Extending the Soar Cognitive Architecture

Conference Paper *in* Frontiers in Artificial Intelligence and Applications · January 2008

Source: DBLP

1 author:

John E. Laird
University of Michigan
234 PUBLICATIONS **8,704** CITATIONS

SEE PROFILE

Extending the Soar Cognitive Architecture

John E. Laird[1]
Division of Computer Science and Engineering, University of Michigan

Abstract. One approach in pursuit of general intelligent agents has been to concentrate on the underlying cognitive architecture, of which Soar is a prime example. In the past, Soar has relied on a minimal number of architectural modules together with purely symbolic representations of knowledge. This paper presents the cognitive architecture approach to general intelligence and the traditional, symbolic Soar architecture. This is followed by major additions to Soar: non-symbolic representations, new learning mechanisms, and long-term memories.

Keywords. Soar, cognitive architecture, reinforcement learning, episodic memory, semantic memory, clustering, mental imagery, emotion.

Introduction

Through the years, there has been substantial evolution and refinement of the Soar architecture (Laird and Rosenbloom, 1996), with eight major versions between 1982 and 2007. During this evolution, the basic approach of pure symbolic processing, with all long-term knowledge being represented as production rules, was maintained and Soar proved to be a general and flexible architecture for research in cognitive modeling across a wide variety of behavioral and learning phenomena (Rosenbloom et al. 1993). Soar also proved to be useful for creating knowledge-rich agents that could generate diverse, intelligent behavior in complex, dynamic environments (Jones et al. 1999; Wray, et al. 2005).

In spite of these successes, it became clear that Soar was missing some important capabilities that we take for granted in humans, many of which have also been ignored by the larger cognitive architecture community. In response, we have made substantial extensions to Soar, adding new learning mechanisms and long-term memories as well as new forms of non-symbolic processing. The motivation for these extensions is to increase the *functionality* of Soar for creating artificial general intelligent systems, but we also expect that these changes this will significantly expand the breadth of human behavior that can be modeled using Soar. Before presenting these extensions, we start with our underlying methodology for understanding and creating artificial generally intelligent systems: cognitive architecture. We then present the traditional version Soar, without the extensions, followed by the extended version of Soar. We conclude with an analysis of this new version of Soar and discussion of future directions.

[1] Corresponding Author: John E. Laird, Division of Computer Science and Engineering, University of Michigan, 2260 Hayward Ave. Ann Arbor, MI 48109-2121; E-mail: laird@umich.edu

1. Cognitive Architecture

During the last twenty years, the field of AI has successfully pursued specialized algorithms for specific problems. What distinguishes generally intelligent entities is their ability to solve not just a single problem using a specific method, but the ability to pursue a wide variety of tasks, including novel tasks, using large bodies of diverse knowledge, acquired through experience, in complex, dynamic environments. This leads us to study the fixed infrastructure that supports the acquisition and use of knowledge: the cognitive architecture. A cognitive architecture consists of:

- memories for storing knowledge
- processing units that extract, select, combine, and store knowledge
- languages for representing the knowledge that is stored and processed

Cognitive architectures distinguish between knowledge that is acquired over time and the fixed cognitive architecture that is common across all tasks. One of the most difficult challenges in cognitive architecture design is to create sufficient structure to support initial coherent and purposeful behavior, while at the same time providing sufficient flexibility so that an agent can adapt (via learning) to the specifics of its tasks and environment. Cognitive architectures must embody strong hypotheses about the building blocks of cognition that are shared by all tasks, and how different types of knowledge are learned, encoded, and used, making a cognitive architecture a software implementation of a general theory of intelligence.

The hypothesis behind cognitive architectures such as Soar and ACT-R (Anderson, 2007) is that there are useful abstractions and regularities above the level of neurally-based theories. This hypothesis plays out both in longer time scales of modeled behavior and in the symbolic representations of knowledge about the world. A related hypothesis is that the structures and discoveries of the symbolic architectures will be reflected in the neurally-based architectures. This is the approach taken in Neuro-Soar (Cho et al., 1991) and the ACT-R/Liebre hybrid (Taatgen et al., 2007) architectures. Probably the most interesting question is whether the extra detail of the neurally-based architectures (and the concurrent additional processing requirements) is necessary to achieve generally intelligent agents, or whether the more abstract architectures sufficiently capture the structures and regularities required for intelligence.

