Control in Act-R and Soar

Todd R. Johnson (JOHNSON.25@OSU.EDU)
Department of Pathology, Division of Medical Informatics
Center for Cognitive Science
The Ohio State University
Columbus, Ohio 43210

Abstract

This paper compares the Act-R and Soar cognitive architectures, focusing on their theories of control. Act-R treats control (conflict resolution) as an automatic process, whereas Soar treats it as a potentially deliberate, knowledge-based process. The comparison reveals that Soar can model extremely flexible control, but has difficulty accounting for probabilistic operator selection and the independent effects of history and distance to goal on the likelihood of selecting an operator. In contrast, Act-R's control is well supported by empirical data, but has difficulty modeling task-switching, multiple interleaved tasks, and dynamic abandoning of subgoals. The comparison also reveals that many of the justifications for each architecture's control structure, such as some forms of flexible control and satisficing, are just as easily handled by both.

Introduction

The last decade has seen the emergence of a variety of cognitive architectures. This is good for cognitive modeling, because architectures provide a ready-made set of tools and theoretical constraints that can assist the cognitive modeling enterprise by constraining the possible models of a set of phenomena or even making the "right" model an obvious consequence of the architectural constraints (Newell, 1990). However, these architectures make many different theoretical distinctions, which can of course have a major influence on the nature of cognitive models supported by each. Despite this, very little work has been done to compare alternative architectures. This paper attempts to rectify this by offering an initial comparison of two of the most well known cognitive architectures: Act-R (Anderson, 1993; Lebiere, 1996) and Soar (Laird, Rosenbloom & Newell, 1986; Laird, Newell & Rosenbloom, 1987; Newell, 1990).

The first goal of this comparison is to identify the similarities and differences between Act-R and Soar by listing each architecture's fundamental theoretical distinctions. The second goal is to assess the empirical support for the major differences in the architectures' control mechanisms. One caveat: since both architectures are products of active research efforts, they are always subject to change. This paper compares the versions of Soar and Act-R available at the end of 1996. One should not assume that this comparison will be accurate for all (or any) future versions of the architectures. However, the major theoretical distinctions in both architectures have been stable for the past decade, so it is likely that portions of this comparison will continue to hold for some time to come.

This is not the first effort to compare and evaluate Soar and Act. Newell, Rosenbloom and Laird (1989) previously compared Soar to Act*, Act-R's precursor. Their goal, however, was more to use Soar and Act as two different examples of cognitive architectures, not to critically evaluate and compare them. The only serious effort to critically evaluate Soar is Cooper and Shallice's (1995) evaluation of Soar as both a psychological theory and an example of the methodology of unified theories of cognition. Their general conclusion is that Soar fairs poorly as a psychological theory and that the unified theory methodology (at least as exemplified by Soar research) does not offer any advantages over traditional psychological research methodology. In contrast to the Soar approach, Anderson and his colleagues have regularly tested Act-R. Many of these results can help us discriminate between Soar and Act-R.

Theoretical Assumptions

This section briefly reviews the theoretical assumptions for Act-R and Soar. These assumptions are summarized in Table 1. Due to space limitations, I only discuss those mechanisms relevant to the comparison of control.

Act-R

The mechanisms proposed in Act-R are based on two foundational assumptions. The first is that "the implementation of Act-R should be in terms of neural-like computation" (Anderson, 1993, p. 12). The second assumption is that "cognition is adapted to the structure of the environment" (Anderson, 1993, p.14). Consequently, many of the mechanisms in Act-R are designed to reflect the statistical nature of the environment.

Act-R is a parallel matching, serial firing production system with a psychologically motivated conflict resolution strategy. Act-R has a declarative memory containing a network of declarative memory elements (DMEs)[1] and a procedural memory containing production rules. DMEs have activation values and associative strengths with other DMEs. The basic cognitive operation in Act-R is a production rule firing. The actions of a production rule in Act-R modify declarative memory—productions cannot directly test or modify procedural memory.

