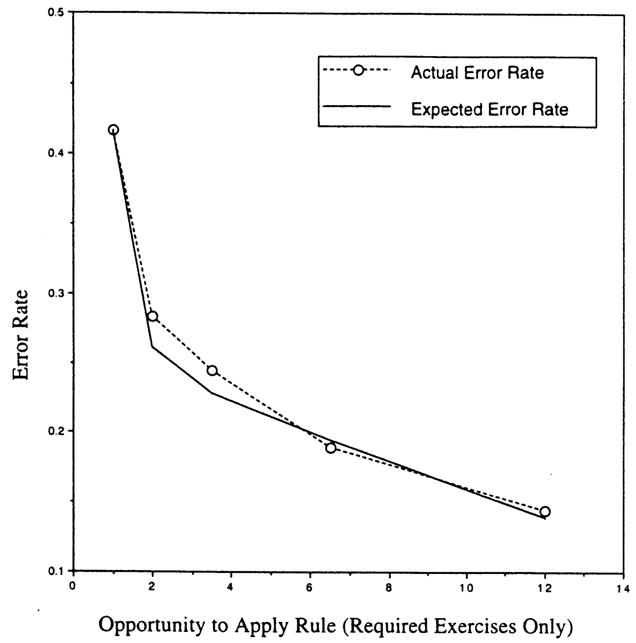
A version of this article was originally presented as part of an Award for Distinguished Scientific Contributions address at the 103rd Annual Convention of the American Psychological Association, New York, NY, August 1995.

Author's note. This research was supported by Grant ONR N0014-90-J-1489 from the Office of Naval Research and Grant SBR 94-21332 from the National Science Foundation.

I would like to thank Marsha Lovett and Lynne Reder for their comments on the article.

Correspondence concerning this article should be addressed to John R. Anderson, Department of Psychology, Carnegie Mellon University, Pittsburgh, PA 15213. For more information on the ACT theory, consult the ACT-R home page on the World Wide Web: http://sands.psy.cmu.edu.

Figure 1
Mean Actual Error Rate and Expected Error Rate Across Successive Rule Applications



Note. From "Student Modeling in the ACT Programming Tutor," by A. T. Corbett, J. R. Anderson, and A. T. O'Brien, 1995, in P. Nichols, S. Chipman, and B. Brennan, *Cognitively Diagnostic Assessment*, Hillsdale, NJ: Erlbaum. Copyright 1995 by Erlbaum. Reprinted by permission.

track the learning of such rules and have found that they improve gradually with practice, as illustrated in Figure 1. Our evidence indicates that underlying the complex, mystical skill of recursive programming is about 500 rules like the one above, and that each rule follows a simple learning curve like Figure 1.

This illustrates the major claim of this article: **All that there is to intelligence is the simple accrual and tuning of many small units of knowledge that in total produce complex cognition. The whole is no more than the sum of its parts, but it has a lot of parts.**

The credibility of this claim has to turn on whether we can establish in detail how the claim is realized in specific instances of complex cognition. The goal of the ACT theory, which is the topic of this article, has been to establish the details of this claim. It has been concerned with three principal issues: How are these units of knowledge represented, how are they acquired, and how are they deployed in cognition?

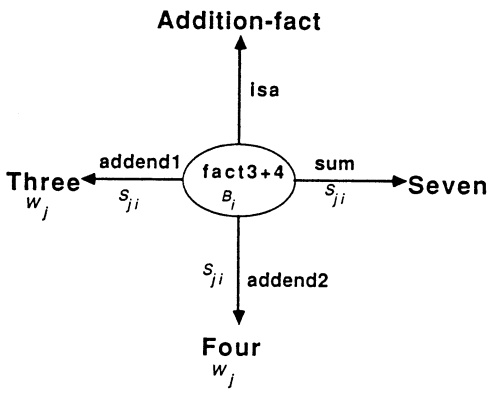
The ACT theory has origins in the human associative memory (HAM) theory of human memory (Anderson & Bower, 1973), which attempted to develop a theory of how memories were represented and how those representations mediated behavior that was observed in memory experiments. It became apparent that this theory only dealt with some aspects of knowledge; Anderson (1976) proposed a distinction between declarative knowledge, which HAM dealt with, and procedural knowledge, which HAM did not deal with. Borrowing ideas from Newell (1972, 1973), it was proposed that procedural knowledge was implemented by production rules. A production-system model called ACTE was proposed to embody this joint procedural–declarative theory. After 7 years of working with variants of that system, we were able to develop a theory called ACT* (Anderson, 1983) that embodied a set of neurally plausible assumptions about how such a system might be implemented and also psychologically plausible assumptions about how production rules might be acquired. That system remained with us for 10 years, but a new system called ACT-R was then put forward by Anderson (1993b). Reflecting technical developments in the past decade, this system now serves as a computer simulation tool for a small research community. The key insight of this version of the system is that the acquisition and deployment processes are tuned to give adaptive performance given the statistical structure of the environment. It is the ACT-R system that we will describe.

Representational Assumptions

Declarative and procedural knowledge are intimately connected in the ACT-R theory. Production rules embody procedural knowledge, and their conditions and actions are defined in terms of declarative structures. A specific production rule can only apply when that rule's conditions are satisfied by the knowledge currently available in declarative memory. The actions that a production rule can take include creating new declarative structures.

