Preliminary Evaluation of Long-term Memories for Fulfilling Delayed Intentions

Justin Li and **John Laird**
University of Michigan
2260 Hayward Street
Ann Arbor, MI 48109-2121 USA
{justinnh,laird}@umich.edu

Abstract

The ability to delay intentions and remember them in the proper context is an important ability for general artificial agents. In this paper, we define the functional requirements of an agent capable of fulfilling delayed intentions with its long-term memories. We show that the long-term memories of different cognitive architectures share similar functional properties and that these mechanisms can be used to support delayed intentions. Finally, we do a preliminary evaluation of the different memories for fulfilling delayed intentions and show that there are trade-offs between memory types that warrant further research.

Introduction

The ability to manage multiple goals has always been a desired capability of artificial intelligence systems. However, there has not been a lot of research on the ability of artificial agents to fulfill delayed intentions — goals that the agent cannot currently act on, but must remember and later recall in order to fulfill. Given the complex environments agents could exist in, the suspension of goals due to either conflicting goals or inopportune environments is not uncommon. The ability to notice that a suspended goal should now be pursued and recall the appropriate actions to take is therefore critical to general artificial agents.

This research on delayed intentions follows recent trends in cognitive architecture research on how agents could use different types of memories. Many architectures incorporate memories of different encodings, interfaces, and longevities. A common theme among them is the separation of short-term and long-term memories. Knowledge in the former is immediately accessible to the agent for further reasoning, but is computationally expensive to maintain and decays quickly if unused; knowledge in the latter is more stable, but requires longer time spans to create or recall. An agent capable of delayed intentions must effectively use both types of memories. On the one hand, the intentions themselves must be stored in long-term memory to persist for any significant length of time. On the other hand, the agent must have quick access to the intentions, such that it can take action at the right time. Understanding delayed intention management thus requires understanding how it relates to the long-term memories of agents and of different architectures.

The contribution of this paper is threefold. First, we define requirements for agents attempting to support delayed intentions with long-term memories. Second, we survey long-term memory systems across multiple cognitive architectures, showing that they can be classified into groups with similar functional profiles. Each of these groups provide different ways of fulfilling delayed intentions. Finally, as an evaluation of these methods, we've implemented agents using the memories in Soar, such that their accuracy and scalability can be compared.

Delayed Intentions in Artificial Agents

The ability to fulfill delayed intentions has been a growing field of study in psychology, where it is called *prospective memory* (McDaniel and Einstein 2007). We use this term loosely to mean both a problem and the capability of agent to solve that problem: how does an agent remember to take action if there is a delay between its intent to act and its ability to act, given that it is actively engaged in background tasks in the mean time? Such prospective memory tasks are common in daily life: the intent to buy milk on the way home after work is one such example, as is the intent to attend a meeting next Wednesday at 3pm. More concretely, we say that an intention has two components: a *target*, the context in which an agent should act; and an *action*, which the agent must perform to fulfill the intention. A *delayed* intention is one which the target for the intention is not present when the intention is formed.

A major challenge in supporting delayed intentions is how to maintain the intentions in memory. As mentioned above, memories in cognitive architectures are often divided by the longevity of the knowledge; conceptually, short-term memory is used for tasks the agent is currently working on. This is often enforced by architectural mechanisms that remove objects from short-term memory when they fall below a certain activation threshold (Chong 2003). Since the length of delay of an intention is unknown, the simplest scheme is to store intentions in long-term memory, then only retrieve them into short-term memory when they are needed. Although it is also possible to keep the intentions in short-

term memory, there are two disadvantages to this approach. First, this violates expected usage of short-term memory — a long-lived agent may have a large number of intentions at the same time, and it is unlikely that the agent will have all intentions in short-term memory while performing other tasks. Second, decision making with large states has been known to be computationally expensive; even an optimized algorithm such as RETE suffers as the number of items it could match grows (Forgy 1979). For these reasons, we only explore methods that keep delayed intentions in long-term memory.

The idea of studying the interaction between goals and agent memory is not new. There has been several studies using ACT-R to model certain delayed intention processes in the human mind. One study (Lebiere and Lee 2002) modeled the Intention Superiority Effect, which describes a phenomenon in humans where intended actions are quicker to retrieve than other memories. Another study (Elio 2006) looked at whether two different models of prospective memory correlated with human data. Both studies rely on the spreading of activation within ACT-R's declarative memory to bring delayed intentions to the agent. Although this is a plausible method for agents to retrieve goals from long-term memory, neither was done in the context of background tasks. The addition of other processing in the agent would disrupt intention structures, allowing intentions to decay in activation. A general approach to delayed intentions requires a more stable method of recalling intentions.