[1] Anderson calls declarative memory elements *chunks*. Since this conflicts with Soar's (nonstandard) usage of the term chunk, I will use the neutral term Declarative Memory Element (DME).

Act-R	Soar
Foundational Assumptions	
Mechanisms should be implemented in neural-like computation	Humans approximate knowledge level systems
Cognition is adapted to the structure of the environment	Humans are symbol systems
Control	
Single goal hierarchy	Single goal hierarchy
Goals originate from task knowledge	Goals are created automatically by the architecture
Adaptive, satisficing conflict resolution	Knowledge-based, least-commitment conflict resolution
Long-Term Memory (LTM)	
Two forms of LTM: Declarative and Procedural	Uniform LTM: All LTM is in procedural form
Procedural memory is represented by production rules	Procedural memory is represented by production rules
Procedural LTM is opaque	LTM is opaque
LTM is permanent	LTM is permanent
Declarative memory is a network with activations and associative strengths	Long-term declarative memory is represented by production rules
Working memory	
Working memory consists of the most active declarative memory elements	Working memory is distinct from LTM
Learning	
The strength of declarative and procedural memory increase as a power function of practice and decrease as a power function of delay	All long-term knowledge arises through chunking
Procedural knowledge is tuned based on experience	
Procedural knowledge is acquired by analogy	
Declarative knowledge is acquired through rules and perception	
Activations and associative strengths are acquired through experience	
Latency Derivation	
Latencies derive from the time needed to match rules, which is the sum of the times needed to match each condition in the rule. The time to match a condition depends on the strength of a rule and the strength of the memory elements it matches.	Constant latency per decision cycle. Overall latency depends on the total number of decision cycles plus the time for external actions.

Table 1: Theoretical distinctions in Act-R and Soar

Act-R repeatedly follows three steps: 1) Instantiate productions whose conditions match DMEs; 2) Select a single instantiation to fire based on Act-R's conflict resolution mechanism; and 3) Fire the selected rule.

Act-R assumes that cognitive behavior is goal-oriented and that goals can give rise to subgoals. A goal in Act-R is a DME that has been pushed onto the architecturally supported goal stack. As with any other DME in Act-R, a goal has a set of attributes (also called slots) and corresponding values, which are just other DMEs. Goals in Act-R typically contain slots that hold the goal's desired and current problem states. Act-R enforces goal-oriented behavior by requiring every rule to match a goal.

Act-R uses a single pushdown (last-in-first-out) goal stack to track goals. The current goal (the one at the top of the stack) is given an activation weight of 1, which is equally divided among the declarative memory elements in the goal's slots.

New goals are created by production rules that create a DME representing the goal and then push the DME onto the goal stack. Goal achievement is signaled by a production rule that pops the goal off the goal stack. In addition, only rules that match the current goal can be instantiated. As discussed later, this restriction leads to some difficulties in modeling flexible control.

Act-R instantiates rules in parallel, but only fires one rule on a given cycle. The time needed to instantiate a production is the sum of the times needed to instantiate each condition in the production. The time to instantiate a condition is a function of the production strength and the activation of the DME to which it matches. This means, that as time goes by during the match process, more and more instantiations will be available to fire. In addition, instantiations will be generated roughly in order of their likelihood of being needed, because rule strength and DME activation increase and decrease as a power law of practice and delay.

Throughout the instantiation process, Act-R must decide whether to wait for additional instantiations or fire the current best instantiation. To determine this, as each instantiation becomes available it is assigned an expected utility $PG-C$, where P is the probability that the goal will be achieved if the production is fired, G is the expected utility of the goal,

and C is the cost of achieving the goal by taking the move specified by the instantiation. Act-R fires the best existing instantiation when the expected improvement of the next instantiation over the current best is less than the estimated cost of the time needed to determine that instantiation.