Declarative knowledge in ACT-R is represented in terms of chunks (Miller, 1956; Servan-Schreiber, 1991) that are schema-like structures, consisting of an isa pointer specifying their category and some number of additional pointers encoding their contents. Figure 2 is a graphical display of a chunk encoding the addition fact that 3 + 4 = 7. This chunk can also be represented textually:

Figure 2
Network Representation of an ACT-R Chunk



```
fact3+4
    isa        addition-fact
    addend1    three
    addend2    four
    sum        seven
```

Procedural knowledge, such as mathematical problem-solving skill, is represented by productions. Production rules in ACT-R respond to the existence of specific goals and often involve the creation of subgoals. For instance, suppose a child was at the point illustrated below in the solution of a multicolumn addition problem:

$$\begin{array}{r} 531 \\ +248 \\ \hline 9 \end{array}$$

Focused on the tens column, the following production rule might apply from the simulation of multicolumn addition (Anderson, 1993b):

```
IF the goal is to add $n1$ and $n2$ in a column
  and $n1 + n2 = n3$
THEN set as a subgoal to write $n3$ in that column
```

This production rule specifies in its condition the goal of working on the tens column and involves a retrieval of a declarative chunk like the one illustrated in Figure 2. In its action, it creates a subgoal that might involve things like processing a carry. The subgoal structure assumed in the ACT-R production system imposes this strong abstract, hierarchical structure on behavior. As argued elsewhere (Anderson, 1993a), this abstract, hierarchical structure is an important part of what sets human cognition apart from that of other species.

Much of the recent effort in the ACT-R theory has gone into detailed analyses of specific problem-solving tasks. One of these involves equation solving by college students (e.g., Anderson, Reder, & Lebiere, in press). We have collected data on how they scan equations, including the amount of time spent on each symbol in the equation.[1] Figure 3 presents a detailed simulation of the solution of equations like $X + 4 + 3 = 13$, plus the average scanning times of participants solving problems of this form (mixed in with many other types of equations in the same experiment). As can be seen in Parts a–c of that figure, the first three symbols are processed to create a chunk structure of the form **x + 4**. In the model, there is one production responsible for processing each type of symbol. The actual times for the first three symbols are given in Parts a–c of Figure 3. They are on the order of 400 milliseconds, which we take as representing approximately 300 milliseconds to implement the scanning and encoding of the symbol and 100 milliseconds for the production to create the augmentation to the representation.[2]

The next symbol to be encoded, the +, takes about 500 milliseconds to process in Part d. As can be seen, it involves two productions, one to create a higher level chunk structure and another to encode the plus into that structure. The extra 100 milliseconds (over the encoding time for previous symbols) reflect the time for the extra production. The next symbol to be encoded (the 3) takes approximately 550 milliseconds to process (see Part e of Figure 3), reflecting again two productions but this time also retrieval of the fact $4 + 3 = 7$. The mental representation of the equation at this point is collapsed into $x + 7$. The = sign is next processed in Part f of Figure 3. It takes a particularly short time. We think this reflects the strategy of some participants of just skipping over that symbol. The final symbol comes in (see Part g of Figure 3) and leads to a long latency reflecting seven productions that need to apply to transform the equation and the execution of the motor response of typing the number key.

The example in Figure 3 is supposed to reflect the relative detail in which we have to analyze human cognition in ACT-R to come up with faithful models. The simulation is capable of solving the same problems as the participants. It can actually interact with the same experimental software as the participants, execute the same scanning actions, read the same computer screen, and execute the same motor responses with very similar timing (Anderson, Matessa, & Douglass, 1995). When I say, "The whole is no more than the sum of its parts but it has a lot of parts," these are the parts I have in mind. These parts are the productions rules and the chunk structures that represent long-term knowledge and the evolving understanding of the problem.

Knowledge units like these are capable of giving relatively accurate simulations of human behavior in tasks such as these. However, the very success of such simulations only makes salient the two other questions that the ACT-R theory must address, which are how did the prior knowledge (productions and long-term chunks) come to exist in the first place and how is it, if the mind is composed of a great many of these knowledge units, that the appropriate ones usually come to mind in a particular problem-solving context? These are the questions of knowledge acquisition and knowledge deployment.

Knowledge Acquisition

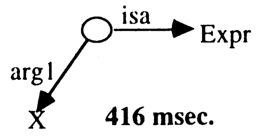
A theory of knowledge acquisition must address both the issue of the origins of the chunks and of the origins of production rules. Let us first consider the origin of chunks. As the production rules in Figure 3 illustrate, chunks can be created by the actions of production rules. However, as we will see shortly, production rules originate from the encodings of chunks. To avoid circularity in the theory we also need an independent source for the origin of the chunks. That independent source involves encoding from the environment. Thus, in the terms of Anderson and Bower (1973), ACT-R is fundamentally a *sensation-*

[1] This involves a scheme wherein participants must point at the part of the equation that they want to read next.

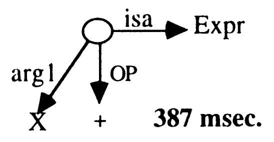
[2] Although our data strongly constrain the processing, there remain a number of arbitrary decisions about how to represent the equation that could have been made differently.

Figure 3
Steps in Solving the Equation x + 4 + 3 = 13

(a) Looking at the X
IF the goal is to solve an equation
  and a variable has been read
  and there are no arguments
THEN store it as the 1st argument.

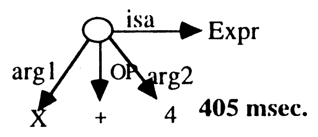


(b) Looking at the +
IF the goal is to solve an equation
  and an operator has been read
  and there is no operator stored



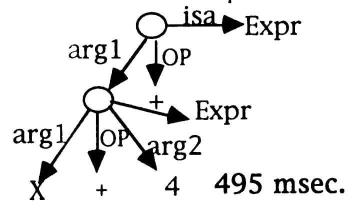
THEN store it as the operator.

(c) Looking at the 4
IF the goal is to solve an equation
  and a number has been read
  and there is no 2nd argument
THEN store it as the 2nd argument.



