Long-Term Declarative Memory for Generally Intelligent Agents

Thesis Proposal

Nate Derbinsky

Table of Contents

1. Introduction

At the conclusion of the summer of 1955, John McCarthy, writing a proposal for a Dartmouth summer research project, coined *Artificial Intelligence* as the study of making machines that "solve kinds of problems now reserved for humans" (McCarthy et al., 1955; Newell, 1991). Over 50 years later, AI has flourished as a research community and many advances have been made, but humans remain sui generis – the only exemplars of generally intelligent, long-lived, learning agents: contrasting an ecosystem of task-specific, short-lived, brittle computer systems (McCarthy, 2007), most persons survive for decades, autonomously contending with, and continually improving performance on, multiple, complex tasks for many hours every day.

Memory systems compose one class of mechanism the human cognitive architecture employs to contend with a dynamic environment in which, amongst other challenges (Laird & Wray, 2010), an individual can only perceive a small part of the complete state of the world at any point in time. A memory system captures, or *encodes*, some aspect of experience; *stores* this information as internal knowledge, potentially changing it over time; and provides for efficient *retrieval* at a later time. Precise and timely knowledge retrievals from this internal, experiential store, when combined with other cognitive mechanisms and processes, functionally serve to robustly enhance a human's situational awareness and decision-making on a variety of tasks, despite a complex, dynamic world.

While human memory is demonstrably not immune to error or bias (Miller, 1956; Owens & Bower, 1979; Schacter, 1999), it inspires AI research by concurrently exhibiting two important functional characteristics. First, human memory doesn't seem to run out of space, despite being bombarded with dense and varying torrents of information, including data that is autobiographical (Tulving, 1983; Laird & Derbinsky, 2009), lexical (Miller, 1995), conceptual (Kolodner, 1983a; Medin & Smith, 1984; Davidsson, 1995; Niles & Pease, 2001); and commonsensical (Lenat, 1995). Furthermore, as human experts amass these vast stores of knowledge, their memory capability does not suffer; rather, they improve performance on domain tasks (Smith et al., 1978), in contrast to many computer models and systems (Minton, 1990; Tambe et al., 1990; Smyth & Cunningham, 1996; Douglass et al., 2009). In summary, despite a deluge of experience, humans, unlike many artificial agents, do not drown; they push forward, bringing to bear their knowledge and reasoning abilities to flourish in challenging and novel situations and tasks.

A review of prior psychological and computational work (Derbinsky & Laird, 2010) suggests that this resilient behavior is due in part to the multiple, dissociated human memory systems: these studies cite the potential for significant functional and computational detriment when utilizing a single memory mechanism for different types of learning tasks. But while research and development of cognitive architectures typically reflects this dissociation strategy (Langley et al., 2009), significant additional work must still be done to understand the role long-term memory systems play in a general cognitive architecture: specifically, and concurrently, (Q1) what *functionality* must memory systems

Table 1. The Degree to which Related Fields of Research Satisfy Requirements of Memory Systems for Generally Intelligent Agents

		REQUIREMENTS					
		R1 Incremental Learning	R2 Comprehensive Learning	R3 Diverse Representation	R4 Scale Efficiently	R5 Effective Access	R6 Task Independence
FIELDS	Cognitive Modeling/ Architecture	☑	☐	☑		☑	☑
	Case-Based Reasoning	☐	☐	☐	☐	☑	
	Information Retrieval/ Databases		☐	☑	☑	☐	☑
	Knowledge Representation	☐		☑	☐	☑	☑

support to achieve human-level intelligence? and (Q2) how can this functionality be *efficiently* supported in long-living agents?

Table 1 summarizes the degree to which fields of related research contribute to these questions. The requirements (R1-R6) are derived from characteristics of the structure of generally intelligent agents, the tasks with which they contend, and the environment in which they are embedded, borrowing heavily from Laird & Wray (2010), and they attempt to capture, along multiple dimensions, critical nuances of the terms *functionality* and *efficiency*. These requirements are then cross-referenced with trends and achievements in related fields of research, wherein an empty cell signifies little-to-no contribution, an open box (☐) signifies partial contribution, and a checked box (☑) symbolizes significant benefaction. We develop this chart in greater detail later (Section 2), but before that presentation we can glean three important points. First, the requirements of memory systems imposed by the context of generally intelligent agents are numerous, challenging, and antithetical in concurrence, such as scaling efficiently (R4), in a task independent fashion (R6), given a diversely represented (R3), comprehensive knowledge store (R3). Second, while it is unsurprising that no individual field can lay claim to concurrently satisfying *all* of these requirements, it is significant to note, however, that no field fully understands R2, the comprehensive spectrum of diverse learning mechanisms that memory systems must support to achieve human-level intelligence, which relates strongly to question Q2 above. Finally, there appear to be opportunities for cross-fertilization to more completely satisfy requirements within a single realm. For instance, we have preliminary evidence (Derbinsky & Laird, 2009; Derbinsky et al., 2010), discussed in greater detail later (Sections 5 and 6), that database and information retrieval techniques are highly effective in efficiently (R4) supporting task-independent (R6) memory systems within a general cognitive architecture, relating to question Q1.

To incrementally contribute to the cognitive architecture literature in memory systems, we propose to improve the functional (R2) and computational (R4) understanding of two human-inspired, well-studied, long-term declarative memory systems, *semantic* and *episodic* (Tulving, 1983), which straddle two extremes of associating context with stored experience. Psychological literature describes semantic memories as including general facts that the agent "knows," independent of the context in which they were originally learned, which can be applied to improve understanding and task performance in

numerous, potentially unrelated situations. In contrast, episodic memories reflect autobiographical, contextualized agent experience that allows an agent to "remember" its own past, such as the consequences of an action in a similar situation and using that knowledge to decide how to act presently.

Discussed in greater detail later (see Section 7), we plan to explore extensions to the episodic and semantic memory modules of the Soar cognitive architecture, which will enhance the functionality of encoding, storage, and retrieval of memories, while efficiently scaling to large stores of knowledge over long agent lifetimes. These extensions begin to address the following two fundamental research questions, which concurrently contribute to our requirements (R2, R4) across the full spectrum of memory operations (encoding, storage, and retrieval):

> What aspects of agent experience should a task-independent memory mechanism *encode* and *store* such as to functionally support performance across a variety of tasks while maintaining reactivity to complex, dynamic environments?
>
> What task-independent regularities of agent experience can efficiently supplement incomplete task-dependent knowledge to improve the expected utility of *retrieved* memories in response to impoverished cues?

In context of episodic memory, we plan to explore the following approaches to these questions, respectively:

X1. **Semantic Pruning of Episodic Encoding**
We plan to explore the degree to which we can reduce long-term episodic storage requirements (R4) in a task-independent fashion (R6) by not encoding historical features and relations of long-term semantic concepts (R2) without significantly degrading the efficacy (R5) of episodic retrievals.

X2. **Efficient Episodic Activation Bias**
We plan to explore task-independent (R6), efficient (R4) methods of incrementally (R1) incorporating varying levels of regularity of agent experience (R2), such as episode-level appraisals and element-level activation, as forms of bias within the retrieval algorithm to improve quality in cases of under-specified cues (R5).

And with semantic memory, we will explore the following approaches, respectively:

X3. **Automatic Semantic Encoding**
We plan to explore the degree to which we can reduce long-term semantic storage requirements (R4) by incrementally (R1) incorporating different agent and environmental regularities (R2) as sources of task-independent (R6) knowledge to inform automatic encoding without significantly degrading the efficacy (R5) of semantic retrievals.

X4. **Efficient Semantic Activation Bias**

We plan to explore task-independent (R6), efficient (R4) methods of incrementally (R1) incorporating regularities of semantic memory requests (R2), such as retrieval history and context, as forms of bias within the retrieval algorithm to improve quality in cases of under-specified cues (R5).

Table 2 summarizes these extensions in context of the requirements (R1-R6) in Table 1 and memory system organization described above (episodic vs. semantic; encoding/storage vs. retrieval). A dot (•) signifies *maintaining* satisfaction of a requirement, while an upward arrow (🡅) indicates that work on the extension will *improve* satisfaction of the requirement. Note that R3, supporting diverse knowledge representations, is satisfied by virtue of extending work on the expressive, relational, symbolic representation in the Soar cognitive architecture, as discussed in greater detail later (Section 3).

Table 2: How Proposed Extensions Relate to Memory System Organization and Requirements

				REQUIREMENTS					
				R1	R2	R3	R4	R5	R6
EXTENSIONS	**SEMANTIC**	*Retrieval*	X4	•	🡅	•	🡅	•	•
		Encoding Storage	X3	•	🡅	•	🡅	•	•
	EPISODIC	*Retrieval*	X2	•	🡅	•	🡅	•	•
		Encoding Storage	X1	•	🡅	•	🡅	•	•

In the remainder of this document, we precisely describe how we plan to explore these extensions in a generally intelligent, artificial agent's dynamic, long-term declarative memory systems as it experiences, and remains reactive to, its environment and contends with multiple, complex tasks. To begin, in Section 2 we more thoroughly discuss artificial memory systems in context of generally intelligent agents, including greater detail about the functional requirements that fall out of the characteristics of environment and task confronting long-living, autonomous, learning agents. Given this foundation, we proceed in Section 3 to frame our proposed approach, focusing on research methodology. We then review our prior work, in which we focused on understanding the computational challenges involved in extending a general cognitive architecture with basic, task-independent episodic (Section 4) and semantic (Section 5) functionality that scales with large bodies of knowledge. In Section 6, we assess our progress and conjointly analyze our prior work in preparation for Section 7, in which we formulate our plan for future research, including proposed functional extensions, evaluation, and timeline.

2. Memory System Requirements

In this section, we dissect memory systems in context of generally intelligent agents. We begin by characterizing this class of agents, enumerating properties of their structure, the types of task with which they contend, and the environment in which they are embedded, with a focus on how these characteristics relate to and constrain their memory mechanisms. Given this breakdown, we set forth functional requirements on artificial memory mechanisms and discuss related research fields and the degree to which their efforts apply to and satisfy these requirements.

2.1 Environment, Task, and Agent Characteristics

Here we enumerate properties of environment, tasks, and agents that lead to requirements for memory mechanisms that support generally intelligent, autonomous agents. This breakdown draws heavily on work by Laird and Wray (2010), in which they develop requirements for cognitive architecture, but specializes the discussion with respect to memory systems embedded within these architectures.

C1. Environment is Diverse with Complex and Interacting Objects

i. The agent can usefully interpret parts of the environment as if it consists of independent objects.
ii. There are many objects.
iii. Objects have numerous, diverse properties.
iv. Some objects share similarities with other objects.

C2. Environment is Dynamic

i. The environment changes independently of the agent.
ii. The environment may change rapidly, relative to agent decision-making.
iii. Environmental dynamics are complex: the agent cannot always accurately predict future states in detail.
iv. Some object properties (C1) change as a consequence of environment dynamics.

C3. Task-Relevant Regularities Exist at Multiple Time Scales

i. Environmental dynamics (C2) are not arbitrary: interactions are governed by physical laws that are constant, often predictable, and frequently lead to recurrence and regularity that impact the agent's ability to achieve goals.
ii. Regularities in environmental dynamics exist at multiple time scales.
iii. Regularities in environmental dynamics lead to regularities in intra- and inter-object property changes (C1, C2).

C4. Tasks can be Complex, Diverse, and Novel

i. Tasks properties and goals are complex.
ii. The agent will contend with numerous tasks during its existence.
iii. The agent will contend with tasks with novel properties and goals.
iv. Tasks vary in the time scales required to achieve them: some are close to the timescale of dynamics in the environment (C2) while others require extended behavior.

C5. Agent/Environment/Task Interactions are Complex and Limited

i. The environment is partially observable: it is impossible for the agents to perceive the entire state of the world.
ii. Agent sensors are noisy and may be occluded by objects (C1) and environmental dynamics (C2), making agent perception incomplete and uncertain.

C6. Agent Computational Resources are Limited

i. The agent has physical limits on its computational resources relative to dynamics of the environment (C2).
ii. Agent interactions (C5) within the complex (C1), dynamic (C2) environment and with complex tasks (C4), given bounded computational resources, make perfect rationality impossible.

C7. Agent Existence is Long-Term and Continual

i. Agent existence is long-term relative to primitive interactions with the environment (C2, C5).
ii. For the duration of its existence, the agent is always present in its environment.

2.2 Requirements for Memory Systems

Based upon the characteristics above, we derive the following requirements to constrain implementations of memory systems for generally intelligent agents. Once again, these borrow heavily from the cognitive architecture requirements of Laird and Wray (2010), but are specialized for memory systems.

R1. Support Incremental, Online Learning

Given that the agent…

i. is continually (C7) embedded within an environment that changes quickly and in complex ways (C2); and
ii. must assimilate and exploit environmental regularities (C3), when and as they become apparent, to effectively contend with diverse ongoing and future tasks (C4);

the agent requires memory systems that…

i. support *incremental* encoding and storage of new information, such that the contents of the agent's internal knowledge cache keep pace with environmental dynamics; and
ii. support *online* knowledge retrievals, such that agent reasoning reflects and takes advantage of its latest observations of the state of the world.

R2. Support Diverse, Comprehensive Learning

Given that the agent…

i. is embedded within a complex environment (C1) for a long-term existence (C7); and
ii. must assimilate and exploit environmental regularities (C3), which occur at varying time scales, including those apparent from a single instance or spread across time, in order to effectively contend with diverse tasks (C4) that entail complex interactions (C5);

the agent requires memory systems that…

i. individually support *diverse* forms of learning, such that optimized mechanisms will efficiently and accurately detect specific types of environmental regularities; and
ii. conjointly support *comprehensive* coverage of learning, such that the agent is broadly sensitive to, as well as able to represent and apply, a wide variety of task-specific knowledge about the world.

R3. Support Diverse Knowledge Representation

Given that the agent…

i. is embedded within a complex environment (C1) for a long-term existence (C7); and
ii. must assimilate and exploit environmental regularities (C3) in order to effectively contend with diverse tasks (C4) that entail complex interactions (C5);

the agent requires memory systems that…

i. support representing *diverse* types of knowledge, including contextualized memories of experiences, as well as more generalized facts, beliefs, and relations about objects in the world.

R4. Scale Efficiently to Large Bodies of Knowledge

Given that the agent…

i. is embedded within a complex environment (C1) that changes quickly (C2) over a long-term, continual existence (C7); and
ii. is contending with diverse tasks (C4) entailing complex interactions (C5);

the agent requires memory systems that…

i. support *efficient* incorporation of new information and access to existing knowledge, such that agent retrievals, drawing from the wealth of available knowledge that arises from environmental and task experience over a long lifetime, are timely, given the rate of environmental dynamics.

R5. Support Effective Access to Knowledge

Given that the agent…

i. must assimilate and exploit environmental regularities (C3) in order to effectively contend with numerous tasks (C4) entailing complex interactions (C5); and
ii. is embedded within a dynamic environment (C2) and is limited with respect to its computational resources (C6);

the agent requires memory systems that…

i. support *effective* access to knowledge about environmental regularities and past task performance, such that retrievals improve the agent's ability to contend with the complexities of its current situation.

R6. Task Independence

Given that the agent…

i. is embedded within a complex environment (C1) that changes quickly (C2) over a long-term, continual existence (C7); and
ii. must assimilate and exploit environmental regularities (C3), given limited computational resources, in order to effectively contend with numerous tasks (C4) that entail complex interactions (C5);

the agent requires memory systems that…

i. encapsulate environmental regularities and interaction complexities that are *independent of task* and that occur at time scales greater than that of the agent, thereby reducing the complexity of learning task-dependent knowledge.

Table 3 illustrates the how the characteristics (C1-C7) of environment, task, and agent, as described above, together impose these requirements (R1-R6) upon memory systems for generally intelligent agents. While all of these characteristics constrain memory systems, it is useful to note that task independence of the mechanism (R6) draws upon the breadth of the challenges with which generally intelligent agents contend, and all requirements are influenced by the constraint of dealing with numerous, complex, and novel tasks (C4), a property that typically does not apply to short-lived, task-dependent systems.

Table 3: The Requirements on Memory Systems for Generally Intelligent Agents Imposed by Characteristics of Environment, Task, and Agent

			CHARACTERISTICS						
			C1 Complex Environment	C2 Dynamic Environment	C3 Regularities in Environment	C4 Complex Tasks	C5 Complex Interactions	C6 Limited Agent	C7 Extended Agent
REQUIREMENTS	R1	Incremental Learning		✔	✔	✔			✔
	R2	Comprehensive Learning	✔		✔	✔	✔		✔
	R3	Diverse Representation	✔		✔	✔	✔		✔
	R4	Scale Efficiently	✔	✔		✔	✔		✔
	R5	Effective Access		✔	✔	✔	✔	✔	
	R6	Task Independence	✔	✔	✔	✔	✔	✔	✔

Table 4: The Degree of Requirements Coverage in Related Fields of Research

		REQUIREMENTS					
		R1 Incremental Learning	R2 Comprehensive Learning	R3 Diverse Representation	R4 Scale Efficiently	R5 Effective Access	R6 Task Independence
FIELDS	Cognitive Modeling/ Architecture	☑	☐	☑		☑	☑
	Case-Based Reasoning	☐	☐	☐	☐	☑	
	Information Retrieval/ Databases		☐	☑	☑	☐	☑
	Knowledge Representation	☐		☑	☐	☑	☑

2.3 Related Fields

The characteristics of environment, task, and agent structure impose significant requirements upon memory systems that must be considered and satisfied *concurrently*. In this section we briefly and broadly discuss related fields and the degree to which their efforts relate to and satisfy these requirements in the context of memory systems. This discussion is summarized in Table 4, a reproduction of Table 1 presented in the introduction to this document.

Cognitive Modeling/Architecture

While it is common for cognitive architectures and models to commit to task independent approaches (R6) for incremental and online encoding (R1) of arbitrary environmental perception (R3) as internal, declarative knowledge, as well as later retrievals using diverse (R5), and often cognitively inspired, methods (Langley et al., 2009), these mechanisms, with few exceptions, such as those architectures, like Soar (Laird & Rosenbloom, 1996), that utilize the Rete algorithm for efficient matching of productions (Forgy, 1982; Doorenbos, 1995), do not scale (R4) to large knowledge bases (Douglass et al., 2009; Douglass & Myers, 2010). Additionally, much work must still be done to explore the full breadth of learning mechanism implementation and integration (R2) that must be in place to effectively capture and apply the variety of task and environmental regularities encountered by long-living agents, including those of autobiographical agent experience (Derbinsky & Laird, 2009) and appraisals (Mariner et al., 2009), as well as statistical regularities in environmental and task demands (Schooler & Anderson, 1997).

