Long-Term Symbolic Learning in Soar and ACT-R

William G. Kennedy (wkennedy@itd.nrl.navy.mil)
Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory, 4555 Overlook Avenue SW
Washington, DC 20385 USA

J. Gregory Trafton (trafton@itd.nrl.navy.mil)
Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory, 4555 Overlook Avenue SW
Washington, DC 20385 USA

Abstract

The characteristics of long-term, symbolic learning were investigated using Soar and ACT-R models of a task to rearrange blocks into specific configurations. Long sequences of problems were run collecting data to answer fundamental questions about long-term, symbolic learning. The questions were whether symbolic learning continues indefinitely, how learned knowledge is used, and whether performance degrades over the long term. It was found that in both systems symbolic learning eventually stopped, ACT-R produced three observable phases of learning, and both Soar and ACT-R suffer from the utility problem of degraded performance with continuous on-line learning.

Introduction

Humans take years to develop the knowledge necessary for reasonably intelligent behavior. Building systems that achieve intelligent behavior is a major goal of the field of Artificial Intelligence (AI), but AI systems have not been run for the equivalent number of years. AI's symbolic learning techniques are typically run only long enough to show performance improvements attributable to the new techniques (Kennedy & De Jong, 2003). In addition, there appears to be a fundamental limit on current AI's symbolic learning techniques. When symbolic learners have been run on series of problems, performance has been found to eventually degrade. This behavior was named the "utility problem" (Minton 1990). On the other hand, humans do not suffer from the utility problem and their long-term learning is familiar to all of us. Understanding the nature and characteristics of learning over the long term in humans is important to achieving intelligent behavior from artificial systems. Explorations into long-term learning that address the utility problem in Soar have been done (Kennedy & De Jong, 2003). This paper reports on research comparing long-term, symbolic learning in Soar and ACT-R.

Learning, or skill acquisition, has been proposed to include the move from problem solving to retrieval (Logan 1988) and to go through three stages (Anderson 1982; Fitts 1964). In the first stage, the cognitive stage, the knowledge is primarily declarative and must be interpreted. General problem solving techniques are employed, such as means-ends analysis, at this initial stage. In the second stage, the associate stage, there is a mix of declarative and procedural knowledge and the problem solving is transitioning from general methods to methods specific to the problem domain. By the third stage, called the autonomous stage, the knowledge is procedural: compiled, fast, and error-free. In this last stage, there is no problem solving and performance improvements are based on psychomotor speedup up to physical limitations (Anderson et al., 2004).

The three stages of behavior have been observed in the complex Kanfer-Ackerman air traffic controller task and correlated with observed behavior on simpler tasks (Ackerman, 1998, 1990). The cognitive stage correlated with general intelligence, perceptual speed correlated with the associate stage, and psychomotor abilities correlated with the autonomous stage. Taatgen and Lee (2003) applied cognitive modeling to this complex task and demonstrated learning in the three stages in one cognitive model.

By long-term symbolic learning (LTSL), we refer to the first two stages, i.e., up to the point where the system has reached steady-state behavior in a symbolic sense.

Computational cognitive models of learning have been successfully built to simulate long-term learning, specifically lifetime learning of arithmetic (Lebiere 1999). AI learning systems typically have not been run long enough to achieve steady-state behavior (Kennedy & De Jong, 2003). One reason is that a performance problem was identified in AI systems with continuous learning by Minton (1990). Problem-solving time grew with the number of productions in the system. His system and others demonstrated degraded performance after 100 or fewer problems were solved. Further research suggested the problem was universal (Holder 1990).

On the other hand, Markovitch and Scott (1988) reported discovering that their symbolic learner's performance actually improved with forgetting. After their system had learned productions (macros) based on 5000 training problems and had established a level of performance in terms of a minimum number of nodes searched, as its learned productions were incrementally removed, its performance improved. Performance peaked when approximately 90 percent of the learned productions had been removed. This surprise finding and the pervasive utility problem indicate weaknesses in the implemented

learning techniques and raise questions about our understanding of the nature of long-term symbolic learning.