2. Traditional Soar

As shown in Figure 1, up until five years ago, Soar has consisted of a single long-term memory, which is encoded as production rules, and a single short-term memory, which is encoded as a symbolic graph structure so that objects can be represented with properties and relations. Symbolic short-term memory holds the agent's assessment of the current situation derived from perception and via retrieval of knowledge from its long-term memory. Action in an environment occurs through creation of motor commands in a buffer in short-term memory. The decision procedure selects *operators* and detects *impasses,* both of which are described below.

At the lowest level, Soar's processing consists of matching and firing rules. Rules provide a flexible, context-dependent representation of knowledge, with their conditions matching the current situation and their actions retrieving information relevant to the current situation. Most rule-based systems choose a single rule to fire at

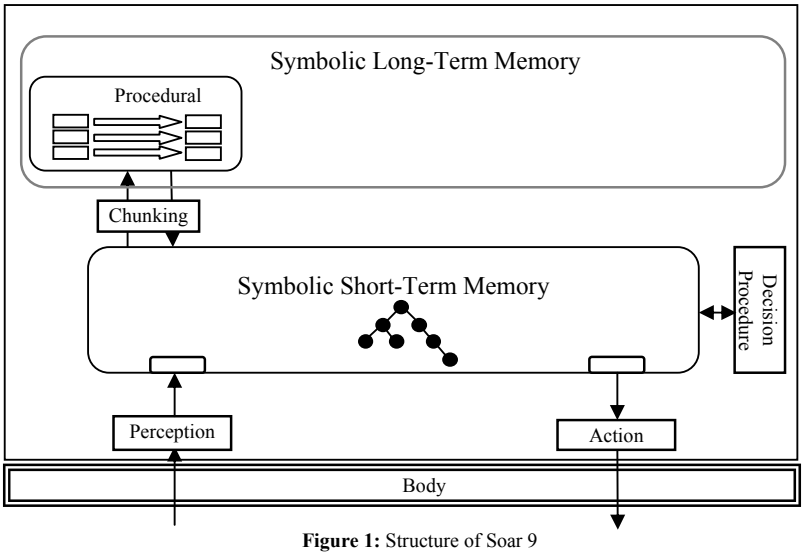

Figure 1: Structure of Soar 9

a given time, and this serves as the locus of choice in the system – where one action is selected instead of another. However, there is only limited knowledge available to choose between rules, namely the conditions of the rules, the data matched by the rules, and possibly meta-data, such as a numeric score, associated with the rules. There is no ability to use additional context-dependent knowledge to influence the decision. Soar allows additional knowledge to influence a decision by introducing *operators* as the locus for choice and using rules to propose, evaluate, and apply operators. Rules act as an associative-memory that retrieves information relevant to the current situation, so there is no need to select between them and thus, in Soar, rules fire in parallel.

The concept of operator is common in AI, but usually involves a monolithic data structure containing the operator's preconditions and actions. However, in Soar, the definition of an operator is distributed across multiple rules. Thus, in Soar, there are rules that *propose* operators that create a data structure in working memory representing the operator and an *acceptable preference* so that the operator can be considered for selection. There are also rules that *evaluate* operators and create other types of preferences that prefer one operator to another or provide some indication of the utility of the operator for the current situation. Finally, there are rules that *apply* the operator by making changes to working memory that reflect the actions of the operator. These changes may be purely internal or may initiate external actions in the environment. This approach supports a flexible representation of knowledge about operators – there can be many reasons for proposing, selecting, and/or applying an operator – some that are very specific and others that are quite general. This representation also makes it possible to incrementally build up operator knowledge structures, so that the definition of an operator can change over time as new knowledge is learned for proposal, selection, and application (Pearson & Laird, 2005).

To support the selection and application of operators and to interface to external environments, Soar has the processing cycle shown in Figure 2.

1. Input. Changes to perception are processed and sent to short-term memory.
2. Elaboration. Rules compute entailments of short-term memory. For example, a rule might test if the goal is to grasp an object, the object's distance, and the agent's reach, and then create a structure signifying whether the object is within reach.