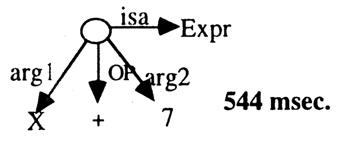
(d) Looking at the +
IF the goal is to solve an equation
  and expression arg1 op arg2 is stored
  and another operator is read
THEN make this expression the first argument

IF the goal is to solve an equation
  and an operator has been read
  and there is no operator stored
THEN store it as the first operator

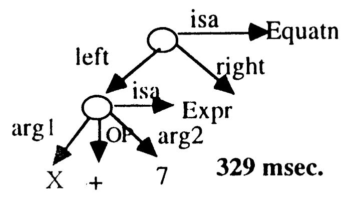


(e) Looking at the 3
IF the goal is to solve an equation
  and a number has been read
  and there is no second argument
THEN store it as the second argument

IF the goal is to solve an equation
  and the current expression is
  (arg + n1) + n2
  and n1 + n2 = n3
THEN make the current expression
  arg + n3



(f) Looking at the =
IF the goal is to solve an equation
  and an equals has been read
THEN make the current structure
  the left hand side



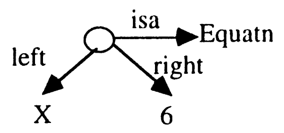
(g) Looking at the 13
IF the goal is to solve an equation
  and a number has been read
  and there is nothing yet on the
    right-hand side
THEN make that the right hand side

IF the goal is to solve an equation
  and there are no more terms in the equation
THEN make the subgoal to transform the equation

IF the goal is to transform an equation
  and the equation is of the form
    arg + n1 = n2
THEN make the equation arg + n1 - n1= n2 - n1

IF the goal is to transform an equation
  and the equation has n1-n1
THEN delete n1-n1.

IF the goal is to transform an equation
  and the equation has n1-n2
  and n2+n3 = n1
THEN replace n1-n2 by n3.



IF the goal is to transform an equation
  and the equation is of the form
    X = n
THEN pop the goal with answer of n

IF the goal is to solve an equation
  and the answer is n
Then type n and pop the goal **1519 msec**

Note. Each panel represents a fixation and gives the average time for that fixation. The production rules are English renditions of the actual ACT-R production rules.

alist theory in that its knowledge structures result from environmental encodings.

We have only developed our ideas about environmental encodings of knowledge with respect to the visual modality (Anderson, Matessa, & Douglass, 1995). In this area, it is assumed that the perceptual system has parsed the visual array into objects and has associated a set of features with each object. ACT-R can move its attention over the visual array and recognize objects. We have embedded within ACT-R a theory that might be seen as a synthesis of the spotlight metaphor of Posner (1980), the feature-synthesis model of Treisman (Treisman & Sato, 1990), and the attentional model of Wolfe (1994). Features within the spotlight can be synthesized into recognized objects. Once synthesized, the objects are then available as chunks in ACT's working memory for further processing. In ACT-R the calls for shifts of attention are controlled by explicit firings of production rules.

The outputs of the visual module are working memory elements called chunks in ACT-R. The following is a potential chunk encoding of the letter H:

```
object
        isa H
        left-vertical    bar1
        right-vertical   bar2
        horizontal       bar3
```

We assume that before the recognition of the object, these features (the bars) are available as parts of an object but that the object itself is not recognized. In general, we assume that the system can respond to the appearance of a feature anywhere in the visual field. However, the system cannot respond to the conjunction of features that define a pattern until it has moved its attention to that part of the visual field and recognized the pattern of features. Thus, there is a correspondence between this model and the feature synthesis model of Treisman (Treisman & Sato, 1990).

A basic assumption is that the process of recognizing a visual pattern from a set of features is identical to the process of categorizing an object given a set of features. We have adapted the Anderson and Matessa (1992) rational analysis of categorization to provide a mechanism for assigning a category (such as H) to a particular configuration of features. This is the mechanism within ACT-R for translating stimulus features from the environment into chunks like the ones above that can be processed by the higher level production system.

With the environmental origins of chunks specified, we can now turn to the issue of the origins of production rules. Production rules specify the transformations of chunks, and we assume that they are encoded from examples of such transformations in the environment. Thus, a student might encounter the following example in instruction:

$$3x + 7 = 13$$

$$3x = 6$$

and encode that the second structure is dependent on the first. What the learner must do is find some mapping between the two structures. The default assumption is that identical structures directly map. In this case, it is assumed the *3x* in the first equation maps onto the *3x* in the second equation. This leaves the issue of how to relate the *7* and *13* to the *6*. ACT-R looks for some chunk structure to make this mapping. In this case, it will find a chunk encoding that *7 + 6 = 13*. Completing the mapping ACT-R will form a production rule to map one structure onto the other:

```
IF the goal is to solve an equation of the form
   arg + n1 = n3
   and n1 + n2 = n3
THEN make the goal to solve an equation
   of the form arg = n2
```

This approach takes a very strong view on instruction. This view is that one fundamentally learns to solve problems by mimicking examples of solutions. This is certainly consistent with the substantial literature showing that examples are as good as or better than abstract instruction that tells students what to do (e.g., Cheng, Holyoak, Nisbett, & Oliver, 1986; Fong, Krantz, & Nisbett, 1986; Reed & Actor, 1991). Historically, learning by imitation was given bad press as cognitive psychology broke away from behaviorism (e.g., Fodor, Bever, & Garrett, 1974). However, these criticisms assumed a very impoverished computational sense of what is meant by imitation.

It certainly is the case that abstract instruction does have some effect on learning. There are two major functions for abstract instruction in the ACT-R theory. On the one hand, it can provide or make salient the right chunks (such as 7 + 6 = 13 in the example above) that are needed to bridge the transformations. It is basically this that offers the sophistication to the kind of imitation practiced in ACT-R. Second, instruction can take the form of specifying a sequence of subgoals to solve a task (as one finds in instruction manuals). In this case, assuming the person already knows how to achieve such subgoals, instruction offers the learner a way to create an example of such a problem solution from which they can then learn production rules like the one above.

