Competence-Preserving Retention of Learned Knowledge in Soar's Working and Procedural Memories

Nate Derbinsky (nlderbin@umich.edu)
John E. Laird (laird@umich.edu)
University of Michigan, 2260 Hayward Street
Ann Arbor, MI 48109-2121 USA

Abstract

Effective management of learned knowledge is a challenge when modeling human-level behavior within complex, temporally extended tasks. This paper evaluates one approach to this problem: forgetting knowledge that is not in active use (as determined by base-level activation) and can likely be reconstructed if it becomes relevant. We apply this model for selective retention of learned knowledge to the working and procedural memories of Soar. When evaluated in simulated, robotic exploration and a competitive, multi-player game, these policies improve model reactivity and scaling while maintaining reasoning competence.

Keywords: large-scale cognitive modeling; working memory; procedural memory; cognitive architecture; Soar

Introduction

Typical cognitive models persist for short periods of time (seconds to a few minutes) and have modest learning requirements. For these models, current cognitive architectures, such as Soar (Laird, 2012) and ACT-R (Anderson et al., 2004), executing on commodity computer systems, are sufficient. However, prior work (Kennedy & Trafton, 2007) has shown that cognitive models of complex, protracted tasks can accumulate large amounts of knowledge, and that the computational performance of existing architectures degrades as a result.

This issue, where more knowledge can harm problem-solving performance, has been dubbed the *utility* problem, and has been studied in many contexts, such as explanation-based learning (Minton, 1990; Tambe et al., 1990), case-based reasoning (Smyth & Keane, 1995; Smyth & Cunningham, 1996), and language learning (Daelemans et al., 1999). Markovitch and Scott (1988) have characterized different strategies for dealing with the utility problem in terms of information filters applied at different stages in the problem-solving process. One common strategy that is relevant to cognitive modeling is selective retention, or *forgetting*, of learned knowledge. The benefit of this approach, as opposed to selective utilization, is that all available knowledge is brought to bear on problem solving, a property that is crucial for model competence in complex tasks. However, it can be challenging to devise forgetting policies that work well across a variety of problem domains, effectively balancing the task performance of cognitive models with reductions in retrieval time and storage requirements of learned knowledge.

In context of this challenge, we present two tasks where effective behavior requires that the model accumulate large amounts of information from the environment, and where over time this learned knowledge overwhelms reasonable computational limits. In response, we present and evaluate novel policies for selective retention of learned knowledge in the working and procedural memories of Soar. These policies investigate a common hypothesis: it is rational for the architecture to forget a unit of knowledge when there is a high degree of certainty that it is not of use, as calculated by base-level activation (Anderson et al., 2004), and that it can be reconstructed in the future if it becomes relevant. We demonstrate that these task-independent policies improve model reactivity and scaling, while maintaining problem-solving competence.

Related Work

Previous cognitive-modeling research has investigated forgetting in order to account for human behavior and experimental data. As a prominent example, memory decay has long been a core commitment of the ACT-R theory (Anderson et al., 2004), as it has been shown to account for a class of memory retrieval errors (Anderson et al., 1996). Similarly, research in Soar investigated task-performance effects of forgetting short-term (Chong, 2003) and procedural (Chong, 2004) knowledge. By contrast, the motivation for and outcome of this work is to investigate the degree to which selective retention can support long-term, real-time modeling in complex tasks.

Prior work shows the potential for cognitive benefits of memory decay, such as in task-switching (Altmann & Gray, 2002) and heuristic inference (Schooler & Hertwig, 2005). In this paper, we focus on improved reactivity and scaling.

We extend prior investigations of long-term symbolic learning in Soar (Kennedy & Trafton, 2007), where the source of learning was primarily from internal problem solving. In this paper, the evaluation domains accumulate information from interaction with an external environment.