The production parameters P and C are tuned based on experience with each rule. The probability of a rule's eventual success P is estimated using a weighted average of the prior probability and the empirical probability, which is the experiential probability derived from a rule's history of success and failure. Rule cost reflects the average cost of eventually achieving the goal by way of the rule. It is estimated using a weighted average of a prior cost and an empirical cost. A rule's strength reflects its log odds of being needed, based on its history of use.

Act-R's conflict resolution scheme is an adaptive satisficing process. It is adaptive because a rule's expected value and the time to instantiate the rule depend on parameters that are changed through the learning mechanisms discussed below. It is satisficing because it stops instantiating rules once it finds one that exceeds the acceptance threshold defined above.

Soar

Soar is based on two foundational assumptions. The first is that humans are (at least to some approximation) knowledge level systems (Newell, 1990, p. 113-117). This means that they apply their knowledge in some rational manner to achieve their goals. The second is that humans are a symbol system (Newell, 1990, p. 113-117). Although Newell admits that the underlying neural level might have substantive effects on the symbol level (Newell, 1990, pp. 113-119), Soar itself has always been based on a strong symbol level and many of its theoretical distinctions reflect this fact.

Soar is a parallel matching, parallel firing rule-based system. Soar's rules represent both procedural and declarative knowledge. Soar's working memory contains only declarative knowledge. Rules cannot test other rules, so they can only match declarative knowledge that other rules have already deposited into working memory. On a given production rule cycle, Soar fires every rule that matches. Rule actions can either propose problem solving operators, register a preference for one or more proposed operators, or make modifications to DMEs in working memory.

All problem solving in Soar is viewed as search in a problem space. A problem space is defined by an initial state, one or more goal states, and a set of operators for transforming states. Soar solves problems by repeating the following steps until the problem is solved or abandoned: 1) Fire rules that propose operators to apply to the current state; 2) Fire rules that register a preference for one or more operators; 3) Select an operator to apply to the current state, then apply it. If no operator can be selected, create an impasse. All of these decisions can be either made by knowledge that is directly encoded in rules or by knowledge generated by searching another problem space.

Soar has a single goal stack that is automatically created and managed by the architecture. Goals arise automatically from an impasse—an architecturally detected lack of direct procedural knowledge that inhibits further progress. Whenever Soar reaches an impasse it automatically generates a subgoal to acquire the relevant knowledge.

Conflict resolution in Soar is concerned with the selection of an operator, not a production rule. Soar fires every production rule that matches in a given cycle, however, only a single operator can be applied to a state at any given time. Unlike most rule-based systems, including Act-R, conflict resolution in Soar is completely knowledge-based. Once operators are proposed, other rules can register preferences for (or against) them. A rule can reject an operator, mark it as the best or worst operator, indicate that it is better or worse than one or more other operators, or indicate that two or more operators are equally good. After this preference phase, Soar determines whether the preferences indicate a single best operator. If so, Soar selects that operator for application to the state[2]. If not, or if there are contradictory preferences, then Soar creates a subgoal to resolve the problem. Like any subgoal, this subgoal is achieved by searching a problem space. Thus, Soar represents the operator tie or contradiction event as declarative knowledge in working memory, which enables Soar's procedural knowledge to detect the event and bring to bear the full problem-solving power of the agent.

All long-term knowledge in Soar arises through chunking, a learning mechanism that compiles the results of subgoal search into a production rule that can produce the result without subgoaling. This implies that Soar can only acquire long-term declarative memory by learning a rule that encodes the appropriate conditions in which the declarative memory will be needed.

Soar uses a constant latency per decision cycle, roughly estimated at 50 msec, based on the minimum amount of time needed to make one deliberate cognitive action. Rules that specify external actions are assumed to take additional time consistent with initiating those actions.

Comparison

Both Soar and Act-R organize control around a single goal hierarchy, but their similarities end there. Act-R's production rules are similar to Soar's operators: both are equivalent to a single operator in a problem space. The goal of conflict resolution in Soar is to select an operator, whereas in Act-R it is to select a rule. Goals in Act-R are created by rules, whereas goals in Soar are created by the architecture in response to an impasse.