The most striking thing about the ACT-R theory of knowledge acquisition is how simple it is. One encodes chunks from the environment and makes modest inferences about the rules underlying the transformations involved in examples of problem solving. There are no great leaps of insight in which large bodies of knowledge are reorganized. The theory implies that acquiring competence is very much a labor-intensive business in which one must acquire one-by-one all the knowledge components. This flies very much in the face of current educational fashion but, as Anderson, Reder, and Simon (1995) have argued and documented, this educational fashion is having a very deleterious effect on education. We need to recognize and respect the effort that goes into acquiring competence (Ericcson, Krampe, & Tesche-Romer, 1993). However, it would be misrepresenting the matter to say that competence is just a matter of getting all the knowledge units right. There is the very serious matter of deploying the right units at the right time, which brings us to the third aspect of the ACT-R theory.

Knowledge Deployment

The human mind must contain an enormous amount of knowledge. Actually quantifying how much knowledge we have is difficult (Landauer, 1986), but we have hundreds of experiences every day, which implies millions of memories over a lifetime. Estimates of the rules required to achieve mathematical competence are in the thousands and to achieve linguistic competence in the tens of thousands. All this knowledge creates a serious problem. How does one select the appropriate knowledge in a particular context? Artificial intelligence systems have run into this problem in serious ways. Expert systems gain power with increased knowledge, but with increases in knowledge these systems have become slower and slower to the point where they can become ineffective. The question is how to quickly identify the relevant knowledge.

Using the rational analysis developed in Anderson (1990), ACT-R has developed a two-pass solution for knowledge deployment. An initial parallel activation process identifies the knowledge structures (chunks and productions) that are most likely to be useful in the context, and then those knowledge structures determine performance as illustrated in our earlier example of the equation solving. The equation solving can proceed smoothly only because of this background activity of making the relevant knowledge available for performance.

Rational analysis posits that knowledge is made available according to its odds of being used in a particular context. Activation processes implicitly perform a Bayesian inference in calculating these odds. According to the odds form of the basic Bayesian formula, the posterior odds of a hypothesis being true given some evidence are

$$\frac{P(H|E)}{P(\bar{H}|E)} = \frac{P(H)}{P(\bar{H})} * \frac{P(E|H)}{P(E|\bar{H})}$$

Posterior-odds = Prior-odds*Likelihood-ratio

or, transformed into log terms,

Log(posterior odds)

= Log(Prior odds) + Log(Likelihood ratio).

Activation in ACT-R theory reflects its log posterior odds of being appropriate in the current context. This is calculated as a sum of the log odds that the item has been useful in the past (log prior odds) plus an estimate that it will be useful given the current context (log likelihood ratio). Thus, the ACT-R claim is that the mind keeps track of general usefulness and combines this with contextual appropriateness to make some inference about what knowledge to make available in the current context. The basic equation is

Activation-Level = Base-level + Contextual-Priming,

where *activation-level* reflects implicitly posterior odds, *base-level* reflects prior odds, and the *contextual-priming* reflects the likelihood ratio. We will illustrate this in three domains—memory, categorization, and problem solving.

Memory

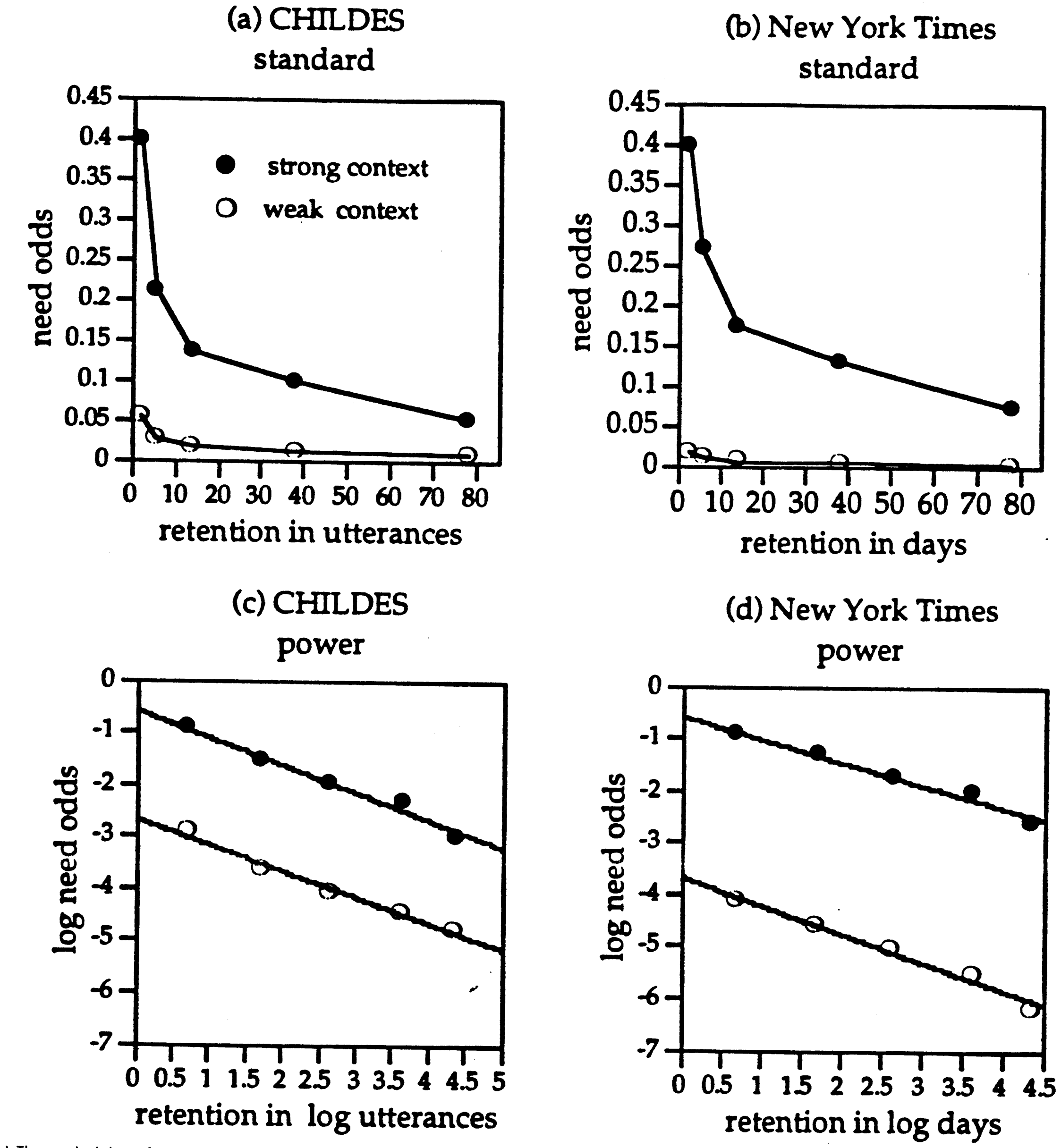
Schooler (1993) did an analysis of how history and context combine to determine the relevance of particular information. For instance, he looked at headlines in the *New York Times*, noting how frequently particular items occurred. In the time period he was considering, the word *AIDS* had a 1.8% probability of appearing in a headline on any particular day. However, if the word *virus* also appeared in a headline, the probability of *AIDS* in that day's headlines rose to 75%. Similarly, he looked at caregiver speech to children in the Child Language Data Exchange System (CHILDES; MacWhinney & Snow, 1990) database. As an example from this database, he found that there was only a 0.9% probability of the word *play* occurring in any particular utterance. On the other hand, if the word *game* also appeared in that utterance, the probability of *play* increased to 45%. Basically, the presence of a high associate serves to increase the likelihood ratio for that item.