Case-Based Reasoning

Case-based Reasoning (CBR: Kolodner, 1992) research focuses on methods for effectively accessing (R5), adapting, and incrementally updating (R1) prior case information to solve specific problems. While work has been done in case-base maintenance methods (Cummins & Bridge, 2009) to combat issues of case utility in large case bases (Smyth & Cunningham, 1996), research focus is typically not applied to online problem solving at the pace of environmental dynamics (R1), nor are most systems evaluated over long lifetimes (R4). Most work is highly applied and task-specific (R6), including static, problem-specific case formats (R3), as well as problem-optimized case retrieval, adaptation, revision, and retention algorithms (R2).

Information Retrieval/Database Management Systems

The Information Retrieval (IR: Singhal, 2001) and Database Management System (DBMS: Ramakrishnan & Gehrke, 1999) research communities have developed substantial literature over the last 50 years (Codd, 1970; Agrawal & Srikant, 1994; Gray et al., 1997; Chaudhuri, 1998; Zobel & Moffat, 2006) on efficient data structures and techniques (R4) for supporting task-independent (R6), expressive queries (R5), on large amounts of diverse data (R3), as well as batch analytical and statistical processing (R2). However, while it is common for problem specifications to detail properties of queries, users, and data services (ex: Chaudhuri et al., 2000), it is rare for these fields to focus on dynamic interactions with complex environments across numerous tasks, and thus they have little to contribute to issues of online (R1) or comprehensive (R2) learning of complex environmental and task regularities, nor the most effective forms of data access (R5) to support agent performance across a variety of tasks.

Knowledge Representation and Reasoning

Knowledge Representation (KR: Davis et al., 1993) is an area of artificial intelligence research that focuses on the epistemological and ontological issues of describing the diversity of the world (R3), paying particular attention to the effects on processes such as reuse (R6) and inference (R1, R5), typically with respect to properties of expressiveness, validity, and efficiency (R4). While there is work applying KR techniques to a variety of problems, such as planning (temporally and spatially), decision-making, and reasoning under uncertainty, the focus is typically not on diverse learning methods (R2) for numerous, novel tasks in complex domains, nor is it frequent for knowledge bases or inference methods to scale (R4) to knowledge stores required for long-living, generally intelligent agents in dynamic environments (Crawford & Kuipers, 1991; Lenat, 1995).

3. Research Approach

To set the stage for discussing our prior and future work relating to the study of the episodic and semantic memory mechanisms in a general cognitive architecture, we discuss here our research approach: we begin with a description of research methodology, exemplifying the process with a description and rationale of our initial research direction; and then proceed with a presentation of Soar, the general cognitive architecture in which we implement and evaluate candidate memory mechanisms, including technical detail relevant to our proposed work and reasons for adopting it as our research platform.

3.1 Research Methodology

As depicted in Figure 1, we adopt an iterative approach to memory system research, decomposed into a directed cycle of *analyzing* prior work, developing *theoretical commitments*, exploring the space of *mechanism*, and performing *evaluation*. In each sub-section below, we more precisely define these phases and, in order to exemplify the process, discuss how this breakdown applied to our prior work.

Analysis

We begin by surveying the significant body of prior work related to memory systems research, including theoretical and experimental psychological/cognitive scientific literature, task/environment decompositions, and relevant computational approaches, such as implemented cognitive models and information retrieval techniques. The results of this phase are two-fold: (1) specific ways in which our work-to-date does not satisfy the memory system requirements for generally intelligent agents and (2) potentially fruitful paths for enhancing our models via new theoretical commitments, as well as apposite means for evaluation and comparison.

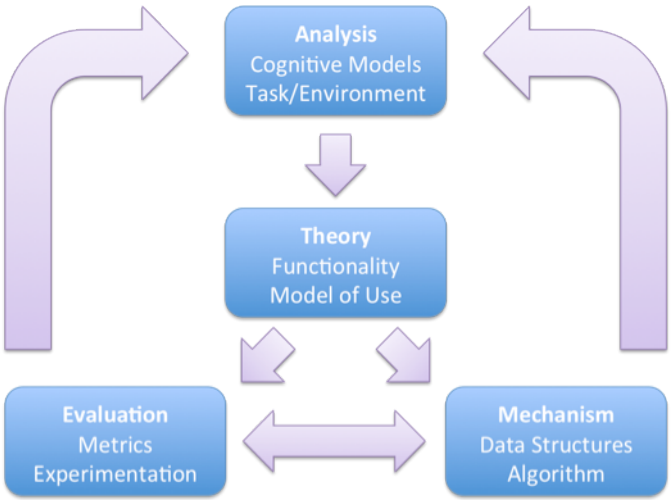

Figure 1. Memory System Research Methodology

Initial Direction. Inherent in memory system research are the clashing goals of *functionality* and *efficiency*. As previously discussed in the context of memory system requirements, we seek on one hand to develop mechanisms that endow agents with flexible (R5) and expressive (R3) access to the full breadth and depth of their experience (R2): ideally, an agent's cognitive architecture would continually soak every last drop of knowledge from each and every experience (R1), ensuring future improvement on all known tasks and the utmost of generalization (R6) and transfer to a wide variety of possible novel problems.

Concurrently, however, we must ensure that computationally bounded agents making use of these memory systems are not hindered in their decision-making and ability to act in the world (R4). At each moment in time, the potential value of seeking and applying prior experience changes, relative to the agent's current situation and goals. Thus an ideal cognitive architecture, one that minimizes computational cost and maximizes the expected value of retrieved knowledge, would continually seek to isolate, maintain, and extract only that prior knowledge that is most directly relevant to the agent's current and expected future context.

As discussed previously (see Table 1), a great deal of cognitive modeling and cognitive architecture work focuses on understanding functionality, while paying very little attention to the degree to which proposed theory and mechanisms will scale to large bodies of knowledge and long agent lifetimes. Thus, as our inaugural focus, we sought to understand the computational challenges involved in extending generally intelligent agents with *basic*, *task-independent* episodic and semantic functionality that scaled with large bodies of knowledge.

As we expatiate below, in order to make tractable progress along this path, our initial intent was to make limited explicit assumptions as to regularities of agent experience, thereby minimizing contributions to R2, while making very constraining assumptions as to mechanism usage and the upper bounds and growth characteristics of computing resources, thereby providing generalize-able contributions to R4.

Theory

Once we have analyzed pertinent sources, we synthesize by posing specific theoretical explanations to explore. Theoretical commitments include both the *functionality* supported by the memory mechanism and a *model of use*.

The functionality of a memory mechanism can be formulated as committing to a set of design decisions within the space of possible computational memory models, forming functional requirements along the dimensions of *Encoding*, *Storage*, and *Retrieval* (see Appendix A).

A model of use entails commitments to expected regularities in agent experience, as dictated by properties of task and environment, which relates to likely patterns of perceived information; regularities of memory mechanism usage, which speaks to expected frequency, composition, and complexity of knowledge queries and manipulations; and constraints on computational time and space resources.

Initial Direction. As discussed in the analysis section, our initial goal was to efficiently scale basically functional mechanisms to large bodies of knowledge over long agent lifetimes. The details and justification of our design decisions, requiring a good amount of additional explanatory build-up, are discussed later for episodic (Section 4) and semantic (Section 5) mechanisms, but our model of use commitments are expatiated here.

We committed to the following two key theoretical assumptions regarding the regularities of agent knowledge and information agents would experience; as discussed below, these presumptions are neither new nor unattested, but have been incorporated in systems research, design, and implementation, including Soar, for decades:

T1. For learning to be tractable across numerous, complex tasks and environments, we assumed agents must re-use mental structures in their internal representation of knowledge: for even complex agents, the number of *distinct* knowledge structures across a long period of time is likely to be much smaller than the *total* number of structures (|distinct structures| $\ll$ |all structures|).

    This observation has been validated in the production matching literature and is key to efficient *sharing*, a technique that reduces match *effort* across productions by reusing data structures to cache intermediate results of common conditions. *For* instance, Doorenbos (1995) demonstrated how sharing can have a dramatic effect on match effort when learning large amounts of long-term procedural knowledge. He showed empirical evidence across seven rule testbed systems that while sharing reduced the number of tokens, data structures representing information about partial rule matching, in the initial, often hand-coded, rule set by a factor of only about 1.5 to 10, this computational saving increased significantly as each system learned new rules, resulting in token reduction ranging from 450-1000 when each system had learned more than 100,000 productions.

T2. Change in the world tends to be local and infrequent: for even complex agents, the degree of structural change experienced when the agent refreshes current perception is likely to be much smaller than the size of the complete agent state (|structural changes| $\ll$ |state structures|).

    This observation is key to optimizations over re-computation of intermediate results in production matching systems. For instance, the Rete algorithm (Forgy, 1982) exploits this assumption by caching all intermediate matching computation, including conjunctive join relations, whereas TREAT (Miranker, 1987) caches only constant matches and re-computes relations on-demand. Depending upon properties of the experimental data set, these design decisions lead to significant trade-offs in production load and match computation (Nayak et al., 1988).

With respect to constraints on computing resources, memory, though not unlimited, is generally considered cheap and plentiful, while time is expensive and limited, and thus our goal was to minimize processing time, possibly at the cost of memory.

In the limit, we proposed efficient support for memory mechanism operations as sub-linear in size of the memory store, which would presumably grow proportionally with the lifetime of the agent, while remaining at most linear in memory consumption.

As an absolute time requirement, we looked to prior experience with real-time agents, which suggested that the Soar cognitive architecture must execute its primitive cycle in 50-100ms to maintain sufficient reactivity in real-world tasks. As a direct consequence, for the agent to maintain online, incremental learning (R1) of perceptual information in complex, dynamic environments, the encoding operation had to occur in fewer than 100ms. However, operations involved in the storage and retrieval of internal knowledge could be spread across multiple cycles, but would likely decline in utility (R5) with increased computing delay.

We reasoned that this relatively unbiased, baseline understanding of efficiency would inform all later explorations, providing a sense of how absolute computation space and time of memory mechanism operations scale with very large knowledge bases on commodity hardware.

Mechanism

Once a set of theoretical commitments is posed, we can begin enumerating and exploring the space of candidate memory mechanisms. This process includes identifying and developing all data structures and algorithms necessary to fully implement the functional requirements, as well as complexity analysis to understand how these processes will scale, while adhering to computational constraints and optimizing for regularities in experiential and usage patterns, as defined in the model of use.

If during this development process we identify a component of theory that is not likely to be satisfied, such as implementing a search of arbitrary run-time data in constant time, we will need to reconsider and reformulate these commitments.

Initial Direction. Details of our initial episodic (Section 4) and semantic (Section 5) mechanisms are detailed later in this document.

Evaluation

Given a set of theoretical commitments, we can also develop evaluation metrics and experiments to measure and analyze the degree to which the theory is satisfied, and the acceptability of any tradeoffs involved, as framed by the model of use.

Specific evaluation metrics include computational resource usage, such as maximum/ average processing time and memory; task-specific performance on a variety of problems, including the degree to which experimental variants improve, or outright enable,

performance; and, where applicable, the degree to which mechanism operation correlates with the predicted outputs of modeled cognitive processes.

If the theory is too vague, such as to not lend to crisp measurement and comparison, we must revise these commitments. Likewise, a mechanism implementation that is not properly instrumented requires revision. Given both theoretical commitments and a set of candidate mechanisms, we can properly evaluate memory systems, leading to subsequent analysis, wherein we compare to other models and implementations and resolve to iteratively revise and enhance the theory and/or mechanism, etc.

Initial Direction. Details of our initial episodic (Section 4) and semantic (Section 5) evaluation are detailed later in this document.

Retrospective

Once begun, we argue that this iterative process serves as a practical, yet principled, top-down method of exploring the infinite space of computational memory mechanisms (Appendix A) while incrementally progressing towards greater coverage of memory system requirements for generally intelligent agents (Table 1).

The resulting contributions, as discussed above, may include data structures, algorithms, and complexity analysis, which not only empower others to reproduce results, but also extend work in a variety of systems; demonstrations of high-level cognitive capabilities, which increase overall agent proficiency and robustness within a variety of tasks and domains; and evaluation benchmarks, which facilitate comparison and progress amongst diverse systems and research avenues.

3.2 Research Architecture

We now shift our discussion from research methodology to Soar (Laird, 2008), the general cognitive architecture in which we are implementing and evaluating candidate memory mechanisms. We begin with an introductory discussion of why we plan to study memory systems in context of an architecture, as opposed to in isolation. We then transition to our reasons for choosing Soar as the research platform, drawing on the characteristics of generally intelligent agents, including the tasks with which they contend and environments in which they are embedded, as well as the resulting requirements imposed upon their memory systems (see Section 2) that we hope to investigate and ultimately satisfy. Finally, we relate the technical details of Soar relevant to our research goals.

The Case for Architecture

A great deal of computational memory model research has focused on particular mechanisms, primarily in isolation of agent architecture, goals, and behavior. For instance, John Anderson's rational analysis work (Anderson, 1990; Anderson, 1991; Schooler & Anderson, 1997; Anderson & Matessa, 1998) posits that since the human cognitive architecture will optimize agent behavior for task performance, the best method for understanding human cognitive behavior lies in environmental and task analysis, rather than attempting to analyze specific human problem-solving methods.

While useful in analyzing, developing, and evaluating individual memory mechanisms, this isolated approach considers only the structure, regularities, and interactions with respect to properties of *environment* and *task*, ignoring those of the *agent* (Simon, 1991). As a consequence, it is difficult to synthesize and apply these specifications within the context of a generally intelligent agent and leaves many questions unanswered, such as what components need be instantiated in a complete architecture; and once endowed with knowledge, how do they connect, interact, and integrate, especially in context of learning, dynamic environments, and extended lifetimes, to form intelligent behavior.

Thus, we plan to study whether there are computational and functional constraints and capabilities that emerge from integrating long-term declarative mechanisms in a general agent architecture and engaging memory-endowed agents in multiple, complex tasks. We now consider which particular architecture is most suitable in context of our research goals.

Architecture Selection

We consider the following two metrics when comparing architectures for evaluating memory mechanisms for generally intelligent agents. First, the architecture must *not*, a priori, theoretically invalidate a requirement for memory systems (see Section 2). If this were the case, no candidate memory mechanism could possibly satisfy the requirement. As an example, consider an architecture in which agent state was represented as a fixed-length binary buffer; given such a constraint, no memory mechanism could possibly satisfy R3, that of supporting diverse knowledge representation. Second, we consider the degree to which the architecture can support experimentation across the dimensions (C1-C7) characterizing generally intelligent agents, the tasks with which they contend, and the environments in which they are embedded. Given these metrics, we discuss below the reasons for which we chose the Soar cognitive architecture (Laird, 2008).

Task-independent (R6), efficient access to (R4)/effective use of (R5) diverse knowledge (R3)

Especially relevant for this work is Soar's considerable history of efficiently representing and bringing to bear large bodies of knowledge to solve diverse problems using a variety of methods (Doorenbos, 1995; Laird & Rosenbloom, 1996). This simultaneous focus on efficiency and generality uniquely distinguishes Soar from other agent architectures.

At one extreme, some systems boast impressive generality and applicability, as exemplified by the impressive number and variety of psychological phenomena captured by ACT-R models (Anderson et al., 2004). However, these systems rarely consider the computational implications of scaling their theoretical commitments to the large knowledge bases accumulated over long agent lifetimes. For instance, the ACT-R declarative memory module has been shown not to scale to large stores of declarative knowledge (Douglass et al., 2009; Douglass & Myers, 2010). However, because ACT-R models primarily focus on explaining details of small time-scale, psychological experiments, research progress is typically unimpeded and, until recently, little attention has been expended into researching the degree to which the details of the ACT-R theory can scale to the conditions with which generally intelligent agents grapple.

Diametrically opposed are systems that demonstrate competency on a constrained set of tasks and knowledge representation, while not contending with the overwhelming quantity and diversity of challenges with which generally intelligent, autonomous agents contend in complex domains. For instance, Cyc (Lenat, 1995) demonstrates comprehensive data integration and hybrid inference capabilities over unparalleled size and scope of knowledge, but suffers debilitating inefficiencies when faced with even simple real-time planning and problem solving tasks. Additionally, some systems are limited in their knowledge representation, such as semantic networks in MicroPsi (Bach, 2003), fixed-sized buffers in ACT-R (Anderson et al., 2004), and productions in classic Soar (Laird & Rosenbloom, 1996), and are therefore likely to demonstrate difficulties effectively extending to the complexities inherent in complex environments and tasks.

In contrast, version 9 of Soar (Laird, 2008), the current iteration of the architecture, implements a fully relational, symbolic representation (R3) across a variety of task-independent (R6) memory systems that have been shown to scale (R4) to large stores of knowledge over long agent lifetimes (Doorenbos, 1995; Derbinsky & Laird, 2009; Derbinsky et al., 2010). For instance, the Soar 9 procedural memory mechanism, inherited from classic Soar, supported the TacAir-Soar system (Jones et al., 1999), which was composed of thousands of symbolic production rules and managed 722 scheduled flights of fixed-wing aircraft flew during an operational training exercise that ran for 48 continuous hours.

Building on Soar's existing implementation, we can tractably study and evaluate memory mechanisms with large data sets, such as the WordNet lexicon (Miller, 1995), and experiences from ecologically valid, complete agents, such as robots in real and simulated environments, over weeks, months, or even years of cognitive real-time (Laird & Derbinsky, 2009). At this time scale, we can also begin to study how memory systems interact with other cognitive mechanisms (R5), such as emotional appraisals and procedural learning, including their relative strengths and limitations in situations and tasks approaching those of long-lived humans. Additionally, Soar's generality of knowledge representation and reasoning allows us to not only study a large spectrum of tasks, but to also extrapolate and apply our results to other systems and architectures.

Support for Complex Tasks (C4), Environments (C1-C3), and Interactions (C5)

The degree to which the agent architecture can support systems integration and deployment directly affects the ability to evaluate memory mechanisms on a variety of tasks in complex environments. While primarily an issue of engineering, this is a valid practical consideration for effectively pursuing our proposed work.

Soar supports a variety of programming languages (such as C++, Java, and Python) on all major operating systems (including Windows, Mac OS, Linux, and iOS) and has been interfaced in diverse execution environments, including RL-Glue (Tanner & White, 2009); game systems, such as ORTS (Wintermute et al., 2007), Infinite Mario (Mohan and Laird, 2010), and Quake (Laird, 2001); and robotics simulation and hardware platforms (Laird, 2009).