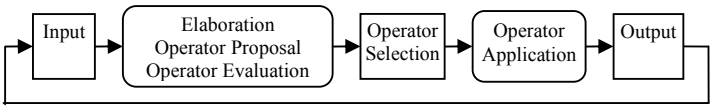

Figure 2: Soar's Processing Cycle

3. Operator Proposal. Rules propose operators that are appropriate to the current situation based on features of the situation tested in the condition of the rules.
4. Operator Evaluation. Rules create preferences for which of the proposed operators should be preferred, based on the current situation and goal. The preferences can be symbolic (A is better than B), or numeric (the estimated utility of A is .73).
5. Operator Selection. A fixed decision procedure combines the generated preferences and selects the current operator. If the preferences are insufficient for making a decision, an *impasse* arises and Soar automatically creates a substate in which the goal is to resolve that impasse. In the substate, Soar recursively uses the same processing cycle to select and apply operators, leading to automatic, reactive meta-reasoning. The impasses and resulting substates provide a mechanism for Soar to deliberately perform any of the functions (elaboration, proposal, evaluation, application) that are performed automatically/reactively with rules.
6. Operator Application. The actions of an operator are performed by rules that match the current situation and the current operator structure. Multiple rules can fire in parallel and in sequence providing a flexible and expressive means for encoding operator actions. If there is insufficient application knowledge, an impasse arises with a substate. This leads to dynamic task decomposition, where a complex operator is performed recursively by simpler operators.
7. Output. Any output commands are passed on to the motor system.

Soar's procedural knowledge, decision-making, and subgoaling provide a strong foundation on to which to build. It has a shared short-term memory where knowledge from perception and long-term memory are combined to provide a unified representation of the current situation. It has a lean decision procedure that supports context-dependent reactive behavior, but also supports automatic impasse-driven subgoals and meta-reasoning. Chunking is Soar's learning mechanism that converts the results of problem solving in subgoals into rules – compiling knowledge and behavior from deliberate to reactive. Although chunking is a simple mechanism, it is extremely general and can learn all the types knowledge encoded in rules (Steier et al., 1987).

3. Extended Soar

In extending Soar, we had two goals: 1. Retain the strengths of the original Soar: a flexible model of control and meta-reasoning along with the inherent ability to support reactive and deliberative behavior and the automatic conversion from deliberate to reactive behavior via chunking. 2. Expand the types of knowledge Soar could represent, reason with, and learn, inspired by human capabilities, but with the primary goal of additional functionality. The extensions fall into two, partially overlapping categories: new non-symbolic representations of knowledge along with associated processing and memory modules, and new learning and memory modules that capture knowledge that is cumbersome to learn and encode in rules.

