

Timing in Episodic Memory: Virtual Characters in Action

Ondřej Burkert¹, Cyril Brom¹, Rudolf Kadlec¹, Jiří Lukavský²

Abstract. In many applications, for instance in role playing games, it is an advantage when “minds” of virtual characters feature an episodic memory system. This system can boost cognitive and learning capabilities of the characters as well as their ability to respond to player’s questions. Recently, several special-purpose memory mechanisms for virtual characters have been published. We have been developing a more generic model, which incorporates hierarchically organized memory for events with gradual forgetting, a component reconstructing plausibly time when an event happened and a spatial memory for “what-where” information. One open question that has not been addressed yet in the context of virtual characters is how precisely should an episodic memory store timing information. To answer this question, we have conducted a study to investigate what time categories people use when asking time-cued questions. We hypothesized that humans prefer using fuzzy categories such as “morning” or “after lunch” rather than exact information and this hypothesis was confirmed. Here, we present the results of the study and overview the part of our memory model that is responsible for timing.

1 INTRODUCTION

Episodic memory [1, 2] is an umbrella term for memory systems operating with representations of personal history of an entity, a term stemming from neuro-psychology. Episodic memory traces are related to particular places and moments, and connected to subjective feelings and current goals. Fundamentally, the episodic memory is being distinguished from the semantic memory and the procedural memory. The former is conceived, more or less, as systems operating with general facts about the world as viewed from the objective perspective. The latter covers processes related to skill learning.

A believable virtual character, or intelligent virtual agent (IVA throughout) is a software agent [3] who imitates human or animal behaviour in a 2D or 3D virtual environment, who seems lifelike and whose actions make sense to the audience allowing them to suspend their disbelief providing convincing portrayal of the personality they expect or come to expect [4]. Recently, it has been argued that episodic memory is one of the key components contributing to the believability, at least for IVAs interacting with humans for more than a couple of minutes,

because it allows the user to understand better the IVA’s history, personality, and internal state: both actual state and past state [5, 6, 7, 8]. Examples of skills an IVA can possess that demand some facet of episodic memory includes, but are not limited to: a) general giving of information to the user based on the past history of the IVA’s interaction with the virtual environment, b) remembering a course of interaction with a human, notably remembering the course of a dialog, c) debriefing after a lesson, d) learning, e) repetition detection and improved reasoning [5].

Many applications require IVAs to have some of these skills, including role-playing computer games, educational simulations, virtual companions, and applications featuring virtual tutors, implying the necessity of augmenting IVAs’ minds with episodic memory systems [9]. Accordingly, several special-purpose episodic memory mechanisms have been presented recently [10, 11, 12, 13]. In our approach, we depart from these solutions in that we aim at a generic, reusable model integrating multiple interacting subsystems.

One of such subsystems is a module estimating time when an event happened. Without the ability of time events, human life would be dull. For instance, imagine yourself recalling morning that you brushed your teeth but not whether this happened a minute ago or three days ago. Virtual characters should be equipped with a similar ability. The proper timing is vital for more believable output of an IVA when attempting, for instance, to summarize last few days into several sentences highlighting the most important events to give a better picture to the user.

Human timing is not perfectly cleared up by psychology yet. Nevertheless, there is an abundance of theories trying to explain many phenomena linked with human timing as well as evidence from experiments. These materials are an important source of inspiration for a developer of such a memory component for timing.

After discussing several theories, Friedman [14] comes to the conclusion that humans use multiple techniques for dating past events. For instance, if you try to recall what you did on December 20, you would automatically convert the date to four days before Christmas and subsequently use this Christmas as an important landmark in the flow of time. You would remember that you were working the two days just before the Christmas resulting in the refinement of the time information to Saturday before the Christmas. Thus you would conclude that you probably went to buy gifts. This way of recall uses reconstructive nature of the memory when general semantic knowledge, knowledge of our lifestyle and personal episodes are combined to refine the recall. But it can also happen that you would answer that you do not know as it was too long ago and the date is too exact. Consider now another example: recalling when you visited your dentist last time. It is likely that you would use again the reconstructive nature of the memory, but

¹ Dept. of Software and Computing Science Education, Charles University in Prague, Ke Karlovu 3, 12118, Prague, Czech Republic.

² Institute of Psychology, Academy of Sciences, Politických vězňů 7, 11000, Prague, Czech Republic

now going from the reason why you went there (tooth ache, long-distance travel) and asking when this cause occurred. Finally, you may recall that it was approximately in May last year, even though the right answer was the 5th June.

Data from various experiments (e.g. [15]) show that the quality of recall can differ on different time scales (hours, days, months). Moreover, the data show that when asked to date an event people usually use various socially agreed general time patterns when speaking about the time (“morning”, “evening”, etc.) instead of from 5:40 to 6:38.