We have explored long-term learning with two of the most widely-used systems implementing substantial computational theories of learning, Soar and ACT-R.

Soar (The Soar Group, 2005) is a symbolic learning system with early success modeling the observed power law of practice (Rosenbloom 1986). In 1990, Newell proposed Soar as a unified theory of cognition (Newell 1990). Soar uses one form of short term memory, called "work memory" and one form of long term memory, "chunks". Soar also uses a single learning mechanism, "chunking", in which the solution to a subproblem is formulated and kept as a production. Consistent with the learning theory that Soar implements, productions are retained forever and working memory is transient. About the same time Soar was proposed as a unified theory of cognition, Soar was demonstrated to suffer from the utility problem (Tambe, Newell, & Rosenbloom, 1990). A successful remedy was to modify the theory and introduce forgetting of learned productions based on a threshold of time since last use (Kennedy & De Jong, 2003).

The ACT family of theories (ACT-R Research Group, 2005) has a long history of integrating and organizing psychological data. The current version, ACT-R, derives important constraints from asking what cognitive processes are adaptive given the statistical structure of the environment (Anderson 1990). It has also been broadly tested in psychological and computational terms.

Within ACT-R, all declarative and procedural knowledge is retained, activation levels are calculated which affects the model's ability to recall the knowledge. Of the productions with activation above a threshold, selection of which one to fire is made based on the productions' expected gain. This conflict resolution process allows ACT-R to learn the effectiveness of productions. However, all productions are maintained and subsequent experience can raise a production's activation.

The work reported here is our investigation of the characteristics of long-term symbolic learning, comparing the behavior of Soar and ACT-R on the same problem domain, Blocks World.

Research Questions

We focus on three fundamental questions associated with the nature of long-term learning as modeled by Soar and ACT-R: whether symbolic learning in these computational cognitive models will go on forever, how the learned knowledge is use in problem solving, and whether the utility problem applies to ACT-R as well as symbolic AI learning systems.

The first question addresses fundamental assumptions. If we built a system and started it learning, it seems inevitable that it would eventually learn so much knowledge as to suffer from the utility problem. That inevitability is based on the expectation that learning continues indefinitely as Newell (1990) believed. The theory behind the ACT-R system acknowledges "great reductions in cognitive involvement" (Anderson 2000) based on obeying the power law (Anderson 1982) although it was also recognized that speed up could continue until it reaches the limitations of the physical system. That suggests that the cognitive portion of learning ends, i.e., not including the psychomotor phase of learning. So, Soar theory predicts that symbolic learning continues indefinitely and ACT-R's theory predicts that symbolic learning eventually stops.

The second question concerns how learned knowledge is used. In Soar, it was found that many, up to about half, of learned productions were never used and that patterns of production use justified excising some productions based on lack of recent use (Kennedy & De Jong, 2003). Anderson and Lebiere (1998) believed that Soar generated too many productions as a result of a general problem with Soar's chunking. Therefore, ACT-R evolved to include more caution in the creations of productions. Both Soar and ACT-R support the learning of new productions based on previous productions which would lead to productions being subsumed and no longer useful.

The third question is whether ACT-R, a cognitive modeling system, suffers from the utility problem that plagues AI's symbolic learning systems. Although ACT-R has been used to model lifetime learning including simulating years of learning by solving tens of thousands of problems over hours of computer time (Lebiere 1999), no discussions of performance problems such as the utility problem were found in the literature.

Answers to these questions will contribute to both the fields of AI and Cognitive Science. For Cognitive Science, results should support the development of a theory of long-term learning that addresses the utility of forgetting. For AI, results should be directly applicable to developing learning robots that are expected to operate autonomously for long periods of time.

Method

To address these research questions, we review previous work on long-term symbolic learning with Soar and other symbolic learning systems and report on experiments using Soar and ACT-R available from their user groups (ACT-R Research Group, 2005; The Soar Group, 2005). Blocks-World domain was used because it was simple to implement, has a large search space, and can be scaled easily by increasing the number of blocks. For both systems, runs were made problems generated randomly, with replacement, from the 30 possible three-block problems that can be solved in one step.