There is also more general work on how agents might manage goals. The life-cycle of goals has been explored in the context of belief-desire-intention (BDI) agents, where goals could be suspended if the context of the goal is invalid, then become an option for the agent to pursue when the context is appropriate. A goal being pursued is first active, and when it is finally completed, it is dropped (Braubach et al. 2005) Although this describes the general progression of delayed intentions, it is unclear what mechanisms agents need to manage goals in this manner, in particular, what the memory systems are used to efficiently find satisfiable goals. There has also been work in augmenting BDI agents with different memory systems, but the interaction between goals and memories were left unspecified (Brom and Lukavský 2008).

To obtain a sufficiently broad sample of long-term memories, we compare the following cognitive architectures in this work, using these primary references: ACT-R (Anderson 2007), CLARION (Sun 2006), GLAIR (Shapiro and Bona 2009), HOMER (Vere 1991), LIDA (Snaider, McCall, and Franklin 2011), Polyscheme (Cassimatis et al. 2007), and Soar (Laird 2008).

Functional Requirements of Delayed Intentions

Before we consider the properties of long-term memories and how they might be used to fulfill delayed intentions, we need to consider the obvious alternative of a separate goal memory. Many architectures and agent frameworks have built-in goal/intention structures. Traditionally, however, these structures are in the form of goal stacks, which favor hierarchical goals in supergoal-subgoal relationships. This assumption suggests that such structures are inappropriate, as delayed intentions may be embedded in background tasks that have no relationship to the intention. This mismatch notwithstanding, recent work has suggested that this restricted form of goal memory is unlikely in humans (Altmann and Trafton 1999). It is more probable that goals are simply another element in the agent's memory, and are treated in the same way. This hypothesis is strengthened by findings of single dissociation between prospective and retrospective memory; that is, the former uses general purpose memories in its operation (Burgess and Shallice 1997). We follow this result and limit ourselves to only using existing memory mechanisms provided by architectures, although using a special purpose memory is a possibility for future study.

With this restriction, we can define the function requirements of an agent capable of fulfilling delayed intentions. We follow the life cycle of an intention (Braubach et al. 2005): suspension and storage, recognition, pursuance, and completion.

Suspension and Storage When an intention is first formed, the agent must store it into long-term memory. This requires that the agent be able to represent both the target and the actions of the intention as well as to add to its long-term memory with this representation.

Recognition In between the formation of the intention and the appearance of the target, the intention may not be immediately accessible; despite this, the agent must recognize that the target to an intention is being presented. The agent's behavior must therefore either be directly influenced by long-term memory structures, or else include accesses and retrievals from memory in a tight loop.

Pursuance When the agent decides to fulfill the intention, it must be able to retrieve the associated actions from its long-term memory.

Completion Finally, once the intention has been completed, the agent must modify its behavior such that it does not attempt to pursue the same intention if the target appears again in the future. This requires the agent to maintain the status of the intention and be able to change its memory to reflect that status.

Properties of Long-Term Memories

Although multiple cognitive architectures have been developed over the years, many share similar long-term memory mechanisms. Particularly relevant for supporting delayed intentions are the following properties of long-term memories:

Encoding What type of knowledge is stored? How is this knowledge represented, both to the agent and to the memory system? Can the agent reason over this knowledge, or is there no declarative representation available for the agent?

Storage Is storage deliberate (agent-initiated) or automatic (architectural)? If storage is deliberate, when is the agent allowed to add to memory? If storage is automatic, what triggers automatic storage?

Retrieval How can the agent retrieve knowledge? What information does the agent need to cue retrieval of knowledge? How can the agent effectively find relevant information, given the rich features of the environment and a large memory?

Modification If previously stored knowledge is wrong, how can the agent correct this knowledge, if at all? Can irrelevant knowledge be removed from memory, and is this process deliberate or automatic?

The major classification we make based on long-term memories is the type of knowledge encoded. Within this division, the similar uses of memory lead to shared solutions in the other questions listed above. In this way, we build up profiles of memory systems, which we then analyze for suitability for supporting delayed intentions. In the following subsections, we examine in detail different implementations of procedural, semantic, and episodic memories. Although there are other types of memories to consider, such as life-long allo-centric spatial memory (Brom and Lukavský 2008), their role in supporting delayed intentions is unclear.

For reference, a summary of the properties of different memory systems is given in Figure 1.