These points can help us with the design of a timing component for an episodic memory for virtual characters. People do not recall time of events exactly (e.g. [14, 16]) which suggests that an IVA does not need to store time information in the exact manner either. The comprehension via time patterns also contributes to parsimonious design of an IVA’s long term memory system since the memory resources can be saved by storing only the required time information. Moreover, the stored time information can mimic better socially agreed time patterns which are used by people in everyday communication.

We have already developed a timing component for an episodic memory for IVAs [17] and integrated it with the rest of our previous episodic memory model [18], producing a novel connectionist model of episodic memory for virtual characters. The important feature of the model is that its timing component is able to learn the time patterns, such as “early afternoon”, automatically based on the IVA’s life style. The time patterns the model learns cluster contextual information with a period of time; for instance, the context of some morning activities with time from 7.00 to 8.30 a.m. Those time patterns are anonymous – they are not labelled as “morning” or “afternoon” directly. The labelling is produced when the IVA uses a pattern as a part of an answer; then the name of the most matching socially agreed time pattern is assigned. For instance, the abovementioned previous cluster would be labelled as “morning”.

In sum, our objective has been to enhance IVAs with the following skills:

1. Learning of internal representation of time patterns.
2. Ability to comprehend human-like time concepts.
3. Ability to retell stories using human-like timing.

However, there were two problems with the model we developed. First, the key parameters – the forgetting rates – remained unspecified. Second, it was not clear whether the time patterns the model learns were believable. The time patterns are used both in the query construction and in the recall. Thus there are two areas of believability to cover. In this paper, we address the former one: the query construction. The issue here is that we did not know which types of time-cued questions are typically asked by users of IVA applications, what time patterns they use and what they are concerned about. Thus, we conducted a questionnaire study to shed light on this issue. We tried to determine which type of time patterns are given by users to IVAs, e.g. what time information is usually carried by a query. The results were incorporated in the internal memory organization allowing IVAs to use these time patterns as the input for the recall. As a side product, the incorporation of the time patterns allowed for easier use of such patterns when an IVA was retelling a story about its past.

The forgetting rates remain as a work-in-progress. There is no consensus in psychology which forgetting curve describes best

behaviour of human memory [19]; nevertheless, there are many psychological works which can help IVA developers to start to address this issue (e.g. [16, 19, 20, 21, 22]). A future work is a thorough believability study on human subjects in which IVAs telling stories about themselves based on their episodic memory would be subject to a kind of Turing test.

The main goal of this paper is to present the findings from the questionnaire study and to review the memory model. The results suggest that people do prefer vaguer time patterns over exact time specification (with some exceptions) and that it is possible to design and implement a timing component compatible with those time patterns.

The paper is organized as follows. Chapter 2 summarizes methods and results of the questionnaire experiments concerning time-cued questions and time patterns. Chapter 3 reviews the prototype of the timing component of the long-term episodic memory followed up by results obtained from several experiments. Additional materials are available on-line on our website³ and in [17].

2 QUESTIONNAIRE

The questionnaire was carried out in three consecutive rounds. We started with asking several people to write down a set of questions which they would ask an imaginary NPC (a non-player character) in a role-playing game. This preliminary phase had 7 participants, all men, undergraduate students, second to fourth grade. From final list we picked questions containing some time-related cues and recalls of past events. Based on these questions, we have created a set of questions for Experiment I. The results of Experiment I were used to refine the scenario and the set of questions for Experiment II.

2.1 Experiment I

Questionnaire and Hypothesis

The questions from the preliminary phase were extended by a set of similar questions with altered specification of time (e.g. “*What did you do yesterday afternoon?*” → “*What did you do on Friday at 13:15?*”). The aim of this alternation was to model various types of time-cues. Supposedly, the questions with vaguer time information would be favoured by the participants. When designing the questionnaire we wanted to cover a larger scope of possible tasks and time precision preferences. Therefore we introduced the participants to a rather general imaginary scenario rather than a more developed specific one. The latter would be also a good option, but it would be necessary to add a lot of additional information to the instruction to describe the setting and related tasks.

Method

The questionnaire was distributed to 30 volunteers (27 men, 3 women). The RPG game experience varied across the sample

³ <http://artemis.ms.mff.cuni.cz/pogamut/tiki-index.php?page=Episodic+memory+for+virtual+agent>

from little/none to large. They were mostly undergraduate students of computer science. They were given approximately 10 minutes for the task.

The participants were introduced to the situation by a short description of the model scenario, which was roughly as follows: *You are a player of an RPG game which takes place in a vast virtual world. There you can move freely and communicate with its inhabitants. Those inhabitants are controlled by the computer and they are able to perceive the surrounding world, reason about it and answer a variety of questions. You have just arrived to a new town and you are going to acquire a quest. To this end, you first talk to a merchant.*

Then the subjects were given a list of questions. The task was to rate these questions on the scale from 1 to 5 where 1 indicated a weird question while 5 indicated a valid question one would ask a virtual character. The exact version of the questionnaire with the task description can be found in Appendix.

