Extending Soar with Dissociated Symbolic Memories

Nate Derbinsky[1] **and John E. Laird**[1]

Abstract. Over long lifetimes, learning agents accumulate large stores of knowledge. To support human-level decision-making, their cognitive architectures must efficiently manage this experience and bring to bear pertinent data to act in the world. Prior psychological and computational work suggests the need for multiple, dissociated memory systems, citing significant functional and computational tradeoffs that arise when implementing a single memory mechanism for different types of learning tasks. In this context, we develop a memory-centric analysis of Soar 9, a general cognitive architecture that incorporates multiple long-term memories. In this analysis, we explore the functional abilities, computational opportunities, and theoretical challenges entailed by integrating a diverse set of symbolic memory systems.

1 INTRODUCTION

A long-lived learning agent facing numerous, complex tasks will experience large amounts of knowledge [4][9]. To support human-level intelligence, an agent's cognitive architecture must encode and store experiences in such a way so that it can retrieve relevant information to make decisions, while remaining reactive to a dynamic environment [12]. To meet these requirements, cognitive architectures employ one or more *memory systems*, mechanisms that efficiently implement a fixed policy to encode, store, and retrieve agent knowledge [8]. The term "fixed" is not intended to suggest non-adaptive, but instead refers to a set computational profile, wherein the architecture commits to restricted retrievals of agent experience in exchange for performance guarantees.

From an evolutionary perspective, Sherry and Shacter [23] have argued for the necessity of multiple, distinct memory systems in animals (including humans). They argue that a memory system will solve some environmental problems, and the justification for different memory systems is that some environmental problems require memory systems that are *functionally incompatible*. As they state, "an efficient habit-learning system will preserve those features of an experience that recur in different episodes and are thus crucial to the learning of a skill… it would appear the major function of the episodic-representational system is to preserve the contextual details that uniquely mark individual experiences" [23]. Computational approaches to memory system implementation offer empirical evidence to support this claim. For instance, O'Reilly states that "there are two incompatible goals that such systems need to solve… one goal is to remember specific information… the other is to extract generalities… the neural solutions to these goals are incompatible" [20].

[1] Computing Science and Engineering Div., Elec. Engineering and Computer Science Dept., Univ. of Michigan, 2260 Hayward St., Ann Arbor, MI, 48109-2121. Email: {nlderbin, laird}@umich.edu.

In this context, we develop a memory-centric, functional analysis of Soar 9. We contextualize the architecture with respect to memory organization posited in cognitive science literature, and offer *computational* arguments for its diverse set of symbolic memory systems. Thus, our discussion focuses on potential incompatibilities of the underlying implementations of different memories and the difficulty of ensuring efficient performance if only a single memory system is used. We discuss one important challenge that arises within the context of multiple memory systems because of the need to support persistence of object identity. We conclude with a discussion of future work for architectural research integrating multiple memory systems.

2 ARCHITECTURAL OVERVIEW

Soar is a cognitive architecture that has been used extensively for developing AI applications and cognitive models. One of Soar's main strengths has been its ability to efficiently represent and bring to bear large bodies of symbolic knowledge to solve diverse problems using a variety of methods [10].

The processes and memory systems of Soar 9 are shown in Figure 1. The declarative short-term memory (termed working memory) contains the agent's representation of its current situation, including representations of environmental perception and actions. All long-term, procedural knowledge is encoded as production rules. Whenever a rule's conditions match working memory structures, the rule is fired and its actions executed. Actions may involve adding or removing structures from working memory, as well as asserting preferences used to initiate and control deliberate action. Soar learns new procedural knowledge through its "chunking" mechanism [11], which

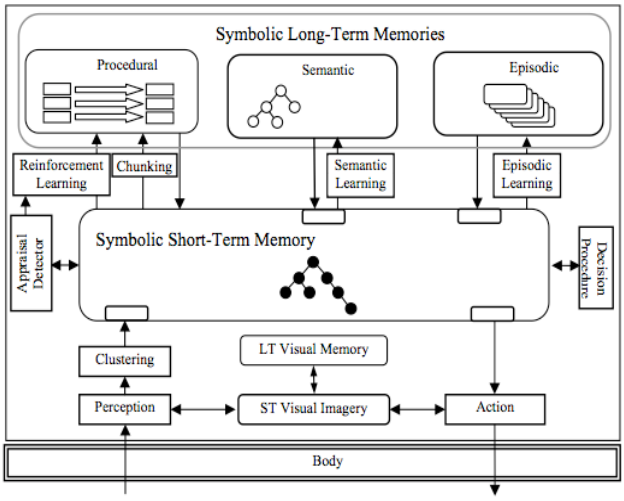

Figure 1. Soar 9: memories (rounded) and processes (box)

monitors problem solving and automatically creates new rules to summarize sub-task results. Procedural control knowledge is tuned via a reinforcement learning mechanism [16].