The Soar Cognitive Architecture

Soar is a cognitive architecture that has been used for developing intelligent agents and modeling human cognition. Historically, one of Soar's main strengths has been its ability to efficiently represent and bring to bear large amounts of symbolic knowledge to solve diverse problems using a variety of methods (Laird, 2012).

Figure 1 shows the structure of Soar. At the center is a symbolic working memory that represents the agent's current state. It is here that perception, goals, retrievals from

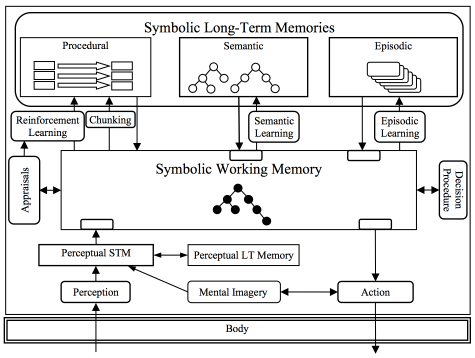

Figure 1: The Soar cognitive architecture.

long-term memory, external action directives, and structures from intermediate reasoning are jointly represented as a connected, directed graph. The primitive representational unit of knowledge in working memory is a symbolic triple (*identifier*, *attribute*, *value*), termed a *working-memory element*, or WME. The first symbol of a WME (identifier) must be an existing node in the graph, whereas the second (attribute) and third (value) symbols may be either terminal constants or non-terminal graph nodes. Multiple WMEs that share the same identifier are termed an "object," and the set of individual WMEs sharing that identifier are termed "augmentations" of that object.

Procedural memory stores the agent's knowledge of when and how to perform actions, both internal, such as querying long-term declarative memories, and external, such as controlling robotic actuators. Knowledge in this memory is represented as if-then rules. The conditions of rules test patterns in working memory and the actions of rules add and/or remove working-memory elements. Soar makes use of the Rete algorithm for efficient rule matching (Forgy, 1982) and retrieval time scales to large stores of procedural knowledge (Doorenbos, 1995). However, the Rete algorithm is known to scale linearly with the number of elements in working memory, a computational issue that motivates maintaining a relatively small working memory.

Soar learns procedural knowledge via chunking (Laird et al., 1986) and reinforcement learning (RL; Nason & Laird, 2005) mechanisms. Chunking *creates new* productions: it converts deliberate subgoal processing into reactive rules by compiling over production-firing traces, a form of explanation-based learning (EBL). If subgoal processing does not interact with the environment, the chunked rule is redundant with existing knowledge and serves to improve performance by reducing deliberate processing. However, memory usage in Soar scales linearly with the number of rules, typically at a rate of 1-5 KB/rule, which motivates forgetting of under-utilized productions.

Reinforcement learning incrementally *tunes existing* production actions: it updates the expectation of action utility, with respect to a subset of state (represented in rule conditions) and an environmental or intrinsic reward signal. A production that can be updated by the RL mechanism (termed in *RL rule*) must satisfy a few simple criteria related to its actions, and is thus distinguishable from other rules. This distinction is relevant to forgetting productions. When an RL rule that was learned via chunking is updated, that rule is no longer redundant with the knowledge that led to its creation, as it now incorporates information from environmental interaction that was not captured in the original subgoal processing.

Soar incorporates two long-term declarative memories, semantic and episodic (Derbinsky & Laird, 2010). Semantic memory stores working-memory objects, independent of overall working-memory connectivity (Derbinsky, Laird, & Smith, 2010), and episodic memory incrementally encodes and temporally indexes snapshots of working memory, resulting in an autobiographical history of agent experience (Derbinsky & Laird, 2009). Agents retrieve knowledge from one of these memory systems by constructing a symbolic cue in working memory; the intended memory system then interprets the cue, searches its store for the best matching memory, and if it finds a match, reconstructs the associated knowledge in working memory. For episodic memory, the time to reconstruct knowledge depends on the size of working memory at the time of encoding, another motivation for a concise agent state.