Schooler (1993) also examined what factors determined the prior odds of an item. One factor was the time since the word last occurred. As the time increased, the odds went down of the word occurring in the next unit of time. This temporal factor serves as the prior odds component of the Bayesian formula. Schooler examined how these two factors combined in his *New York Times* database and the child-language database. The results are displayed in Figure 4. Parts (a) and (b) show that both the presence of a high associate ("strong context" in Figure 4) and the time since last appearance ("retention" in Figure 4) affect the probability of occurrence in the critical unit of time. It might appear from parts (a) and (b) of Figure 4 that the time factor has a larger effect in the presence of a high associate. However, if one converts the odds scale to log odds and the time scale to log time (see Anderson & Schooler, 1991, for the justification) we get the functions in parts (c) and (d) of Figure 4. Those figures show parallel linear functions representing the additivity that we would expect given the Bayesian log formula above.

The interesting question is whether human memory is similarly sensitive to these factors. Schooler (1993) did an experiment in which he asked participants to complete word fragments, and he manipulated whether the fragments were in the presence of a high associate or not as well the time since the target word had been seen. The data are displayed in log–log form in Figure 5. They once again show parallel linear functions, implying that human memory is combining information about prior odds and likelihood ratio in the appropriate Bayesian fashion and is making items available as a function of their posterior probability.[3] Schooler's demonstration is one of the clear-

[3] These linear functions on a log scale imply that latency is a power function of delay on an untransformed scale. See Rubin and Wenzel (1994) for a discussion.

Figure 4
Probability of a Word Occurring



Note. (a) The probability of a word occurring in the next utterance as a function of the number of utterances since its last occurrence; (b) the probability of a word occurring in the day's headlines as a function of the number of days since its last occurrence. Separate functions are plotted for probability in the presence of high and low associates. Parts (c) and (d) replot this data probability to log odds and time to log time. From "Memory and the Statistical Structure of the Environment," by L. J. Schooler, 1993, p. 58, unpublished doctoral dissertation. Reprinted by permission. CHILDES = Child Language Data Exchange System.

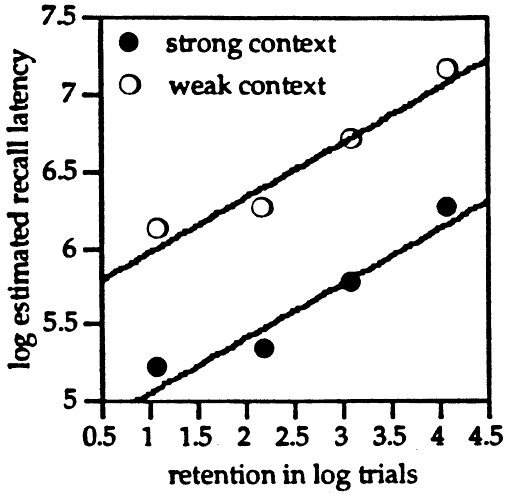
est and most compelling demonstrations of the way the mind tunes access to knowledge to reflect the statistical structure of the environment.

The statistical structure is represented in ACT-R by making the activation of a chunk structure, like the one in Figure 2, a function of activation received from the various elements in the environment plus a base-level activation. The momentary activation of chunk **i** in ACT-R is

$$\mathbf{A_i} = \mathbf{B_i} + \sum_{\mathbf{j}} \mathbf{W_j S_{ji}},$$

where $\mathbf{B_i}$ is the base-level activation of chunk **i**, $\mathbf{W_j}$ is the weighting of contextual chunk **j**, and $\mathbf{S_{ji}}$ is the strength

Figure 5
Log Latency to Complete a Word Fragment as a Function of Time Since the Word Was Seen



Note. Data are plotted separately for fragment completion in the presence of strong versus weak associates. From "Memory and the Statistical Structure of the Environment," by L. J. Schooler, 1993, p. 82, unpublished doctoral dissertation. Reprinted by permission.