Procedural Memory

Procedural memory is the memory for how an agent should behave. Conceptually, knowledge in procedural memory can often be described by if-then rules, as actions which the agent should take if certain conditions are true. Framed in this way, any learning agent must ultimately modify its procedural memory, either directly or indirectly. The direct method of altering behavior is to change procedural memory via the addition or removal of rules. Often, however, there are other architectural mechanisms which dictate the application of rules, such as whether one rule should be applied instead of others. Changing these preferences would also alter the agent's behavior, but in a less direct manner.

Since procedural memory is intimately tied to the behavior of the agent, the exact encodings used in procedural memory are as numerous as the number of cognitive architectures. Despite this difference, architectures share function similarities in their procedural memory. First, procedural memories all modify the state of the agent's short-term memory; the "rules" in procedural memory match structures in short-term memory, after which the actions of the rule will apply (barring architectural constraints on rule firing). Although this may include buffers from which knowledge in other long-term memory knowledge can be retrieved, there is often no direct access to semantic and episodic memory.

Second, although the rules are encoded differently, procedural memory is often implicit: the agent has no declarative access to the rules. Although some architectures allow agents to directly modify their procedural memory (e.g. GLAIR), this is uncommon. Instead, most architectures support a limited mechanism for adding knowledge to their procedural memory, often in the form of compressing several reasoning steps into one (e.g. ACT-R, Soar). The primary effect of either learning method is that the agent becomes more efficient: solving the same problem a second time takes less resources than the first run through. In addition to modeling learning by contiguity (Anderson 2007), procedural learning could also be used to learn semantic knowledge through a process called data-chunking (Rosenbloom, Laird, and Newell 1987). With this approach, agents represent semantic knowledge procedurally.

However, the lack of declarative access also means that procedural memory is append-only, with the agent unable to modify or remove rules once learned. To the casual observer, this may suggest that agents cannot recover from errors in learning. In reality, however, architectures often have other mechanisms to compensate (Laird 1988). For example, the agent might learn other rules which control the application of the first one. Errors might also be correctable using subsymbolic mechanisms, such as reinforcement learning or neural networks, which can devalue rules such that they never apply.

These properties of procedural memory create potential drawbacks in its use for supporting delayed intentions. The inability for an agent to remove rules makes it difficult to modify memory after an intention is fulfilled. Since rules in many architectures persist for the lifetime of the agent, whenever the target to a fulfilled intention appears, the agent will attempt to act on it again. This creates a burden on the system, and can become costly when an agent has fulfilled thousands of intentions. A second drawback to using procedural memory is the lack of declarative access to rules. In the example of buying milk, the agent might find itself visiting the grocery store for a separate errand, when it is desirable to also complete the delayed intention. Without declarative access to procedural memory, however, the agent lacks even the knowledge that it has an intention, nevermind what the target or action of the intention is. Similarly, the agent would not be able to remove an intention if it is no longer relevant.

Semantic Memory

Semantic memory encodes the common-sense notion of knowledge. It stores generalized facts and statements about the world, without contextual information of when the agent learned this knowledge. Traditionally within the artificial intelligence community, the distinction between short-term working memory and long-term semantic memory is vague. Logic-based architectures often use a single memory to store both the agent's state as well as semantic domain knowledge; however, this is neither psychologically plausible nor efficient, as reasoning by the agent often slows down as short-term memory size increases (Wang and Laird 2006). Some architectures continue to combine short-term working memory and long-term semantic memory in the same system (e.g. GLAIR, HOMER), with no distinction between the two. Even for such systems, there is often the concept of retrievals from memory, in contrast to the

Architecture	Procedural Memory		Semantic Memory		Episodic Memory	
	Declarative	Removable	Storage	Retrieval method	Storage	Retrieval Method
ACT-R	No	No	Automatic	Under-specified query	*N/A*	*N/A*
CLARION	No	No	Automatic	Associative search	Automatic	By time
GLAIR	Yes	Yes	Deliberate	Inference	*N/A*	*N/A*
HOMER	Yes	*unknown*	Automatic	Inference	Automatic	Automatic Association
LIDA	No	No	Automatic	Associative search	Automatic	Associative Search
Polyscheme	No	No	Deliberate	Various	*N/A*	*N/A*
Soar	No	No	Deliberate	Under-specified query	Automatic	Over-specified query

Table 1: Summary of the capabilities of memory systems in different cognitive architectures. Procedural memory is declarative if the agent can reason about rules, and removable if rules can be deleted. For semantic and episodic memories, storage is how the agent stores knowledge to memory, and retrieval method is the type of searches possible. An under-specified query is one where the memory object must contain all searched-for attributes, while an over-specified query allows attributes to be missing. Some notes on particular architectures and mechanisms. CLARION, due to its dual representation memory, can perform both symbolic matching as well as a subsymbolic associative search. HOMER's episodic memory includes a daemon that notifies the agent whenever current events have occurred before. LIDA's sparse distributed memory uses associative search, the details of which can be found elsewhere (Kanerva 1993). The Polyscheme architecture has multiple specialist modules, each of which may implement separate knowledge retrieval algorithms.

automatic matching of procedural rules. The requirement of agent deliberation in making the retrieval is what unites what we call semantic memory.