Agent reasoning in Soar consists of a sequence of decisions, where the aim of each decision is to select and apply an *operator* in service of the agent's goal(s). The primitive *decision cycle* consists of the following phases: encode perceptual input; fire rules to elaborate agent state, as well as propose and evaluate operators; select an operator; fire rules that apply the operator; and then process output directives and retrievals from long-term memory. Unlike ACT-R, multiple rules may fire in parallel during a single phase. The time to execute the decision cycle, which primarily depends on the speed with which the architecture can match rules and retrieve knowledge from episodic and semantic memories, determines agent reactivity. We have found that 50 msec. is an acceptable upper bound on this response time across numerous domains, including robotics, video games, and human-computer interaction (HCI) tasks.

There are two types of persistence for working-memory elements added as the result of rule firing. Rules that fire to apply a selected operator create *operator-supported* structures. These WMEs will persist in working memory until deliberately removed. In contrast, rules that do not test a selected operator create *instantiation-supported* structures, which persist only as long as the rules that created them match. This distinction is relevant to forgetting WMEs.

As evident in Figure 1, Soar has additional memories and processing modules; however, they are not pertinent to this paper and are not discussed further.

Selective Retention in Working Memory

The core intuition of our working-memory retention policy is to remove the augmentations of objects that are not actively in use and that the model can later reconstruct from long-term semantic memory, if they become relevant. We characterize WME usage via the base-level activation model (BLA; Anderson et al., 2004), which estimates future

usefulness of memory based upon prior usage. The primary activation event for a working-memory element is the firing of a rule that tests or creates that WME. Also, when a rule first adds an element to working memory, the activation of the new WME is initialized to reflect the aggregate activation of the set of WMEs responsible for its creation. The base-level activation of a WME is computed as:

$$A = \ln\left(\sum_{j=1}^{n} t_j^{-d}\right)$$

where n is the number of memory activations, t_j is the time since the jth activation, and d is a free decay parameter. For computational efficiency, history size is bounded: each working-memory element maintains a history of at most the c most recent activations and the activation calculation is supplemented by an approximation of the more distant past (Petrov, 2006). This model of activation sources, events, and decay is task independent.

At the end of each decision cycle, Soar removes from working memory each element that satisfies all of the following requirements, with respect to τ, a static, architectural threshold parameter:

R1. The WME was not encoded directly from perception.
R2. The WME is operator-supported.
R3. The activation level of the WME is less than τ.
R4. The WME augments an object, *o*, in semantic memory.
R5. The activation of all augmentations of *o* are less than τ.

We adopted requirements R1-R3 from Nuxoll, Laird, and James (2004), whereas R4 and R5 are novel. Requirement R1 distinguishes between the decay of representations of perception, and any dynamics that may occur with actual sensors, such as refresh rate, fatigue, noise, or damage. Requirement R2 is a conceptual optimization: as operator-supported WMEs are persistent, while instantiation-supported structures are direct entailments, if we properly manage the former, the latter are handled automatically. This means that if we properly remove operator-supported WMEs, any instantiation-supported structures that depend on them will also be removed, and thus our mechanism only manages operator-supported structures. The concept of a fixed lower bound on activation, as defined by R3, was adopted from activation limits in ACT-R (Anderson et al., 1996), and dictates that working-memory elements will decay in a task-independent fashion as their use for reasoning becomes less recent/frequent.

Requirement R4 dictates that our mechanism only removes elements from working memory that can be reconstructed from semantic memory. From the perspective of cognitive modeling, this constraint on decay resembles a working memory that is in part an activated subset of long-term memory (Jonides et al., 2008). Functionally, requirement R4 serves to balance the degree of working-memory decay with support for sound reasoning. Knowledge in Soar's semantic memory is persistent, though may change over time. Depending on the task and the model's knowledge-management strategies, it is possible that any removed knowledge may be recovered via deliberate reconstruction from semantic memory. Additionally, knowledge that is not in semantic memory can persist indefinitely to support model reasoning.