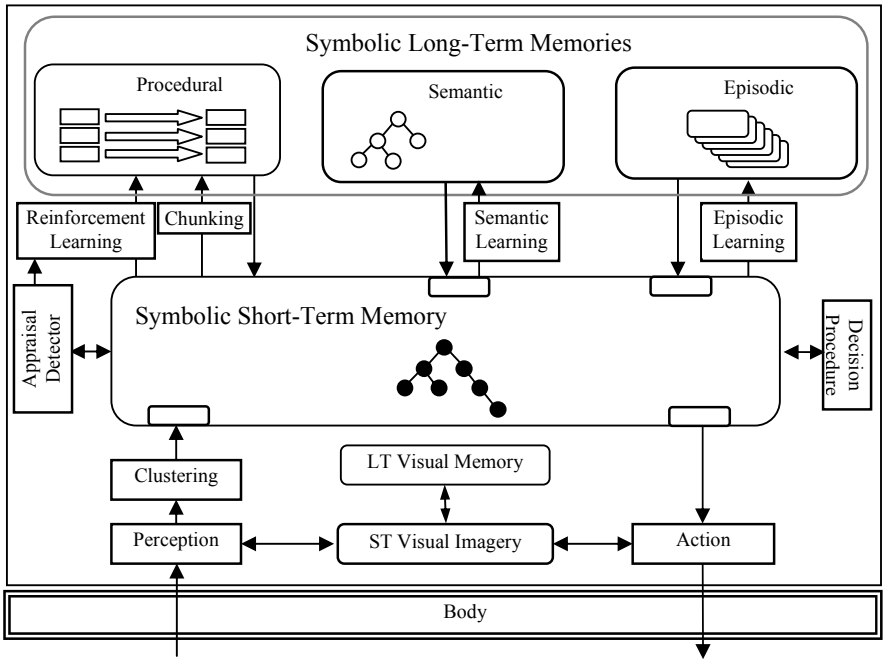

Figure 3: Soar 9

Figure 3 shows the structure of Soar, version 9. All of the new components have been built, integrated, and run with the traditional Soar components; however, as of yet there is not a single unified system that has all the components running at once. The major additions include: working memory activation, which provides meta-information about the recency and usefulness of working memory elements; reinforcement learning; which tunes the numeric preferences of operator selection rules; the appraisal detector, which generates emotions, feelings, and an internal reward signal for reinforcement learning; semantic memory, which contains symbolic structures representing facts; episodic memory; which contains temporally ordered "snapshots" of working memory; a set of processes and memories to support visual imagery, which includes depictive representations in which spatial information is inherent to the representation; and clustering, which dynamically creates new concepts and symbols.

Soar's processing cycle is still driven by procedural knowledge encoded as production rules. The new components influence decision making indirectly by retrieving or creating structures in symbolic working memory that cause rules to match and fire. In the remainder of this section, we will give descriptions of these new components and discuss briefly their value and why their functionality would be very difficult to achieve by the existing mechanisms.

3.1. Working Memory Activation

Inspired by ACT-R (Anderson 2007), we added activation to Soar's working memory (Chong, 2003; Nuxoll et al., 2004). Activation provides meta-information in terms of the recency of a working memory element and its relevance, which is computed based

on when the element matched rules that fired. This information is not used to determine which rules to fire, as Soar fires all rules that match, but it is stored as part of episodic memories, biasing their retrieval so that the episode retrieved is the most relevant to the current situation. Empirical results verify that working memory activation significantly improves episodic memory retrieval (Nuxoll & Laird, 2007). In the future, we expect that working memory activation will be used in semantic memory retrieval and emotion. Working memory activation requires architectural changes because it must access and maintain information that is only available to the architecture (when working memory elements are created and matched).

3.2. Reinforcement Learning

Reinforcement learning (RL) involves adjusting the selection of actions in an attempt to maximize reward. In early versions of Soar, all preferences for selecting operators were symbolic, so there was no way to represent or adjust such knowledge; however, we recently added numeric preferences, which specify the expected value of an operator for the current state (Wray & Laird, 2003). During operator selection, all numeric preferences for an operator are combined, and an epsilon-greedy algorithm is used to select the next operator. This makes RL in Soar straightforward – it adjusts the actions of rules that create numeric preferences for selected operators (Nason & Laird, 2005). Thus, after an operator applies, all of the rules that created numeric preferences for that operator are updated based on any new reward and the expected future reward, which is simply the summed numeric value of the numeric preferences for the next selected operator. RL in Soar applies across all goals, including impasse-generated subgoals. One intriguing aspect of RL in Soar is that the mapping from situation and operator to expected reward (the value-function) is represented as collections of rules. Only those rules that match the current situation participate in the selection of an operator, and there can be many rules contributing estimates of future reward for a single operator. This representation supports varying degrees of coverage and hierarchical representations, which can greatly speed up learning (Wang & Laird, 2007).

RL would be very difficult to implement in Soar using only chunking. RL applies to every operator selection, on every decision, even when there is no impasse, while chunking only learns rules through impasses. In addition, RL modifies existing rules by changing the values of numeric preferences, while chunking only adds new rules. In fact, RL and chunking are quite complementary because when there are no selection rules, an impasse arises and problem solving in a subgoal can generate initial preferences for the tied operators. Chunking then creates rules to generate these initial preferences in the future, and RL then tunes the values as experience accumulates.