Outside of the encoding of knowledge, the semantic memories of different architectures are remarkably similar. Agents often have the ability to create new short-term structures to be stored into semantic memory, although some systems have mechanisms which automatically store new information into semantic memory (e.g. ACT-R). For retrieval, agents can retrieve elements of semantic memory either directly or through some interface in working memory. A retrieval requires the agent to create a cue, which is a description of the desired properties of the object stored in memory. Semantic memory then returns an object that includes all the properties described, subject to architectural bias if multiple objects match the description. Finally, agents can also change the contents of semantic memory, or optionally entirely remove knowledge from the store.

An advantage of using semantic memory for delayed intentions is that it provides a declarative representation to the agent, a capability not afforded by procedural memory. The ability for the agent to modify semantic memory also promises to at least reduce the cost of completing intentions, as obsolete intentions can simply be removed. However, the separation of semantic memory from short-term memory — on which procedural memory matches — raises a different problem. Although the agent has a declarative representation of the intention, it cannot be matched directly by procedural memory. The intention must therefore be retrieved before the target appears to allow the agent to act on it. This problem is exacerbated by the requirement that semantic memory searches cannot be over-specified, meaning that the agent must know a subset of the target or the action for efficient retrieval. A possible solution to this problem is to relax the search constraint; however, this requires the search to process superset queries (Terrovitis et al. 2006), making it strictly more expensive. A general solution to this problem remains an area of research.

Episodic Memory

Episodic memory was first described by Tulving (Tulving 1983) as a memory for the experiences of an agent. What distinguishes episodic memory from other memories is that it contains temporal information — all knowledge is ordered by time. This ordering allows agents to "mental time travel", that is to re-experience previous episodes. Although the use of episodic memory in artificial agents is the least explored of the long-term memories discussed, the general consensus is that it allows the agent to capture knowledge that it may not know is important at the time. This includes the possible need to revisit previous experiences and several other advantages (Nuxoll and Laird 2007), although many remain unexplored in literature. Similar, although forgetting or otherwise removing information from episodic memory is understood to be useful, a systematic investigation remains to be done.

As its name suggests, the different episodic memory systems are united in their inclusion of a timestamp for each object. That aside, the main motivation for episodic memory function is the ability to retrieve unknown contextual knowledge. Automatic storage of episodes is essential for this, as is the ability to search through the episode store associatively. By this, we mean that the query for the search can be over-specified — the episode returned may contain only some of the attributes described. For example, the intention to meet at Wednesday at 3pm could be retrieved by both a search for "3pm" as well as a search for "Thursday 3pm", as the "3pm" attribute is shared by the query and the result. The agent can therefore agent find similar objects in its history, which the HOMER architecture directly provides with a daemon. Of the different episodic memories, the sparse distributed memory of LIDA deserves special mention, as its retrieval iterates through potential objects until it stabilizes, resulting in a gist representation which may not match any single object in memory. Other architectures also store temporal information: the activation of ACT-R's declarative memory serves as a rough indication of whether an object was seen recently, while the GLAIR

architecture uses temporal logic as contextual information on its internal states. However, neither architecture offers any mechanisms that take advantage of the temporal element outside of what their semantic memories already provide.

As with semantic memory, using episodic memory for delayed intentions allows the agent to eliminate the life-time cost associated with using only procedural memory. Episodic memory also provides the agent with a declarative presentation of intentions for reasoning, another advantage it shares with semantic memory. The associative nature of episodic memory search allows the agent to find intentions with the currently perceived features as a target. This, however, presents a two related problems. First, because, the associative nature is not discriminative, a search could return an already-fulfilled intention as it could ignore the "unfulfilled" status of the intention. Second, since the episode of the intention's formation remains in episodic memory, that episode could be returned despite the intention being fulfilled later in the agent's history. This is because knowledge is only true at the time of storage, but not necessarily at the time of retrieval. Solving these problem would require extra processing for the agent, or perhaps a more careful use of episodic memory.