Requirement R5 supplements R4 by providing partial support for the *closed-world assumption*. R5 dictates that either all object augmentations are removed, or none. This policy leads to an object-oriented representation whereby procedural knowledge can distinguish between objects that have been cleared, and thus have no augmentations, and those that simply are not augmented with a particular feature or relation. R5 makes an explicit tradeoff, weighting more heavily model competence at the expense of the speed of working-memory decay. This requirement resembles the declarative module of ACT-R, where activation is associated with each chunk and not individual slot values.

Empirical Evaluation

We extended an existing system where Soar controls a simulated mobile robot (Laird, Derbinsky, & Voigt, 2011). Our evaluation uses a simulation instead of a real robot because of the practical difficulties in running numerous, long experiments in large physical spaces. However, the simulation is quite accurate and the Soar rules (and architecture) used in the simulation are exactly the same as the rules used to control the real robot.

The robot's task is to visit every room on the third floor of the Computer Science and Engineering building at the University of Michigan. For this task, the robot visits over 100 rooms and takes about 1 hour of real time. During exploration, it incrementally builds an internal topographic map, which, when completed, requires over 10,000 WMEs to represent and store. In addition to storing information, the model reasons about and plans using the map in order to find efficient paths for moving to distant rooms it has sensed but not visited. The model uses episodic memory to recall objects and other task-relevant features during exploration.

In our experiments, we aggregate working-memory size and maximum decision time for each 10 seconds of elapsed time, all of which is performed on an Intel i7 2.8GHz CPU, running Soar v9.3.1. Because each experimental run takes 1 hour, we did not duplicate our experiments sufficiently to establish statistical significance and the results we present are from individual experimental runs. However, we found qualitative consistency across our runs, such that the variance between runs is small as compared to the trends we focus on below.

We make use of the same model for all experiments, but modify small amounts of procedural knowledge and change architectural parameters, as described here. The baseline model (A0) maintains all declarative map information both in Soar's working and semantic memories. A slight modification to this baseline (A1) includes hand-coded rules to prune away rooms in working memory that are not required for immediate reasoning or planning. The experimental model (A2) makes use of our working-memory retention policy and we explored different values of the base-level decay rate (c=10 and τ=-2 for all models).

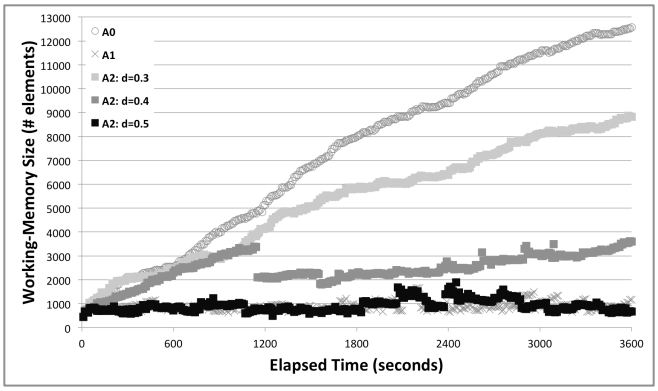

Figure 2: Model working-memory size comparison.

Figure 2 compares working-memory size between conditions A0, A1, and A2 over the duration of the experiment. We note first the major difference in working-memory size between A0 and A1 after one hour, when the working memory of A1 contains more than 11,000 fewer elements, more than 90% less than A0. We also find that the greater the decay-rate parameter for A2, the smaller the working-memory size, where a value of 0.5 qualitatively tracks A1. This finding suggests that our policy, with an appropriate decay, keeps working-memory size comparable to that maintained by hand-coded rules.