At regular intervals, the architecture automatically stores a snapshot of working memory in episodic memory [2]. The agent can retrieve a stored episode by creating a cue of working memory structures. The episodic memory retrieval mechanism searches the memory and selects the episode that best matches the cue (biased by recency), and then reconstructs the episode in working memory (in a special area so that the retrieved episode is not confused with current experience).

Soar's semantic memory supports storage and retrieval of concepts, facts, and relations, independent of the context in which they were originally experienced. An agent retrieves concept knowledge from semantic memory into working memory using a cue analogous to that of episodic memory.

As for overall classification, Soar's symbolic memories are distinguished as short-term, where transient structures representing an agent's understanding of the current situation are stored in working memory, and long-term, where persistent structures are maintained that are potentially relevant to future situations. The long-term symbolic memories can be divided into systems of procedural and declarative knowledge, wherein declarative information is further sub-divided as either semantic or episodic [25]. Within the architecture, all three of these memories have a parallel structure, where each one is structurally accessed independently. Soar integrates long-term memory retrievals, procedural control knowledge, and a fixed decision cycle to form a reactive, dual-process model [24]. This organization contrasts strongly with frameworks like Tulving's SPI, where there is an *embedding* or hierarchical organization between memory systems [30][31]. In Soar, there can be interactions between the modules, such as the retrieval from semantic can be used to cue the retrieval of knowledge from episodic, and in general, structures from any memory (and perception) can be stored and used to cue retrievals from other memories. In the next sections, we will characterize and functionally analyze each of these memory systems in detail.

3 WORKING MEMORY

The Soar cognitive architecture represents an agent's current situation in a short-term, declarative, symbolic memory system. Unlike many psychological models of short-term memory [23], Soar's working memory is not an activated portion of knowledge in long-term memories, but is a separate store. Furthermore, the contents of Soar's working memory, despite its label as "short-term," are not forgotten as a result of inactivity or decay [19].

Soar's working memory entails no active processing, but instead serves as a common representation substrate for procedural reasoning, initiating external action, and cueing retrievals for other long-term memories. It is composed of an arbitrarily large set of *mental entities* and associated symbolic augmentations [22], represented in a single connected graph. A mental entity can represent an object in the world (present, past, or hypothetical) or a mental concept, such as an "idea," with no experienced physical parallel. Mental entities can be created from the external environment (either via perception or motor system feedback) or the internal memory systems (such as the firing of a production rule or an episodic/semantic retrieval) and the representation does not distinguish as to knowledge provenance. Augmentations, implemented as an arbitrarily large set of symbolic attribute-value pairs for each mental entity, provide a fully general, relational description language. With these theoretical commitments, working memory serves functionally to allow the agent to symbolically represent arbitrary and novel combinations and compositions of experienced and hypothetical mental entities during its decision-making.

4 PROCEDURAL MEMORY

Soar's procedural memory stores knowledge, in the form of production rules, about when and how to perform both internal and external actions [17]. Production rule representation is asymmetric: the antecedent, termed "conditions" or left-hand-side (LHS), concisely specifies a pattern to be matched against working memory, represented as a conjunctive set of variablized working memory structures. When the antecedent successfully matches working memory, the consequent of the rule is executed. The consequent is also termed "actions" or right-hand-side (RHS), represented as a conjunctive set of variablized working memory modifications. These representational semantics are in contrast to *associative* memory processing, in which retrievals supplement agent state by *completing* (with respect to a matching metric) a cue from prior experience [24].

As is typical of production systems, procedural retrievals in Soar involves determining which productions successfully match working memory. The result of the procedural memory system retrieval, the *match set*, is the set of productions, each paired with the set of working memory structures that satisfies its conditions. However, whereas many architectures, such as ACT-R [1], implement a conflict resolution mechanism with respect to the match set and fire a single rule each primitive cycle, Soar executes *all* rule instantiations in the match set in parallel. Consequently, procedural retrieval allows the agent to have an arbitrary, global control policy over its representational state, and can do this efficiently even for large bodies of knowledge. The reason is that Soar's procedural memory is encoded as rules with local pattern matching, for which there exist efficient, scalable matching algorithms [3][5].