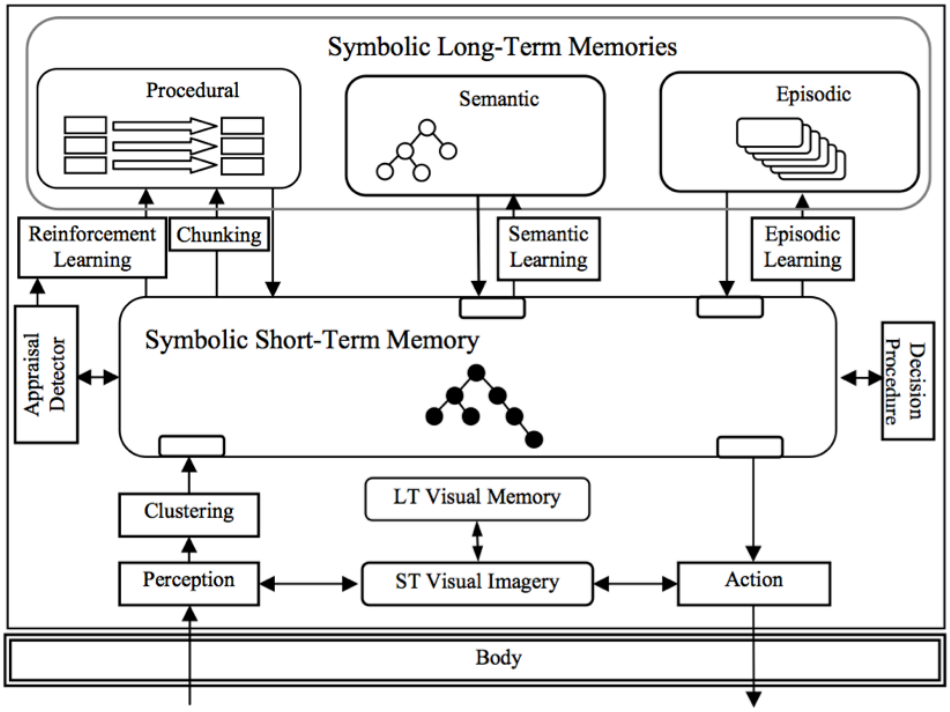

Figure 2. Soar 9

This capability sharply contrasts agent architectures that are limited by language/system (such as the Lisp requirement of ACT-R: Anderson et al., 2004), partial implementions (such as Clarion: Sun, 2006), or theoretically posed frameworks (such as LIDA: Franklin & Patterson, 2006; and Icarus: Stracuzzi et al., 2009).

The Soar Cognitive Architecture

We now discuss the technical and theoretical details of Soar relevant to our research of memory mechanisms. Throughout this discussion, we make reference to how foundations are laid within the architecture for satisfying the requirements (R1-R6) of memory systems for generally intelligent agents.

Figure 2 shows the structure of Soar (Laird, 2008): symbolic short-term memory holds the agent's assessment of the current situation, derived from perception and via retrieval of knowledge from its symbolic long-term memory systems; action in an environment occurs through creation of motor commands in a buffer in short-term memory; and the fixed decision procedure selects operators and detects impasses.

At an abstract level, Soar's symbolic representation of present state and topological integration of long-term declarative and procedural knowledge is not unique: this archetypal arrangement is similar to that of many other cognitive architectures, especially those that are either designed to model human behavior, such as ACT-R (Anderson et al., 2004), LIDA (Franklin & Patterson, 2006), and Clarion (Sun, 2006), or are inspired by human behavior, such as Icarus (Stracuzzi et al., 2009). Although there are many commonalities in these systems, there are also significant differences in design decisions of and theoretical commitments to knowledge representation, memory system functionality,

and learning. For instance, while Soar's short-term memory representation, as detailed below, is manifested as an arbitrarily complex symbolic graph, ACT-R maintains a fixed set of constant-sized, symbolic buffers and Clarion integrates symbolic and connectionist representations. These structural departures often reflect differences in research goals and phenomena of study.

Although Soar has many additional learning mechanisms and processes, the most relevant aspect of its design to the study and evaluation of the episodic and semantic long-term, declarative memory systems is the functionality and structure of its symbolic short-term knowledge store, termed "working memory." As previously stated, Soar's working memory contains an agent's dynamic internal state, including perceptual data, situational awareness, current goals and intentions, and motor commands. Functionally, working memory represents arbitrary and novel combinations and compositions of symbols (R3). It also functions as a common substrate upon which the cognitive architecture applies agent control knowledge, represented as production rules, to symbolically represent, reason with, and retrieve long-term knowledge (Derbinsky & Laird, 2010). Soar's working memory is implemented as a directed, connected graph of working memory elements (WMEs). Each WME is a symbolic triple, consisting of an *identifier* (a node or vertex), *attribute* (a link or edge label), and *value* (a node or terminal value), and working memory as a whole is defined as a set, such that no two WMEs can have the same identifier, attribute, and value.

To balance environmental reactivity (R1) with rich access to large bodies of knowledge (R5) while contending with arbitrarily complex problems (R6), Soar adopts the *problem space hypothesis* (Newell & Simon, 1972) as a core theoretical commitment. According to this conjecture, problem solving in a task is defined as generalized search in a *problem space*. A problem space is composed of a set of *states* and a set of *operators*, which transform one state to the next. At each state in the problem space, knowledge is used to evaluate the operators that are available in the current state and determine the next-best operator. As the agent contends with multiple tasks and problems, it is possible that problem solving may extend over multiple problem spaces.

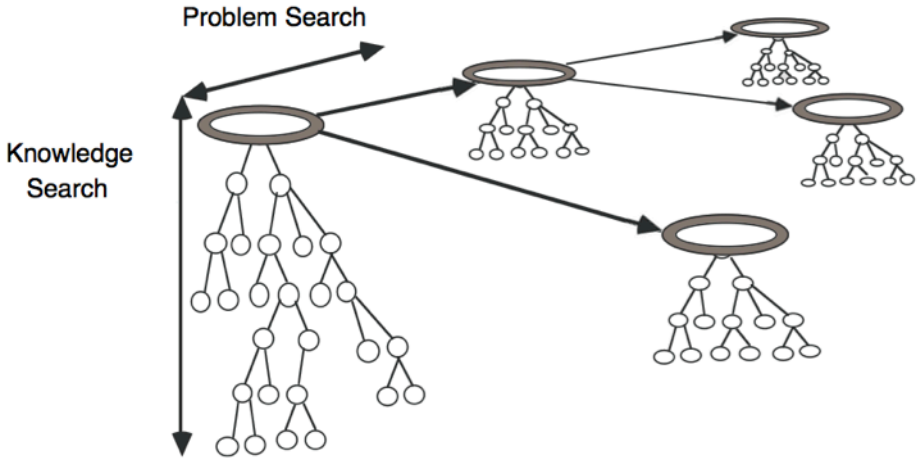

Figure 3. Search Spaces in Problem Solving

The process of extracting directly available knowledge from a knowledge base and making it available to the generative search process at the problem space level is called *knowledge search* (Newell 1990; Strosnider & Paul, 1994). This process is depicted in Figure 3, where states are represented as shaded oval rings, operators as edges connecting these rings, and knowledge base(s) as the graphical trees stemming from each ring. Note that while the depth of the knowledge base below each state is limited, illustrating a finite quantity of immediate information, the plane upon which states are situated is potentially infinite in scope, demonstrating the possibility of an expansive problem space, resulting from the generative process of transforming states through operator application. Note also that search through problem space is not guaranteed to exhibit cyclicity: search progress is made as operators transform state and, given environmental dynamics and the resulting changing availability of operators within problem space(s), there is no guarantee of the ability to reach a prior state directly or indirectly via application of one or more operators.

There are many types of knowledge search that are computationally unbounded, such as generalized logical inference and structural graph matching. Soar, however, firmly commits to the concept of *bounded rationality* (Simon, 1991), which contends that rationality of agent decision making is limited by information availability, mechanisms in the cognitive architecture, and finite time, relating to dynamic pressures of the environment. Consequently, Soar distinguishes generative search in problem space from bounded search of immediate knowledge in long-term memories, and thus any unbounded computation over knowledge is not incorporated within architectural retrieval mechanisms, but must be instead formulated as deliberate problem search, such that task-dependent agent control knowledge can be brought to bear to prune the search space while the agent maintains continual reactivity with its complex, dynamic environment.

A related commitment of the Soar cognitive architecture is the strict division of task-independent architectural mechanisms from task-dependent agent knowledge. This separation contains a strong analogy within the computer architecture world, in which designers strive to optimize system utility, as measured by such factors as production cost, potential for broad utilization, energy consumption, and application speed, by shifting the balance between those mechanisms *fixed* in highly efficient and potentially parallel hardware and the range of functionality *supported* by the less efficient, user-accessible software instruction set. When applied to knowledge search, this optimization process appeals to the difficulties, well studied in the database and information retrieval communities, involved in maintaining a computationally efficient search over run-time domain knowledge (Strosnider & Paul, 1994). The resulting intuition, well studied in computational theory, is that the greater the degree to which the complexity of a generalized search process can be constrained, the greater the potential for efficient implementation via highly optimized data structures and algorithms. Soar applies this tenet by efficiently encoding in fixed mechanisms, such as memory systems, the task-independent domain knowledge, such as environmental regularities at time scales that approach or exceed the life of the agent, which may improve the quality and performance of dissociative knowledge retrieval processes (R6), while maintaining universal computation at the level of problem search (Derbinsky & Laird, 2010), where experience and task-dependent agent knowledge may be brought to bear incrementally to prune complex task solving (Laird & Wray, 2010).

4. Prior Work: Episodic Memory

We now begin a review of our prior work, in which we sought to understand the computational challenges involved in extending generally intelligent agents with basic, task-independent memory mechanisms that scaled with large bodies of knowledge. We discuss our work relating to episodic memory here, and continue to semantic memory in Section 5. Then, in Section 6, we present our proposal for future work, in which we explore research questions relating to expanded functionality in both of these memory systems.

This section begins with a motivational description of the episodic memory system, including a small amount psychological background; proceeds with a discussion of related work; continues to a functional specification of a computational memory model; describes novel data structures and algorithms we developed to efficiently implement this memory system; and concludes with evaluation.

4.1 Motivation

As first described by Tulving, episodic memory captures historical knowledge contextualized in agent experience (Tulving, 1983). 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:

E1. Architectural: episodic retrievals are available for all tasks and the process of storing memories does not compete with knowledge-based reasoning (related to R6, the task-independent requirement of memory systems for generally intelligent agents).
E2. Automatic: episodic memories are stored without deliberation. Reasoning can only indirectly influence episodic storage, such as through deliberate rehearsal (related to R1, the requirement for incremental, online learning).
E3. Autonoetic: retrieved episodic memories are distinguished from current sensing.
E4. Autobiographical: retrieved episodes are represented in the context in which they were originally experienced.
E5. Temporally indexed: retrieved episodes include meta-data providing temporal context with respect to other episodes.

Nuxoll has done some work (2007) to postulate as to and demonstrate some of the functional roles episodic memory may serve in context of a general cognitive architecture, such as facilitating virtual sensing, action modeling, and retroactive learning.

In context of the memory requirements for generally intelligent agents, incorporating episodic memory in a cognitive architecture contributes significantly to R2, the support of diverse, comprehensive learning, by incrementally (R1) providing an agent rich access to a contextualized, temporally indexed, internal store of its prior experience. However, scaling

(R4) effective access (R5) to this knowledge in a task-independent fashion (R6) over long agent lifetimes poses a significant challenge.

4.2 Functional Specification

To develop a baseline understanding of the efficiency challenges involved in affording agents the functional benefits of episodic memory over long lifetimes, we adopted the high-level design decisions described by Nuxoll and Laird (2007). Here we describe those generally, followed by a mapping onto Soar, as depicted in Figure 4.

The *episodic storage* process automatically encodes a subset of agent state (E1, E2) at regular intervals (E2) and temporally indexes this knowledge within the episodic store (E5). This process does not modify or generalize stored episodic knowledge, and thus episodic knowledge grows strictly monotonically, faithfully capturing the full extent of agent experience. To retrieve episodic memories, the agent deliberately constructs a cue (E1), partially specifying relevant contextual features within the episode. The *cue matching* process selects a single best match from the episodic store, defined as the most recent qualitative nearest-neighbor, and the retrieved episode is fully *reconstructed* (E4) in a special buffer (E3), such that the agent can reason about this knowledge without confusing current sensing and past experience.

In Soar, as illustrated in Figure 4, agent state is represented in its Working Memory as a directed, connected graph. Architecturally specified subsets of this graph form the episode to be encoded, the cue to be matched, and the reconstructed episode to be reasoned over. The requirement to support this arbitrary graph structure had significant implications for the underlying mechanism implementation, as discussed later. A simpler representation, such as a vector or propositional representation, would have made it possible to develop simple and very fast implementations of episodic memory, but at significant cost in expressability and generality (R3). By adopting this representational requirement, however, we developed an underlying implementation that is independent of other details of Soar, thereby generalizing to other architectures whose dynamic data can be fully captured by a graph-based representation.

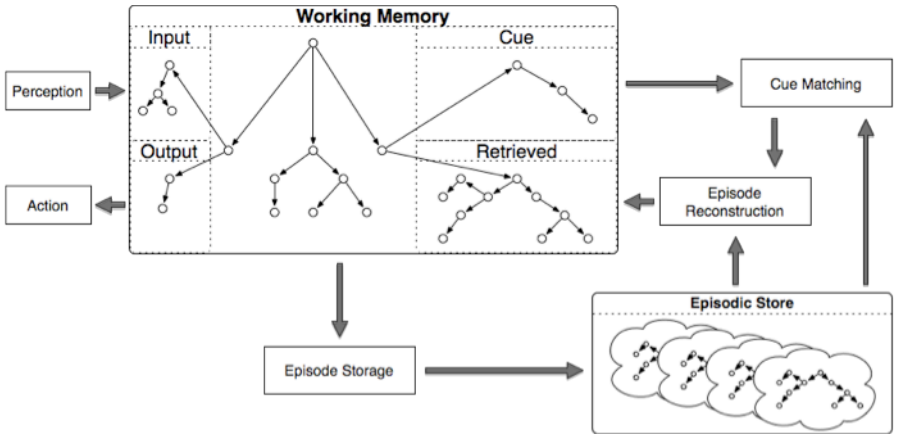

Figure 4. Integration of Episodic Memory in Soar

4.3 Related Work

To contextualize our work, we first discuss the case-based reasoning research related to our functional specification and then evaluate an existing episodic memory system. In both cases we focus on the degree to which the work satisfies the base requirements (R1-R6) for memory systems, as well as those specific to episodic memory (E1-E5).

Related Case-Based Reasoning Work

Episodic memory research is closely related to studies in case-based reasoning (CBR). The goal of CBR is to optimize task performance given a case-base, where each case consists of a problem and its solution (Kolodner, 1992). In CBR systems, however, case structure is typically pre-specified, case-base size is either fixed or grows at a limited rate, and the cases usually do not have any inherent temporal structure. In contrast, an episodic store grows with experience, accumulating snapshots of an agent's experiences over time. An agent endowed with this memory can retrieve relevant episodes to facilitate reasoning and learning based upon prior events.

Efficient nearest-neighbor algorithms have been studied in CBR for qualitative and quantitative retrieval (Lenz & Burkhard, 1996; Stottler et al., 1989; Wess et al., 1994). The underlying algorithms and data structures supporting these algorithms, however, typically depend upon a relatively small and/or static number of case/cue dimensions, and do not take advantage of the temporal structure inherent to episodic memories.

Considerable work has been expended to explore heuristic methods that exchange reduced competency for increased retrieval efficiency (Smyth & Cunningham, 1996), including refined indexing (Daengdej et al., 1996; Fox & Leake, 1995), storage reduction (Wilson & Martinez, 2000), and case deletion (Patterson et al., 2003). Many researchers achieve gains through a two-stage cue matching process that initially considers surface similarity, followed by structural evaluation, such as Forbus et al. (1995).

The requirement of dealing with time-oriented problems has been acknowledged as a significant challenge within the CBR community (Combi & Shahar, 1997), motivating work on temporal CBR (T-CBR) systems (Patterson et al., 2004), and research on the representation of and reasoning about time-dependent case attributes (Jære et al., 2002), as well as preliminary approaches to temporal case sequences (Ma & Knight, 2003; Sánchez-Marré et al., 2005). However, existing T-CBR work does not deal with accumulating an episodic store, nor does it take advantage of temporal structure for efficient implementations.

EM: A Generic Memory Module for Events

Some systems and architectures make conjectures about long-term stores of prior experience, such as CLARION (Sun, 2003; Sun, 2006), and temporal beliefs, such as ICARUS (Stracuzzi et al., 2009). However, it is relatively rare to find a fully implemented, task-independent, episodic memory system. Here we analyze one such existing system.

EM (Tecuci & Porter, 2007) is a generic store to support episodic memory functionality in a variety of systems, including planning, classification, and goal recognition. EM is an

external component with an API, wherein host systems must implement a thin interface layer. The term "episode" in EM defines a sequence of actions with a common goal and is represented as a triple: context ("general setting" of the episode), content (ordered set of the events that make up the episode), and outcome (a domain/task-specific evaluation of the result of the episode). Though meaningful in systems like planners, this representational constraint is inappropriate for generally intelligent agents, as it may be difficult to pre-define action sequences and outcome evaluation functions for long-living agents that must contend with multiple, possibly novel, tasks.

EM queries are partially defined episodes and a single evaluation dimension. EM utilizes a two-stage evaluation scheme, whereby a constant number (5) of potential matches are found and then compared using a relatively expensive semantic matcher. While Tecuci and Porter have shown results for learning in short (250 episode), single-task domains, it is unclear whether the underlying algorithms and data structures will scale to agents with many orders of magnitude more episodes.

4.4 Efficient Implementation

We have developed and analytically described novel data structures and algorithms for efficient, task-independent, graph-based episodic memory systems that satisfies the functional requirements above (Derbinsky & Laird, 2009). We decompose discussion of this work into three episodic operations: *storage*, *cue matching*, and *reconstruction*.

Episodic Storage

Given an agent whose present state is represented as a connected, directed graph, we define episodic storage as the process of encoding, at a given point in time, all information necessary to recreate that state at a later time. By automatically associating a unique temporal identifier with this captured data (E5), a storage process supports an architectural (E1), automatic (E2), and autobiographical (E4) episodic memory system.

To satisfy this definition, a naïve storage mechanism simply records all nodes and edges that comprise agent state during each episode. While sufficient, this approach has unsatisfying computational resource requirements: encoding time and space, per episode, linear in the size of the state graph. Prior work by Nuxoll and Laird (2007), however, exploits two key observations to improve this computational profile.

