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 been a dearth of research on the ability of artificial agents to fulfill delayed intentions — goals that the agent cannot immediately act on, but must remember and later recall in order to fulfill. Given the complex environments that agents could exist in, conflicting goals and inopportune surroundings may often require goals to be set aside. The ability to notice that a suspended goal should now be pursued and to recall the appropriate actions to take is therefore critical to general artificial agents.

A critical element of delayed intentions is that an agent must first form an intention, then later act on the same intention. This persistence over time requires the agent to keep track of its internal state — or in plainer language, to maintain memories of past events. Making the assumption that delayed intentions are not privileged by the architecture or otherwise different from other objects, intentions must be stored into and retrieved from memory through the same mechanisms as other memory structures.

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). A prospective memory task is one in which an agent is actively engaged in other background tasks between the formation of the intent to act and its ability to do so. Critical to this definition is that the intention is not immediately accessible to the agent when it must act, resulting in the conundrum of remembering to remember the intention. Such 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.

The need to remember and retrieve structures when fulfilling delayed intentions suggests that some kind of memory system is involved. Many cognitive architectures incorporate memories of different encodings, interfaces, and longevities; consistently among these variations is the separation of long-term and short-term memories. Knowledge in the former is less susceptible to removal, but requires longer time spans to create or recall; knowledge in the latter can be quickly manipulated by the agent for further reasoning, but is computationally expensive to maintain. 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. Since the length of delay of an intention is unknown, the simplest scheme is to store intentions in long-term memory and only retrieve them into short-term memory when they are needed. Although it is 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 is increased (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 human cognition. 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, as well as treating intentions differently from other objects. 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 storing 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 are needed by agents to manage goals in this manner — in particular, how memory systems can efficiently find satisfiable goals. There has also been work in augmenting BDI agents with different memory systems, but the interaction between goals and memories was left unspecified (Brom and Lukavský 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 merely another element in the agent's memory, and are treated in the same way. The findings of single dissociation between prospective and retrospective memory — that the former uses general purpose memories in its operation — strengthens this position (Burgess and Shallice 1997). While we do not rule out the possibility of a special-purpose goal memory, this work is motivated by the hypothesis that a goal memory is not necessary for delayed intentions. We therefore limit ourselves to only using existing general purpose memory mechanisms provided by architectures, with the hope that it will provide insight into delayed intentions management and the usage of memories in agents.

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 the representation to its long-term memory.

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 division we make between 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.

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). For reference, a summary of the properties of different memory systems is given in Table 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, where the "then" part encodes actions the agent should take when the "if" part is true. The close link between procedural memory and agent behavior means that any learning agent must ultimately modify its procedural memory. Directly, this can be done by changing procedural memory via the addition or removal of rules. Indirectly, the agent could also change the preferences that 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.

Although architectures use different encodings for their rules in procedural memory, there are functional similarities between the different implementations of 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 into which knowledge from other long-term memories 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.

A result of the lack of declarative access is that agents are unable to modify or remove rules once learned, rendering procedural memory append-only. To compensate for this, architectures often have other mechanisms to recover from incorrectly learned rules (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 features of procedural memory prescribe a general method to use it for storing delayed intentions. An agent must learn a rule that contains both the target and the action of the intention. The semantic knowledge stored in this rule is conditioned on the target, such that the rule will fire when the target is perceived. Since the application of the rule provides the agent with the necessary actions to take, no deliberate reasoning is necessary to recall the intention. After the intention is fulfilled, the agent must then prevent itself from pursuing the intention again. For architectures where the removal of rules is possible, this is trivial; for other architectures, the exact method for this suppression would be architecture-dependent.

Although the general method above would allow agents to fulfill delayed intentions, there are drawbacks especially for architectures where agents have no declarative access to rules. Since rules in many architectures persist for the lifetime of the agent, the number of irrelevant rules would monotonically increase. This creates a burden on the system, and can become expensive in both time and space after agents have fulfilled thousands of intentions. Further more, since agents do not know what rules are in memory, they cannot determine what intentions they have already made. In the example of buying milk, an 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, never mind what the target or action of the intention is.

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

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.

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 automatic matching of procedural rules. The necessity 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 (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 target, 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.

As much as semantic memory seems like a natural extension of short-term memory, there is one crucial difference: procedural memory can only directly access the latter but not the former. Thus, although delayed intentions could be stored in and retrieved from semantic memory, agents would need a separate mechanism to trigger the search for delayed intentions in memory. There does not appear to a uniform way of doing so across architectures; as such, this framework for using semantic memory to store delayed intentions remains to be fully specified.

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 effective 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" through re-experiencing 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 captures knowledge that agents may not know is important at the time. This provides multiple advantages to agents (Nuxoll and Laird 2007), although many remain unexplored in literature. Similarly, although variations in episodic memory — such as forgetting or otherwise removing information — 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 contex-

tual 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 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.

Episodic memory, because it was designed to capture all information, allows for a very different approach to using it for delayed intentions. Taking advantage of the partial matching ability of episodic memory, agents query for unfulfilled intentions using all the environmental features it currently perceives as the cue. The associative nature of the search then returns the intention whose target has the most features that overlap with the agent's state. Thus, the act of recognizing a target and retrieving the action is combined into the single process of searching episodic memory. Together with the automatic capture of knowledge, episodic memory reduces the amount of deliberate reasoning required by the agent.