The task is set up so that one learned production can solve one problem, which would not be true for more complex problems or problem domains. That simplicity allows us to isolate long term learning characteristics. The expectation is that if we find characteristics within one-step problems, those same characteristics will manifest themselves in more complex models.

The version of Soar used was 8.6.1 for Windows, run within the Soar Java Debugger (in text view). The ACT-R system, version 6 [R145] was run without modification on several desktop PCs and Macintosh computers. Version 6 implements the latest theory on the compilation of sequentially firing productions into new productions. For both systems, default parameters were used except: for Soar, learning was turned on, and for ACT-R, the latency factor (:lf) was set to 0.4 and the declarative retrieval threshold (:rt) was set to -1, both to keep the memory of the problem available during problem solving. We also enabled subsymbolic calculations (:esc T). To run long series of problems in Soar, command files were generated and run in batch. For ACT-R, a Lisp problem generator was run calling ACT-R to run each problem in the series. Traces of the systems' behavior were analyzed off-line.

Task Description

The general task was to rearrange a small set of named blocks on a table from one configuration to another. As an example, a problem is to change the configuration of block-A on block-B with block-B and block-C on the table to a tower configuration with block-C on block-B, block-B on block-A, and block-A is on the table, as shown in Figure 1. Moves consist of selecting an available block and moving it to the table or on top of another block. The criteria to move a block is that the block be clear, i.e., not having another block on top of it, and the destination also be clear. The table is always a legal destination.

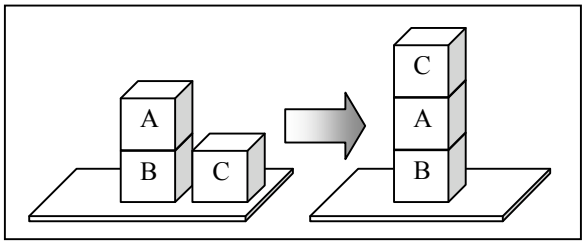

Figure 1: A Blocks World Problem.

Cognitive Models in Soar and ACT-R

Cognitive models of this task were implemented in Soar and ACT-R. Both systems blindly select legal moves with no prior knowledge. There is no planning involved because that was to be learned. Both systems chose from available legal moves based on its knowledge.

When either system achieved the goal configuration, it learned the effective move. In Soar, the solution to the subproblem of choosing the best move for the problem resulted in a "chunk", the new production. In ACT-R, the move was saved as a "chunk" in declarative memory. (Note the different uses of the term "chunk".) When a solution was retrieved, a new production compiling that information was generated. After several recreations of the same production, in accordance with the ACT-R theory, its utility would be increased enough to compete successfully with the solution retrieval production and fire (Anderson, et al 2004).

For Soar, three models of Blocks World are provided with the Soar software (The Soar Group, 2005). The look-ahead model was used because it was the only one that learns. The model representation specifies the current and goal states based on what each block is on top of, either the table or another block. Soar sequentially selects and applies moves of a block to a new location until the current configuration matched the goal. When Soar does not have the knowledge to immediately select a move, it establishes a subproblem to decide the move to make. The solution to a subproblem is saved as learned production, a Soar "chunk", which eliminates the need to repeat the solving of the subproblem. Soar's chunks are not written for specific blocks but are generalized to descriptions of any blocks meeting specific on-top and clear conditions.

A similar model was developed for ACT-R following the approach common in the ACT-R community (e.g., Lebiere 1999). The ACT-R model uses its vision module to read in a problem's initial and goal configurations in terms of what the blocks are on. It then attempts to recall a previous move from the current configuration to the goal. If it finds such a move, that move is executed. If it does not, a legal move is created based on any block that is available to be moved and any possible destination. After making the move, the resulting configuration is evaluated as to whether it achieved the goal. The move achieving the goal is saved as a solution for the problem of achieving the goal from the previous configuration. The model randomly tries moves and recognizes and then saves those that achieve the goal.

Both systems begin by conducting general problem solving, note actions that achieve the goal of solving a problem, and save that knowledge for future use. Both immediately save the knowledge as production. Soar makes that production immediately available for use and ACT-R requires the production to be generated several times to raise its activation enough to be fired.