3.3. Emotion

The functional and computational role of emotion is open to debate; however, in the last twenty years there has been substantial research on appraisal theories of emotion (Roseman & Smith, 2001). These theories propose that an agent continually evaluates a situation and that evaluation leads to emotion. The evaluation is hypothesized to take place along multiple dimensions, such as goal relevance (is this situation important to my goals?), goal conduciveness (is this situation good or bad for my goals?), causality (who caused the situation?), control (can I change the situation?), and so on. These

dimensions are exactly what an intelligent agent needs to compute as it pursues its goals while interacting with an environment. Thus, we have created a computational implementation of a specific appraisal theory (Scherer, 2001) in Soar, represented by the appraisal detector in Figure 3. In Soar, appraisals lead to emotions, emotions influence mood, and mood and emotion determine feelings (Marinier & Laird, 2007). Individual appraisals produce either categorical or numeric values, which combine to form an intensity of the current feeling. This intensity becomes the intrinsic reward (Singh et al., 2004) for reinforcement learning, which significantly speeds learning (Marinier & Laird, 2008). A major goal of our future work is to explore how emotion, mood, and feeling can be used productively with other modules (such as retrieval from long-term memory and decision making), as well as in interacting with other agents.

It is possible to implement similar theories of emotion in Soar without modifying the architecture (Gratch & Marsella, 2004); however, in these approaches, Soar is used more as a programming language than as a cognitive architecture – all of the important functionality of the system comes from procedural knowledge encoded in Soar as opposed to the structure of the architecture. This makes it impossible for emotions, mood, and feelings to directly influence other architectural modules, which we expect is critical to many aspects of emotion, mood, and feelings.

3.4. Semantic Memory

In addition to procedural knowledge, which is encoded as rules in Soar, there is declarative knowledge, which can be split into things that are known, such as facts, and things that are remembered, such as episodic experiences. Semantic learning and memory provides the ability to store and retrieve declarative facts about the world, such as tables have legs, dogs are animals, and Ann Arbor is in Michigan. This capability has been central to ACT-R's ability to model a wide variety of human data and adding it to Soar should enhance our ability to create agents that reason and use general knowledge about the world. In Soar, semantic memory is built up from structures that occur in working memory. A structure from semantic memory is retrieved by creating a cue in a special buffer in working memory. The cue is then used to search for the best partial match in semantic memory, which is then retrieved into working memory.

There has been a significant body of research on acquiring semantic knowledge through Soar's chunking mechanism, under the label of *data chunking* (Rosenbloom, 2007). Although it is possible, it is not easy. Data chunking requires pre-existing knowledge about the possible structures that can exist in working memory, and the agent has to interrupt its current task processing to deliberately force an impasse in which chunking can learn the appropriate rules. Moreover, because the knowledge is encoded in rules, retrieval requires an exact match of the cue, limiting the generality of what is learned. These factors made it difficult to use data chunking in new domains, begging the question as to how it would naturally arise in a generally intelligent agent.

3.5. Episodic Memory

In contrast to semantic memory, which contains knowledge independent of when and where it was learned, episodic memory contains memories of what was experienced over time (Tulving, 1983). In Soar, episodic memory includes specific instances of the structures that occur in working memory at the same time, providing the ability to remember the context of past experiences as well as the temporal relationships between

experiences (Nuxoll & Laird, 2007). An episode is retrieved by the deliberate creation of a cue, which is a partial specification of working memory in a special buffer. Once a cue is created, the best partial match is found (biased by recency and working memory activation) and retrieved into a separate working memory buffer (to avoid confusion between a memory and the current situation). The next episode can also be retrieved, providing the ability to replay an experience as a sequence of retrieved episodes.

Although similar mechanisms have been studied in case-based reasoning, episodic memory is distinguished by the fact that it is task-independent and thus available for every problem, providing a memory of experience not available from other mechanisms. Episodic learning is so simple that it is often dismissed in AI as not worthy of study. Although simple, one has only to imagine what life is like for amnesiacs to appreciate its importance for general intelligence (Nolan, 2000). We have demonstrated that when episodic memory is embedded in Soar, it enables many advanced cognitive capabilities such as internal simulation and prediction, learning action models, and retrospective reasoning and learning.

Episodic memory would be even more difficult to implement using chunking than semantic memory because it requires capturing a snapshot of working memory and using working memory activation to bias partial matching for retrieval.