Figure 3 compares maximum decision-cycle time in msec., between conditions A0, A1, and A2 as the simulation progresses. The dominant cost reflected by this data is time to reconstruct prior episodes that are retrieved from episodic memory. We see a growing difference in time between A0 and A2 as working memory is more aggressively managed (i.e. greater decay rate), demonstrating that episodic reconstruction, which scales with the size of working memory at the time of episodic encoding, benefits from selective retention. We also find that with a decay rate of 0.5, our mechanism performs comparably to A1. We note that without sufficient working-memory management (A0; A2 with decay rate 0.3), episodic-memory retrievals are not tenable for a model that must reason with this amount of acquired information, as the maximum required processing time exceeds the reactivity threshold of 50 msec.

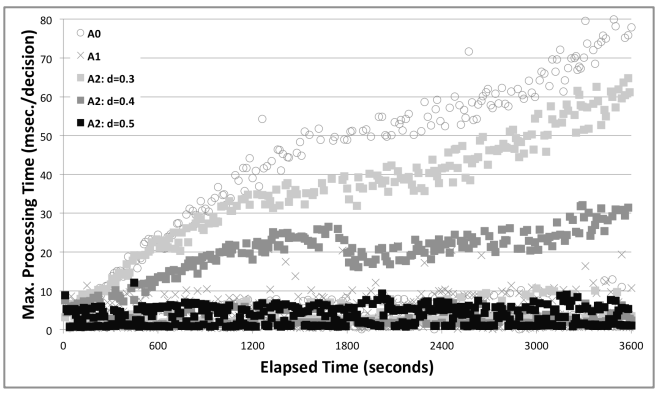

Figure 3: Model maximum decision time comparison.

Discussion

It is possible to write rules that prune Soar's working memory; however, this task-specific knowledge is difficult to encode and learn, and interrupts deliberate processing.

In this work, we presented and evaluated a novel approach that utilizes a memory hierarchy to bound working-memory size while maintaining sound reasoning. This approach assumes that the amount of knowledge required for immediate reasoning is small relative to the overall amount of knowledge accumulated by the model. Under this assumption, as demonstrated in the robotic evaluation task, our policy scales even as learned knowledge grows large over long trials. We note that since Soar's semantic memory can change over time and is independent of working memory, our selective-retention policy does admit a class of reasoning error wherein the contents of semantic memory are changed so as to be inconsistent with decayed WMEs. However, this corruption requires deliberate reasoning in a relatively small time window and has not arisen in our models. While the model completed this task for all conditions reported here, at larger decay rates ($\geq$0.6) the model thrashed because map information was not held in working memory long enough to complete deep look-ahead planning. This suggests additional research is needed on either adaptive decay-rate settings or planning approaches that are robust in the face of memory decay.

Selective Retention in Procedural Memory

The intuition of our procedural-memory retention policy is to remove productions that are not actively used and that the model can later reconstruct via deliberate subgoal reasoning, if they become relevant. We utilize the base-level activation model to summarize the history of rule firing.

At the end of each decision cycle, Soar removes from procedural memory each rule that satisfies all of the following requirements, with respect to parameter τ:

R1. The rule was learned via chunking.
R2. The rule is not actively firing.
R3. The activation level of the rule is less than τ.
R4. The rule has not been updated by RL.

We adopted R1-R3 from Chong (2004), whereas R4 is novel. Chong was modeling human skill decay, and did not delete productions, so as to not lose each rule's activation history. Instead, decayed rules were prevented from firing, similar to below-utility-threshold rules in ACT-R. R1 is a practical consideration to distinguish learned knowledge from "innate" rules developed by the modeler, which, if modified, would likely break the model. R2 recognizes that matched rules are in active use and thus should not be forgotten. R3 dictates that rules will decay in a task-independent fashion as their use for reasoning becomes less recent/frequent. We note that for fixed parameters (d and τ) and a single activation, the BLA model is equivalent to the use-gap heuristic of Kennedy and Trafton (2007). However, the time between sequential rule firings ignores firing frequency, which the BLA model incorporates.