The procedural memory system learns knowledge via chunking and reinforcement learning (RL) mechanisms [11][16]. The former creates a new production when results are created in sub-tasks by compiling a trace of production firings. The conditions of the resulting rule is composed of structures that were necessary for producing the result, and the actions are the results. Functionally, chunking is a learning mechanism that converts deliberate search in subtasks into reactive rules that are accessed via match over procedural memory: it converts agent deliberation into reaction.

While Soar's chunking mechanism efficiently refines general procedural knowledge to apply to more specific situations, as they are experienced, it is write-only and does not change learned rules. Soar's RL mechanism, however, incrementally tunes production actions to reflect an expectation of action *performance*, with respect to an environmental or intrinsic reward signal [27]. Functionally, the RL mechanism enables the agent to incrementally improve its decision-making by reflecting relevant past experience, without the need for a full model of the results of its actions in a potentially stochastic environment.

5 SEMANTIC MEMORY

The functional purpose of a semantic memory is to efficiently retrieve declarative facts about a mental entity that have been experienced by an agent, independent of the context in which they were originally learned [29]. The existence of this memory is based on the assumption that some aspects of experience are re-usable over time, independent of how situations may differ temporally, spatially, or with respect to other contextual distinctions relative to an agent's state and goals.

It is possible to represent generic facts about mental entities both in procedural and working memory, but both approaches would appear to incur significant performance costs. Representing content-addressable, generic facts in productions, termed *data chunking* [21], requires the number of rules to be combinatorial in the number of relevant features for each mental entity. For instance, if a mental entity, such as a mental representation of a person, was described by 20 symbolic features, such as age, weight, hair colour, etc., data chunking would require 2^{20} (about 1 million) rules to provide access to the person based upon any combination of the features. Naively representing generic facts in working memory will require much less space, but match time proportional with the number of mental entities [28]. For instance, in Figure 3 we illustrate how match time per semantic retrieval increases as the number of mental entities increases (here, each mental entity has 20 symbolic features). When using the procedural matching mechanism, retrieval time increases exponentially with the number of entities, versus logarithmic increase using a dedicated semantic matcher.

The source of this efficiency rift has to do with the types of problems these memories are designed to address. The result of a successful semantic retrieval is a single mental entity, with its associated declarative facts. Thus, the data structures and algorithms optimized for semantic retrievals can unfold incrementally and locally, dynamically re-ordering constraints to find the *first* match. In contrast, Soar's procedural matching seeks *all* sets of satisfactory working memory structures. Thus, efficient production matchers [5] typically implement *global* caching techniques, spread over a *static* discrimination network, to efficiently index working memory structures that could apply to *any* condition. This analysis lends credence to the possibility that a semantic memory system may be functionally incompatible with a procedural memory system.

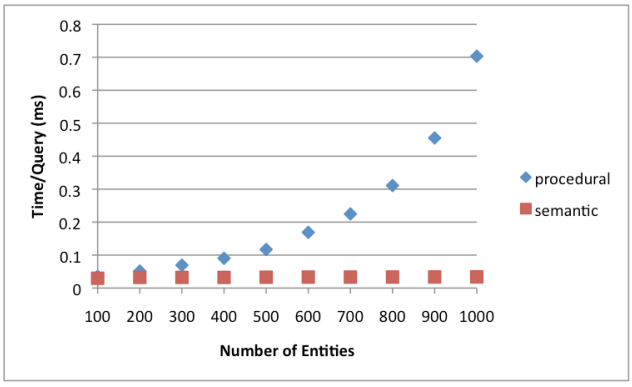

Figure 3. Semantic matching time vs. number of entities

Over a long lifetime, it is conceivable that a learning agent, engaging in multiple, complex tasks, will need to incrementally store and have efficient access to large amounts of semantic knowledge, such as in common-sense/ontological [13] and lexical databases [4][15]. Consequently, to achieve the aforementioned functionality, Soar implements a semantic memory system distinct from working and procedural systems. The knowledge in semantic memory is based on structures originally in working memory and the primitive representation is the same in semantic memory as in working memory. To maintain overall agent reactivity, retrievals from the semantic memory system, specified as feature cues in working memory, are restricted to bounded matching and thus any complex reasoning, such as inductive or deductive processing, is retained for deliberate action. Semantic learning is incremental and stored knowledge can change over time.