First, for learning to be tractable across numerous, complex tasks and environments, agents must *re-use* mental structures (T1) in their representation of knowledge and thus, for even complex agents, the number of *distinct* structures encoded across a long period of time is likely to be much smaller than the total number of encoded structures ($|\text{distinct structures}| \ll |\text{all structures}|$). Nuxoll and Laird extended this idea to episodic memory by maintaining and referencing a global data structure, termed the *Working Memory Tree*, of all distinct structures that had ever been encoded by the storage mechanism. With this insight, episode storage can be conceptually understood as generating bags of pointers to distinct structures in an experiential, incrementally and monotonically growing, temporally global data structure, which benefits the overall implementation by compressing the

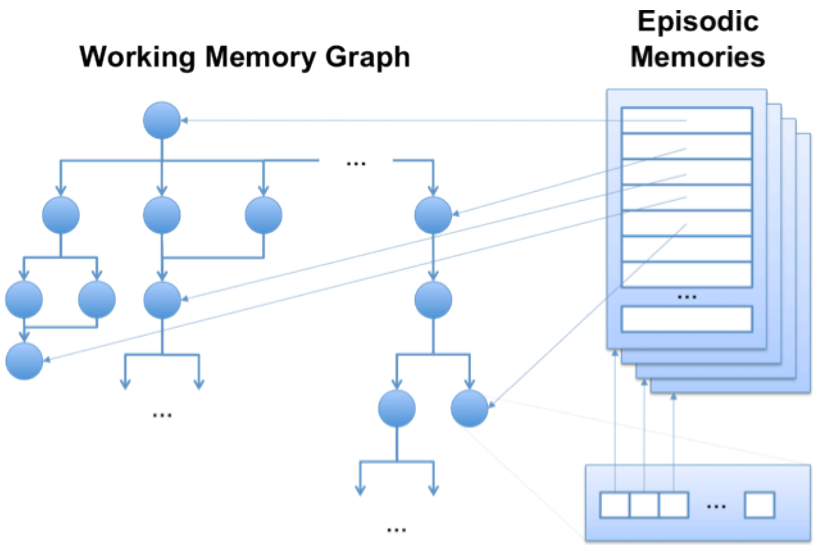

Figure 5. Exploiting Structural Re-Use in Episodic Storage

representation of repeated structures. Whereas Nuxoll and Laird compacted agent state to a tree with distinct edge labels, we extended this work to support a fully relational representation, termed the *Working Memory Graph* (Figure 5).

The second observation is that change in the world tends to be local and infrequent (T2): for even complex agents, the number of structural *changes* between episodes is likely to be much smaller than the size of the complete agent state (|structural changes| $\ll$ |episode structures|). Given this insight, Nuxoll and Laird converted their explicit episodic representation (Figure 5) to an implicit representation (Figure 6), associating with each distinct global data structure node a list of temporal range(s) during which the associated mental structure was present in agent state. This transformation reduces computational complexity to space and time linear in structural *change*, as opposed to absolute number. We extended this work to efficiently support the *Working Memory Graph* (Figure 6).

Cue Matching

Given an episodic store, as illustrated in Figure 6, and a cue, represented by an acyclic graph partially specifying relevant contextual features, we define cue matching as identifying the most temporally recent episode that shares the greatest number of symbolic features structurally in common with the cue. This operation functionally supports task-independent (E1) access to episodic knowledge.

To satisfy this definition, a naïve cue matching mechanism performs a graph-match between the cue and each episode in the store, beginning with the most recent and concluding once a perfect match is found or once all episodes are considered and ranked. While sufficient, this approach has unsatisfying computational resource requirements: potentially exponential time, with respect to average episode size, for each graph-match and a linear growth, with respect to the number of episodes in the store, in the number of evaluations. With the following three strategies, we have built on prior work to improve the tractability of both of the aforementioned issues.

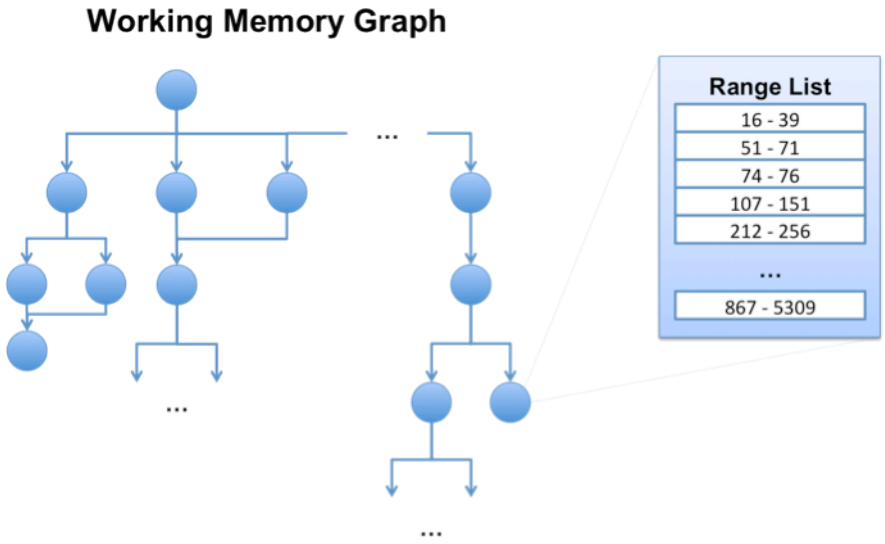

Figure 6. Exploiting Structural Stability in Episodic Storage

The first optimization is to only consider episodes for evaluation that contain at least one feature in common with the cue. We can easily obtain this candidate list by cross-referencing the cue with the working memory graph and incrementally merging pertinent lists of temporal ranges.

An important extension to this technique is to recognize that the degree to which candidate episodes match a cue will *only* change at the endpoint of a temporal range. Termed *Interval Search* (Nuxoll & Laird, 2007), this optimization minimizes candidate episode evaluation. For example, consider the feature expanded in Figure 6. Were this node a cue feature, we could potentially avoid evaluating episodes 868 through 5308 (a savings of over 4000 evaluations!), as they are implicitly known, without exception, to all contain this feature.

The final optimization is to minimize the frequency of potentially combinatorial graph-match evaluations by implementing a two-stage matching policy, a common technique (for example: Forbus et al., 1995). The key observation here is that a candidate episode only has the potential to *structurally* match a cue if it contains all *surface* cue features, where a surface feature is defined as a distinct, directed path from root to leaf without relational consideration. Given the working memory graph, we developed a mapping from surface feature matching to satisfaction of a set of disjunctive normal form (DNF) boolean equations. Furthermore, we developed and implemented a novel discrimination network, termed the *DNF Graph*, which efficiently and incrementally maintains satisfaction of a set of DNF formulas during each endpoint of Interval Search (Derbinsky & Laird, 2009).

The consequent of these optimizations is a cue-matching algorithm that is guaranteed to satisfy our functional specification, but which attempts to minimize temporal linear scanning and combinatorial structural evaluation by processing only changes between candidate episodes.

Episode Reconstruction

Given an episodic store, as illustrated in Figure 6, and a valid temporal id of an episode, we define reconstruction as the process of faithfully reproducing all working memory structures that originally composed that episode. This process is tantamount to an interval intersection query: collect all working memory graph structures that started before and ended after a particular point in time.

To efficiently support this operation, we implemented a Relational Interval Tree (Kriegel et al., 2000), which is a mapping of the interval tree data structure onto Relational Database Management System (RDBMS) B+-tree indexes and SQL queries. As with the standard interval tree, intersection queries execute in time logarithmic with the number of stored intervals. Because intervals represent working memory structure changes, this growth characteristic is sufficiently sub-linear with respect to the number of episodic memories.

4.5 Evaluation

We implemented these algorithms and data structures as the episodic memory module in the Soar cognitive architecture. Our initial empirical evaluation served to validate performance models of the primitive memory operations (see Appendix B), as well as to lend evidence that our memory system implementation, while growing intractably in the worst-case (linear in the number of episodes), performs within reasonable limits in practice.

Our empirical evaluation domain, a competitive tile-based game, was particularly stressful with respect to the size of agent state (over 2500 features), as well as the degree of dynamics per episode (70-90% of perceptual features change each episode). In this game, we have demonstrated sub-100msec. retrievals for a variety of cues; sub-10msec. storage to capture about 1-2 KB/episode; and sub-50msec. reconstruction on commodity hardware after 1 million episodes, which amounts to days or weeks of cognitive real-time.

In addition to providing baseline efficiency results, our mechanism, embedded within a cognitive architecture, supports a variety of independent research, including the possibility of an agent *learning* control over internal memory mechanisms (Gorski & Laird, 2009) and situations in which episodic memory, in conjunction with other components of a general cognitive architecture, serves as a valuable repository of action modeling knowledge (Laird et al., 2010; Xu & Laird, 2010).

5. Prior Work: Semantic Memory

In this section, we continue our discussion of our prior work, focusing now on the semantic memory mechanism. We begin with a motivational description of the memory mechanism; continue to a functional specification of a computational memory model, relating to the ACT-R declarative memory module; describe novel data structures and algorithms we developed to efficiently implement this memory system; and conclude with evaluation.

5.1 Motivation

The functional purpose of a semantic memory is to efficiently retrieve declarative facts about mental concepts that have been experienced by an agent, independent of the context in which they were originally learned (Tulving, 1983). The functional benefit of this memory is premised 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.

In context of the memory requirements for generally intelligent agents, incorporating semantic memory in a cognitive architecture contributes significantly to R2, the support of diverse, comprehensive learning, by providing an agent rich access to general, non-contextualized information about the world in which it is embedded. However, as with episodic memory, scaling (R4) effective access (R5) to this knowledge in a task-independent fashion (R6) over long agent lifetimes poses a significant challenge.

5.2 Functional Specification

Arguably the most prolific exemplar of semantic memory in the cognitive architecture research literature is the declarative memory (DM) module in ACT-R (Anderson et al., 2004). In the ACT-R DM, semantic knowledge is represented as a set of *chunks*, which are collections of labeled *slots* that have symbolic *values*. To retrieve declarative knowledge, a production rule issues a request to the DM by populating the declarative buffer with positive and negative slot-value pairs. These pairs are interpreted as hard constraints that either must be met (positive) or must not be met (negative). Given this request, the module searches the store for matching candidate chunks. If any are found, the DM module, given default parameter settings, indicates a successful retrieval, selects randomly amongst the candidate chunks and reconstructs it in the appropriate buffer. The module also supports the use of non-symbolic aspects of current context and prior chunk retrievals as a form of activation to bias selection amongst candidate chunks, functionality that is used in many cognitive models. If no perfect match is found, the default behavior of the DM module is to report a retrieval failure.

We adopted this interface and developed an abstract problem formulation of symbolic semantic memory retrievals (Derbinsky et al., 2010). Appendix C includes details of this formulation, including a mapping to the ACT-R declarative memory module.

5.3 Efficient Implementation

Short-lived agents and typical cognitive models have very modest semantic memory requirements. In these cases, naïve data structures and algorithms, despite inefficiencies, suffice for semantic retrievals. However, prior work (Douglass et al., 2009; Douglass & Myers, 2010) has shown that cognitive models of complex tasks require more substantial amounts of semantic knowledge, such as a large subset of the WordNet lexicon (Miller, 1995), and that the existing ACT-R retrieval mechanism does not scale. Likewise, it is conceivable that over a long lifetime, a learning agent, engaging in multiple, complex tasks, will need to incrementally store and have efficient access to large amounts of semantic knowledge, as exemplified by the Cyc (Lenat, 1995) and SUMO (Niles & Pease, 2001) ontological knowledge bases.

Given the functional specification described above, we have developed a baseline analytical and empirical understanding of the efficiency challenges involved in affording agents the functional benefits of semantic memory over long lifetimes (Derbinsky et al., 2010). To address many of these problems, we developed system-independent methods for efficient retrieval functionality that support most of the ACT-R DM specification. Additionally, while we have not achieved the full functionality of ACT-R activation, we have moved towards that goal by formulating and efficiently supporting a simpler class of activation bias.

Basic Retrievals

In this section we consider the retrieval problem without incorporation of sub-symbolic activation. Appendix C provides an abstract decomposition of the retrieval problem and Appendix D provides detailed implementation detail and analysis.

Without considering sub-symbolic activation, the basic retrieval problem is very similar to a constrained form of a *subset* query on *set-values*, which has been widely studied in database and information retrieval (IR) communities (Terrovitis et al., 2006). In its general form, the worst-case time cost is known to be linear in the sum of the number of candidate elements for each cue augmentation, though clever indexing methods have shown massive average-case improvements in real-world data. We make use of tried-and-true methods in this literature.

Our core indexing structure is an *inverted index* (Zobel & Moffat, 2006), a reverse hash providing efficient access to elements that contain any single augmentation. We also order cue processing based upon inverted index statistics of candidate cardinality, a common technique in database query optimization (Chaudhuri, 1998). Incorporating these lightweight components, retrievals are bound by the number of candidate elements of the most constraining positive cue augmentation.

Preliminary Incorporation of Activation Bias

A major contribution of the ACT-R DM module to cognitive modeling is the sub-symbolic influence of the current context and prior retrievals as a form of activation bias for declarative retrievals (Anderson et al., 2004). This functionality, however, has been shown to come at a significant computational cost that does not scale to large declarative memories (Douglass et al., 2009; Douglass & Myers, 2010).

While we have not achieved the functionality of all aspects of ACT-R's activation scheme, we have made progress by formulating (see Appendix C) and efficiently supporting (see Appendix E) a simpler class of activation bias. This class is constrained by two key requirements: (1) activation values of semantic elements are static unless altered by a relatively infrequent update and (2) the activation update process must be *locally efficient*, intuitively meaning it is bounded in both the number of elements addressed and the amount of computation expended. Given these constraints, minor modifications to the previously described indexing structures and algorithms (see Appendix E) yield efficient support for semantic retrievals.

5.4 Evaluation

We implemented these algorithms and data structures as the semantic memory module in the Soar cognitive architecture and have collected initial empirical evidence that our mechanism, while growing intractably (linear in the number of semantic memories) in the worst-case, performs within reasonable limits in practice. As detailed in Appendix F, we evaluated the system on a scalable, synthetic data set, as well as the entirety of the WordNet 3 lexicon (Miller, 1995). For successful retrievals on data sets scaling to millions of semantic memories, our mechanism achieves sub-millisecond retrievals, which are two orders of magnitude faster than previously reported comparable results (Douglass et al., 2009).

In addition to providing baseline efficiency results, our mechanism, embedded within a production-level cognitive architecture, has already supported independent research into diverse mechanisms for action modeling (Laird et al., 2010) and will likely enhance future work in a variety of areas, including natural language processing (Lonsdale & Rytting, 2001) and instructable autonomous agents (Huffman, 1994).

6. Prior Work: Analysis

In this section, we synthesize and analyze our prior work on episodic (Section 4) and semantic (Section 5) memory systems, including a summary of contributions, pitfalls, and avenues to guide future work. This discussion contributes to the *analysis* phase of our research methodology (see Section 3.1), setting the foundation for Section 7, in which we detail our plans for future work.

As previously discussed (see Section 3.1), the intended contribution of our initial research was to establish a baseline understanding, in the field of cognitive architecture (see Table 1 in the introduction to this document), of how computational episodic and semantic memory models for generally intelligent agents scale with large amounts of knowledge (R4) while providing the minimal functionality needed for real-world tasks. As an initial approach, we made minimal assumptions (T1, T2) about regularities in environmental information, agent knowledge, and mechanism usage, while shouldering requirements involving faithful storage over long lifetimes and exact retrievals given constraining computational resources.

Our work to date demonstrates analytically and empirically that faithful storage (especially for episodic memory) and exact retrievals are intractable in the general case for long-lived agents. However, we have formulated the computational problems involved for episodic (Derbinsky & Laird, 2009) and semantic (Derbinsky et al., 2010) memory models and have implemented mechanisms that demonstrate practical performance in isolated study, as well as support crucial functionality as a part of an integrated cognitive architecture in independent research.

Our experience thus far suggests that future work will need to concurrently walk two paths: improving functionality and investigating diverse methods to combat computational intractability.

6.1 Improving Functionality

While our initial work minimized memory mechanism functionality and concentrated on matters of computational efficiency, it is time to push forth our understanding of requirement R2, functional support for diverse, comprehensive learning, as informed by numerous sources of related research, including the following two shortcomings of our work-to-date.

One inadequacy of our existing work is an insufficient understanding of what information a generally intelligent agent needs to incrementally encode within its experiential stores such as to succeed in numerous tasks while exploring and remaining reactive to its complex, dynamic environment. For instance, in our work with semantic memory (see Section 5), while most semantic knowledge was preloaded from large datasets, such as WordNet (Miller, 1995), we delegated encoding of experiential information to deliberate agent action, as controlled by task-specific procedural knowledge. This approach is incomplete and unsatisfying along several dimensions, including a lack of theory for the

origins of this procedural knowledge, especially for novel tasks. By contrast, we opted to automatically encode an architecturally defined subset of agent state in our work with episodic memory (see Section 4). While this policy for accumulating episodic knowledge has been demonstrated as sufficient to support numerous general cognitive capabilities (Nuxoll, 2007), it is likely unsustainable computationally over long agent lifetimes (Nuxoll et al., 2010), as it imposes a monotonically increasing storage requirement and thus makes efficient retrievals difficult (Laird & Derbinsky, 2009). As described in Section 7, we plan to examine these issues in extensions X1 and X3.

Another inadequacy of our prior work is that we don't understand what sources of task-independent regularities in agent experience can serve to supplement incomplete task-dependent agent knowledge such as to increase the expected utility of memory retrievals in the case of impoverished cues. In both our work with semantic and episodic memory mechanisms, we condition retrievals on structured cues, and, in the case that multiple memories match these cues with equal cardinality, bias the result towards recency of memory access (as opposed, for instance, a random selection from amongst all candidates). While this approach was computationally efficient, [subjectively] intuitive, and shallowly reflective of psychological studies in human memory bias (Schacter, 1999), it ignored a space of knowledge sources that, when integrated with task-independent memory retrieval mechanisms, may increase task performance across a variety of problems in complex domains. As described in Section 7, we plan to further explore this space, with respect to both episodic and semantic memory mechanisms, in extensions X2 and X4.

6.2 Combating Computational Intractability

While we are striving to improve our understanding of comprehensive functionality (R2), however, we must simultaneously maintain computational efficiency, scaling to large stores of knowledge over long agent lifetimes (R4). Our analytical and empirical work to date demonstrates that the fundamental computational problems facing memory retrievals, while practically efficient for many use cases, are unbounded and seemingly intractable in the general case for large stores of knowledge. To combat this finding, we plan to simultaneously investigate numerous paths.