Results

For each research question, previous research, experimental results, analysis, and observations are discussed.

Does symbolic learning continue indefinitely?

Few researchers have run symbolic learning systems on long series of problems. TacAir-Soar with thousands of productions has been run for hours but does not employ learning (Jones 1994). ACT-R has been used to model life-time learning on thousands of arithmetic problems with ACT-R run for hours (Lebiere 1999), but its performance was not discussed. We therefore assume its performance was not an issue. When we ran both Soar and ACT-R, on Blocks World problems, symbolic learning ended as shown in Figures 2 and 3. The learning curves shown are plots of the cumulative number of new productions in the system against problem number. Plots of the average of five runs for both systems are shown.

Both plots have the same general shape, but their units are very different. Figure 2 shows Soar's learning over only 50 problems while Figure 3 shows ACT-R's learning over

2500 problems. The scale is so different because Soar learns its last production on problem 15 and ACT-R's last production was learned on problem 1751. Because Soar learns very general productions, it does not need to see every possible problem in the domain before learning stops. Our ACT-R model must see every problem to compile a production. The number of productions learned is also very different. All of the Soar runs learned exactly eight productions, all general. ACT-R eventually learned 34-37 productions, one for each problem plus a few due to model coding.

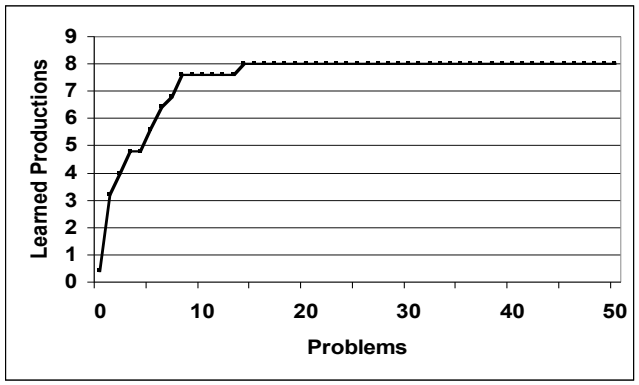

Figure 2: Productions Learned in Soar

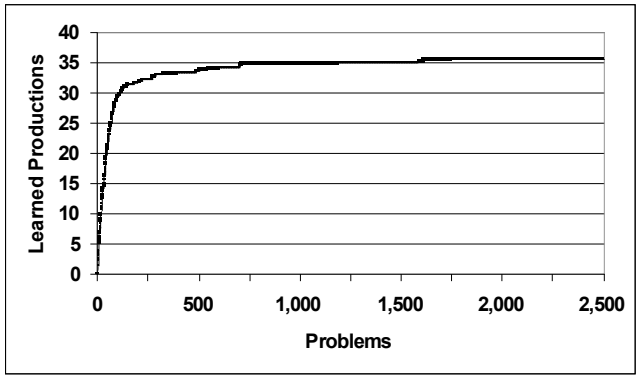

Figure 3: Productions Learned in ACT-R

Although symbolic learning in both Soar and ACT-R stopped, there are differences in the system's behavior. Soar learned productions that are generalized while ACT-R's productions are specific to individual block names. Therefore, Soar needs far fewer productions to cover the same deductions. In addition, the ACT-R model includes its vision module which introduces noise resulting in additional production learning which is not material to the model. As a result, when the learning ends varies between the systems, but the fact that both stop learning is significant.

How is learned knowledge used?

Learning is based on the expectation of future use of the knowledge learned. But, how and when is it used?

A possible set of patterns of use based on when productions were learned was discussed in Kennedy & De Jong (2003). That pattern was that (1) productions learned early are used more frequently than productions learned later, (2) the most recently learned productions are used more frequently, and (3) production use is independent of when learning occurred.

Previous experiments with Soar demonstrated that in some problem domains only about half of Soar's learned productions were ever used and that although there was an overall relatively constant number of productions used on each problem, more recently learned productions were used (Kennedy & De Jong, 2003).