6 EPISODIC MEMORY

As first described by Tulving, episodic memory captures historical knowledge contextualized in agent experience [29]. Whereas semantic knowledge encodes what an agent "knows," episodic knowledge captures an historical stream of what an agent "remembers." Functionally, Tulving discusses the following requirements of an episodic memory system:

R1. Architectural: episodic retrievals are available for all tasks and the process of storing memories does not compete with knowledge-based reasoning.
R2. Automatic: episodic memories are stored without deliberation. Reasoning can only indirectly influence episodic storage, such as through deliberate rehearsal.
R3. Autonoetic: retrieved episodic memories are distinguished from current sensing.
R4. Autobiographical: retrieved episodes are represented in the context in which they were originally experienced.
R5. Temporally indexed: retrieved episodes include meta-data providing temporal context with respect to other episodes.

Nuxoll [18] postulates as to some of the functional roles episodic retrievals may serve, such as facilitating virtual sensing, action modeling, and retroactive learning.

Some work [2] has been done to understand the computational challenges involved in meeting the functional requirements of an episodic memory system. The complex algorithms involved would *seem* to suggest that no combination of working, procedural, and semantic memory systems could achieve efficient episodic retrievals. However, we observe that a crucial paradigm for efficient retrievals, "only processing changes," holds true not only of episodic memory, but also procedural. In fact, in the following analysis we develop a mapping on the basis of encoding, storage, and matching, demonstrating that in Soar, the matching required for procedural memory forms a *proper subset* of the matching required for episodic memory.

When working memory adds structures, efficient implementations for both procedural and episodic memory systems must process and store the event. The two systems differ, however, at the removal of a structure. Production conditions apply only to current agent state, and thus procedural memory need not keep a persistent log of historical structures.

Episodic memory, however, permanently stores the time at which the removal took place and indexes the event, ensuring the ability to faithfully and efficiently reconstruct the episode at any point in the future.

When a new production rule is added to procedural memory, Soar's implementation optimizes its caching structures and discrimination network to efficiently match against the new "cue" (encoded as the rule's LHS). When a cue is supplied to episodic memory, a very similar process takes place (the cache/discrimination network is termed a Disjunctive Normal Form graph in [2]). However, whereas procedural memory considers only current working memory structures, episodic filters *all* pertinent historical working memory changes, until finding a match. When a match is found in these systems, Soar fires the production rule instantiation or reconstructs the episodic memory, respectively.

Analogous to the semantic memory analysis, Soar's implementation of episodic memory dictates that retrievals result in a single episode and, if possible, a single graph-match between cue and episode. If we relax these constraints, changing nothing else about the system specification, episodic retrieval is functionally equivalent to procedural matching over a log of pertinent historical working memory additions and removals. In fact, we have implemented a proof-of-concept episodic memory matcher using Soar's working and production memory systems, where episodes, stored as an in-memory list of working memory changes, are compared to a cue, instantiated as conditions in a set of production rules.

Production matching suffers from an exponential worst-case complexity, with respect to the size of working memory. By extension, an episodic retrieval is *strictly* harder, with the potential for a linear increase with respect to the number of experienced episodes, which, over long agent lifetimes, is likely to be much greater than the size of a single episode. Thus, as discussed in the conclusion of [2], a bounded, task-independent episodic memory system must likely adopt heuristic schemes that provide tighter bounds. This finding offers some compelling evidence for the dissociation between episodic and procedural memory systems.

To afford agents the functional benefits of episodic memory over long lifetimes [9], Soar 9 implements the memory system described in [2]. The episodic learning mechanism automatically encodes a subset of working memory (R1, R2) at regular intervals (R2) and temporally indexes this knowledge within the episodic store (R5). The agent deliberately retrieves from episodic memory by constructing a cue (R1), partially specifying relevant contextual features within the episode. The retrieved episode is fully re-constructed (R4) in a special region of working memory (R3), such that the agent does not confuse current sensing and past experience.

7 SUMMARY OF MEMORY SYSTEMS

Thus far, we have analysed the functionality of the symbolic memory systems in the Soar 9 cognitive architecture. To summarize, working memory allows the agent to symbolically represent, reason with, and retrieve long-term knowledge about arbitrary and novel combinations and compositions of mental entities. The procedural memory represents knowledge about when and how to perform internal and external actions.

Soar also contains two declarative long-term memories. Semantic memory provides context-independent, content-addressability to persistent knowledge about mental entities. Episodic memory provides content-accessibility, but contextualized within autobiographical agent experiences. For both of these memory systems, we have shown some computational evidence that procedural memory, while general enough to represent their contents, is likely to be unable to do so in an efficient manner required for reactivity with a dynamic environment over long agent lifetimes.