3.6. Visual Imagery

All of the previous extensions depend on Soar's existing symbolic short-term memory to represent the agent's understanding of the current situation and with good reason. The generality and power of symbolic representations and processing are unmatched and our ability to compose symbolic structures is a hallmark of human-level intelligence. However, for some constrained forms of processing, other representations can be much more efficient. One compelling example is visual imagery (Kosslyn et al., 2006), which is useful for visual-feature and visual-spatial reasoning. We have added a set of modules to Soar that support visual imagery (Lathrop & Laird, 2007), including a short-term memory where images are constructed and manipulated; a long-term memory that contains images that can be retrieved into the short-term memory; processes that manipulate images in short-term memory, and processes that create symbolic structures from the visual images. Although not shown, these extensions support both a depictive representation in which space is inherent to the representation, as well as an intermediate, quantitative representation that combines symbolic and numeric representations. Visual imagery is controlled by the symbolic system, which issues commands to construct, manipulate, and examine visual images.

With the addition of visual imagery, we have demonstrated that it is possible to solve spatial reasoning problems orders of magnitude faster than without it, and using significantly less procedural knowledge. Visual imagery also enables processing that is not possible with only symbolic reasoning, such as determining which letters in the alphabet are symmetric along the vertical axis (A, H, I, M, O, T, U, V, W, X, Y).

3.7. Clustering

One additional capability that has been missing from Soar is a mechanism that detects statistical regularities in the stream of experiences and automatically creates new symbolic structures that represent those regularities, providing a mechanism for automatically generating new symbols and thus concepts that can be used to classify

perception. Few existing cognitive architectures have the capability to create new symbolic structures – they are a prisoner of the original encodings provided by a human programmer. To remedy this shortcoming, we have added a clustering module to Soar that is based on research by Richard Granger (Granger 2006). The underlying algorithms are derived from the processing of thalamocortical loops in the brain, where it appears there is clustering and successive sub-clustering of inputs using winner-take-all circuits. Although we do not yet have a general implementation of clustering for all types of perception in Soar, we have used clustering to create new symbolic structures that enrich that state representation and speed reinforcement learning.

3.8. Integration of New Modules in Soar

As mentioned earlier, the processing cycle of Soar did not require any changes to accommodate the new modules beyond invoking them at appropriate times. Working memory activation is updated when rules fire, while reinforcement learning is invoked when a new operator is selected and clustering is invoked during input processing. The retrieval for semantic and episodic memories is performed during the input phase, so that the results are available during the next processing cycle. The visual imagery system is treated as if it was an external module, with commands to the system creating by rules and then executed during the output phase, with new interpretations of the images created during the input phase. Although the modules are invoked at specific points in the processing cycle, they could run asynchronously. For example, when a cue is created for episodic memory, the retrieval processes could run in parallel with rule matching and firing. We have maintained the more rigid approach because it simplifies development and debugging, and because there was no computational advantage to having asynchronous threads before the advent of multi-core processors.

Although knowledge encoded in rules still controls behavior through the selection and application of operators, the new modules influence behavior through the working memory structures they create. For example, if a list of instructions is stored in semantic memory, with the appropriate rules, they can be retrieved, and lead to the selection of operators that interpret and execute them. Reactivity is maintained by rules that propose and prefer operators to process unexpected external events.

4. Discussion

4.1. Non-symbolic processing in Soar

One of the major changes in extending Soar has been the introduction of non-symbolic processing. In Soar, clustering is *subsymbolic*, where non-symbolic perceptual structures are combined together to create symbols. All the other non-symbolic processing is co-symbolic – it either controls symbolic processing (similar to ACT-R's use of non-symbolic processing), or in the case of visual imagery, provides an alternative to symbolic processing by providing a representational media for reasoning about the world. In contrast, Clarion has extensive subsymbolic processing (Sun 2006) where it supports neural networks processing for some of its reasoning in addition to symbolic processing. Below is a chart of the different functional uses of non-symbolic processing in Soar.