The use of the learned productions was expected to be uniformly distributed due to the environment, a uniform distribution of the problems. To look for patterns in production use, the problems in which productions were used were plotted by problem number. However, no patterns of use were discernable. The statistics of production use confirm uniform use. Over five runs of 250 problems each, Soar learned eight new productions but used only four on each run. Those four productions were each used 25 percent of the time (standard deviation was less than 0.05 percent). Over five runs of 2500 problems, the ACT-R model learned 36 productions and used 26 in four runs and 27 in the other. Over the 2500 problems, each production was each used 3.85 percent of the time (standard deviation was less than 1.25 percent). (Perfectly uniform use would have been 1/26.2 = 3.81 percent of the time.)

On closer examination, a pattern of use, stages of learning, in ACT-R was observed. Anderson described three stages of skill acquisition, cognitive stage, associative stage, and autonomous stage (Anderson 2000). ACT-R model running these simple Blocks-World problems demonstrated three stages of learning. Figure 4 shows, per problem, the number of moves created during problem solving, retrievals of previously successful moves, and firing of learned productions per 100 problems. The number of productions fired per 100 problems became approximately 100, or one production per problem in the third stage.

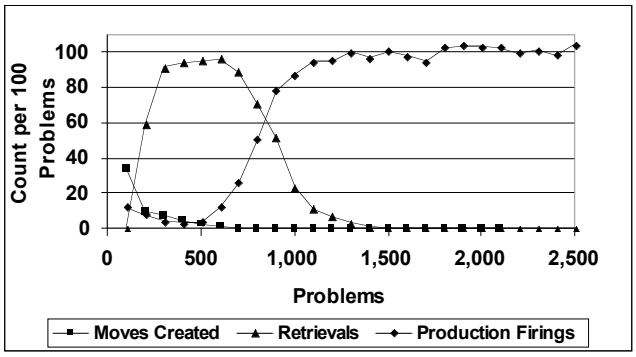

Figure 4. ACT-R Stages of learning

Figure 4 clearly shows three stages of learning. The first is the period when the model is solving problems by creating moves based only on what were legal moves. There was no knowledge in the model guiding the generation of moves other than their legality. It also shows the period where the model relied on the retrieval of previous solutions from memory to solve the current problem. Finally, it shows the

use of compiled knowledge in productions, now procedural, rather than the retrieval of declarative facts retrieved from memory.

These runs demonstrated that both systems transitioned from problem solving to using learned knowledge. Soar did so immediately and ACT-R took longer. Both systems reused learned knowledge. Soar used half its learned productions and ACT-R used approximately three-fourths.

Does ACT-R suffer from the utility problem?

In the same year Soar was proposed as a unified theory of cognition, it was confirmed to suffer performance degradation with continued learning known as the utility problem (Tambe 1990). One component of Newell's cognitive theory implemented in Soar is that productions are the only form of long-term memory and all productions are kept indefinitely. Relaxing this premise led to statistically significant improvements in Soar's performance in long-term learning experiments (Kennedy & De Jong, 2003).

To test whether ACT-R suffered from the same problem, timing data was collected during long runs of the system on Blocks World problems. Multiple runs were made on personal computers running the Windows operating system with the standalone version of ACT-R version 6 [R145] (Allegro Common Lisp) and on Apple G4 and G5 machines running Mac OS X and Macintosh Common Lisp (MCL) version 5. No changes were made to the Lisp garbage collection parameters. Time and memory consumption was available from the Lisp implementation (implementation specific) and from ACT-R. Two times were associated with solving a problem in ACT-R. The first was the human's response time calculated by ACT-R as a function of the Blocks-World model and ACT-R's sub-symbolic modeling. Figure 5 shows the average ACT-R run times for five runs averaged over 10 problems. The other time available is Lisp's internal clock measured in "ticks". This measurement is implementation dependent and captures the time associated with garbage collection. For this implementation, 1000 ticks equal 1 second. Figures 6 and 7 show the Lisp time and memory resources used for five runs averaged over 10 problems.