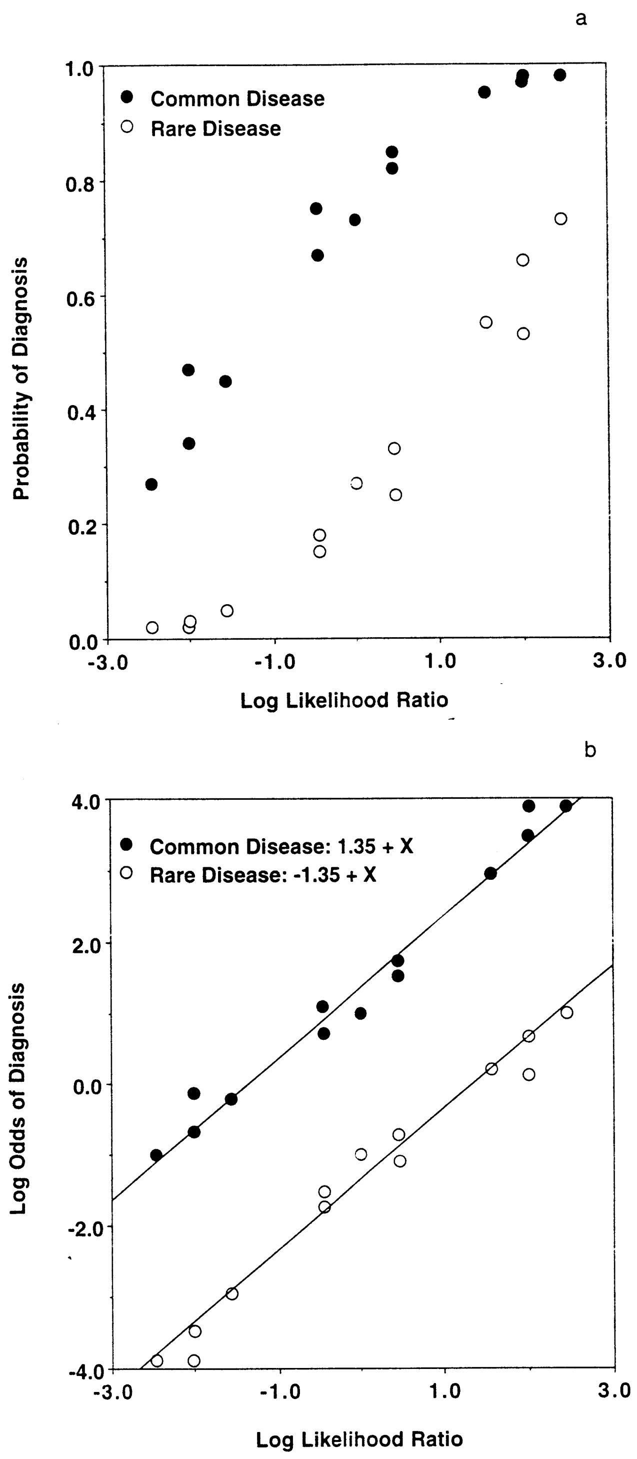
of association between chunk **j** and chunk **i**. For example, if *three* and *four* were chunks in an addition problem, they would be the contextual chunks (the **j**'s) priming the availability of the relevant fact (the **i**) that 3 + 4 = 7.

Categorization

Figure 6 plots some data from Gluck and Bower (1988) in a way the data are not often represented. Gluck and Bower had participants classify patients as having a rare or a common disease given a set of symptoms. Gluck and Bower manipulated what was in effect the likelihood ratio of those symptoms given the disease. Figure 6a plots the probability of the diagnosis of each disease as a function of the log likelihood ratio. It shows that participants are sensitive to both base rates of the disease and to the likelihood ratio of those symptoms for that disease. There has been considerable interest in the categorization literature in participants' failure to respond to base rates, but Figure 6 clearly illustrates that they are often very sensitive. Figure 6b plots the same data with a transformation of the choice probability into log odds. It shows that participants are perfectly tuned to the likelihood ratios, showing slopes of 1.

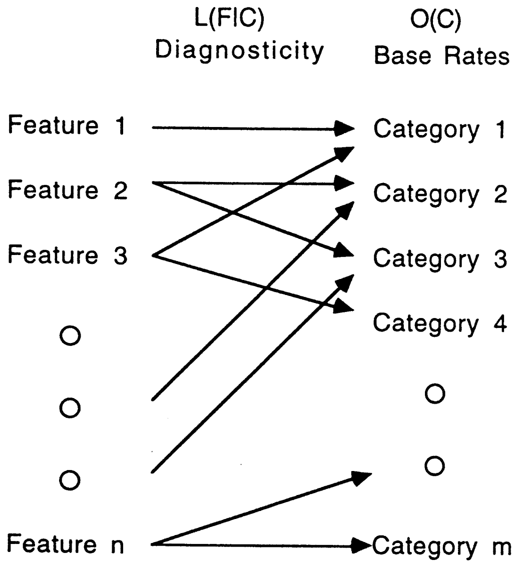
Figure 7 shows the categorization network that we have built into ACT-R to produce categorization data such as those of Gluck and Bower (1988). Various features in chunks spread activation to various categories according to the likelihood of that feature given the category. Categories have base-level activations to reflect their prior frequencies. The resulting activations reflect the log posterior probability of the category. The most active category is chosen to classify the stimulus.

Figure 6
Probability of Diagnosis



Note. (a) Probability of diagnosing a rare versus a common disease as a function of the likelihood ratio of the symptoms. (b) The dependent measure transformed into log odds. From "From Conditioning to Category Learning: An Adaptive Network Model," by M. A. Gluck and G. H. Bower, 1988, *Journal of Experimental Psychology: General, 117*. Copyright 1988 by the American Psychological Association. Adapted by permission of the author.

Figure 7
The Categorization Network Used to Classify Stimuli



Note. The stimulus features spread activation to various categories as a function of their diagnosticity. L(F|C) are the log likelihood ratios and the O(C) are the log prior odds of the categories.