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. Of the architectures compared, the different memories of Soar has the least overlap in functionality, and therefore provides a solid basis for comparison. We constrain ourselves to existing mechanisms provided by the 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 and as the establishment of baselines for future research.

Procedural Memory

Soar's procedural memory is representative of the class of procedural memories described above. The memory encodes knowledge in if-then rules, which match on and modify Soar's short-term memory. Like many other architectures, Soar agents do not have declarative access to its rules, and thus cannot remove them when appropriate. The general method of using procedural memory applies well to Soar agents, leaving only the problem of modifying memory after the intention has been fulfilled. Following previous work (Laird 1988), the agents instead learn a second rule that negates the results of the first. 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.

Using Soar's procedural memory in this manner inherits the disadvantages of general method — namely, that it fails to provide the agent with a declarative representation of existing intentions and imposes computation costs on the agent when rules have to be suppressed. Furthermore, data-chunking in Soar is an expensive process, requiring multiple cycles of decision during which the agent could not perform other tasks. This presents problems when fast response times to the background event is required. On the positive side, the learned intention-specific rule guarantees 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 semantic memory of Soar bears no major differences from the general description of semantic memories above: it is a knowledge store separate from short-term memory, requiring deliberate action to store or retrieve items. As noted, agents using semantic memory for delayed intentions need a method of trigger a search for unfulfilled intentions. 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 from semantic memory 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 remains. Furthermore, 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

Soar's episodic memory differs from the general concept of episodic memory in several ways. First, Soar's episodic memory automatically captures snapshots of agent's short-term memory; this allows intentions to be automatically stored into memory without deliberate action by the agent. Additionally, retrieval from Soar's episodic memory is biased by recency — all else being equal between two episodes, the more recent one will be returned. Complementing this bias is an architectural mechanism to restrict the search to only the most recent version of objects. This prevents agents from recalling episodes containing intentions from before they were fulfilled, and has the additional benefit of reducing the number of episodes to be searched. 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 until no intentions are applicable). Using episodic memory is also expensive in other ways: since recognizing the target also requires search, a retrieval must be attempted 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 compare 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 consider the issue of scaling and look at whether the agent remains reactive while coping with large numbers of intentions.

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 and targets, as well as the length for which targets are presented and for which 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. For the semantic memory agent, to avoid the need for prediction, we keep all identifiers of unfulfilled intentions in short-term memory. This is equivalent to the best case where the agent predicts all targets perfectly; an agent in the average case would perform worse due to prediction errors.

Within this domain, the results of the different long-term memories are shown in Table 2. Note that 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 background tasks, 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; the procedural and semantic memory

Agent	Intentions Fulfilled
Working memory	86.16 %
Episodic memory	78.61 %
Semantic memory	50.66 %
Procedural memory	44.78 %

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 targets appearing while the agent is engaged.

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. Episodic memory, on the 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 retrieve 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 and ignore the pressures of background tasks. 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 show how the agents scale as more intentions are fulfilled. Note that the semantic memory agent stays consistently below 1 millisecond; 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. Similarly, since the episodic memory agent must iterate through all intentions until it finds one that is unfulfilled,

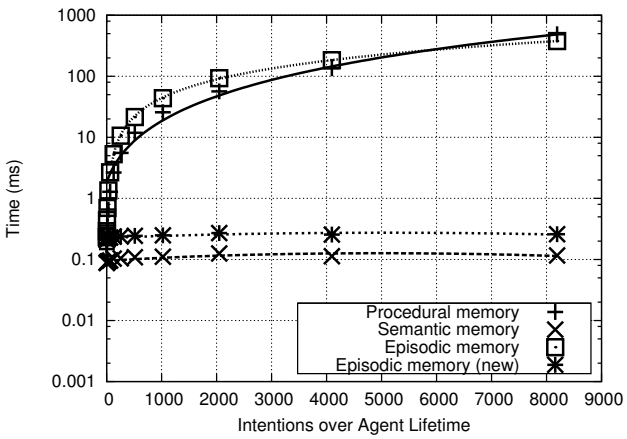

Figure 1: Time needed for different agents to retrieve a single unfulfilled intention. The working memory agent is not included as its intentions are already in working memory. Note that the semantic memory and the new episodic memory agents have response times of under a millisecond. The results of quadratic regressions on the data is shown in Table 3.

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
Episodic memory (new)	-1.880e-09	1.912	0.225

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.

the retrieval time therefore grows roughly linearly with the number of intentions. In the process of this work, however, we discovered that this linear factor is larger than it needs to be. In particular, the search algorithm also considers intentions which could not possibly be returned due to the recency bias. The new episodic memory search is more than three orders of magnitude faster than the original, making it comparable in speed to the semantic memory agent.

Conclusions and Future Work

This work has shown that agents could use long-term memory for the purpose of fulfilling delayed intentions, although work remains to be done for it to be truly useful. 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 does not require constant retrievals to recall intentions within the correct context. On the other hand, the performance of episodic memory is the closest to that of short-term memory, but may interfere with the background task by monopolizing the use of episodic memory. As these mechanisms were not designed for retrieving delayed

intentions, they fail to accurately recall intentions without disrupting the background task. We are hopeful that there are opportunities to leverage the advantages of the different memories to create a single, superior delayed intention system. One particularly intriguing possibility is that target recognition is not trigged by intention-specific mechanism at all, but by more general-purpose indicators. Previous research has explored the use of the spreading of activation (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.

Acknowledgements

The authors acknowledge the funding support of the Office of Navy Research under grant number N00014-08-1-0099.

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.

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 6th 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.