Requirement R4 attempts to retain only those rules that the model cannot regenerate via chunking, a process that compiles existing knowledge applied in subgoal reasoning. Chunked rules that have been updated by RL encode expected utility information, which is not captured by other learning mechanisms. Because this information is difficult, if not impossible, to reconstruct, these rules are retained.

Empirical Evaluation

We extended an existing system (Laird et al., 2011) where Soar plays Liar's Dice, a multi-player game of chance. The rules of the game are numerous and complex, yielding a task that has rampant uncertainty and a large state space (millions-to-billions of relevant states for games of 2-4 players). Prior work has shown that RL allows Soar models to significantly improve performance after playing a few thousand games. However, this involves learning large numbers of RL rules to represent the state space.

The model we use for all experiments learns two classes of rules: RL rules, which capture expected action utility, and symbolic game heuristics. Our experimental baseline (B0) does not include selective retention. The first experimental modification (B1) implements our selective-retention policy, but does not enforce requirement R4 and is thereby comparable to prior work (Kennedy & Trafton, 2007; Chong, 2004). The second modification (B2) fully implements our policy. We experiment with a range of representative decay rates, including 0.999, where rules not immediately updated by RL are deleted (c=10, τ=-2 for all).

We alternated 1,000 2-player games of training then testing, each against a non-learning version of the model. After each testing session, we recorded maximum memory usage (Mac OS v10.7.3; dominated, in this task, by procedural memory), task performance (% games won), and average decisions/task action. We do not report maximum decision time, as this was below 6 msec. for all conditions (Intel i7 2.8GHz CPU, Soar v9.3.1). We collected data for all conditions in at least three independent trials of 40,000 games. For conditions that used selective retention, we were able to gather more data in parallel, due to reduced memory consumption (six trials for d=0.35, seven for remaining).

Figure 4 presents average memory growth, in megabytes, as the model trains, where the error bars represent ±1 standard deviation. For all models, the memory growth of games 1-10K follows a power law (r^2≥0.96), whereas for

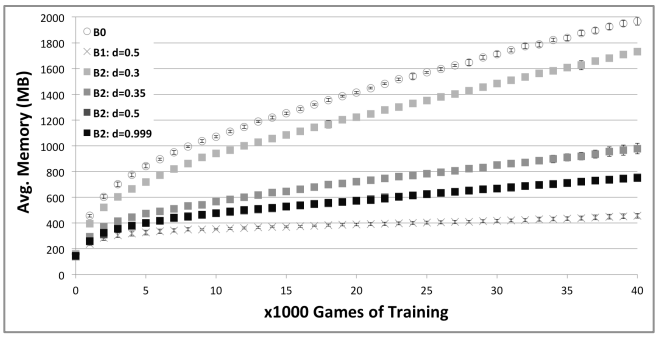

Figure 4. Avg. memory usage ±1 std. dev. vs. games played.

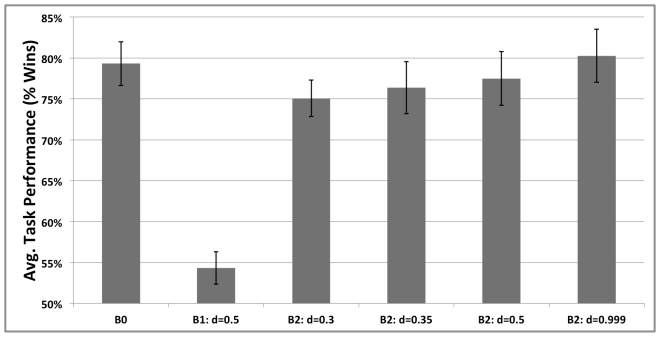

Figure 5. Avg. task performance ±1 std. dev.

11-40K, growth is linear (r^2≥0.99). These plots indicate that memory usage for the baseline (B0) and the slowly decaying model (B2, d=0.3) is much greater, and faster growing, than models that more aggressively decay. It also shows that there is a diminishing benefit from faster decay (e.g. d=0.5 and 0.999 for B2 are indistinguishable).