The distinction between task-initiated goals and architecturally initiated goals becomes important when we consider the architectures' conflict resolution strategies, which are the major theoretical difference between control in Soar and Act-R. Act-R uses an automatic conflict resolution strategy that selects an action (a rule) based on the match time of the available rules and the expected utility of each rule. Other than the match time and expected utility, which depend on the history of declarative and procedural memory, no other knowledge can influence conflict resolution. In contrast,

[2] If the preferences indicate a single set of equivalent operators, then Soar will choose one of those operators at random.

Soar uses an open, knowledge-based conflict resolution strategy that selects an action (an operator) based on all available knowledge.

What kind of empirical support is there for these different conflict resolution strategies? Soar's control scheme was designed around two general characteristics of human cognition. First, is the observation that people behave flexibly. Newell argues that problem-solving methods (such as the weak methods) emerge during problem solving as a function of available knowledge and task demands (Newell, 1990). Soar's conflict resolution strategy is a least-commitment control scheme that naturally supports this kind of behavior. In fact, some of the earliest work in Soar showed that many of the weak methods emerge by combining Soar's control strategy, a few independent bits of general problem-solving knowledge, and various kinds of knowledge of the task domain (Laird et al., 1986).

Research that is more recent explores the use of Soar for modeling interleaved tasks and interaction with the external world (Nelson, Lehman & John, 1994). Soar's ability to respond to any goal in the goal hierarchy along with its ability to replace goals anywhere in its goal stack, make it well-suited for modeling interleaved tasks. Likewise, Soar can respond to changes in its environment by immediately detecting the changes and switching to a different goal.

The second characteristic in support of Soar is that people generally behave in a rational manner to achieve their goals. According to Newell, intelligent systems behave according to the principle of rationality, which states that "the system takes actions to attain its goals, using all the knowledge that it has." (Newell, 1990, p. 50) Soar's knowledge-based conflict resolution strategy with its automatic impasses supports this kind of behavior.

We must next consider whether Act-R can account for the same phenomena. At first glance, it seems that this might be difficult for Act-R. Since its conflict resolution strategy always picks an action regardless of the number of competing actions and how closely those actions are ranked, only a very limited amount of knowledge is used to select actions.

On closer inspection, however, the situation for Act-R is not as bleak. According to Act-R, cognitive skill acquisition begins with the deliberate interpretation of instructions and examples[3], which are then proceduralized by the analogy mechanism into production rules that directly specify appropriate actions. This means that Act-R must begin to solve a task by placing declarative representations of actions into working memory. Once these actions are in working memory, any available knowledge can be brought to bear on them, including knowledge generated by problem solving in a subgoal. For Act-R to reason in this way, it must have a general set of rules for recalling instructions about the task. Once Act-R proceduralizes some of its knowledge, the general rules will compete with the newly formed rules, which means that Act-R will reason deliberately on some trials and automatically on others. However, if the task environment changes such that the procedural knowledge is no longer appropriate, Act-R will again fall back to deliberate reasoning. Thus, it seems that Act-R is capable of displaying at least some flexible, knowledge-based behavior.

Unlike Soar, Act-R's approach to flexible behavior is not directly supported by the architecture. Soar's conflict resolution mechanism would essentially need to be programmed into Act-R's rules. However, the implications of this difference are unclear. B. Chandrasekaran (personal communication) has argued that the Soar architecture might emerge from a lower level architecture because of the need to do problem solving. It is possible that Act-R is one such lower level architecture.

One potential source of difficulty for modeling flexible behavior in Act-R, is that it instantiates only rules that match the current goal. This severely limits Act-R's flexibility in responding to dynamic internal or external changes, because rules related to the current goal have complete control of problem solving, including when to surrender control.