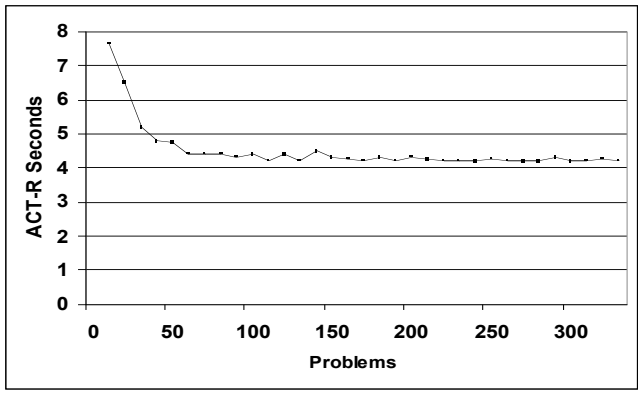

Figure 5. ACT-R Model Times for Blocks World Problems

Figure 5 shows a downward trend in the average time to solve individual problems. This trend is the expected output of the ACT-R model of improved performance with experience.

However, all of the systems running our model of the Blocks World eventually failed (froze). Failure ranged from as early as 340 problems on the Apple G4 (desktop) with 384MB of memory to over 32000 problems on a PC with 1GB of memory. Figures 5-7 are from the Apple G4 runs. Figure 6 shows the increasing Lisp ticks prior to failure. Figure 7 shows the corresponding memory allocation.

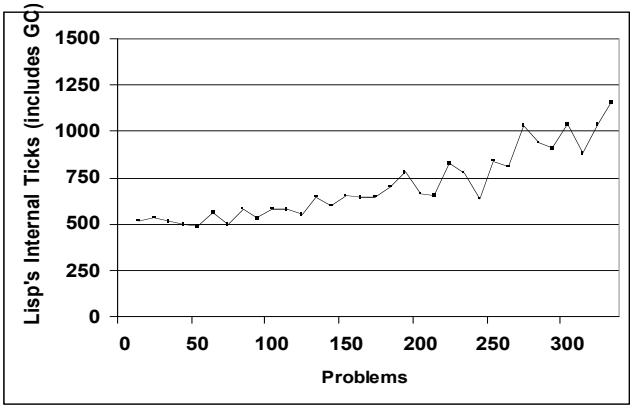

Figure 6. ACT-R Run Times in Lisp "ticks"

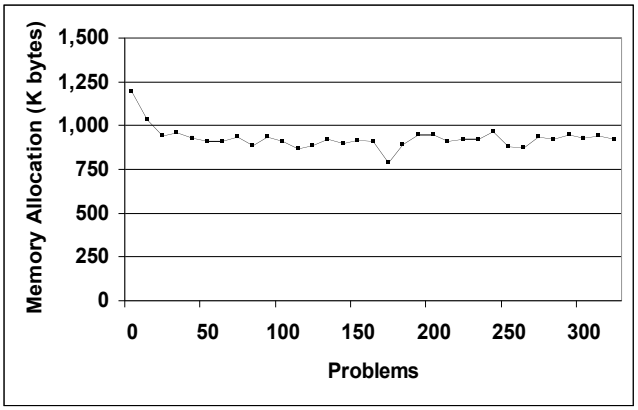

Figure 7. ACT-R Memory Allocation

While the ACT-R model's output shows a downward trend in response time consistent with human performance data, the calculations necessary to run the model frequently required garbage collection and all runs eventually failed.

The failure of ACT-R on long series of problems appears to be independent of hardware and operating system (we used both Macintoshes and PCs) and Lisp implementations (we used Allegro and Macintosh Lisps). Other long runs of ACT-R (Lebiere 1999) used a different version of ACT-R which did not include production compilation nor include the use of the vision module of ACT-R. Therefore, although ACT-R models human performance, the system suffers from the utility problem like other symbolic learning systems. It is unclear whether the utility problem exists at the implementation level (e.g., Lisp), the architectural level (e.g., ACT-R version 6), or the theory level.