The memory mechanisms in Soar, while biologically and psychologically inspired, are not intended to model the details of human memory systems. Rather, these systems provide efficient access to classes of agent experience, supporting a broad range of cognitive capabilities and increasingly rational agent behaviour. However, they do not currently support many of the memory phenomena detailed in psychological literature. For instance, retrieval from semantic memory is not biased by additional structures in working memory, as in ACT-R [1], nor does episodic memory support spontaneous retrievals.

8 INTER-MEMORY OBJECT IDENTITY

By integrating multiple memory systems in Soar 9, we gain many functional benefits, but some new issues arise. In this section, we detail one such issue, *persistent object identity*: the problem of managing distinct, persistent objects over multiple memory systems.

Historically [10], Soar contained only a working memory and a procedural memory, with only a single learning mechanism, chunking. In this context, mental entities were temporary, persisting only as long as their representation was maintained in working memory. Consequently, Soar implemented a *weak* commitment to object identity [7], wherein mental entities in working memory during procedural retrievals were distinguishable only through context, with respect to the agent's state, and features.

However, with the integration of the semantic memory system in Soar 9, mental entities have gained persistence and need to be distinguishable from each other over time to functionally support semantic relations. For instance, consider an agent reasoning about its car ("should I buy a new hybrid now, or run this one into the ground?"). To make an environmentally friendly, yet economical, decision, the agent must be able to store, retrieve, reason with, and update semantic knowledge about *its* car, vehicles in general, and other cars with which the agent has had personal experience (such as a prior car, or that of a friend/co-worker or advertisement). With a weak commitment to object identity, knowledge updating would introduce ambiguity between these mental entities over time, depending upon their features and context (such as if two past vehicles were both painted red). Furthermore, as knowledge in any/all of the multiple symbolic memories can reference the same mental entities, there is an analogue to the symbol grounding problem, motivating a *strong* commitment to object identity [22].

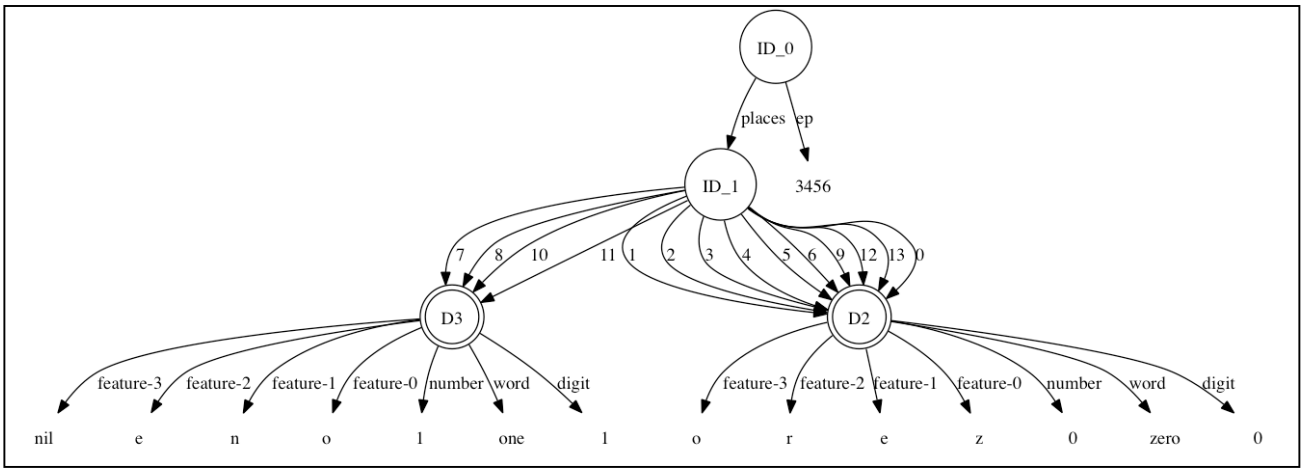

Figure 4. Generalized counting example

Soar 9 implements a theory of *learned* object identity. Mental entities are initially represented using a weak or *transitory* identity, but, once added to semantic memory, transition to a strong or *persistent* identity. This theory has important implications for memory system learning and retrievals.