Next, we must consider the evidence in support of Act-R and consider how Soar might account for it. Several results support the Act-R account. First, it is well known that people satisfice—we tend to set an acceptance threshold and then pick the first action that rates above that threshold. This is modeled in Act-R by forcing a decision at each cycle based on comparison among the expected utility of an action, the estimated cost of searching for additional actions, and the estimated gain of the next action. For example, if Act-R is given only a set of rules for making the moves in the Tower of Hanoi along with a rule for detecting the goal state, it will quickly select one of the moves and execute it, without doing any internal lookahead search. In contrast, given the same knowledge, Soar (along with its default knowledge) will do exhaustive depth-first lookahead search until it finds a solution, at which point it will directly solve the problem. Johnson, Zhang and Wang (1994) have produced a modified set of Soar default rules that enables Soar to solve problems with very limited lookahead search, however, the psychological validity of their approach has not been adequately tested.

It is important to understand that Act-R's architectural mechanism for producing satisficing behavior does not apply outside a single rule selection. It is also unclear whether people actually satisfice at the rule matching level proposed by Act-R. However, it is clear that people often satisfice at a higher level by deliberately considering and evaluating options until, at some point, they decide to act rather than continue searching. As with Soar, it is possible to model this behavior in Act-R by deliberately evaluating declarative representations of moves and taking the first move that rises above an acceptance threshold; however, this bypasses Act-R's architectural support for satisficing.

Anderson has also argued that Act-R's conflict resolution mechanism can be used to model speed-accuracy trade-offs by altering the threshold that it uses to decide when to stop matching additional rules. (Anderson, 1993, p. 62) However, this also applies only to the selection of a single rule, not to more extended deliberate decisions. Thus, neither Soar's nor Act-R's conflict resolution mechanisms directly support satisficing during deliberate search. This does not necessarily indicate a flaw with either architecture. Given the computational importance of satisficing, it seems likely

[3] Soar adopts the same view.

that cognition developed to support satisficing behavior at many different levels, using many different mechanisms, including ones that are more deliberate.

Although the source of satisficing behavior in Soar and Act-R are somewhat different, at present, the evidence does not appear to favor either one. This is in part a consequence of the generality of the evidence supporting satisficing behavior. More detailed quantitative evidence might discriminate between the two architectures.

A second body of evidence supports the use of expected utility and instantiation time in Act-R's conflict resolution strategy. Anderson, Kushmerick, and Lebiere (1993) showed that the distribution of an individual's choices from among a set of moves reflects the expected utility of those moves. Act-R models probabilistic move selection by adding a random amount of noise to the expected utilities of each rule instantiation. This is difficult to model in Soar because Soar's control strategy is largely deterministic. The only exception occurs when two or more operators are given indifference preferences (which is meant to indicate that those operators are equally good), in which case Soar will randomly select from among the operators. It is possible to use indifference preferences to implement probabilistic operator selection. Suppose one wants to model a situation in which operator A is twice as likely to be chosen as operator B. By proposing two A operators, one B operator, and making all of them indifferent, Soar will have a 2/3 chance of selecting an A operator, but only a 1/3 chance of selecting B.

There are four problems with the indifference technique. First, it violates the semantics of Soar's indifference preference, which is supposed to mean that the operators are equally good. Second, it requires one to avoid using many of Soar's preferences, such as those that indicate that one operator is better than another, because using such preferences would automatically exclude one or more operators from consideration. Third, since all operators are made indifferent, Soar will never generate an operator tie impasse, effectively bypassing Soar's knowledge-based conflict resolution strategy for selecting among operators. Finally, the technique must be augmented with a theory of learning that shows how chunking can learn new rules that change the distribution of operators in a way that reflects the operators' expected utility. Given that Soar can (theoretically) learn any production that a programmer can write, it seems likely that such a learning theory is possible, but it is unclear how natural or psychologically valid the theory will be.

Anderson, Kushmerick, and Lebiere (1993) also showed that the time to select a rule correlates with instantiation time, not the number of alternative rules. This implies that subjects do not evaluate all available moves, but instead take the first move that exceeds some threshold of acceptability. This is a direct prediction of Act-R's satisficing conflict resolution strategy, which sequentially considers rules, roughly in order of their likelihood of being needed, until it finds an acceptable one.