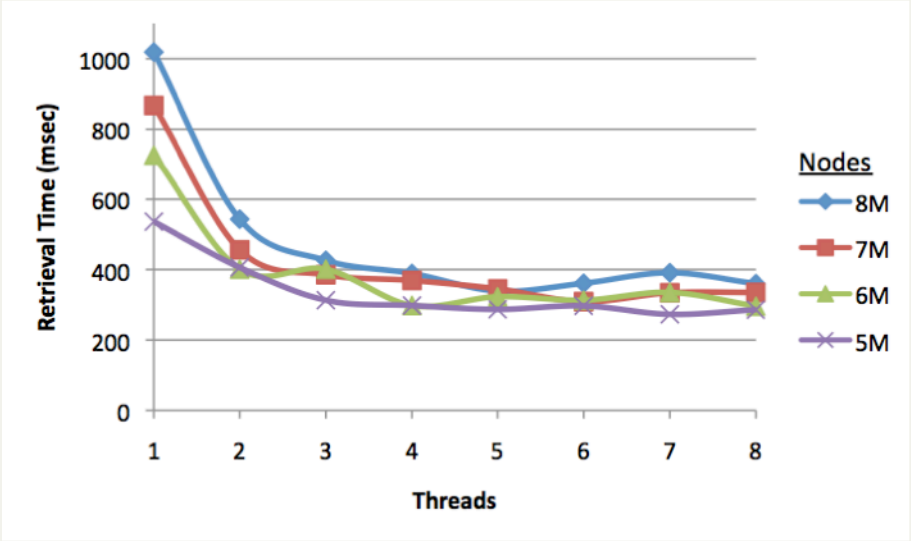

Figure 8. Preliminary Experimentation with Parallel Semantic Matching

For inspiration we can look to the human brain, the sole exemplar of a generally intelligent cognitive architecture, which employs massive parallelism across its network of more than a hundred billion neurons. Thus, one natural path for exploration is the degree to which components of episodic and semantic retrieval processes lend to highly concurrent algorithms.

The most recent and directly applicable work in this realm was done by Douglass and Myers (2010), in which they applied the highly concurrent, message-passing Erlang language to calculation of ACT-R chunk activation updates over a very large store of semantic knowledge. Their empirical results demonstrated that concurrent processing yielded significant improvements in retrieval time, converting a crucial process from quadratic growth with respect to the number of semantic concepts to a linear growth. Unfortunately, absolute retrieval times, given significant processing power (8 cores), were still at least an order of magnitude greater than that required for generally intelligent agents in dynamic domains (440 msec. over ~54k concepts, scaling to 10.9 sec. over ~1.3M concepts) and it is unclear whether additional processing cores would significantly improve this result.

We have also done preliminary empirical experimentation with parallelizing semantic retrievals. Figure 8 presents unpublished retrieval time data, measured in milliseconds, where we have modified our retrieval algorithm (see Appendix D) to divide search of candidate semantic concepts across a variable number of computing cores (1-8). This data represents a concurrent version of the "Failure Sweep" experiment presented in Appendix F, wherein the retrieval mechanism is required to consider 50% of the semantic concepts in the store, whose total size ranges from 5-8 million (labeled as "nodes" in the figure). Note that this implementation was prototyped externally to Soar, in Java (versus Soar's native C++), and thus absolute retrieval times (~300-1000 msec.) are non-comparable with those in other sections. When one core is utilized, we reproduce a linear growth in retrieval time with respect to the number of semantic concepts ($R^2 \approx 0.996$) and, although this is

preliminary work, we did find that once sufficient parallel processing was available (~4 cores), retrieval time was approximately constant (~300-400 msec.). However, without increasing the number of semantic concepts beyond the practical limits of main memory (tens of gigabytes), we did not see a continually linear decrease in retrieval time from increasingly adding cores. Additionally, as with Douglass and Myers, our absolute retrieval times were not sufficiently efficient for dynamic environments.

These studies suggest that while parallelism may serve a role to ameliorate retrieval time as knowledge stores scale, it is likely not a silver bullet at the grain size of large processors with current memory architectures on commodity hardware and thus other avenues must be explored.

7. Future Work

In this section, we detail our plan for future work, as guided by our research methodology (see Figure 7, a reproduction of Figure 1 presented in Section 3). We begin by discussing our overall evaluation strategy, including describing and justifying a complex domain, cognitive robotics, which we plan to utilize as a common evaluation test-bed. We then present four extensions that will functionally enhance encoding, storage, and retrieval of memories in Soar's episodic and semantic memory modules, while efficiently scaling to large stores of knowledge over long agent lifetimes. Finally, we present our proposed timeline to complete this work.

7.1 Evaluation Strategy

One major challenge of our proposed work is there are no accepted benchmarks or metrics for comparing task-independent memory systems in context of generally intelligent agents. Thus, principled evaluation of the quality and progression of our work is a continuous research goal in and of itself. Given this context, we have devised the following evaluation strategy, which entails concurrently assessing our future work on two fronts, focusing on the degree to which we improve satisfaction of requirements R2 (diverse, comprehensive learning) and R4 (scaling to large bodies of knowledge).

With each proposed research extension, we first plan to develop focused benchmarks to thoroughly characterize the computational requirements of our implementations, much as we did with our prior work on episodic (see Section 4) and semantic (see Section 5) memory mechanisms, such as to evaluate the degree to which we maintain reactivity in complex, dynamic environments (R4). In addition, we plan to incorporate the following complex domain, cognitive robotics, as a cumulative, thematic test bed across all mechanism explorations, such as to characterize how our work contributes to task performance within a complex environment. By qualitatively and quantitatively assessing

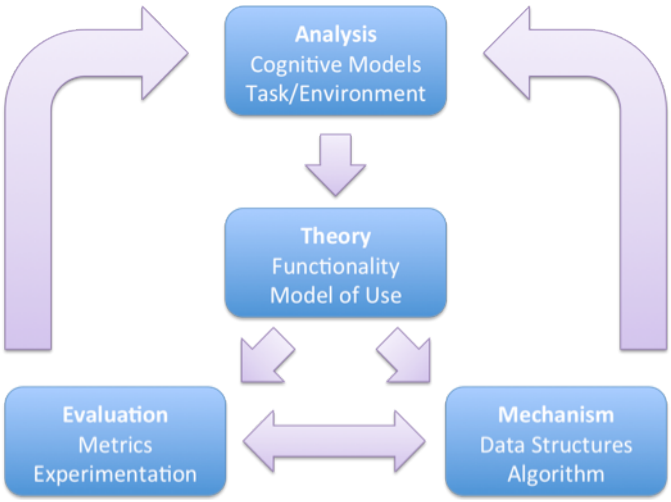

Figure 7. Memory System Research Methodology

agent performance across a number of tasks within this domain, we intend for this evaluation component to serve as a measure of the degree to which we are improving coverage of diverse learning mechanisms for generally intelligent agents (R2).

Cognitive Robotics

As depicted in Figure 9, we have been extending work on instructable autonomous agents (Huffman, 1994; Huffman & Laird, 1995) as applied to cognitive robotics (Laird, 2009). This is an especially interesting domain for memory research because it embodies all of the characteristics of tasks and environments in context of generally intelligent agents (see Section 2): not only do robots in real-world (C1) missions contend with dynamic (C2), partially observable (C5), and stochastic (C3) environments, but computationally limited (C6), autonomous agents controlling robots must contend with arbitrary tasks and missions (C4) while incorporating into their reasoning and planning large amounts of diverse knowledge from prior missions (C7) or databases, immediate state, rules of engagement and doctrine, short- and long-term goals, as well as situated instruction from other agents or humans.

In context of both simulation and physical robotics hardware, we plan to evaluate the efficacy of memory mechanisms on large-scale scenarios inspired by search & rescue and building clearing missions. These situations will involve a number of interesting characteristics:

> **Environmental Complexity (C1)**. The agent will be situated in a building, subdivided into many rooms, where adjacent rooms are connected via doorways (see Figure 9, for instance). Rooms will contain numerous, diverse objects with a variety of features (including physical properties, such as color, size and weight, as well as more abstract properties, such as semantic classifications) and relations (such as spatial location and orientation). Some object properties will be shared amongst objects and some properties (in isolation or combination) may be exploited to benefit task performance, resulting in varying degrees of task-specific and task-independent ambiguity.

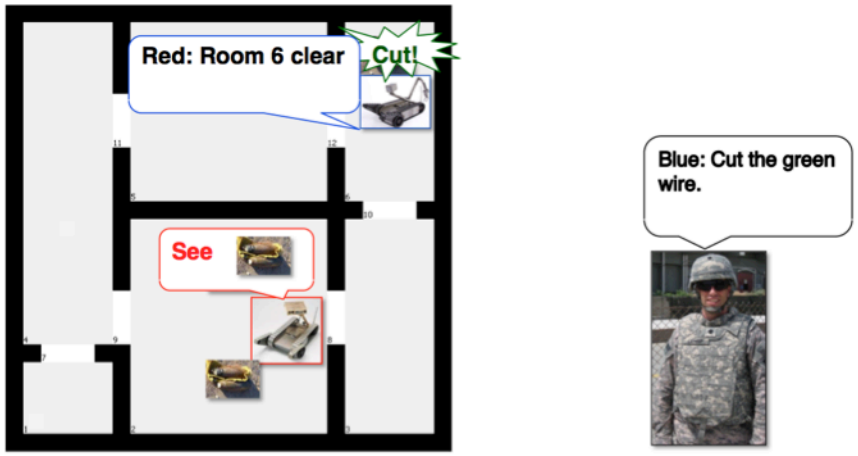

Figure 9. Cognitive Robotics Domain

Environmental Dynamics (C2). Properties of objects will change both as a result of agent action, such as the agent physically moving the object, as well as independently, such as a light going out due to severed power. As exemplified, changes will occur at varying time scales.

Environmental Regularities (C3). Regularities in environmental complexity, such as frequency of the occurrence of objects, and dynamics, such as object detonation, will be parameterized, and thus regularities will exist that the agent can detect, depending upon the capabilities of its cognitive mechanisms, and potentially exploit to succeed in tasks.

Diverse Tasks (C4). The agent will be tasked with numerous problems. Dependencies will exist in these missions, such that expertise in some tasks, such as navigation, will affect performance in other tasks, such as retrieving a particular object. Task goals will vary in temporal duration and complexity, such that some objectives, such as "go to room X," are simply defined and short-term, whereas others will be long-term, such as "never go to room Y," or complex in definition, such as "avoid dangerous situations."

Limited Interactions (C5). The building structure will be sufficiently complex such as to introduce significant partial observability: the agent will not be able to perceive objects or doorways in rooms outside its present context.

Limited Agent (C6). The agent will run on commodity hardware and make decisions with respect to real-time input from its environment.

Extended Agent (C7). As the number of objects and rooms scale, the agent will require a long lifetime in order to complete its mission objectives. The agent's existence will be continuous: a priori knowledge (preloaded from knowledge bases) will only be supplemented by that which is learned from environmental and task experience, which may include interactive instruction from human(s) or other agent(s).

The types of tasks we plan to explore include task-dependent situational assessment (such as to avoid environmental dangers and prioritize task progress), locating objects (work that could apply to locating human victims in unsafe or disaster situations) and manipulating objects (work that could apply to building maintenance or clearing, such as during times of questionable structural integrity). As we explore the extensions discussed below, the degree to which the agent improves performance in these tasks will empirically evidence its increased coverage of diverse learning mechanisms (R2) while maintaining reactivity to environmental dynamics given large stores of knowledge and a long agent lifetime (R4).

Note that while we plan to implement our work on physical robots, our focus will not be on low-level issues of motor control and perception, but instead on evaluating the degree to

which highly functional memory mechanism lend to generally useful cognitive capabilities, such as virtual sensing (reasoning about information not directly accessible to robot sensors), management of goals (such as progress towards mission directives), and generalization through categorization (such as by reducing human direction through access to declarative stores of knowledge, including doctrine and ontological world data).

7.2 Functional Extensions

We now propose four extensions to our prior work that will functionally enhance encoding, storage, and retrieval of memories in Soar's episodic and semantic memory modules, while efficiently scaling to large stores of knowledge over long agent lifetimes. As discussed in Section 6.1, these explorations are motivated by shortcomings of our work to date, particularly with respect to requirement R2, functional support for diverse, comprehensive learning, and are intended to contribute to the following two research questions:

> What aspects of agent experience should a task-independent memory mechanism *encode* and *store* such as to functionally support performance across a variety of tasks while maintaining reactivity to complex, dynamic environments?
>
> What task-independent regularities of agent experience can efficiently supplement incomplete task-dependent knowledge to improve the expected utility of *retrieved* memories in response to impoverished cues?

Extensions X1 and X2, respectfully, propose work in context of episodic memory and X3 and X4 deal with semantic memory. In each case, we are guided by our research methodology (see Figure 7), and thus we present informative analysis related to the intended exploration; our proposed approach and the theoretical commitments entailed; as well as relevant projected evaluation metrics/experiments and data structures/algorithms.

X1. Semantic Pruning of Episodic Encoding

Analysis. As an agent explores its environment, contending with numerous tasks, episodic knowledge increases with change in the agent's representation of its present state (Laird & Derbinsky, 2009). Concurrently, as the agent encounters facts about the world, its long-term semantic knowledge grows with new conceptual and ontological information. Over long agent lifetimes, these knowledge stores pose significant computational challenges with respect to raw memory consumption, as well as the computation time required to index and search this data.

Two cognitive theories inspire investigation as to how the human cognitive architecture may functionally contend with these challenges. First, the weak version of the theory of cognitive economy (Collins & Quillian, 1969; Collins & Loftus, 1975) suggests that details of object instances need not be re-encoded at multiple levels of conceptual hierarchies. For example, if an agent is newly taught that "birds have wings," this information *need* not be "copied" as a feature of previously encountered instances of birds, nor as new instances are encountered, but instead may be inferred as a property of the hierarchical structure. The

weak version of this theory does not preclude duplication of information, but suggests that generalization through processing over distal hierarchies may lead to greater cognitive utility than mass duplication of local features and relations. The second theory posits that human episodic memory relies heavily on reconstruction (Williams & Hollan, 1981; Kolodner, 1983b; Hassabis & Maguire, 2007); thus rather than simply recalling an episode from a store of "snapshots," previous experience is pieced together from disparate sources as a form of information processing.

Theory. We plan to explore an episodic encoding policy optimization that shares surface-level similarities with these theories wherein historic features and relations of long-term semantic concepts in agent state are not directly encoded in episodic memory, but rather only "pointers" to knowledge available within the semantic store. As an immediate consequence, this proposed extension would reduce the amount of knowledge episodic memory must manage, thereby improving storage and retrieval performance (R4). This policy also requires that, upon retrieval, the agent must deliberately reconstruct certain details on-demand from the present contents of semantic memory, if they are pertinent to the current situation and task. Since the agent must rely upon current semantic knowledge, which may have changed since episodic encoding, there is the potential for the agent to incorrectly reconstruct the experienced situation, which may hold consequences for present task performance.

Evaluation. A significant research question is the degree to which the computational performance gains of this proposed policy will outweigh the potential for task performance detriment when integrating the contents of two distinct long-term memory stores. In addition to focused synthetic benchmarking to characterize space reduction and query time improvement, we plan to incorporate this extension as an added variable in the cognitive robotics experimentation, which will contextualize computation gains in task performance degradation.

X2. Efficient Episodic Activation Bias

Analysis. Given the nature and function of a memory system, it is often the case that an agent querying a memory mechanism is not able to specify a cue that uniquely distinguishes one memory from another. In this situation, the memory mechanism must attempt to "predict" (Anderson, 1990), based upon task-independent environmental, task, and agent regularities, what memory will be most useful.

In context of semantic memory, a good set of these regularities has been flushed out in the literature (Anderson & Schooler, 1991), as discussed in X4. However, given that we have limited experience with artificial agents utilizing a task-independent episodic memory system in complex tasks (Nuxoll, 2007), it is unclear whether this is a functionally sufficient, let alone optimal, set of functions to bias an episodic retrieval mechanism. We can, however, look to humans, who appear to incorporate multiple, complex forms of bias in retrieval of episodic and declarative memories, including emotion (Dolan, 2002) and central arousal (Davis & Whalen, 2001).

Theory. While significant work must be done to determine which specific episodic bias functions are functionally required for generally intelligent agents, we plan to begin the investigation by understanding how to efficiently support two classes of retrieval bias, representing the extremes with respect to memory granularity, within a task independent episodic memory mechanism: activation of individual structures within an episode/cue and activation of entire episodes.

The first class supports work such as that by Nuxoll (2007), in which he showed some evidence that incorporating the working memory activation (Chong, 2003; Nuxoll et al., 2004) of cue features, a measure relating to procedural memory activity, as a bias may contribute to improved task performance on certain retrievals. The second class supports recent work on integrating appraisal theory as a form of emotional modeling in a general cognitive architecture (Marinier et al., 2009).

By making progress on understanding the computational requirements (R4) of these two classes of episodic activation bias, we will support future experimentation to discover the optimally functional set (R2).

Evaluation. As an initial evaluation, we plan to exhaustively benchmark both classes of activation on synthetic data sets, as we did with our prior work with semantic memory activation (Derbinsky et al., 2010). Additionally, we will implement and evaluate task-dependent appraisals within the cognitive robotics domain as a demonstration of real-world use, and an initial exploration of task benefit. For instance, consider a search & rescue agent exploring and assessing rooms within a building for individuals requiring assistance. As it encounters rooms, it will associate with its experience a potentially complex set of appraisals, including goal relevance and novelty. After exploring the scene, when planning how to expend its time and energy, we hypothesize that episodic retrievals biased by these measures, driven by task-dependent control knowledge, will result in improved task performance over an unbiased baseline.

X3. Automatic Semantic Encoding

Analysis. Given a sequential presentation of instances, some important conceptual findings include useful clusters of the instances into categories, an intensional definition for each category, and a hierarchical organization for the category (Michalski & Stepp, 1983). This type of conceptual information about the world supports a number of important cognitive functions, including linguistics, communication, and inference (Davidsson, 1995). Unfortunately, incremental formation of concepts is a complex problem that has tasked and, in the general case, eluded the machine learning community for decades (Gennari et al., 1989).

Anderson and Matessa (1998) describe an appealing approach to categorization, a subset of the problem of conceptual clustering, wherein incremental accumulation of conceptual information is not *directly* served by an architectural primitive, but rather *emerges* from problem-solving within ACT-R (Anderson et al., 2004), a general cognitive architecture with sufficient learning capabilities to capture relevant regularities in perceived instances.

A crucial component of Anderson and Matessa's model is the automatic encoding of declarative information. The ACT-R theory posits an architectural semantic encoding policy, in which changes to module buffers, including perception and goals, results in automatic storage of new declarative knowledge. This encoding policy is computationally cheap, as no complex decisions are made to decide the "best" elements to encode, but relies on task knowledge to construct cues and comprehensive activation bias during retrieval to sift through and select the most pertinent knowledge chunk amongst potentially large amounts of declarative knowledge.