Our Approaches

In this section we describe how we use the different memories to fulfill delayed intentions in the Soar cognitive architecture. We constrain ourselves to mechanisms provided by the current memories, and use architecture-specific mechanisms to build a functional system. Since the full design space of delayed intention systems has not been explored, these approaches should not be taken as the best way to overcome the challenges listed above, but as first attempts towards potential solutions.

Procedural Memory

To use procedural memory for delayed intentions, the agent must learn a data-chunk which contains both the target and the action of the intention. This is done by the agent internally creating the target features, then deciding to perform the actions because these features are perceived; the architecture compiles this reasoning process into a chunk, which is added to procedural memory. The agent then requires no deliberate reasoning to recognize the target, as the learned rule will apply automatically when the target is perceived. Similarly, the application of the rule provides the agent with the necessary actions to take, which makes retrieval unnecessary. Since the Soar architecture does not provide declarative access to procedural memory, the agent cannot delete the rule. Instead, to overcome the problem of modifying memory after the intention has been completed, the agent learns a second rule that negates the results of the first rule. Learned by a similar process, this rule is conditioned on the first rule being applied, and has the action of rejecting the actions of the first rule. When a previous target appears, the agent therefore attempts to act on the intention (the first rule), but then stops itself in the attempt (the second rule). The combined effect is that the agent only acts on intentions a single time.

This use of procedural memory inherits the disadvantages of procedural memory in general, namely, it does not provide the agent with a declarative representation of existing intentions and it imposes computation costs on the agent when rules have to be suppressed. Further more, data-chunking is an expensive process, taking multiple cycles of decision time, during which the agent could not process other tasks. On the positive side, the learned intention-specific rule guarantee that the actions of the intention are available to the agent at the right time, without extra retrievals from long-term memory.

Semantic Memory

The similarities between short-term memory and semantic memory means that the agent does not need to specially encode the intention. However, merely storing the intention in semantic memory is not sufficient for the agent to act on it. This is because the rules in procedural memory — which govern the agent's behavior — applies based only on structures in working memory, but cannot apply based on knowledge in semantic memory. The agent therefore needs a separate mechanism to trigger the retrievals of intentions from semantic memory. To do this, we fall back on procedural memory to provide the retrieval query. To properly encode the intention, the agent must learn a data-chunk in addition to storing the intention into semantic memory. This rule is conditioned on both the target of the intention, as before, as well as an identifier unique to the intention, which could be used to retrieve the intention directly. To recognize the target, the agent periodically predicts what features it might perceive in the near future, then retrieves the identifiers of intentions whose target contains those features. When the target does appear, the agent uses the unique identifier to retrieve the intention efficiently, from which it determines the actions to take. Finally, after the intention is fulfilled, the agent removes the identifier from working memory, thus preventing the intention-specific rule from applying again.

Although using semantic memory allows the agent to reason over formed intentions, falling back on data-chunks to provide a retrieval query means that the computational cost of data-chunking remain. Further more, the agent must perform an extra retrieval for the intention's action after the data-chunk applies, which again expends resources. The main benefit of using semantic memory is that it provides a long-term mapping between intentions and unique identifiers, which can be used to directly prevent rules from applying a second time, without the need to learn a second rule. The alternative of only using semantic memory for this mapping eliminates the cost of retrieving the action, but the other costs remain.

A final drawback of this approach is that it requires an unspecified capability to predict features of the environment. Although this is possible using episodic memory, by recalling a similar episode and using the previous outcome, the prediction errors are likely to be large. Additionally, it is unclear when the agent should make predictions and retrieve intentions, although literature provides some suggestions (Sellen et al. 1997). We leave both questions open, and

merely conclude that solutions would benefit this approach greatly.

Episodic Memory

Episodic memory, because it was designed to capture all information, requires a very different approach. Taking advantage of the automaticity of episodic memory, the agent does not have to take any action with the newly created intention; the intention is automatically stored into episodic memory. Instead of relying on rules to detect whether the target to an intention is present, the agent searches through its episodic memory for an unfulfilled intention with a target containing all the environmental features it currently perceives. The associative nature of the search returns to the agent the intention with the most feature overlap. To prevent episodic memory from returning an episode before the intention was fulfilled, we allow the agent to restrict the search to only the most recent version of objects. This guarantees that only episodes containing the latest states of intentions will be retrieved. Thus, the act of recognizing a target and retrieving the action is combined into the single process of searching episodic memory. After the agent fulfills the intention, it updates the status of the intention in working memory, which then gets automatically stored into episodic memory again, where the search restriction above prevents it from being retrieved in reaction to a different target.