Soar can produce similar behavior through use of a deliberate satisficing technique as described above. Since this satisficing technique can select an operator without considering all operators, the selection time will depend on the time needed to evaluate only those operators considered before one is selected. This is sufficient to reproduce the general behavior, but it is unclear whether it can model the detailed quantitative data. The predicted times from the Soar and Act-R models stem from different sources. The Act-R model depends on the instantiation time for each rule, whereas the Soar model depends on the number of decision cycles needed to evaluate each operator. If we assume that Soar uses a simple evaluation metric, then each evaluation will take a constant number of decision cycles. In contrast, rule instantiation latencies in Act-R are governed by the number of conditions in each rule, and in the strength of each rule and the matched memory elements.

Lovett and Anderson (1996) showed that the likelihood of selecting a rule instantiation is sensitive to the rule's history of success and distance to goal. In general, people prefer moves that take them closer to the goal and have a higher likelihood of success. In particular, they showed that history and distance to goal independently affect the likelihood of selecting a move. Once subjects gain experience with a rule, their experience will affect their likelihood of selecting instantiations of that rule, regardless of the rule instantiation's distance to goal. This supports Act-R's assumption that history is kept with each rule, without regard for the context in which the rule fires.

Before considering Act-R 3.0's explanation of these phenomena, we must first look at Lovett and Anderson's experimental task in detail. Lovett and Anderson used the building sticks task (BST) in which subjects had to build a stick of a desired length by adding or subtracting sticks of three different lengths. For instance, given building sticks of length 1, 2, and 10, and a desired stick length of 5, a subject could solve the problem by adding two sticks of length 2 and one stick of length 1. Alternatively, the subject could solve the problem by first selecting the stick of length 10, then subtracting two sticks of length 2 and one of length 1. The first solution (2 + 2 + 1 = 5) is called the undershoot strategy, because the initial stick selection is less than the desired length. The second solution (10 - 2 - 2 - 1 = 5) is called the overshoot strategy, because the initial stick selection is longer than the desired length. In problems in which both strategies are applicable, subjects tend to select the strategy that gets them closer to the desired length. In the above example, subjects would tend to select undershoot because 5 is closer to 2 than it is to 10. Lovett and Anderson also showed that the likelihood of selecting a strategy was influenced by the magnitude of a problem's bias, which they defined as the difference between the distance to goal for the best undershoot move and the distance to goal for the best overshoot move. In the example above, the bias is 3 5 = -2. A negative bias indicates a bias toward undershoot, whereas a positive bias indicates overshoot. As the absolute magnitude of the bias increases, so does the likelihood of selecting the corresponding strategy. Finally, the probability that a strategy will succeed (based on its history) affects its likelihood of being selected regardless of problem bias. Although some BST problems can be solved by either undershoot or overshoot, some can only be solved by one of the strategies. Furthermore, problems can be designed that are biased toward one strategy, but solved by the other.

To model this data, Lovett and Anderson (personal communication) have proposed a model that contains both distance-specific rules, which include a test for distance in their conditions, and general rules which apply regardless of distance. The model contains four production rules: closer-overshoot, general-overshoot, closer-undershoot, and general-undershoot. The closer-x rules suggest x only when x moves closer to the goal than the competing moves. The general-x rules are identical to the rules used in the original model: they propose overshoot or undershoot moves regardless of relative distance. In initial tests, this model appears to explain the phenomena. The computation of distance for BST is assumed to be directly available from perception, so sensitivity to the magnitude of the bias is simulated by adding perceptual noise. The model will initially tend to prefer moves according to the bias, because closer-x rules are given a higher prior probability of success, reflecting subjects' past experience that similarity increases probability of success. The independence of history and distance to goal is also achieved, because the model tends to use both the general-x and closer-x rules, although initially, general-x rules have a lower probability of being used. If on average, the general-x rules have more success than the closer-x rules, the general-x rules will have a higher probability of being selected, regardless of distance to goal.