Conclusions and Implications

We conclude that Soar and ACT-R have demonstrated similar long-term learning characteristics. The learning of symbolic productions on long series of problems from a finite problem domain eventually stops. We draw no conclusions on domains where an infinite number of problems are possible. On the use of learned productions, there may be some differences between Soar and ACT-R, but our long runs of Blocks World problems did not show them. We found that ACT-R demonstrates three phases of learning: cognitive problem solving, recall of previous solutions, and finally, proceduralization of knowledge. These phases relate to the three stages described by Anderson (2000). Finally, both Soar and ACT-R suffer from the utility problem in that performance degrades with continued learning and can result in system failure.

This work demonstrates that the two major, current cognitive models of learning, Soar and ACR-R, need revision to better address long-term learning. Human cognition does not "fail" with long-term learning. To support long-term, on-line learning that would be expected of an intelligent system, both architecture require changes, possibly as simple as including a permanent forgetting mechanism. The removal of low-use productions showed a statically significant improvement in Soar's performance (Kennedy & De Jong, 2003). Finally, for humans, the "power law of forgetting" (Anderson 2000), i.e., the decay of some memories, may support or even be necessary to allow continued learning and improved performance. Most studies of forgetting have focused on declarative knowledge. A study of the loss of procedural memories should be undertaken.

Acknowledgments

This work was performed while the first author held a National Research Council Research Associateship Award at the Naval Research Laboratory. The views and conclusions contained in this document should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U. S. Navy.

References

Ackerman, P. I. (1990). A correlation analysis of skill specificity: learning, abilities, and individual differences. *Journal of experimental psychology: Learning, memory, and cognition*. 15(5) 883-901.

Ackerman, P. I. (1998). Determinants of individual differences during skill acquisition: Cognitive abilities and information processing. *Journal of experimental psychology: General*. 117(3) 288-318.

ACT-R Research Group. (2005) Retrieved August 4, 2005, from http://act-r.psy.cmu.edu/

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

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

Anderson, J. R. (2000). *Learning and memory: an integrated approach*. 2nd ed. Hoboken, NJ: John Wiley & Sons.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review 111,* (4) 1036-1060.

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

Holder, L. B. (1990). The general utility problem in machine learning. *Proceedings of the Seventh International Conference on Machine Learning* (pp. 402-410). San Mateo: Morgan Kaufmann.

Jones, R., Laird, J. and Nielsen, P., (1994). Coordinated Behavior of Computer Generated Forces in TacAir-Soar. *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Orlando, FL.

Kennedy, W. G. & De Jong, K. A. (2003). Characteristics of long-term learning in Soar and its application to the utility problem. *Proceedings of the Twentieth International Conference on Machine Learning* (pp. 337-344). Menlo Park: AAAI Press.

Lebiere, C. (1999). The dynamics of cognition: An ACT-R model of cognitive arithmetic. *Kognitionswissenschaft.*, 8 (1), pp. 5-19.

Logan, G. D. (1988). Toward an instance theory of automatization. *Psychological Review 96*, 492-527.

Markovitch, S. & Scott, P. D. (1988). The role of forgetting in learning. *Proceedings of the fifth international conference on machine learning* (pp. 725-750). San Mateo: Morgan Kaufmann.

Minton, S. (1990). Quantitative results concerning the utility of explanation-based learning, *Artificial Intelligence, 42*, 363-392.

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

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

The Soar Group (2005) Retrieved July 7, 2005, from http://sitemaker.umich.edu/soar

Tambe, M., Newell, A., & Rosenbloom, P.S. (1990) The problem of expensive chunks and its solution by restricting expressiveness. *Machine Learning. V*, pp. 299-348.

Taatgen, N. A. (2002). A model of individual differences in skill acquisition in the Kanfer-Ackerman Air Traffic Control Task. *Cognitive Systems Research*, 3(1), 103-112.

Taatgen, N.A. & Lee, F.J. (2003). Production Compilation: A simple mechanism to model Complex Skill Acquisition. *Human Factors*, 45(1) 61-76.

Tambe, M., Newell, A, & Rosenbloom, P. S. (1990). The problem of expensive chunks and its solution by restricting expressiveness. *Machine Learning, V.* pp. 299-348.