As with semantic memory, our approach inherits the benefit of a declarative representation. We also overcome the problem of retrieving obsolete episodes by maintaining pointers to the most recent version of objects, which the agent restricts its search to. This leaves the problem of ignored "unfulfilled" attributes, which would require the agent to exhaustively iterate through matching intentions until an unfulfilled one is found (or no intentions are applicable). Using episodic memory is also expensive in other ways: since recognizing the target also requires search, a search must be done whenever the features of the environment changes to ensure that a target has not been missed. This cost grows quickly as the search becomes more expensive from frequently changing environments and long agent lifetimes.

Preliminary Evaluation

As a preliminary evaluation of how the different long-term memories perform in fulfilling delayed intentions, we implemented three agents in Soar emphasizing the advantages of the architecture's procedural, semantic, and episodic memories. Additionally, we implemented an agent using only working memory, which keeps all unfulfilled intentions in working memory and uses intention-independent rules to directly reason over them. Since this agent does not need to expend resources storing and retrieving intentions from working memory, it serves as an ideal to measure the other agents against.

We evaluate the agents on two dimensions. First, we look at the accuracy of the system, whether it allows agents to fulfill delayed intentions while otherwise engaged. Second, we look at whether the systems scale, and look at whether the agent remains reactive while coping with large numbers of intentions.

Agent	% Intentions Fulfilled
Working memory	82.66
Episodic memory	81.10
Procedural memory	40.42
Semantic memory	9.33

Table 2: The average accuracy of agents using different long-term memories, across multiple parameter settings. The working memory agent is the ideal; it fails to fulfill 100% of its intentions due to certain cues appearing while the agent is engaged.

Accuracy Evaluation

To evaluate how successfully an agent can fulfill delayed intentions, we designed an environment which probabilistically provides agents with intentions. Each intention has a target (which may be the same target as previous intentions) and an action, that of specifying the intention associated with the target. A certain number of timesteps after the intention is presented, the environment probabilistically presents targets to intentions. These targets persist for a short period before disappearing. Additionally, the environment presents the agent with background tasks, which also last for some number of timesteps. To simulate different degrees of cognitive load, we vary the frequency with which the environment presents intentions, targets, and cues, as well as the length for which targets are presented and background tasks persist (or rather, we vary the underlying distribution from which the lengths are drawn). We assume that the background task is more urgent, and therefore it restricts the time available to the agent for the storage and retrieval of intentions.

Within this domain, the results of the different long-term memories are shown in Table 2. Note that the working memory agent keeps all intentions in short-term memory. This is equivalent to the best case where the agent predicts all targets perfectly; an average case agent would perform worse due to prediction errors. Also, although the working memory agent represents the ideal agent, it does not achieve perfect performance. This is due to a subset of the targets only appearing while the agent is working on the background task, such that the agent did not have time to act on the intention. Compared to the working memory agent, only the episodic memory agent achieves similar levels of performance, while the procedural and semantic memory agents fall behind. The mediocre performance of the latter two memory systems comes from the strict requirement of retrieval via under-specified queries. Since the retrieval of any intention from these memories requires some knowledge of the target involved, the agent must spend time learning a production rule to provide this information, time which the agent does not have when background tasks are frequent. Semantic memory in particular incurs an extra penalty due to the need to retrieve the action of the intention after the data-chunk has applied. Episodic memory, on the

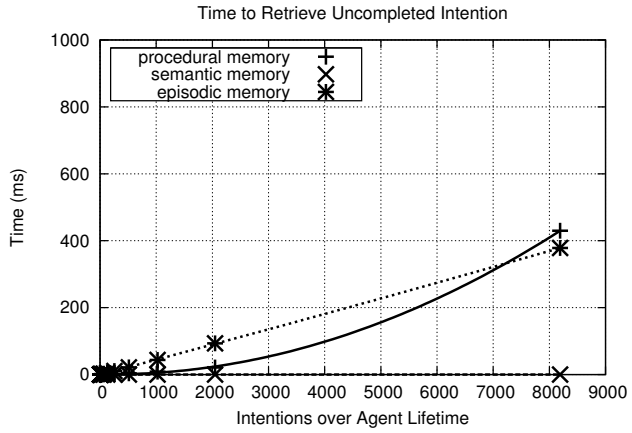

Figure 1: Time needed for different agents to retrieve a single unfulfilled intention. The working memory agent is not included as its intentions are already in working memory. Note that the semantic memory agent is practically indistinguishable from the x-axis. The equations from regression are shown in Table 3

other hand, was designed to retrieve objects with even the smallest association with the query, and therefore reacts to over-specification without problems. However, it fails to perform as well as working memory, since the memory system can only return one intention at a time, while the working memory agent can simultaneously respond to multiple targets.