Modeling the independent influence of history and distance to goal presents a challenge to Soar. Soar can easily make use of distance to goal information, however, Soar does not automatically maintain history of success information for each operator. One could program Soar to remember the number of successes and failures for overshoot and undershoot and then use an operator evaluation metric that combines distance to goal with history information; however, it is unlikely that subjects deliberately keep such counts. Another possibility is to use a model that attempts to categorize each problem as an overshoot or undershoot problem. Soar-based Symbolic Concept Acquisition (SCA) can perform such a task and has been shown to produce graded performance with respect to accuracy and response time (Miller & Laird, 1996). SCA tends to respond faster and more accurately to concepts that are similar to frequently encountered concepts. To use SCA, one could use each BST problem and solution as a training example for category learning. New problems are solved by categorizing the current problem using the classification rules acquired during previous attempts. Continued success with one strategy will result in a number of classification rules for that strategy and relatively few rules for the alternative. Thus, the system will be more likely to classify new examples in terms of the successful strategy, although this depends on the similarity to previously categorized examples. Of course, one would need to construct a detailed SCA model of BST to adequately evaluate this solution.

Conclusions

This comparison of control reveals two problem areas for Soar. It is difficult to see how Soar can account for probabilistic move selection as well as the independent effects of history and distance to goal on the likelihood of selecting a move. In contrast, Act-R's control mechanism appears to be well supported by empirical data, but does not appear to support the range of flexible control supported by Soar. The comparison also reveals that many of the justifications for each architecture's control structure, such as flexible control and satisficing, are just as easily handled by both.

Acknowledgments

I thank the members of the Soar and Act-R research community for their detailed comments on earlier drafts of this paper. This research was supported in part by grants N00014-95-1-0241 and N00014-96-1-0472 from the Office of Naval Research, Cognitive and Neural Sciences and Technology Division.

References

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

Anderson, J. R. (1990). *The Adaptive Character of Thought.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J. R. (1993). *Rules of the Mind.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., Kushmerick, N., & Lebiere, C. (1993). Navigation and Conflict Resolution. In J. R. Anderson (Ed.), *Rules of the Mind* (pp. 93-119). Hillsdale, NJ: Lawrence Erlbaum Associates.

Cooper, R., & Shallice, T. (1995). Soar and the case for unified theories of cognition. *Cognition, 55*, 115-149.

Johnson, T. R., Zhang, J., & Wang, H. (1994). Bottom-up recognition learning: a compilation-based model of limited-lookahead learning. In A. Ram & K. Eiselt (Eds.), *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 469-474): Lawrence Erlbaum Associates.

Laird, J., Rosenbloom, P., & Newell, A. (1986). *Universal Subgoaling and Chunking*: Kluwer Academic Publishers.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence, 33*, 1-64.

Lebiere, C. (June 1996). Act-R: A Users Manual [On-line]. Available: ftp://ftp.andrew.cmu.edu/pub/act-r/ftp/release/beta/ACTR3TXT/Manual.rtf

Lovett, M. C., & Anderson, J. R. (1996). History of success and current context in problem solving: Combined influences on operator selection. *Cognitive Psychology, 31*(2).

Miller, C. S., & Laird, J. E. (1996). Accounting for graded performance within a discrete search framework. *Cognitive Science, 20*(4), 499-537.

Nelson, G., Lehman, J. F., & John, B. E. (1994). Integrating cognitive capabilities in a real-time task, *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 658-663). Hillsdale, NJ: Lawrence Erlbaum Associates.

Newell, A. (1990). *Unified Theories of Cognition.* Cambridge, MA: Harvard University Press.

Newell, A., Rosenbloom, P. S., & Laird, J. E. (1989). Symbolic Architectures for Cognition. In M. I. Posner (Ed.), *Foundations of Cognitive Science* (pp. 93-131). Cambridge, MA: MIT Press.