Problem Solving

In her dissertation on strategy selection in problem solving, Lovett (1994) developed what she called the *building sticks task* that is illustrated in Figure 8. Participants are told that their task is to construct a target stick, and they are given various sticks to work with. They can either choose to start with a stick smaller than the target stick and add further sticks to build up to the desired length (called the *undershoot* operator) or to start with a stick longer than the target and cut off pieces equal to various sticks (called the *overshoot* operator). This task is an analog to the Luchins waterjug problem (Luchins, 1942). Participants show a strong tendency to select the stick that gets them closest to their goal. In terms of production rules, it is competition between two alternative productions:

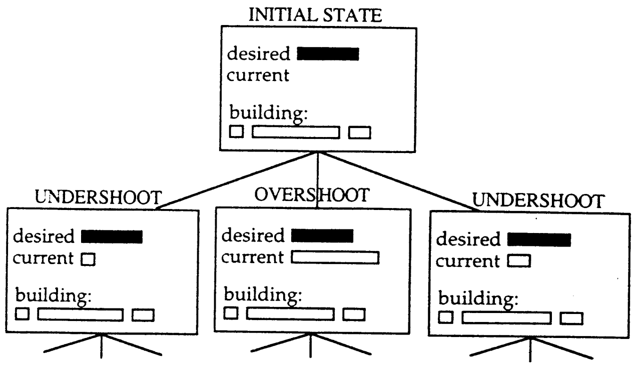
Overshoot

```
IF the goal is to solve the building sticks task
    and there is no current stick
    and there is a stick larger than the goal
THEN add the stick
    and set a subgoal to subtract from it.
```

Undershoot

```
IF the goal is to solve the building sticks task
    and there is no current stick
    and there is a stick smaller than the goal
THEN add the stick
    and set a subgoal to add to it.
```

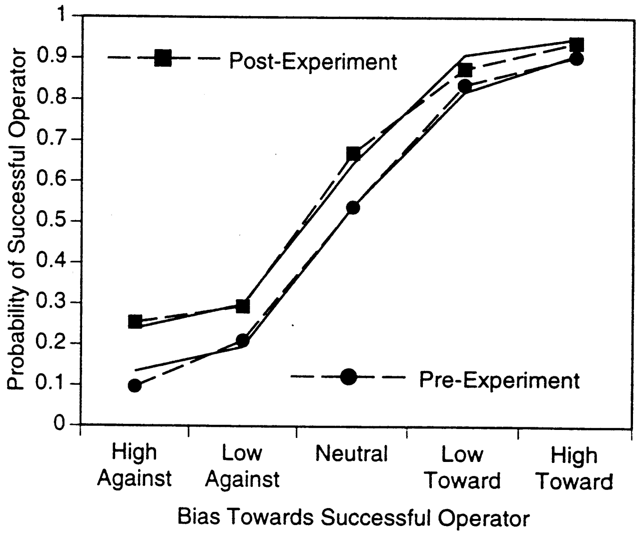
Figure 8
Initial and Successor States in the Building Sticks Task



Note. From "The Effects of History of Experience and Current Context on Problem Solving," by M. C. Lovett, 1994, p. 38, unpublished doctoral dissertation. Reprinted by permission.

Lovett (1994) gave participants experience such that one of the two operators was more successful. Figure 9 shows their tendency to use the more successful operator before and after this experience as a function of the bias toward the operator (determined by how close the stick gets one to the goal relative to the alternative sticks). There are clear effects of both factors. Moreover, Lovett was

Figure 9
Probability of Using an Operator as a Function of the Bias Towards That Operator Both Before and After Experiencing Success With That Operator



Note. From "The Effects of History of Experience and Current Context on Problem Solving," by M. C. Lovett, 1994, p. 87, unpublished doctoral dissertation. Reprinted by permission.

able to model these data by assuming that participants were combining their experience with the operators (serving as prior odds) with the effect of distance to goal (serving as likelihood ratio). In the domain of question-answering strategies, Reder (1987, 1988) had earlier shown a similar combination of information about overall strategy success with strategy appropriateness.

ACT-R estimates the log odds that a production calling for an operator will be successful according to the formula

$$\textbf{Log Odds(Operator)} = \textbf{Log(Prior Odds)} + \textbf{Context Appropriateness},$$

where the prior odds reflect the past history of success and context appropriateness reflects how close the operator takes one to the goal. When multiple productions apply, ACT-R selects the production with the highest expected gain.

Summary

Whether it is selecting what memory to retrieve, what category to place an object in, or what strategy to use, participants and ACT-R are sensitive to both prior information and information about appropriateness to the situation at hand. Although it is hardly a conscious process, people seem to combine this information in a way that is often optimal from a Bayesian perspective. It is this capacity that enables people to have the right knowledge at their fingertips most of the time. Although Bayesian inference is nonintuitive and often people's conscious judgments do not accord with it (even when their behavior does—see Figures 5, 6, and 8), it is really a very simple mechanism. Thus, we achieve great adaptiveness in knowledge deployment by simple statistical inference.

The Metaphor of Simon's Ant

In *The Sciences of the Artificial,* Simon (1981) described a situation in which an ant produced a very complex path across the terrain of a beach. A person observing only the path itself might be inclined to ascribe a great deal of intelligence to the ant. However, it turned out that the complexity of the path is really produced by the complexity of the terrain over which the ant was navigating. As Simon wrote, "An ant, viewed as a behaving system, is quite simple. The apparent complexity of its behavior over time is largely a reflection of the complexity of the environment in which it finds itself" (p. 64). Simon argued that human cognition is much the same—a few relatively simple mechanisms responding to the complexity of the knowledge that is stored in the mind. In Simon's analogy, the complex behavior of the ant maps onto human cognition, the ant's navigating mechanisms map onto basic cognitive mechanisms, and the complexity of the beach maps onto the complexity of human knowledge.

ACT-R fully endorses Simon's (1981) analogy but also carries it one degree further in analyzing the complexity of the knowledge we as humans possess. In this application of the analogy, the complex behavior of the ant maps onto the knowledge we have acquired, the ant's navigating mechanisms maps onto our relatively simple learning processes, and the complexity of the beach maps onto the complexity of our environment. Under this analysis, complex human cognition is just a simple reflection, once removed, of its environment, even as the ant's navigation is directly a simple reflection of its environment.