A system implementing weak object identity benefits from highly generalizable learned knowledge. For example, variablized production rules refer to *any* mental entity that satisfies contextual and feature constraints. A stronger commitment to object identity implies that learning mechanisms must identify, and thus learned knowledge must precisely express, the entities to which the learned knowledge extends. Theoretically, because Soar 9 implements a progressive strategy of object identity, it can start with general knowledge to guide action. As the agent gains knowledge of persistent objects, related knowledge in other memory systems will become more specific. Extending the legacy of chunking, Soar functionally benefits from knowledge becoming increasingly specific to particular circumstances. However, this places a strong burden on the agent and the architecture to implement an effective policy for learning mental entity persistence.

From a performance perspective, memory retrievals in a reasoning system implementing weak object identity scale with the number of contextually identical objects (with respect to matching semantics). This is evidenced by the multi-valued attribute problem in production systems [28] and column/value indexing in relational databases. For instance, posing the query "find me all cars that have exactly 50MPG highway fuel efficiency" will depend on the number of cars asserted in the system. In a system implementing strong object identity, however, memory retrievals will scale with the number of *persistent* objects, which can be efficiently addressed via hashing ("find me the Prius!").

We have evaluated the performance implications of object identity in Soar's episodic retrievals using a generalized counting domain. The task requires a sequence of retrievals, in which the agent relationally represents an incrementing counter in a particular numeric base. For instance, in Figure 4, the agent is representing the decimal number 3456 in base 2. Here, node D2 represents a zero (0), node D3 represents a one (1), and the numeric edges connecting node ID_1 to D2 and D3 indicate that when 14 positional "places" are used to represent 3456 in binary ("00110110000000"), the 7th, 8th, 10th, and 11th digits (zero-based) will be one's, and the rest zero's. In our experimental setup, we can vary the number of episodes (the maximum number to which the agent counts), the number of positional digits used to represent the number, and the numeric base in which the value is relationally represented. Additionally, we can associate an arbitrary number of features with each digit object.

In Figure 5 we plot total query time versus an increasing numeric base in the generalized-counting task. We hold constant the number of features-per-digit (0) and the total number of episodes (50,000). The cue is constructed, per numeric base, to represent the digit in the least significant digit of the greatest episode. For instance, the decimal number 49,999 in octal (141517) has the digit "7" in its 0th place. In each cue, the digit entity was either transitory or persistent. Because Soar's episodic memory system searches episodes in order of decreasing recency [2], these cues match the first candidate episode. What we see is that while the time to retrieve a query in the persistent case remains nearly constant, the time to retrieve in the transitory case increases linearly with the numeric base. This is because the semantics of the transitory cue ask for any digit in the 0th place, and this ambiguity increases with the number of digits dictated by the numeric base.

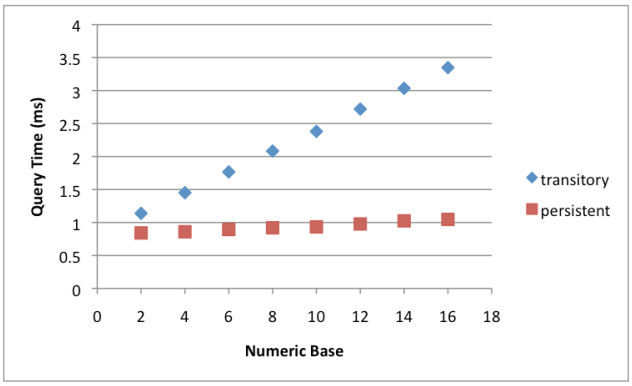

Figure 5. Time (ms) retrieving the most recent episode vs. numeric base

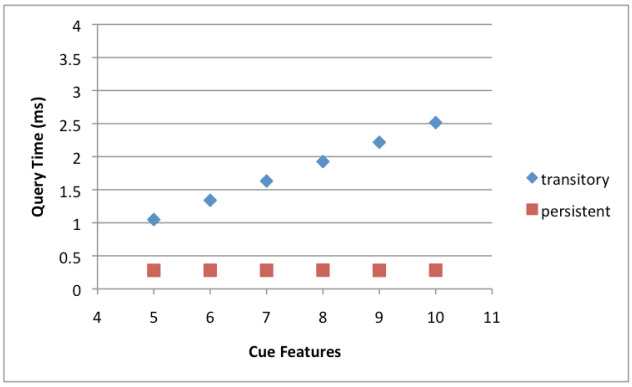

Figure 6. Time (ms) retrieving the most recent episode vs. cue size

Similarly, if we perform the same experiment but hold constant the numeric base (16) and vary the features-per-digit (5-10), as in Figure 6, we find that the cue features have no effect on query time for a persistent mental entity, but linearly increase retrieval time with the transitory entity because the system must match a linearly increasing number of constraints.