Non-symbolic Processing	Function
Numeric Preferences	Control operator selection
Reinforcement Learning	Learn operator selection control knowledge
Working memory activation	Aid long-term memory retrieval
Visual Imagery	Represent images and spatial data for reasoning
Appraisals: Emotions & Feelings	Summarize intrinsic value of situation – aid RL
Clustering	Create symbols that capture statistical regularities

4.2. Analysis of Memories and Learning in a Cognitive Architecture

The second major change is the addition of new memories and learning systems. Below we summarize the major dimensions of variations in the memory and learning systems:

Memory/Learning System	Source of Knowledge	Representation of knowledge	Retrieval of knowledge
Chunking	Traces of rule firings in subgoals	Rules	Exact match of rule conditions, retrieve actions
Clustering	Perception	Classification networks	Winner take all
Semantic Memory	Working memory existence	Mirror of working memory object structures	Partial match, retrieve object
Episodic Memory	Working memory co-occurrence	Episodes: Snapshots of working memory	Partial match, retrieve episode
Reinforcement Learning	Reward and numeric preferences	Numeric preferences	Exact match of rule conditions, retrieve preference
Image Memory	Image short-term memory	Image	Deliberate recall based on symbolic referent

4.3. Future work

The addition of these new modules only scratches the surface of the potential research, with explorations of the interactions and synergies of these components being the next order of business. Some of the interactions are obvious, such as using RL to learn the best cues for retrieving knowledge from episodic memory, or using episodic memory for retrospective learning to train RL on important experiences. Other interactions require deeper integrations, such as how emotion and visual imagery are captured and used in episodic memory, how knowledge might move between episodic and semantic memories, how emotions and feelings influence decision making, or how emotion and working memory activation might influence storage and retrieval of knowledge in episodic and semantic memory. Finally, we need to study the impact of these extensions on Soar's status as a unified theory of cognition (Newell 1990).

4.4. Final thoughts

So far, the explorations in psychology and AI through the space of cognitive architectures have been extremely fruitful. Research on ACT-R has led to comprehensive computational theories of a wide variety of human phenomena, including brain activity (Anderson, 2007), while research on EPIC has made great strides in modeling the detailed interactions of cognition with perception and action (Kieras & Meyer, 1997). However, these architectures and others have ignored many of the cognitive capabilities we are now studying in Soar (episodic memory, emotion,

visual imagery), and it has taken us close to twenty years to study them. Why is this? The simple answer is that our architectures are prisoners of the tasks we study. To date, most of the research on cognitive architecture has arisen from studying short-term tasks – similar to those that arise in a psychology laboratory setting where there is minimal emotional engagement and little need for personal history. The challenge for the future is to develop general artificial systems that can pursue a wide variety of tasks (including novel ones) over long time scales, with extensive experiential learning in dynamic, complex environments. As is evidenced from this paper, my belief is that research in cognitive architecture provides the building blocks to meet this challenge.

Acknowledgments

The author wishes to thank the current and former graduate students and research programmers who have contributed to the extensions to Soar described in this paper: Ron Chong, Karen Coulter, Michael James, Scott Lathrop, Robert Marinier, Shelley Nason, Andrew Nuxoll, Jonathan Voigt, Yongjia Wang, and Samuel Wintermute.

References

Anderson, J. R. (2007) *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press.

Cho, B., Rosenbloom, P. S. & Dolan, C. P. (1991) Neuro-Soar: A neural-network architecture for goal-oriented behavior, *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, 673-677. Chicago, IL.

Chong, R. (2003) The addition of an activation and decay mechanism to the Soar architecture. *Proceedings of the Fifth International Conference on Cognitive Modeling,* Pittsburgh, PA.

Granger R (2006) Engines of the brain: The computational instruction set of human cognition. *AI Magazine* 27: 15-32.

Gratch, J. & Marsella, S. (2004) A Domain-independent Framework for Modeling Emotion. Cognitive Systems Research, 5:269-306, 2004.

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V. (1999) Automated intelligent pilots for combat might simulation. *AI Magazine,* 20(1), 27-41.

Kieras, D. & Meyer, D. E. (1997) An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, 12, 391-438.

Kosslyn, S. M., Thompson, W. L., & Ganis, G. (2006). *The Case for Mental Imagery.* New York, New York: Oxford University Press.

Laird, J. E., & Rosenbloom, P.S. (1996) The evolution of the Soar cognitive architecture. In T. Mitchell (ed.) *Mind Matters,* 1-50.