In a nutshell, ACT-R implies that declarative knowledge is a fairly direct encoding of things in our environment; procedural knowledge is a fairly direct encoding of observed transformations; and the two types of knowledge are tuned in to their application by encoding the statistics of knowledge use in the environment. What distinguishes human cognition from the cognition of other species is the amount of such knowledge and overall cognitive architecture in which this is deployed (particularly the ability for organizing behavior according to complex goal structures).

REFERENCES

Anderson, J. R. (1976). *Language, memory, and thought.* Hillsdale, NJ: Erlbaum.

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

Anderson, J. R. (1990). *The adaptive character of thought.* Hillsdale, NJ: Erlbaum.

Anderson, J. R. (1993a). Problem solving and learning. *American Psychologist, 48,* 35–44.

Anderson, J. R. (1993b). *Rules of the mind.* Hillsdale, NJ: Erlbaum.

Anderson, J. R., & Bower, G. H. (1973). *Human associative memory.* Washington, DC: Winston and Sons.

Anderson, J. R., Boyle, C. F., Corbett, A., & Lewis, M. W. (1990). Cognitive modelling and intelligent tutoring. *Artificial Intelligence, 42,* 7–49.

Anderson, J. R., Corbett, A. T., Koedinger, K., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of Learning Sciences, 4,* 167–207.

Anderson, J. R., Farrell, R., & Sauers, R. (1984). Learning to program in LISP. *Cognitive Science, 8,* 87–130.

Anderson, J. R., & Matessa, M. (1992). Explorations of an incremental, Bayesian algorithm for categorization. *Machine Learning, 9,* 275–308.

Anderson, J. R., Matessa, M., & Douglass, S. (1995). The ACT-R theory of visual attention. In *Proceedings of the Seventeenth Annual Cognitive Science Society,* 61–65.

Anderson, J. R., Reder, L. M., & Lebiere, C. (in press). Working memory: Activation limitations on retrieval. *Cognitive Psychology.*

Anderson, J. R., Reder, L. M., & Simon, H. A. (1995). *Applications and misapplications of cognitive psychology to mathematics education.* Manuscript submitted for publication.

Anderson, J. R., & Reiser, B. J. (1985). The LISP tutor. *Byte, 10,* 159–175.

Anderson, J. R., & Schooler, L. J. (1991). Reflections of the environment in memory. *Psychological Science, 2,* 396–408.

Cheng, P. W., Holyoak, K. J., Nisbett, R. E., & Oliver, L. M. (1986). Pragmatic versus syntactic approaches to training deductive reasoning. *Cognitive Psychology, 18,* 293–328.

Corbett, A. T., Anderson, J. R., & O'Brien, A. T. (1995). Student modeling in the ACT Programming Tutor. In P. Nichols, S. Chipman, & B. Brennan (Eds.), *Cognitively diagnostic assessment* (pp. 19–41). Hillsdale, NJ: Erlbaum.

Ericcson, K. A., Krampe, R. T., & Tesche-Romer, C. (1993). The role of deliberate practice in the acquisition of expert performance. *Psychological Review, 100,* 363–406.

Fodor, J. A., Bever, T. G., & Garrett, M. F. (1974). *The psychology of language.* New York: McGraw-Hill.

Fong, G. T., Krantz, D. H., & Nisbett, R. E. (1986). Effects of statistical training on thinking about everyday problems. *Cognitive Psychology, 18,* 253–292.

Gluck, M. A., & Bower, G. H. (1988). From conditioning to category learning: An adaptive network model. *Journal of Experimental Psychology: General, 117,* 227–247.

Landauer, T. K. (1986). How much do people remember? Some estimates of the quantity of learned information in long-term memory. *Cognitive Science, 10,* 477–493.

Lovett, M. C. (1994). *The effects of history of experience and current context on problem solving.* Unpublished doctoral dissertation, Carnegie Mellon University, Pittsburgh, PA.

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

MacWhinney, B., & Snow, C. (1990). The child language data exchange system: An update. *Journal of Child Language, 17,* 457–472.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63,* 81–97.

Newell, A. (1972). A theoretical exploration of mechanisms for coding the stimulus. In A. W. Melton & E. Martin (Eds.), *Coding processes in human memory* (pp. 373–434). Washington, DC: Winston.

Newell, A. (1973). Production systems: Models of control structures. In W. G. Chase (Ed.), *Visual information processing* (pp. 463–526). New York: Academic Press.

Pirolli, P. L., & Anderson, J. R. (1985). The acquisition of skill in the domain of programming recursion. *Canadian Journal of Psychology, 39,* 240–272.

Posner, M. I. (1980). Orienting of attention. *Quarterly Journal of Experimental Psychology, 32,* 3–25.

Reder, L. M. (1987). Strategy selection in questions answering. *Cognitive Psychology, 19,* 90–138.

Reder, L. M. (1988). Strategic control of retrieval strategies. In G. H. Bower (Ed.), *The psychology of learning and motivation* (pp. 227–259). New York: Academic Press.

Reed, S. K., & Actor, C. A. (1991). Use of examples and procedures in problem solving. *Journal of Experimental Psychology: Learning, Memory, & Cognition, 17,* 753–766.

Rubin, D. C., & Wenzel, A. E. (1994, November). *100 years of forgetting: A quantitative description of retention.* Paper presented at the 35th Annual Meeting of the Psychonomics Society, St. Louis, MO.

Schooler, L. J. (1993). *Memory and the statistical structure of the environment.* Unpublished doctoral dissertation, Carnegie Mellon University, Pittsburgh, PA.

Servan-Schreiber, E. (1991). *The competitive chunking theory: Models of perception, learning, and memory.* Unpublished doctoral dissertation, Carnegie Mellon University, Pittsburgh, PA.

Simon, H. A. (1981). *The sciences of the artificial* (2nd ed.). Cambridge, MA: MIT Press.

Treisman, A. M., & Sato, S. (1990). Conjunction search revisited. *Journal of Experimental Psychology: Human Perception and Performance, 16,* 459–478.

Wolfe, J. M. (1994). Guided search 2.0: A revised model of visual search. *Psychonomic Bulletin & Review, 1,* 202–238.