Scalability Evaluation

A long-lived agent may form a large number of intentions over its lifetime. For a delayed intention system to be useful, it must remain efficient when dealing with many intentions both fulfilled and unfulfilled. To simulate this, we focus solely on increasing the number of intentions the agent must handle. The agents are presented with n intentions and the targets for $n - 1$ of them, such that a single unfulfilled intention remains. We then present the agent with the target of the remaining intention and measure the time required for the agent to fulfill it. Since the intended action is simple, the major factor in this metric is the amount of time it takes for the agent to retrieve the correct intention. Note that all the intentions presented to the agent have the same target — intuitively, this creates the worst case for the agents, as the target cannot be used to distinguish between intentions. The working memory agent is not shown, since the agent can directly access the actions of intentions stored in working memory.

The amount of time required to retrieve the last intention is show in Figure 1, and the regressions in Table 3 shows how the agents scale as more intentions are fulfilled. Note that the semantic memory agent has retrieval times very close to the x-axis. This implies that semantic memory is not affected at the scale of 10,000 intentions. The poor performance of procedural memory is as detailed above: since lots of working memory elements are added and removed for every intention, the agent must do work at least linear to the number of intentions, plus extra overhead cost. Finally, for the episodic memory agent, the agent must iterate through all intentions until it finds one that is unfulfilled. While this is true in general, we also discovered an inefficiency in Soar's episodic memory search which forces even the best case scenario to be linear in the number of intentions. Figure 2 contains some initial results in overcoming this inefficiency, showing roughly 25x speed-up in the best case. Note that this was evaluated outside the domain above, and are therefore not comparable with Figure 1. Nonetheless, this suggests that further work on improving episodic memory for delayed intentions has potential.

Agent	Ax^2 +	Bx +	C
Procedural memory	6.680e-06	-0.002	0.892
Semantic memory	-5.454e-10	5.936e-06	0.119
Episodic memory	2.135e-07	0.044	-0.256

Table 3: Regression for time different agents need to retrieve an unfulfilled intention. All agents scale on the same order as a quadratic function, although the quadratic term is small for the semantic memory agent.

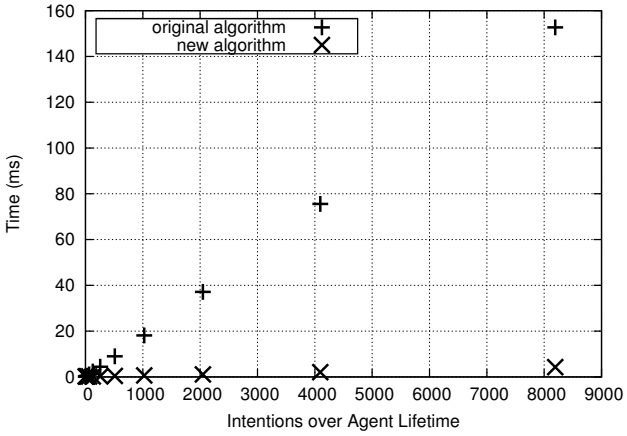

Figure 2: Time required to retrieve the last intention stored from episodic memory. Note that this evaluation was done as the best case for both implementations, and is not related to the domain in Figure 1.

Conclusions and Future Work

This evaluation shows that agents could use long-term memory for the purpose of fulfilling delayed intentions. While procedural memory is ill-suited due to its implicit and append-only nature, the declarative memories fare better. Semantic memory does poorly in accuracy because of the need to learn both procedural rules and semantic knowledge, but scales very well with the number of intentions. On the other hand, the performance of episodic memory matches that of short-term memory, but scales poorly. We are hopeful that more work on efficient episodic memory retrieval mechanisms will alleviate this problem, and that there are opportunities to leverage the advantages of the different memories to create a single, superior delayed intention

system.

Additionally, this work has focused on using existing mechanisms — the ability to data-chunk and the associative properties of search — to recognize the targets of intentions. As these mechanisms were not designed for this purpose, they impose large costs in terms of time and computation. It is possible that target recognition is not trigged by intention-specific mechanism at all, but by more general-purpose indicators. One possibility is the spreading of activation as has been explored in previous research (Elio 2006), where the appearance of the target causes the activation of the intention, making it likely to be retrieved. Psychology literature also suggests several mechanisms which leads agents to notice particular events, one example being a discrepancy between the expected and actual processing times (Whittlesea and Williams 2001). The agent then attributes this discrepancy to the event being a target to an intention, and therefore performs a retrieval. This serves as a middle ground between the rule-based promptings and the repeated deliberate queries used here, and has the potential to strike the balance between the need for preparation and the need for information capturing.