Finally, in Figure 7, we repeat the first experiment, but force the matcher to consider all candidate episodes in the episodic store. In this case, a cue with a persistent entity improves the selectivity of the matcher, requiring consideration of (total episodes)/(numeric base) episodes and thus it speeds up with larger bases. For instance, the matcher had to consider about 3k episodes in hexadecimal, 5k in decimal, and 25k in binary. No such speed up is achieved with the transitory entity, where the matcher has to consider the full 50,000 episodes.

9 CONCLUSIONS & FUTURE WORK

This paper provided a memory-centric analysis of the Soar 9 cognitive architecture. We functionally dissected the multiple symbolic memory systems in Soar 9, contextualizing the theoretical commitments within existing psychological research into memory and contributing computational credence to theories of dissociation and memory organization.

Many important integration issues remain for future investigation. As indicated, learning *when* to commit to persistent object identity is crucial for an effective balance of efficient knowledge access and learned knowledge transfer. Right now the agent performs deliberate storage. One possible route is to incorporate working memory activation or appraisal [14] meta-data as measure of entity importance, and commit when some threshold is exceeded.

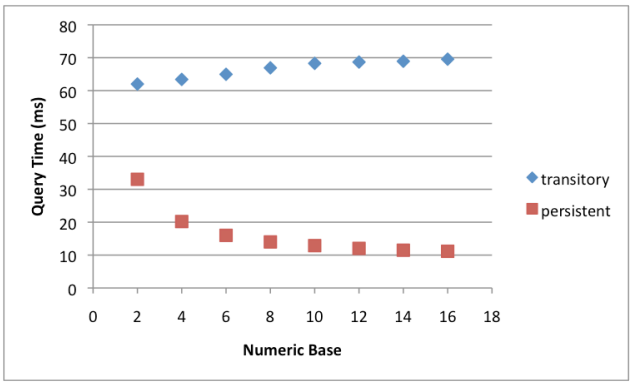

Figure 7. Time (ms) to traverse all candidate episodes

Another issue is how regularities in perceptual data, such as audition or vision, can/should be captured in semantic memory [8]. This is the very challenging problem of knowing *when* to assign a persistent symbol to non-symbolic information.

Of additional concern is how to represent and reason about inconsistent knowledge from differing memory systems. For instance, if the agent is hypothesizing about a persistent mental entity in working memory and invokes an episodic retrieval about that entity, how should Soar capture the varying source of data (if at all)?

Also, while this paper has focussed on the symbolic memory systems in Soar, the architecture has memories that store short- and long-term visual knowledge [8] and future research may incorporate other modalities as well. Interesting questions arise as to how the architecture should manage, reason about, and simulate persistent mental entity commitments across facilities computing in multiple modalities.

Finally, a cornerstone of Soar's commitment to these dissociated memory systems is that controlled retrievals from multiple systems can bolster relevant knowledge coverage, and thus endow the agent with new cognitive capabilities and improved rationality. However, initial investigation [6] into an agent learning to use even a single memory system, in a very simple environment, suggests that significant work remains in developing proper representations and algorithms to support rational decision-making, using multiple memory systems in complex environments.

REFERENCES

[1] Anderson, J.R., Bothell, D.B., Michael D., Douglass, S., Lebiere, C., Qin, Y.: An Integrated Theory of the Mind. In: Psychological Review, Vol. 111 (4), pp. 1036-1060 (2004)

[2] Derbinsky, N., Laird, J.E.: Efficiently Implementing Episodic Memory. In: Proceedings of the 8th International Conference on Case-Based Reasoning (ICCBR), pp. 403-417 (2009)

[3] Doorenbos, R.B.: Production Matching for Large Learning Systems. PhD Thesis, Carnegie Mellon (1995)

[4] Douglass, S., Ball, J.: Large Declarative Memories in ACT-R. In: Proceedings of the 9th International Conference of Cognitive Modeling (ICCM), (2009)

[5] Forgy, C.L.: Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. In: Artificial Intelligence, Vol. 1, pp. 17-37 (1982)

[6] Gorski, N.A., Laird, J.E.: Learning to Use Episodic Memory. In: Proceedings of the 9th International Conference on Cognitive Modeling (2009)

[7] Khoshafian, S.N., Copeland, G.P.: Object Identity. In: Proceedings of the 1986 Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA), pp. 406-416 (1986)