Figure 5 presents average task performance after 1,000 games of training, where the error bars represent ±1 standard deviation. This data shows that given the inherent stochasticity of the task, there is little, if any, difference between the performance of the baseline (B0) and decay levels of B2. However, by comparing B0 and B2 to B1, it is clear that without R4, the model suffers a dramatic loss of task competence. For clarity, the model begins by playing a non-learning copy of itself and learns from experience with each training session. While the B0 and B2 models improve from winning 50% of games to 75-80%, the B1 model improves to below 55%. We conclude that a selective-retention policy that only incorporates production-firing history (e.g. Chong, 2004; Kennedy & Trafton, 2007) will negatively impact performance in tasks that involve informative interaction with an external environment. Our policy incorporates both rule-firing history and rule reconstruction, and thus retains this source of feedback.

Finally, Figure 6 presents average number of decisions for the model to take an action in the game after training for 10,000 games. In prior work (e.g. Kennedy & Trafton, 2007), this value was a major performance metric, as it reflected the primary reason for learning new rules. In this work, each decision takes very little time, and so the number of decisions to choose an action is not as crucial to task performance as the selected action. However, these data show that there exists a space of decay values (e.g. d=0.35) in which memory usage is relatively low and grows slowly (Figure 4), task performance is relatively high (Figure 5), and the model makes decisions relatively quickly (Figure 6).

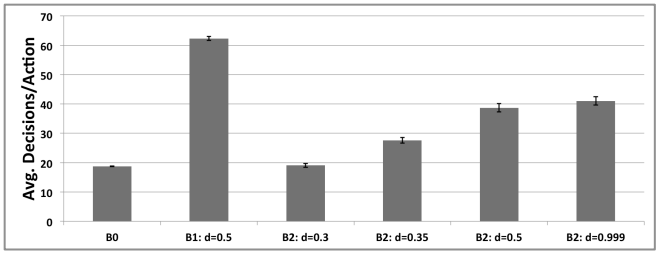

Figure 6. Avg. decisions/task action ±1 std. dev.

Discussion

This work contributes evidence that we can develop models that improve using RL in tasks with large state spaces. Currently, it is typical to explicitly represent the entire state space, which is not feasible in complex problems. Instead, Soar learns rules to represent only those portions of the space it experiences, and our policy retains only those rules that include feedback from environmental reward. Future work needs to validate this approach in other domains.

Concluding Remarks

This paper presents and evaluates two policies for effective retention of learned knowledge from complex environments. While forgetting mechanisms are common in cognitive modeling, this work pursues this line of research for functional reasons: improving computational-resource usage while maintaining reasoning competence. We have presented compelling results from applying these policies in two complex, temporally extended tasks, but there is additional work to evaluate these policies, and their parameters, across a wider variety of problem domains.

This paper does not address the computational challenges associated with efficiently implementing these policies. Derbinsky and Laird (2012) present and evaluate algorithms for implementing forgetting via base-level activation.

Acknowledgments

We acknowledge the funding support of the Air Force Office of Scientific Research, contract FA2386-10-1-4127.

References

Altmann, E., Gray, W. (2002). Forgetting to Remember: The Functional Relationship of Decay and Interference. *Psychological Science, 13 (1)*, 27-33.

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.

Anderson, J. R., Reder, L., Lebiere, C. (1996). Working Memory: Activation Limitations on Retrieval. *Cognitive Psychology, 30*, 221-256.

Chong, R. (2003). The Addition of an Activation and Decay Mechanism to the Soar Architecture. *Proc. of the 5th Intl. Conf. on Cognitive Modeling* (pp. 45-50).

Chong, R. (2004). Architectural Explorations for Modeling Procedural Skill Decay. *Proc. of the 6th Intl. Conf. on Cognitive Modeling*.