References

Altmann, E. M., and Trafton, J. G. 1999. Memory for goals: An architectural perspective. In *Proceedings of the 21st Annual Conference of the Cognitive Science Society (CogSci)*.

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

Braubach, L.; Pokahr, A.; Moldt, D.; and Lamersdorf, W. 2005. Goal representation for BDI agent systems. In Bordini, R.; Dastani, M.; Dix, J.; and Seghrouchni, A., eds., *Programming Multi-Agent Systems (ProMAS-3)*, volume 3346 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. 44–65.

Brom, C., and Lukavský, J. 2008. Episodic memory for human-like agents and humanlike agents for episodic memory. In *Papers from the AAAI Fall Symposium on Biologically Inspired Cognitive Architectures (BICA)*.

Burgess, P. W., and Shallice, T. 1997. The relationship between prospective and retrospective memory: Neuropsychological evidence. In Conway, M. A., ed., *Cognitive Models of Memory*. MIT Press.

Cassimatis, N.; Bugajska, M.; Dugas, S.; Murugesan, A.; and Bello, P. 2007. An architecture for adaptive algorithmic hybrids. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, 1520–1526. AAAI Press.

Chong, R. 2003. The addition of an activation and decay mechanism to the Soar architecture. In *Proceedings of the 5th International Conference on Cognitive Modeling (ICCM)*.

Elio, R. 2006. On modeling intentions for prospective memory performance. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society (CogSci)*, 1269–1274.

Forgy, C. L. 1979. *On the Efficient Implementation of Production Systems*. Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, PA, USA.

Kanerva, P. 1993. Sparse distributed memory and related models. In *Associative neural memories*. New York, NY, USA: Oxford University Press, Inc. 50–76.

Laird, J. E. 1988. Recovery from incorrect knowledge in Soar. In *Proceedings of the National Conference on Artificial Intelligence*. Cambridge, MA, USA: MIT Press. 615–620.

Laird, J. E. 2008. Extending the Soar cognitive architecture. In *Proceedings of the 1st Conference on Artificial General Intelligence (AGI)*.

Lebiere, C., and Lee, F. J. 2002. Intention superiority effect: A context-switching account. *Cognitive Systems Research* 3(1):57–65.

McDaniel, M. A., and Einstein, G. O. 2007. *Prospective Memory: An Overview and Synthesis of an Emerging Field*. Sage.

Nuxoll, A. M., and Laird, J. E. 2007. Extending cognitive architecture with episodic memory. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI)*, 1560–1565. Vancouver, Canada: AAAI Press.

Rosenbloom, P. S.; Laird, J. E.; and Newell, A. 1987. Knowledge level learning in soar. In *Proceedings of the sixth National conference on Artificial intelligence - Volume 2*, AAAI'87, 499–504. AAAI Press.

Sellen, A. J.; Louie, G.; Harris, J. E.; and Wilkins, A. J. 1997. What brings intentions to mind? An in situ study of prospective memory. *Memory* 5:483–507.

Shapiro, S. C., and Bona, J. P. 2009. The GLAIR cognitive architecture. In *Biologically Inspired Cognitive Architectures (BICA)*.

Snaider, J.; McCall, R.; and Franklin, S. 2011. The lida framework as a general tool for agi. In *Proceedings of the 4th Conference on Artificial General Intelligence (AGI)*.

Sun, R. 2006. *The CLARION Cognitive Architecture: Extending Cognitive Modeling to Social Simulation*. Cambridge University Press.

Terrovitis, M.; Passas, S.; Vassiliadis, P.; and Timos, S. 2006. A combination of trie-trees and inverted files for the indexing of set-valued attributes. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*.

Tulving, E. 1983. *Elements of Episodic Memory*. Oxford University Press.

Vere, S. A. 1991. Organization of the basic agent. *SIGART Bulletin* 2(4):164–168.

Wang, Y., and Laird, J. E. 2006. Integrating semantic memory into a cognitive architecture. Technical report, University of Michigan.

Whittlesea, B. W. A., and Williams, L. D. 2001. The discrepancy-attribution hypothesis: I. the heuristic basis of feelings and familiarity. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 27(1):3–13.