Theory. We plan to explore and evaluate a small space of automatic, task-independent semantic encoding policies in Soar, which will supplement its learning capabilities (R2). The base policy will emulate that of ACT-R: all changes to agent state constitute new semantic knowledge. We also plan to integrate *temporal stability*, the length of time a structure persists, which may help distinguish transient structures from those more permanent in nature, and *situational focus*, which may serve as task-independent measure of task-dependent importance, as heuristics by which to ameliorate computational space burden (R4).

Mechanism. We plan to extend the episodic working memory graph (see Section 4) to incrementally summarize the distinctiveness of concept features that have occurred in the agent's representation of state. For instance, if a particular concept always has a particular feature "color," then this structure would have a relatively high measure of temporal stability.

We also plan to incorporate Soar's working memory activation (Chong, 2003; Nuxoll et al., 2004) as a measure of situational focus. This process maintains, for each structure in working memory, a measure of procedural activity, which is a summary of the prior history of rules that have tested or created the structure.

Evaluation. As an initial evaluation, we plan to compare memory required to store semantic knowledge and time to retrieve memories using focused, synthetic data sets. We also plan to survey the literature (such as Anderson & Matessa, 1992) for existing categorization data sets on which we can further evaluate our space of encoding policies. In context of cognitive robotics experimentation, we plan to compare, with respect to memory requirements and task performance, the efficacy of hand-coded semantic knowledge, such as representations of persistent room organization and stable object features, with those stores produced by our space of automatic encoding policies.

X4. Efficient Semantic Activation Bias

Analysis. Anderson and Schooler's rational analysis of declarative memory led to empirical evidence that for a memory system to optimally estimate the odds that a particular memory will be needed, it likely requires sensitivity to particular sub-symbolic, structural regularities in the environment (Anderson, 1990), including a comprehensive treatment of

concept retrieval history (Anderson & Schooler, 1991) and associations with present context (Schooler & Anderson, 1997).

The ACT-R cognitive architecture (Anderson et al., 2004) declarative memory model implements a comprehensive activation suite, including retrieval history (termed base-level activation) and context (termed spreading activation). This activation functionality is crucial to numerous cognitive models, but has been shown not to scale to large semantic stores (Douglass et al., 2009; Douglass & Myers, 2010).

Theory. We plan to investigate and evaluate computational methods for incorporating base-level and spreading activation functionality into a semantic retrieval mechanism that scales with large stores of knowledge. This will extend our previous work on semantic activation bias functions (see Section 5), complimenting the literature's existing understanding of activation functionality (R2) with a new understanding of scalable methods (R4).

Mechanism. Douglass and Myers (2010) have shown preliminary empirical evidence that parallelism can significantly improve spreading activation calculation in large semantic stores. However, even massive concurrency did not bound their calculations sufficiently for autonomous agents to remain reactive to dynamic environments. Thus, we plan to formally and analytically evaluate approximation methods, including bounding the effects of activation updates (Berthold et al., 2009), such as degree of fan-out, and allowing limited error into high-dimensional, nearest-neighbor matching (Andoni & Indyk, 2008).

Evaluation. To evaluate these activation function variants, we plan to measure and balance efficiency and functionality. First, we will measure the degree to which the computational resource usage of these algorithms, including average/maximum retrieval time and indexing data structure space, scales to large semantic stores, such as the size of WordNet (Miller, 1995), Cyc (Lenat, 1995), or SUMO (Niles & Pease, 2001). We plan to formally characterize these growth rates, as well as empirically evaluate usability in large synthetic and real-world data sets.

Second, as we explore approximations to theoretical activation bias functions, we will evaluate the degree of model fidelity degradation, as measured by apposite experiments in the literature. For example, one well-studied problem, applicable to contextual activation spreading, is the Word Sense Disambiguation (WSD) task (Navigli, 2009). A particular formulation of this problem measures WSD accuracy given both a set of input words, each potentially contributing to context, and a machine-readable dictionary, such as WordNet (Miller, 1995). We plan to make use of existing WSD algorithms in this space, such as Lesk (Lesk, 1986) and those using semantic networks (Tsatsaronis et al., 2007; Tsatsaronis et al., 2008), as task performance baselines, as well as comparisons to predictions made by theoretical activation models, which will serve to characterize the absolute degree of model fidelity, taking into account relative loss of task functionality.

We also plan to investigate the role semantic retrievals play in the cognitive robotics domain, and how activation bias impacts agent behavior, especially when we introduce

approximate algorithms. For instance, semantic memory is a likely candidate for storing semi-static task experience, such as symbolic map structures. We plan to compare how well the agent is able to contend with large, complex maps when this type of information is retrieved from Soar's short-term working memory, its contextual episodic memory, and its associative semantic memory. Computationally, we expect differences in storage space and retrieval time and that with large environments with many semantic structures, short-term memory retrievals will not scale. Qualitatively we hypothesize that differences in biases between the retrieval algorithms for these mechanisms will lead to dominant strategies over multiple tasks in this complex domain. We also plan to investigate the degree to which the Word Sense Disambiguation results can apply to situated instruction work, possibly resulting in the reduction of the need for specificity in instruction and the increased expressivity of the vocabulary.

Summary

Table 5 (reproduced from Table 2 in the introduction to this document) summarizes the proposed extensions both with respect to memory system organization (episodic vs. semantic; encoding/storage vs. retrieval) and how we will contribute to satisfaction of the requirements of memory systems for generally intelligent agents.

Table 5: How Proposed Extensions Relate to Memory System Organization and Requirements

				REQUIREMENTS					
				R1	R2	R3	R4	R5	R6
EXTENSIONS	SEMANTIC	*Retrieval*	X4	•	🡅	•	🡅	•	•
		Encoding Storage	X3	•	🡅	•	🡅	•	•
	EPISODIC	*Retrieval*	X2	•	🡅	•	🡅	•	•
		Encoding Storage	X1	•	🡅	•	🡅	•	•

Each extension will build on prior work on the episodic and semantic memory systems in Soar, and will thus satisfy requirements R1 (online/incremental learning), R3 (diverse representation), R5 (effective access), and R6 (task independence). In addition, each will incrementally add to supporting a diverse, comprehensive set of learning mechanisms (R2) that scale with large stores of knowledge over long agent lifetimes (R4).

7.3 Timeline

In this section we discuss our timeline for the proposed research and evaluation in this thesis.

We first note that we will not have to develop the software and hardware infrastructure for the cognitive robotics domain from scratch, but will instead build on existing work and collaborate with others as a part of a larger project. Consequently, we do not expect that our proposed evaluation strategy will significantly hamper our research efforts, nor affect our timeline.

Next, we admit that there is a moderate amount of interplay within our proposed extensions. For instance, an automatic semantic encoding policy (X3) will likely impact the amount of knowledge with which the retrieval mechanism must contend (X4) and the degree to which episodic retrieval efficacy is degraded given semantic pruning (X1). While these interdependencies will ideally lend to interesting conclusions in the final evaluative synthesis, we recognize the potential for danger in context of a timely and decomposable thesis progression. We have therefore devised the following serialized schedule that provides for modular research progress, while incrementally preserving many of the cumulative interdependencies of the thesis.

As demonstrated in our prior work, the semantic bias (X4) algorithmic component relies minimally only upon characterizing the size and composition of the knowledge store to study. While this is also true of episodic bias (X2), the relative complexity of memory representation (a semantic key-value set versus an episodic graph) and amount of prior evaluative work (significant for semantic, nearly non-existent for episodic) make efficient episodic bias a significantly more difficult problem. Thus, as previously detailed, we plan to begin with X4. For evaluation, we plan to begin with synthetic preloaded datasets (as demonstrated by our initial work in Section 5), deliberately encoded knowledge within the cognitive robotics domain, and apposite studies from existing literature, such as the Word Sense Disambiguation task.

To enhance our evaluation of X4, the natural next step is to investigate stores, across numerous tasks, produced whilst exploring the space of automatic semantic encoding policies (X3). While we do not expect that we will have to overcome significant algorithmic challenges to implement these encoding mechanism modifications, we do see the potential for this work will yield significant added functionality to Soar agents, result in significant synergies with X4, and lend to a more holistic inspection of the semantic memory system.

Given efficient semantic activation bias (X4) and an automatic semantic encoding policy (X3), we can independently proceed to either episodic encoding (X1) or retrieval (X2). We expect that some of the algorithmic components of retrieval will transfer from the semantic work (such as concurrency and approximation infrastructure), but that there will still be very difficult problems with which to contend. In contrast, the encoding component is likely to be relatively straightforward algorithmically, but may pose a significant evaluation challenge in order to characterize the policy over numerous tasks given a space of semantic knowledge defined via X3 and X4.

If extensions X3 and/or X4 require significantly more time than planned, the final work segment lends a natural path to conclude this thesis. Semantic pruning of episodic encoding (X1) will yield an interesting exploration of the interplay between semantic and episodic mechanisms, while the episodic bias (X2) extension is most easily and cleanly detached for future work.

October 2010 – January 2011

- Efficient Semantic Activation Bias (X4)

January 2011 – April 2011

- Automatic Semantic Encoding (X3)

April 2011 – September 2011

- Semantic Pruning of Episodic Encoding (X1)
- Efficient Episodic Activation Bias (X2)

October 2011 – March 2012

- Thesis data analysis, writing, and defense.

Appendix A. Memory Model Space

This appendix details an initial characterization (Derbinsky & Gorski, 2010) of the space of memory systems, borrowing heavily from and generalizing Nuxoll's breakdown of the space of episodic memory systems (2007). We define a memory system implementation as a commitment to features from within the space defined by these dimensions, represented as (encoding, storage, retrieval).

A.1 Encoding

- **Initiation** – Initiation encompasses the event conditions that trigger the encoding and storage processes. These events may condition upon fixed architectural characteristics of state (such as a temporal frequency) or may be accessible to agent control knowledge.

- **Determination** – Once initiated, the memory mechanism selects features of agent state (or derivation thereof) that compose the knowledge to be stored, as well as any additional context (temporal, spatial, etc) that may also be associated with the knowledge.

A.2 Storage

- **Granularity** – Stored experience varies with the grain size at which knowledge can accessed and modified. This may range from minute (such as the symbol level), to moderate (an episode), to coarse (such as the entire knowledge store).

- **Dynamics** – Knowledge in the memory system may change over time, such as to bias retrieval or forget knowledge. The mechanisms that cause this change may be fixed, condition upon agent knowledge, or deliberate agent action.

A.3 Retrieval

- **Accessibility** – Experience encoded within the memory system may vary in the degree to which it is exposed to other architectural mechanisms, such as to maintain overall agent reactivity. For instance, a declarative long-term memory may allow for enumeration of all stored memories.

- **Initiation** – Initiation encompasses the event conditions that trigger the retrieval process. As with encoding initiation, these events may condition upon fixed characteristics of state or may be accessible to agent knowledge/control.

- **Cue Determination** – Once initiated, the memory system composes agent state, knowledge, context, and/or [possibly inaccessible] meta-data to select or create a retrieval cue.

- **Selection** – When supplied a cue, the memory system implements a policy for how stored knowledge is matched with respect to the cue, which may be restricted by time,

computation, and/or number of results, as well as include bias from agent state, context, and/or meta-data.

- **Result** – When the memory system selects stored experience for retrieval, it may arbitrarily represent the knowledge, associated context, and aspects of the retrieval process, such as match quality, for agent inspection.

Appendix B. Detailed Episodic Memory Evaluation

This appendix contains detailed information about our initial evaluation of episodic memory in Soar, termed Soar-EpMem (Derbinsky & Laird, 2009), including the evaluation domain, as well as performance models of cue matching and reconstruction operations.

B.1 Evaluation Domain

TankSoar, exemplified in Figure B1, is a pre-existing domain that has been used extensively in evaluating other aspects of Soar and was used in the original episodic memory research in Soar (Nuxoll & Laird, 2007). In TankSoar, each Soar agent controls an individual tank that moves in a discrete 15x15 two-dimensional maze. Agents have only limited sensing of the environment and their actions include turning, moving, firing missiles, raising shields, and controlling radar. A TankSoar agent has access to a rich set of environmental features through its senses, including smell (shortest-path distance to the nearest enemy tank), hearing (the sound of a nearby enemy), path blockage, radar feedback, and incoming missile data.

TankSoar includes an agent named *mapping-bot* that builds up an internal map of the environment as it explores. The *mapping-bot* agent's working memory contains about 2,500 elements. Over 90% of these elements comprise a static map of its environment. A large proportion of the remaining WMEs (usually 70-90%) are related to perception and they typically change within one episode. For the experiments described below, a new episode was stored every time an action was performed in the environment, which is approximately every primitive decision cycle in Soar. The properties of this agent, especially the large working memory and the large number of WMEs changing per episode, make TankSoar an atypically stressful domain for episodic memory experimentation.

The tests were run on an Intel 2.8GHz Core 2 Duo processor with 4GB RAM. The Soar-EpMem episodic store was managed using version 3 of the SQLite in-process relational

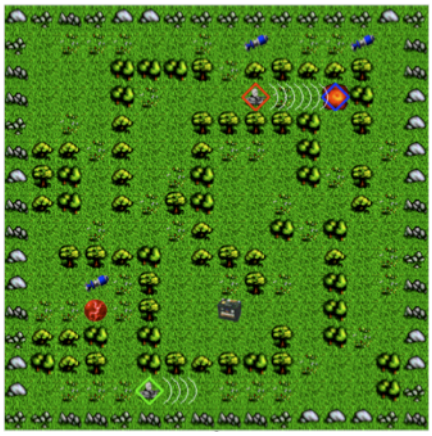

Figure B1. TankSoar

database engine. The tests described below involved one million *mapping-bot* episodes, averaged over ten trials.

B.2 Cue Matching Evaluation

To compare Soar-EpMem cue matching performance with theoretical bounds, we developed the following model to reflect the effects of operational algorithms and data structures:

$$Cue\ Match = DNF + Interval\ Search + Graph\ Match$$
$$DNF = (X_1)(\log_2[\ U * R\ ])(L)$$
$$Interval\ Search = (X_2)(1/T)(Distance)(\Delta)$$

The constants in the equations (X_1, X_2) reflect linear scaling factors for a given computer. To derive these values for our experimental setup, we performed 100 isolated executions of primitive operations (*DNF* and *Interval Search*) on data collected from 10 trials of *mapping-bot* data at 10 time points (100K, 200K, ... 1M). We collected the necessary episode statistics (described below) and performed linear regressions to fit data points for 15 different queries. Low performance timers (resolution was 1µs) caused most model noise.

The *Cue Match* operation comprises *DNF* construction and *Interval Search*. The former is linearly dependent upon the logarithmic growth of the average number, *U*, of historically unique internal and leaf nodes multiplied by *R*, the total number of stored intervals, as well as linearly dependent upon *L*, the number of literals associated with the cue nodes. In our tests (see Figure B2), we found X_1 to be 4.33µs (R^2=0.996). In experiments with *mapping-bot* to one million episodes, results depended greatly on the cue. For all cues that did not reference "squares" on the agent's internal map, *DNF* operation time was constant and below 8 ms. Cues containing references to map "squares" (and thus referring to over 250 underlying structures) brought this upper bound to 55.1 ms.

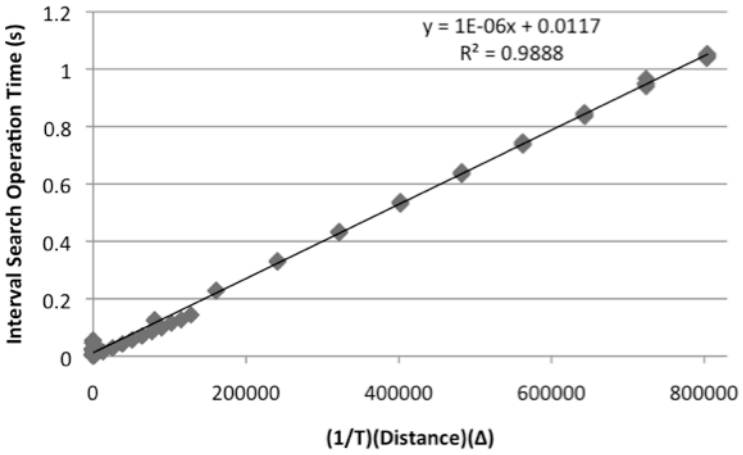

Figure B2. Cue Matching DNF Regression Data

The *Interval Search* operation is expressed as a proportion of relevant cue node intervals. *T* represents the total number of episodes recorded. *Distance* represents the temporal difference between the current episode and the best match. *Δ* represents the total number of intervals relevant to the cue. Intuitively, the farther back in time we must search for an episode, the more intervals we must examine. This ratio could be re-written as the product the minimal relative co-occurrence probability of the cue nodes, and the total number of changes experienced to date by these cue nodes. In our tests (see Figure B3), the X_2 constant was 1.29μs (R^2=0.989). We found absolute operation times depended greatly on the supplied cue. For cues that did not compel distant searches, *Interval Search* was constant with an upper bound of 2.5 ms. With cues crafted to force a linear scan of the episodic store, time increased linearly to a maximum of 1.03 seconds over one million episodes.

Since the linear factors, *L* and *Δ*, grow proportionally to changes in agent working memory, the first phase of Soar-EpMem cue matching achieves the lower bound of growing linearly with agent changes. The *Graph Match* operation, however, is much more difficult to characterize. CSP backtracking depends upon cue breadth, depth, structure (such as shared internal cue nodes), and corresponding candidate episodes, but can be combinatorial in the worst case (though our two-phase matching policy attempts to minimize this cost). We have not extensively evaluated this component, but we expect a studied application of heuristic search will effectively constrain graph- match in the average case.

B.3 Episode Reconstruction Evaluation

To compare Soar-EpMem reconstruction performance with theoretical bounds, we developed the following model to reflect the effects of operational algorithms and data structures:

$$Reconstruction = RI\text{-}tree + Collect$$
$$RI\text{-}tree = (X_3)(\log_2 R)$$
$$Collect = (X_4)(M)(1 + \log_2 U)$$

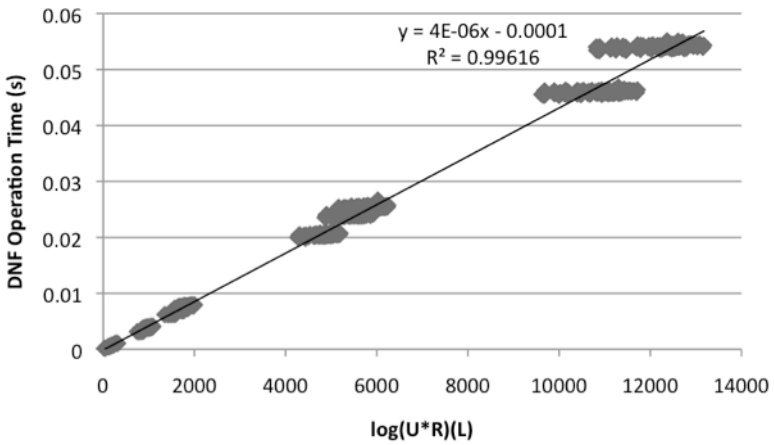

Figure B3. Cue Matching Interval Search Regression Data

To validate our model we performed 100 isolated executions of primitive operations (*RI-tree* and *Collect*) on the same data collected for cue matching (10 trials, *mapping-bot*, 10 time points from 100K to 1M episodes). We collected the necessary statistics (described below) for 50 episodes selected randomly (5 per 10,000 episodes through the first 100,000 episodes of execution) and performed linear regressions to fit data points.