Daelemans, W., Van Den Bosch, A., Zavrel, J. (1999). Forgetting Exceptions is Harmful in Language Learning. *Machine Learning, 34*, (pp. 11-41).

Derbinsky, N., Laird, J. E. (2009). Efficiently Implementing Episodic Memory. *Proc. of the 8th Intl. Conf. on Case-Based Reasoning* (pp. 403-417).

Derbinsky, N., Laird, J. E. (2010). Extending Soar with Dissociated Symbolic Memories. *Symposium on Human Memory for Artificial Agents, AISB* (pp. 31-37).

Derbinsky, N., Laird, J. E. (2012). Computationally Efficient Forgetting via Base-Level Activation. *Proc. of the 11th Intl. Conf. on Cognitive Modeling*.

Derbinsky, N., Laird, J. E., Smith, B. (2010). Towards Efficiently Supporting Large Symbolic Declarative Memories. *Proc. of the 10th Intl. Conf. on Cognitive Modeling* (pp. 49-54).

Doorenbos, R. B. (1995). Production Matching for Large Learning Systems. Ph.D. Diss., Computer Science Dept. Carnegie Mellon, Pittsburgh, PA.

Forgy, C. L. (1982). Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence, 19 (1)*, 17-37.

Kennedy, W. G., Trafton, J. G. (2007). Long-term Symbolic Learning. *Cognitive Systems Research, 8*, 237-247.

Laird, J. E. (2012). *The Soar Cognitive Architecture*, MIT Press.

Laird, J. E., Derbinsky, N., Tinkerhess, M. (2011). A Case Study in Integrating Probabilistic Decision Making and Learning in a Symbolic Cognitive Architecture: Soar Plays Dice. *Papers from the 2011 Fall Symposium Series: Advances in Cognitive Systems* (pp. 162-169).

Laird, J. E., Derbinsky, N., Voigt, J. R. (2011). Performance Evaluation of Declarative Memory Systems in Soar. *Proc. of the 20th Behavior Representation in Modeling & Simulation Conf.* (pp. 33-40).

Laird, J. E., Rosenbloom, P. S., Newell, A. (1986). Chunking in Soar: The Anatomy of a General Learning Mechanism. *Machine Learning, 1 (1)*, 11-46.

Markovitch, S., Scott, P. D. (1988). The Role of Forgetting in Learning. *Proc. of the 5th Intl. Conf. on Machine Learning* (pp. 459-465).

Minton, S. (1990). Qualitative Results Concerning the Utility of Explanation-Based Learning. *Artificial Intelligence, 42*, 363-391.

Nason, S., Laird, J. E. (2005). Soar-RL: Integrating Reinforcement Learning with Soar. *Cognitive Systems Research, 6 (1)*, 51-59.

Nuxoll, A., Laird, J. E., James, M. (2004). Comprehensive Working Memory Activation in Soar. *Proc. of the 6th Intl. Conf. on Cognitive Modeling* (pp. 226-230).

Petrov, A. (2006). Computationally Efficient Approximation of the Base-Level Learning Equation in ACT-R. *Proc. of the 7th Intl. Conf. on Cognitive Modeling* (pp. 391-392).

Schooler, L., Hertwig, R. (2005). How Forgetting Aids Inference. *Psychological Review, 112 (3)*, 610-628.

Smyth, B., Cunningham, P. (1996). The Utility Problem Analysed: A Case-Based Reasoning Perspective. *LNCS, 1168*, 392-399.

Smyth, B., Keane, M. T. (1995). Remembering to Forget: A Competence-Preserving Case Deletion Policy for Case-Based Reasoning Systems. *Proc. of the 13th Intl. Joint Conf. on Artificial Intelligence* (pp. 377-382).

Tambe, N., Newell, A., Rosenbloom, P. S. (1990). The Problem of Expensive Chunks and its Solution by Restricting Expressiveness. *Machine Learning, 5*, 299-349.