[8] Laird, J.E.: Extending the Soar Cognitive Architecture. In: Proceedings of the 1st Conference on Artificial General Intelligence (AGI), pp. 224-235 (2008)

[9] Laird, J.E., Derbinsky, N.: A Year of Episodic Memory. In: Workshop on Grand Challenges for Reasoning from Experiences, 21st IJCAI (2009)

[10] Laird, J.E., Rosenbloom, P.: The Evolution of the Soar Cognitive Architecture. Mind Matters: A Tribute to Allen Newell, pp. 1–50 (1996)

[11] Laird, J.E., Rosenbloom, P., Newell, A.: Chunking in Soar: The Anatomy of a General Learning Mechanism. In: Machine Learning, Vol. 1 (1), pp. 11-46 (1986)
[12] Laird, J.E., Wray III, R.E.: Cognitive Architecture Requirements for Achieving AGI. In: Proceedings of the 3rd Conference on Artificial General Intelligence (AGI), (2010)
[13] Lenat, D.B.: CYC: A Large-Scale Investment in Knowledge Infrastructure. In: Communications of the ACM, Vol. 38 (11), pp. 33-38 (1995).
[14] Marinier, R., Laird, J.E.: Emotion-Driven Reinforcement Learning. In: Cognitive Science (2008)
[15] Miller, G.A.: WordNet: A Lexical Database for English. In: Communications of the ACM, Vol. 38 (11), pp. 39-41 (1995).
[16] Nason, S., Laird, J.E.: Soar-RL: Integrating Reinforcement Learning with Soar. In: Cognitive Systems Research, Vol. 6 (1), 51-59 (2005)
[17] Newell, A.: Unified Theories of Cognition (1994)
[18] Nuxoll, A.: Enhancing Intelligent Agents with Episodic Memory. PhD Dissertation, University of Michigan (2007)
[19] Nuxoll, A., Laird, J.E., James, M.: Comprehensive Working Memory Activation in Soar. International Conference on Cognitive Modeling (ICCM), Poster (2004)
[20] O'Reilly, R.C.: Modeling Integration and Dissociation in Brain and Cognitive Development. In: Processes of Change in Brain and Cognitive Development: Attention and Performance, Munakata, Y. & Johnson, M.H. (Eds.), Vol. 21, pp. 375–402 (2006)
[21] Rosenbloom, P.S.: A Cognitive Odyssey: From the Power Law of Practice to a General Learning Mechanism and Beyond. In: Tutorials in Quantitative Methods for Psychology, Vol. 2 (2), pp. 38-42 (2006)
[22] Santore, J.F., Shapiro, S.C.: A Cognitive Robotics Approach to Identifying Perceptually Indistinguishable Objects. In: Anchoring Symbols to Sensor Data, Papers from the AAAI Workshop, Technical Report WS-04-03, Coradeschi, S. & Saffiotti, A. (Eds), pp. 1-9 (2004)
[23] Sherry, D.F., Shacter, D.L.: The Evolution of Multiple Memory Systems. In: Psychological Review, Vol. 94 (4), pp. 439-454 (1987)
[24] Smith, E.R., DeCoster, J.: Dual-Process Models in Social and Cognitive Psychology: Conceptual Integration and Links to Underlying Memory Systems. In: Personality and Social Psychology Review, Vol. 4 (2), pp. 108-131 (2000)
[25] Squire, L.R., Knowlton, B., Musen, G.: The Structure and Organization of Memory. In: Annual Review of Psychology, Vol. 44, pp. 453-495 (1993)
[26] Strosnider, J.K., Paul, C.J.: A Structured View of Real-Time Problem Solving. In: AI Magazine, Vol. 15 (2), pp. 45-66 (2004)
[27] Sutton, R.S., Barton, A.J.: Reinforcement Learning: An Introduction (1998)
[28] Tambe, M., Kalp, D., Rosenbloom, P.S.: An Efficient Algorithm for Production Systems with Linear-Time Match. In: Tools with Artificial Intelligence (TAI), pp. 36-43 (1992)
[29] Tulving, E.: Elements of Episodic Memory (1983)
[30] Tulving, E.: Multiple Memory Systems and Consciousness. In: Human Neurobiology, Vol. 6 (2), pp. 67-80 (1987)
[31] Tulving, E.: Organization of Memory: Quo Vadis? In: The Cognitive Neurosciences, Gazzaniga, M.S. (Ed), pp. 839-853 (1995)