Total time for episode *reconstruction* is the sum of two operations: *RI-tree* and *Collect*. *RI-tree* refers to the process of extracting pertinent intervals from the Relational Interval Tree. The logarithmic dependent variable, *R*, refers to the total number of ranges in the RI-tree structure. In our experiments, the X_3 constant was 2.55μs (over 70% R^2). After one million episodes, we recorded the upper bound of *RI-tree* operation time as 0.1 ms.

The *Collect* operation refers to cross-referencing pertinent episode intervals with structural information in the *Working Memory Graph*. This process depends upon the average number, *U*, of historically unique internal and leaf nodes, as well as the number of elements, *M*, comprising the episode to be reconstructed. With *mapping-bot* we regressed an X_4 value of 1.6μs. Because episode size does not vary greatly in *mapping-bot* (2500-2600 elements, typically), the dominating linear factor, *M*, highlighted noise in the experimental data and thus R^2 was 73%. After one million episodes, we recorded an upper bound of 22.55 ms for the *collection* operation with episodes ranging from 2521-2631 elements.

If we assume a constant or slowly growing average episode size, the *M* factor can be considered a constant and thus *Reconstruction* becomes the linear sum of logarithmic components *R* and *U*. Both *R* and *U* increases result from changes in agent working memory. Thus, under these assumptions, Soar-EpMem episode *reconstruction* achieves the lower bound of growing linearly with agent changes.

Appendix C. Semantic Memory Retrieval Formulation

This appendix contains detailed information about our formulation of the semantic memory retrieval problem (Derbinsky et al., 2010), including a mapping to the ACT-R declarative memory module and extensions for a class of activation bias.

C.1 Basic Problem Formulation

We define a semantic memory store as a set of *elements*. A semantic element is decomposed into a set of symbolic *augmentations*. For example, consider the following example semantic store, in which the letters A-D identify elements and lower-case Greek letters represent augmentations:

A: {α, β, ε, φ}
B: {α, ε}
C: {γ}
D: {γ, φ}

We define a symbolic retrieval *cue* as having a required *positive* component and an optional *negative* component, each of which is expressed as a set of symbols (corresponding to the augmentations of a particular semantic store). For instance, consider the following retrieval cue, corresponding to the example store above, consisting of both positive (+) and negative (–) components: +{α, ε}, –{γ}. The positive set specifies augmentations that an element *must* contain, and the negative set those that it *must not* contain.

Given a semantic store and a cue, we define the *result* of a semantic retrieval to be a single element from the store, including all augmentations, that satisfies the constraints represented by the cue. Thus, the result of the example cue and the example store would either be element A or B (with respective augmentation set {α, β, ε, φ} or {α, ε}). A retrieval is considered a *success* if there exists a result (as with our example) and a *failure* otherwise.

C.2 Mapping to ACT-R DM

We now map the ACT-R DM to our abstract formulation. First, without loss of generality, we interpret the chunk type as a slot-value pair. Next, since we are considering qualitative matching (equality is defined as symbolic equivalence), each distinct slot-value pair can be equivalently represented as a single, composite symbol (by concatenating the slot label and value with a unique separating character). Since slot-value pair order is arbitrary, a chunk instance can be equivalently represented as a set of [composite] symbols. In ACT-R, all chunks of a given type must contain values for the same set of slots and a chunk type can only have one slot of a given label; without loss of generality, we eliminate both of these constraints. Given the analysis above, a chunk maps to a semantic memory element, and slot-value pairs to augmentations.

We apply a similar analysis to semantic retrieval requests, with distinct slot-value pairs compressed to a single composite symbol. If we require that equivalent slot-value pairs in chunks and retrieval requests resolve to the same composite symbols, then the set of

positive tests form the positive cue component and the negative tests the negative component.

With this analysis, we claim that the symbolic ACT-R DM retrieval interface is an instance of our problem formulation. Thus, results from our work extend to ACT-R models, and any other system that can be similarly mapped.

C.3 Extension: Activation Bias

To integrate activation bias in our problem formulation, we extend our definition of a semantic memory element to include a numerical *activation value*, as exemplified below by the numbers in square brackets:

A [1.41]: {α, β, ε, φ}
B [1.73]: {α, ε}
C [3.14]: {γ}
D [2.72]: {γ, φ}

We refine our previous definition of a retrieval result as an element from the DM, including all augmentations, that satisfies the constraints represented by the cue *and* has the maximal activation value. Given the example cue (+{α, ε}, −{γ}) and this expanded DM, the result is now unambiguously B (and its associated augmentations), as it has a greater activation value than A.

Appendix D. Supporting Basic Semantic Memory Retrievals

This appendix discusses indexing structures and processes to efficiently support a large class of symbolic semantic memory retrievals, without consideration of sub-symbolic activation, accompanied by a brief computational complexity analysis. We decompose our description into the required positive cue component, followed by the negative.

D.1 Positive Cue Component

To review, the positive cue component for symbolic semantic retrievals is a non-empty set of augmentations that an element must contain. To assist in our analysis, we define R_p as the elements that contain an augmentation p and, accumulated over all p in P, R to be the bag of candidate elements (which may contain duplicates, if an element contains more than one augmentation, p, in P).

Indexing

Building on this prior work, the primary indexing structure for our mechanism is an inverted table of semantic elements, combined with cached frequency statistics. The structure contains a sorted list of each augmentation, p, in the semantic store, each paired with a sorted list of elements in which they are contained as well as the size of this list, R_p. We note that this structure roughly doubles the size of the store and can be updated very efficiently as the store changes. Consider the following index over the example store in Appendix C:

α (2): [A, B]
β (1): [A]
γ (2): [C, D]
ε (2): [A, B]
φ (2): [A, D]

Algorithm

To retrieve based only on the positive cue component, we first generate a sorted list, Q, of all augmentations p in P, keyed ascending on R_p, which requires $|P|$ queries on the inverted index. Q represents a specialized query plan, sorted in ascending order of candidate element list size. With the example positive component above, Q is either [α,β] or [β,α] (as $R_\alpha = R_\beta$), and we use the former for the remainder of this analysis.

Next, we pop the first augmentation from Q (α) and retrieve a pointer, w, to the head of the element list in the inverted index (initially referring to the first element, A). Note that since this list is updated incrementally with changes to the semantic store, we do not have to compute this list in response to the query. Iterating over the remaining augmentations in Q ([β]), we verify, using the original store, that w satisfies all remaining positive constraints. If so, return w and *success*. Otherwise, increment w to point to the next element in the inverted index and retry verification. If no element successfully verifies, the retrieval is a *failure*.

Analysis

In the worst case, this retrieval mechanism grows linearly with $|E|$. However, the small amount of indexing and query optimization bounds element iteration to $\min(R_p)$, the set of elements containing the most selective positive query augmentation. Furthermore, we only need to fully examine this list in the *failure* case.

D.2 Negative Cue Component

The negative cue component for symbolic semantic retrievals is an optional set of augmentations that a retrieval must not contain.

We have struggled with how to efficiently support this type of constraint given our problem formulation. What makes this component difficult is that given a large DM with a sparse distribution of augmentations, it can be prohibitively expensive to maintain an index of the elements *not* containing an augmentation, analogous to issues surrounding the closed-world assumption and negated conditions in production matching (Doorenbos, 1995).

Initial Integration

Currently, we integrate this functionality with the positive cue component above by special-casing negative augmentations. First, $|R'_n|$, the number of candidate elements that do not contain a particular augmentation n, equals $(|E| - |R_n|)$, the total number of elements less the number of elements that do contain the augmentation. This quantity can be computed efficiently and used to order Q with negative augmentations. Second, because we cannot efficiently enumerate R'_n, w is initialized as the head of the list of the first positive augmentation in Q. Finally, when verifying a candidate element, we simply invert the result of the set-inclusion query on E.

Analysis

Using this approach, our mechanism loses a major performance benefit. This forfeiture arises when there exists an augmentation in the negative component that is more selective than any positive component augmentation, which is probably not uncommon. While we are theoretically able to integrate this functionality, we have neither implemented nor evaluated this work empirically in Soar, and plan to address this deficiency in the future.

Appendix E. Supporting Efficient Activation Bias

This appendix first defines the class of activation update processes we can efficiently support, and then discusses how we achieve this functionality.

E.1 Efficient Activation Bias Updates

Just as semantic memory must support efficient updates to elements and augmentations, so too must it support efficient updates to activation values. In this context, for large stores, we propose that an activation value update process must be *locally efficient*. An activation update process is locally efficient if it satisfies two properties: (1) the update can affect the activation value of at *most* a constant number of elements and (2) updating the activation value of an element takes time strictly sub-linear in the number of semantic elements.

The locally efficient activation update process we implemented in Soar is a straightforward mechanism to bias retrievals towards recency. After each successful retrieval, the activation value of the retrieved element is updated to be one greater than the previously largest activation value. This update process is local, as it only changes a single element per retrieval, and it is efficient, as the largest activation value can be cached to avoid any search over E.

In ACT-R, chunk activation includes retrieval history (base-level), current context (spreading), partial matching, and noise. Both the base-level approximation and permanent noise computations appear to be local, so it should be possible to extend our approach to cover those components. However, transient noise, partial matching, and spreading activation appear to be global to the elements of the store, which suggests significant further theoretical and engineering research are necessary to develop locally efficient mechanisms. Douglass and Myers (2010) have shown initial evidence that highly concurrent semantic networks may afford significant performance gains for global updates.

E.2 Efficient Support

The most direct method of integrating activation values in our efficient algorithm is to sort the candidate list (w) by activation values on demand. This approach, henceforth referred to as Scheme I, suffers from retrieval times that are *always* dependent upon augmentation selectivity, as the candidate list must be fully computed to be sorted.

Another method of integrating activation values, Scheme II, is to maintain, for each augmentation, an element list sorted by activation value. Thus, w is sorted in order of activation, independent of augmentation selectivity. However, the time required for updating activation values is dependent upon the number of different augmentations an element can have (its augmentation cardinality), and for large cardinalities, this cost can be prohibitive.

Our approach to integrating activation values combines these schemes by exploiting an assumption that most elements will have "small" augmentation cardinality. Given this information, we explain how we can extend our implementation to yield efficient retrievals

and then we validate our assumption empirically by studying three large, commonly used knowledge bases.

Our Approach

If an element has small augmentation cardinality, Scheme II is efficient, independent of semantic store size. If few elements must be sorted per retrieval, Scheme I is efficient, independent of element augmentation cardinality. To resolve this tension between augmentation cardinality and element selectivity, we apply these schemes on a per-element basis: we apply Scheme II when an element has small augmentation cardinality, and otherwise apply Scheme I. What we describe here are the data structure modifications and additional processing necessary to efficiently implement this split strategy.

First, we introduce a threshold parameter, *t*, which represents a small value of augmentation cardinality. By default, we integrate activation bias as described in Scheme II above. However, if the augmentation cardinality of a particular element is greater than *t*, we associate a one-time special "infinity" (∞) activation value with all its augmentations and maintain a separate list associating the element with its activation value, per Scheme I. For instance, if *t*=3, we would have a list wherein [A=1.41] and our inverted index would contain the following information:

α (2): [A=∞, B=1.73]
β (1): [A=∞]
γ (2): [C=3.14, D=2.72]
ε (2): [A=∞, B=1.73]
φ (2): [A=∞, D=2.72]

By default, an update to an element's activation value will involve updating a small number of references (≤*t*) throughout the inverted index. For elements with augmentation cardinality greater than *t*, such as A, we need only update this value once, thereby bounding the update to constant time and addressing the weakness of Scheme II.

During retrieval, as we are populating the list of augmentations, *Q*, which is sorted by activation level, we may now encounter one or more infinite activations at the head of the list. If so, we perform a lookup for its true activation level and execute insertion sort into a second, special list, *Q*'. We then merge *Q* and *Q*' to form our query plan. Notice that if the size of *Q*' is small (i.e. few elements have augmentation cardinality greater than *t*), this process is cheap and independent of augmentation selectivity, the weakness of Scheme I. Thus, if we can select an appropriate value of t, we will achieve efficient activation bias support.

Validation

To validate that our split strategy works well on real data sets, we studied three large, commonly used knowledge bases (KBs): SUMO (Niles & Pease, 2001), OpenCyc (Lenat, 1995), and WordNet (Miller, 1995). For each KB, we extracted the number of features of each named entity. Each distribution was unimodal and exhibited strong right skew, suggesting that while most elements had a similar feature size, there were rare cases with

exceptionally large cardinalities. Then, we sampled from these distributions to form synthetic data sets that were reasonably large (5040 elements) and empirically valid in augmentation cardinality. We then collected empirical retrieval data, summarized in Table D1, showing that for each KB there was a range over the value of *t* that optimally balanced the performance effects of cue selectivity and augmentation cardinality. For two of the KBs, we could efficiently employ Scheme II above for more than 99% of elements, versus only about 93% for the SUMO data set.

Important components of this analysis for future examination are (1) automatically selecting a value of t for a given DM and (2) tuning this value online for changing DM contents. As to the former, we see in Table E1 that the optimal threshold typically covers greater than 90% of the elements using augmentation cardinality, but that value is not constant across data sets. Further analysis of the KBs may uncover why this is the case and suggest better factors for prediction. As for the latter, we expect that caching t in indexing structures will allow the algorithm to adapt in real time, while maintaining efficient retrievals.

Table E1: Optimal Thresholds.

Data Set	Optimal *t* Range	Element Coverage
SUMO	50 – 70	92.78 – 93.86%
OpenCyc	40 – 60	99.17 – 99.74%
WordNet	20 – 40	99.50 – 99.90%

Appendix F. Detailed Semantic Memory Evaluation

In this appendix we detail the evaluation of our implementation of semantic memory (see Appendices D & E). We implemented our data structures and algorithms as the Semantic Memory long-term, symbolic memory system in the Soar cognitive architecture (Laird, 2008). We used version 3 of the SQLite in-process relational database engine to manage the semantic store and all experimental results were run on a 2.8GHz Core 2 Extreme processor with 4GB of RAM.

Our final evaluation spans two data sets: (1) the WordNet 3 lexicon and (2) a scalable synthetic benchmark of our design. WordNet offers a large, ecologically valid knowledge base with which we can compare to previous results in this space (Douglass et al., 2009). Our synthetic dataset offers us the ability to exhaustively benchmark our retrieval mechanism on arbitrarily large semantic stores.

F.1 WordNet

As with Douglass et al. (2009), we used the WN-LEXICAL WordNet 3 data conversion (Emond, 2006). The data set has over 820K chunks, which includes over 212K word/sense combinations. Once imported, Soar's semantic store, including all indexing structures, is about 400MB.

Our first experiment was to verify (a) that retrieval time was independent of augmentation selectivity and (b) that the activation bias was processed efficiently in under-specified cues. We performed semantic retrievals on 100 randomly chosen, single-augmentation cues, averaged over 10 trials. Retrieval time was 0.1887 msec. each (0.0216 std. deviation).

Our next experiment focused on larger cues. We randomly chose 10 nouns and formed a cue from their full sense description. Retrieval time was an average of 0.2973 msec. over 10 trials each (0.0108 std. deviation).

Douglass et al. (2009) used a derived subset of the WN-LEXICAL dataset, so direct replication of their work is difficult. They reported retrievals of about 40 msec. with cues of 1-4 augmentations on a declarative memory with about 232.5k chunks. Our results show 100x faster retrievals on a comparable set of cues scaling to a 3x larger DM.

F.2 Synthetic Data

In addition to running on a known data set, we tested our implementation more exhaustively to measure how it scales with much larger semantic stores. We developed a scalable, synthetic dataset generator and, in Table F1, we list statistics of the data sets we used as they scale with *k*, the size control parameter:

Table F1: Synthetic Statistics

k	Elements	Store Size (MB)
7	5,040	3.00
8	40,320	27.81
9	362,880	291.95
10	3,628,800	2048.00

While we have a DM generator, we do not have a model of what are typical cues used to access a semantic store and how those cues could interact with the performance profile of the semantic retrieval mechanism. For instance, we do not know how selective the cues are likely to be, meaning how many elements, termed candidates, could possibly satisfy any part of the cue. Furthermore, we do not know the proportion of cues that will have no perfect matches. To allow us to test these different interactions, we constructed the stores so that we can generate cues with independently controlled selectivity. In each KB, there are $k!$ elements and each element has augmentation cardinality of $(k+1)$. For $i = 2 \ldots k$, the ith augmentation of an element has selectivity $(k!/i)$. The 0th augmentation of each element is shared by all elements and the 1st augmentation is unique.

Selectivity Sweep

Our first question is whether the semantic retrieval mechanism provides bounded retrievals for under-specified cues, independent of the number of candidate elements. For each distinct augmentation in the store, we constructed a cue and measured retrieval time. As depicted in Figure F1, we found nearly constant-time retrievals within each data set, independent of augmentation selectivity, measuring just under 0.4 msec. for k=10.

Cue Sweep

Our next question is whether combinations of augmentations result in complex cues that adversely affect retrieval time. We constructed all possible lengths of cues using all combinations of augmentation selectivity and measured retrieval time. As shown in Figure F2, the only factor affecting retrieval time within a data set was the number of augmentations in the cue ($R^2 \approx 1$), achieving a maximum of about 0.5 msec. for k=10.