Lathrop, S.D., and Laird, J.E. (2007). Towards Incorporating Visual Imagery into a Cognitive Architecture. *Eighth International Conference on Cognitive Modeling*.

Marinier, R. P., & Laird, J. E. (2007) Computational Modeling of Mood and Feeling from Emotion. *Proceedings of 29th Meeting of the Cognitive Science Society.* 461-466. Nashville: Cognitive Science Society.

Marinier, R. P., & Laird, J. E. (2008) Emotion-Driven Reinforcement Learning. (under review for *Cybernetics and Systems*, will know decision before final copy is due)

Nason, S., & Laird, J. E. (2005) Soar-RL: Integrating reinforcement learning with Soar. *Cognitive Systems Research*, 6(1), 51-59.
Newell, A. (1990) *Unified Theories of Cognition*. Harvard University Press.
Nolan, C. (2000) *Memento,* New Market Films.
Nuxoll, A. M., Laird, J. E., & James, M. (2004) Comprehensive working memory activation in Soar, *Sixth International Conference on Cognitive Modeling*, 226-230.
Nuxoll, A. M., & Laird, J. E. (2007) Extending cognitive architecture with episodic memory. *Proceedings of the 21st National Conference on Artificial Intelligence.*
Pearson, D. J., & Laird, J. E., Incremental Learning of Procedural Planning Knowledge in Challenging Environments, *Computational Intelligence*, 2005, 21(4), 414:439.
Roseman, I. & Smith, C. A. (2001) Appraisal theory: Overview, Assumptions, Varieties, Controversies. In Scherer, Schorr, & Johnstone (Eds.) *Appraisal processes in Emotion: Theory, Methods, Research*, 3-19. Oxford University Press.
Rosenbloom, P. S. (2006) A cognitive odyssey: From the power law of practice to a general learning mechanism and beyond. *Tutorials in Quantitative Methods for Psychology*, 2(2), 38-42.
Rosenbloom, P. S., Laird, J. E., & Newell, A. (1993) *The Soar papers: Research on Integrated Intelligence.* MIT Press, Cambridge, MA.
Scherer, K. R. (2001) Appraisal considered as a process of multi-level sequential checking. In K. R. Scherer, A. Schorr, & T. Johnstone (Eds.) *Appraisal processes in Emotion: Theory, Methods, Research.* 92-120. Oxford University Press.
Singh, S., Barto, A. G., & Chentanez, N. (2004) Intrinsically motivated reinforcement learning. *18th Annual Conference on Neural Information Processing Systems (NIPS).*
Steier, D.S., Laird, J.E., Newell, A., Rosenbloom, P.S., Flynn, R., Golding, A., Polk, T.A., Shivers, O., Unruh, A., Yost, G.R. (1987) Varieties of Learning in Soar. Proceedings of the Fourth International Machine Learning Workshop.
Sun, R. (2006). The CLARION cognitive architecture: Extending cognitive modeling to social simulation. In: Ron Sun (ed.), Cognition and Multi-Agent Interaction. Cambridge University Press, New York.
Taatgen, N. A., Juvina, I., Herd, S., Jilk, D., & Martens, S. (2007) Attentional blink: An internal traffic jam? Eighth International Conference on Cognitive Modeling.
Tulving, E. (1983) *Elements of Episodic Memory*. Oxford: Clarendon Press.
Wang, Y., and Laird, J.E. (2007) The Importance of Action History in Decision Making and Reinforcement Learning. *Proceedings of the Eighth International Conference on Cognitive Modeling*. Ann Arbor, MI.
Wang, Y., and Laird, J.E. 2006. Integrating Semantic Memory into a Cognitive Architecture. CCA-TR-2006-02, Center for Cognitive Architecture, University of Michigan.
Wray, R. E & Laird, J.E. (2003) Variability in Human Behavior Modeling for Military Simulations. *Proceedings of the 2003 Conference on Behavior Representation in Modeling and Simulation.* Scottsdale, AZ.
Wray, R. E., Laird, J. E., Nuxoll, A., Stokes, D., & Kerfoot, A. (2005) Synthetic adversaries for urban combat training. *AI Magazine*, 26(3), 82-92.