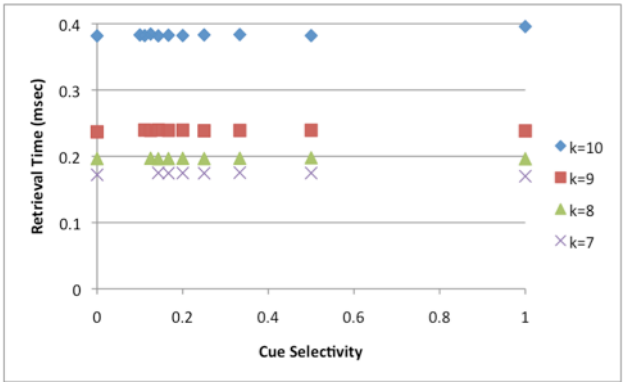

Figure F1. Synthetic Selectivity Sweep Results

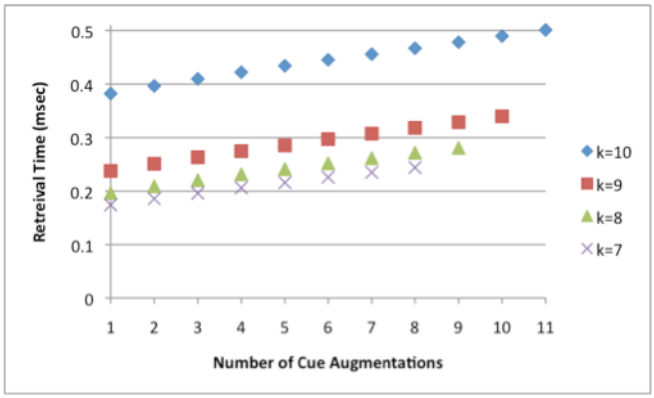

Figure F2: Synthetic Cue Sweep Results

Failure Sweep

For our mechanism, retrieval failure is the algorithmic worst-case, as it must examine and fail to verify all candidate elements. We constructed our last experiment to measure retrieval time for cues that fail only after examining significant proportions of the elements in the KB. While our mechanism minimizes the chance of this situation, these results are useful to set an expectation for the unlikely worst-case retrieval time in any given semantic store. As shown in Figure F3, the number of inspected candidate elements was the only factor affecting retrieval time, independent of the data set. Because the time is linear in the number of candidates, and not the total number of KB elements, our mechanism, for even worst worst-case cues, scales to arbitrarily large data sets when cue augmentations are sufficiently selective.

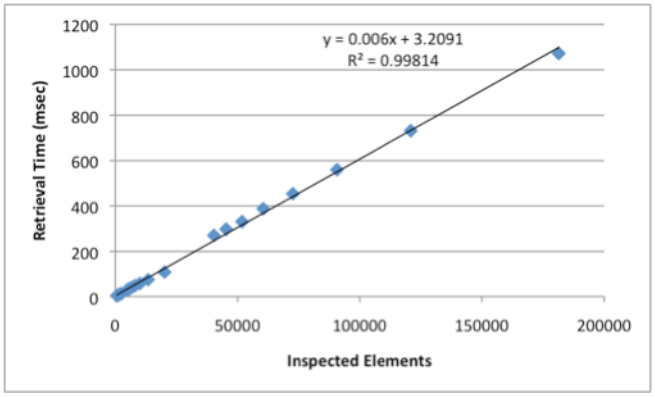

Figure F3: Synthetic Failure Sweep Results

References

Agrawal, R. Srikant, R. (1994). Fast Algorithms for Mining Association Rules in Large Databases. *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB).*

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

Anderson, J. R. (1991). The Place of Cognitive Architectures in a Rational Analysis. *Architectures for Intelligence*. Hillsdale, NJ: Lawrence Erlbaum Associates.

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

Anderson, J. R., Matessa, M. P. (1992). Explorations of an Incremental, Bayesian Algorithm for Categorization. *Machine Learning* 9. 275-308.

Anderson, J. R., Matessa, M. P. (1998). The Rational Analysis of Categorization and the ACT-R Architecture. *Rational Models of Cognition*. 197-217. Oxford: Oxford University Press.

Anderson, J. R., Schooler, L. J. (1991). Reflections of the Environment in Memory. *Psychological Science* 2. 396-408.

Andoni, A., Indyk, P. (2008). Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Communications of the ACM* 51 (1). 117-122.

Bach, J. (2003). The MicroPsi Agent Architecture*. Proceedings of the 5th International Conference on Cognitive Modeling (ICCM)*.

Berthold, M. R., Brandes, U., Kötter, T., Mader, M., Nagel, U., Thiel, K. (2009). Pure Spreading Activation is Pointless. *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM)*.

Chaudhuri, S. (1998). An Overview of Query Optimization in Relational Systems. *Proceedings of the 17th ACM Symposium on the Principles of Database Systems (PODS)*.

Chaudhuri, S., Narasayya, V., Agrawal, S. (2000). Automated Selection of Materialized Views and Indexes for SQL Databases. *Proceedings of the 26th International Conference on Very Large Databases (VLDB)*.

Chong, R. (2003). The Addition of an Activation and Decay Mechanism to the Soar Architecture. *Proceedings of the 5th International Conference on Cognitive Modeling (ICCM)*.

Collins, A. M., Quillian, M. R. (1969). Retrieval Time from Semantic Memory. *Journal of Verbal Learning and Verbal Behavior* 8. 240-247.

Collins, A. M., Loftus, E. F. (1975). A Spreading-Activation Theory of Semantic Processing. *Psychological Review* 82 (6). 407-428.

Combi, C., Shahar, Y. (1997). Temporal Reasoning and Temporal Data Maintenance in Medicine: Issues and Challenges. *Computers in Biology and Medicine* 27 (5). 353-368.

Crawford, J. M., Kuipers, B. J. (1991). Algernon – A Tractable System for Knowledge Representation. *SIGART Bulletin* 2 (3). 35-44.

Cummins, L., Bridge, D. (2009). Maintenance by a Committee of Experts: The MACE Approach to Case-Base Maintenance. *Proceedings of the 10th International Conference on Case-Based Reasoning (ICCBR)*.

Daengdej, J., Lukose, D., Tsui, E., Beinat, P., Prophet, L. (1996). Dynamically Creating Indices for Two Million Cases: A Real World Problem. *EWCBR LNCS* 1168. 105-119.

Davidsson, P. (1995). On the Concept of Concept in the Context of Autonomous Agents. *Proceedings of the Second World Conference on the Fundamentals of Artificial Intelligence*.

Davis, R., Shrobe, H., Szolovits, P. (1993). What is a Knowledge Representation? *AI Magazine* 14 (1). 17-33.

Davis, M., Whalen, P. J. (2001). The Amygdala: Vigilance and Emotion. *Molecular Psychiatry* 6. 13-34.

Derbinsky, N., Gorski, N. A. (2010). Exploring the Space of Computational Memory Models. *Proceedings of the Symposium on Human Memory for Artificial Agents (36th AISB)*.

Derbinsky, N., Laird, J. E. (2009). Efficiently Implementing Episodic Memory. *Proceedings of the 8th International Conference on Case-Based Reasoning (ICCBR)*.

Derbinsky, N., Laird, J. E. (2010). Extending Soar with Dissociated Symbolic Memories. *Proceedings of the Symposium on Human Memory for Artificial Agents* (*36th AISB)*.

Derbinsky, N., Laird, J. E., Smith, B. (2010). Towards Efficiently Supporting Large Symbolic Delcarative Memories. *Proceedings of the 10th International Conference on Cognitive Modeling (ICCM)*.

Dolan, R. J. (2002). Emotion, Cognition, and Behavior. *Science* 298. 1191-1194.

Doorenbos, R. B. (1995). *Production Matching for Large Learning Systems*. PhD Thesis, Carnegie Mellon.

Douglass, S., Ball, J., Rodgers, S. (2009). Large Declarative Memories in ACT-R. *Proceedings of the 9th International Conference on Cognitive Modeling (ICCM)*.

Douglass, S. A., Myers, C. W. (2010). Concurrent Knowledge Activation Calculation in Large Declarative Memories. *Proceedings of the 10th International Conference on Cognitive Modeling (ICCM)*.

Emond, B. (2006). WN-LEXICAL: An ACT-R Module Built from the WordNet Lexical Database. *Proceedings of the 7th International Conference on Cognitive Modeling (ICCM)*.

Forbus, K., Gentner, D., Law, K. (1995). MAC/FAC: A Model of Similarity-based Retrieval. *Cognitive Science* 19 (2). 141-205.

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

Fox, S., Leake, D. (1995). Using Introspective Reasoning to Refine Indexing. *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*.

Franklin, S., Patterson, F. G. Jr. (2006). The LIDA Architecture: Adding New Modes of Learning to an Intelligent, Autonomous, Software Agent. *Integrated Design and Process Technology* (*IDPT-2006)*.

Gennari, J. H., Langley, P., Fisher, D. (1989). Models of Incremental Concept Formation. *Artificial Intelligence* 40. 11-61.

Gorski, N. A., Laird, J. E. (2009). Learning to Use Episodic Memory. *Proceedings of the 9th International Conference on Cognitive Modeling (ICCM)*.

Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., Pirahesh, H. (1997). Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *Data Mining and Knowledge Discovery* 1. 29-53.

Hassabis, D., Maguire, E. A. (2007). Deconstructing Episodic Memory with Construction. *Trends in Cognitive Sciences* 11. 299-306.

Huffman, S. B. (1994). *Instructable Autonomous Agents*. PhD Thesis, University of Michigan.

Huffman, S. B., Laird, J. E. (1995). Flexibly Instructable Agents. *Artificial Intelligence Research* 3. 271-324.

Jære, M., Aamodt, A., Skalle, P. Representing Temporal Knowledge for Case-Based Prediction. *ECCBR LNCS* 2416. 174-234.

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., Koss, F. V. (1999). Automated Intelligent Pilots for Combat Flight Simulation. *AI Magazine* 20 (1). 27-41.

Kolodner, J. L. (1983a). Maintaining Organization in a Dynamic Long-term Memory. *Cognitive Science* 7 (4). 243-280.

Kolodner, J. L. (1983b). Reconstructive Memory: A Computer Model. *Cognitive Science* 7 (4). 281-328.

Kolodner, J. (1992). An Introduction to Case-Based Reasoning. *Artificial Intelligence Review* 6 (1). 3-34.

Kriegel, H., Pötke, M., Seidl, T. (2000). Managing Intervals Efficiently in Object-Relational Databases. *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB)*.

Laird, J. E. (2001). It Knows What You're About to Do: Adding Anticipation to a Quakebot. *Proceedings of the Fifth International Conference on Autonomous Agents*.

Laird, J. E. (2008). Extending the Soar Cognitive Architecture. *Proceedings of the First Conference on Artificial General Intelligence (AGI)*.

Laird, J. E. (2009). Towards Cognitive Robotics. *SPIE Defense and Sensing Conferences*.

Laird, J. E., Derbinsky, N. (2009). A Year of Episodic Memory. *Proceedings of the Workshop on Grand Challenges for Reasoning from Experiences* (*21st IJCAI)*.

Laird, J. E., Rosenbloom, P. (1996). The Evolution of the Soar Cognitive Architecture. *Mind Matters: A Tribute to Allen Newell*, Mahwah: Lawrence Erlbaum Associates, Inc.

Laird, J. E., Wray III, R. E. (2010). Cognitive Architecture Requirements for Achieving AGI. *Proceedings of the 3rd Conference on Artificial General Intelligence (AGI)*.

Laird, J. E., Xu, J. Z., Wintermute, S. (2010). Using Diverse Cognitive Mechanisms for Action Modeling. *Proceedings of the 10th International Conference on Cognitive Modeling (ICCM).*

Langley, P., Laird, J. E., Rogers, S. (2009). Cognitive Architectures: Research Issues and Challenges. *Cognitive Systems Research* 10. 141-160.

Lenat, D. (1995). CYC: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM* 38 (11). 33-38.

Lenz, M., Burkhard, H. (1996). Case Retrieval Nets: Basic Ideas and Extensions. *KI 1996 LNCS* 2689. 246-260.

Lesk, M. (1986). Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. *Proceedings of the 5th International Conference on Systems Documentation (SIGDOC-86)*.

Lonsdale, D., Rytting, C. A. (2001). Integrating WordNet in NL-Soar. *Proceedings of the Workshop on WordNet and Other Lexical Resources: Applications, Extensions, and Customizations (2nd NAACL).*

Ma, J., Knight, B. (2003). A Framework for Historical Case-Based Reasoning. *ICCBR LNCS* 2689. 246-260.

Marinier, R. P., Laird, J. E. (2008). Emotion-Driven Reinforcement Learning. *Proceedings of the 30th Annual Conference of the Cognitive Science Society (CogSci)*.

Marinier, R. P., Laird, J. E., Lewis, R. L. (2009). A Computational Unification of Cognitive Behavior and Emotion. *Journal of Cognitive Systems Research* 10 (1). 48-69.

McCarthy, J., Minsky, M. L., Rochester, N., Shannon, C. E. (1955). *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence*. <http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html>.

McCarthy, J. (2007). From Here to Human-Level AI. *Artificial Intelligence* 171 (18). 1174-1182.

Medin, D. L., Smith, E. E. (1984). Concepts and Concept Formation. *Annual Review of Psychology* 35. 113-138.

Michalski, R. S., Stepp, R. (1983). Learning from Observation: Conceptual Clustering. *Machine Learning: An Artificial Intelligence Approach*. Tioga: Palo Alto, CA.

Miller, G. A. (1956). The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information. *Psychological Review* 63 (2). 81-97.

Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM* 38 (11). 39-41.

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

Miranker, D. P. (1987). TREAT: A Better Match Algorithm for AI Production Match Systems. *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI)*.

Mohan, S., Laird, J. E. (2010). Relational Reinforcement Learning in Infinite Mario. *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*.

Navigli, R. (2009). Word Sense Disambiguation: A Survey. *ACM Computing Surveys* 41 (2).

Nayak, P., Gupta, A., Rosenbloom, P. (1988). Comparison of the Rete and Treat Production Matchers for Soar (A Summary). *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI)*.

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

Newell, A. (1991). OH 227. *Oral History Interview by Arthur L. Norberg*. Pittsburgh, Pennsylvania. Charles Babbage Institute, University of Minnesota, Minneapolis.

Newell, A., Simon H. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.

Niles, I., Pease, A. (2001). Towards a Standard Upper Ontology. *Proceedings of the Second International Conference on Formal Ontology in Information Systems (FOIS)*.

Nuxoll, A. M. (2007). *Enhancing Intelligent Agents with Episodic Memory*. PhD Thesis, University of Michigan.

Nuxoll, A. M., Laird, J. E. (2007). Extending Cognitive Architecture with Episodic Memory. *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI)*.

Nuxoll, A. M., Laird, J. E., James, M. (2004). Comprehensive Working Memory Activation in Soar. *Proceedings of the 6th International Conference on Cognitive Modeling (ICCM)*.

Nuxoll, A. M., Tecuci, D., Ho, W. C., Wang, N. (2010). Comparing Forgetting Algorithms for Artificial Episodic Memory Systems. *Proceedings of the Symposium on Human Memory for Artificial Agents (36th AISB)*.

Owens, J., Bower, G. H. (1979). The "Soap Opera" Effect in Story Recall. *Memory & Cognition* 7 (3). 185-191.

Patterson, D., Galushka, M., Rooney, N. (2004). An Effective Indexing and Retrieval Approach for Temporal Cases. *Proceedings of the 17th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*.

Patterson, D., Rooney, N., Galushka, M. (2003). Efficient Retrieval for Case-Based Reasoning. *Proceedings of the 16th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*.

Ramakrishnan, R. Gehrke, J. (2003). *Database Management Systems*. 3rd. McGraw-Hill, Inc.

Sánchez-Marré, M., Cortés, U., Martínez, M., Comas, J., Rodríguez-Roda, I. (2005). An Approach for Temporal Case-Based Reasoning: Episode-based Reasoning. *ICCBR LNCS* 2620. 465-476.

Schacter, D. L. (1999). The Seven Sins of Memory: Insights from Psychology and Cognitive Neuroscience. *American Psychologist* 54 (3). 182-203.

Schooler, L., Anderson, J. (1997). The Role of Process in the Rational Analysis of Memory. *Cognitive Psychology* 32 (3). 219-250.

Simon, H. (1991). Cognitive Architectures and Rational Analysis: Comment. *Architectures for Intelligence*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Singhal, A. (2001). Modern Information Retrieval: A Brief Overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* 24.

Smith, E. E., Adams, N., Schorr, D. (1978). Fact Retrieval and the Paradox of Interference. *Cognitive Psychology* 10. 438-464.

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

Stottler, R., Henke, A., King, J. (1989). Rapid Retrieval Algorithms for Case-Based Reasoning. *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI)*.

Stracuzzi, D., Li, N., Cleveland, G., Langley, P. (2009). Representing and Reasoning over Time in a Unified Cognitive Architecture. *Proceedings of the 9th International Conference on Cognitive Modeling (ICCM)*.

Strosnider, J. K., Paul, C. J. (1994). A Structured View of Real-Time Problem Solving. *AI Magazine* 14 (2). 45-66.

Sun, R. (2003). A Tutorial on CLARION 5.0.
http://www.cogsci.rpi.edu/~rsun/sun.tutorial.pdf

Sun, R. (2006). The CLARION Cognitive Architecture: Extending Cognitive Modeling to Social Simulation. *Cognition and Multi-Agent Interaction*.

Tambe, M., Newell, A., Rosenbloom, P. S. (1990). The Problem of Expensive Chunks and its Solution by Restricting Expressiveness. *Machine Learning* 5. 299-348.

Tanner, B., White, A. (2009). RL-Glue: Language-Independent Software for Reinforcement-Learning Experiments. *Journal of Machine Learning Research* 10. 2133-2136.

Tecuci, D., Porter, B. (2007). A Generic Memory Module for Events. *Proceedings of the 20th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*.

Terrovitis, M., Passas, S., Vassiliadis, P., Sellis, T. (2006). A Combination of Trie-trees and Inverted Files for the Indexing of Set-valued Attributes. *Proceedings of the 15th Conference on Information and Knowledge Management (CIKM)*.

Tsatsaronis, G., Vazirgiannis, M., Androutsopoulos, I. (2007). Word Sense Disambiguation with Spreading Activation Networks Generated from Thesauri. *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*.

Tsatsaronis, G., Varlamis, I., Vazirgiannis, M. (2008). Word Sense Disambiguation with Semantic Networks. *Proceedings of the 11th International Conference on Text, Speech and Dialogue*.

Tulving, E. (1983). *Elements of Episodic Memory*. Oxford: Clarendon Press.

Wess, S., Althoff, K., Derwand, G. Using k-d Trees to Improve the Retrieval Step in Case-Based Reasoning. *EWCBR LNCS* 837. 167-181.

Williams, M. D., Hollan, J. D. (1981). The Process of Retrieval from Very Long-Term Memory. *Cognitive Science* 5. 87-119.

Wilson, D., Martinez, T. (2000). Reduction Techniques for Instance-Based Learning Algorithms. *Machine Learning* 38 (3). 257-286.

Wintermute, S., Xu, J., and Laird, J. E. (2007). SORTS: A Human-Level Approach to Real-Time Strategy AI. *Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*.

Xu, J. Z., Laird, J. E. (2010). Instance-Based Online Learning of Deterministic Relational Action Models. *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*.

Zobel, J., Moffat, A. (2006). Inverted Files for Text Search Engines. *ACM Computing Surveys* 38 (2).