Results

Figure 1 depicts the average response with standard deviation for each question. The evidence that would support our hypothesis, that exactly timed questions would score lower than vaguer questions, was not strong enough. Nevertheless, we were able to distinguish *post hoc* several categories among the questions which can explain some unpredicted ratings. The scenario concerned the acquisition of a quest, which can explain high values for questions related with criminality, orcs, guards and healing potions (six questions – see Appendix –, mean = 4.14, SD = 0.95). But we have also found that most of the artificially added questions with the accurate time information were rejected (four questions, mean = 1.79, SD = 0.96) as well as closely personal questions (three questions, mean = 2.23, SD = 1.18) as they were not important to the interrogator. On the other hand, personal questions with some relation to a possible quest or the overall situation were rated high (three questions, mean = 3.91, SD = 0.96).

We have not found any notable relation between the amount of RPG game experience of participants and the ratings.

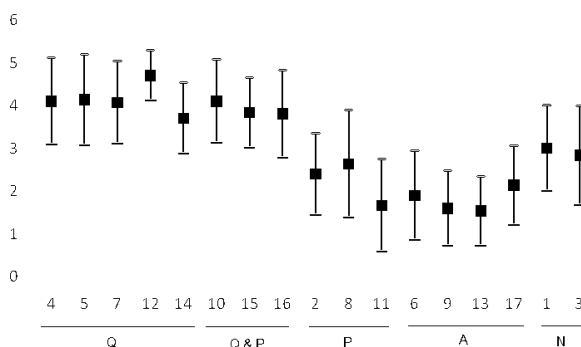


Figure 1. Mean question ratings. Whiskers indicate standard deviation from the average in the middle. Y axis represents rating (1 indicates a weird question, 5 a valid question). X axis contains numbers of questions as they appeared in the questionnaire. First 5 questions are quest related (Q), next 3 are personal and quest related (Q & P). Then follow 3 personal questions (P), 4 artificial questions (A) and 2 neutral (N).

Discussion

Experiment I showed that we are pursuing the right direction. Nevertheless, its results were strongly influenced by the proposed scenario situated in an RPG game, which usually includes goblin raids, magic potions etc. Thus we have conducted another experiment with modified settings as well as with the altered set of questions.

2.2 Experiment II

Questionnaire and Hypothesis

The previous results allow us to form the following hypothesis. We divide the time questions into three categories based on the specificity of the time information:

1. Exact time is given or demanded & thorough recollection is demanded: an answer is supposed to contain exact time information specified in numbers, e.g. 13th April morning or at 13:17; a question is supposed to contain such time cues (“What were you doing the 13th April morning?”)
2. Neither category 1 nor 3: e.g. questions targeting events more than one week old *plus* demanding their dating on the scale of parts of a day (the morning of last Christmas), etc.
3. Vague time cue is given & recollection is easy: i) questions that contain as a time cue a recent “past day” time concept such as “yesterday” or a combination of a “multi hour” fuzzy time concept (MHTC) plus a recent “past day” time concept such as “yesterday morning”; ii) a question on a repetitive activity that contains as a time cue a particular day of week or a particular day of week plus MHTC (e.g. Monday evenings); iii) a question that contains a vague “multi day” time cue without specifying more detail such as “last week”, “recently”, “in last few days”; or iv) a when-question that can be answered easily and using vague time concepts (“When were you on your last holidays?” – “Last winter.”)

The assignment of questions into the categories is necessarily subjective. Three of the authors of this paper (CB, OB, JL) voted on each question (i.e. they selected a category number or “I don’t know”) and only questions on which at least two of them agreed upon the category and the last voted “I don’t know” or his preferred category for this question differed from the other votes at most by 1, was assigned into that category. Several questions were removed from the questionnaire in the design phase based on the result of this voting procedure. We did not use variants of the same question adapted for categories 1 or 3 in order to conceal the hypothesis to the participants.

We presume that people would prefer questions from category 3 over questions from category 2 over questions from category 1.

Method

The questionnaire was distributed by email to former participants of Summer School of Mathematics and Physics which is a camp for talented young interested in science.

Another group of participants was formed by undergraduate students of computer science and psychology and the last group consisted of several graduates. The response ratio was about 30%. We have gathered 24 responses (7 women and 17 men ranging from mostly 19 to 28 years, with 3 older). All subjects were distinct from the subjects of Experiment I.

This time we wanted to avoid tight coupling with a concrete situation and an environment. Thus we have described the world more vaguely and proposed several scenarios to stimulate imagination of participants:

There are several towns in the virtual world. The situation can fall, for instance, into one of those scenarios:

1. *You live in one of the towns and you go regularly to the pub. You engage in a conversation with a newcomer trying to get some news and personal information about him.*
2. *You have come to a new location and figure out how things go here, what are they up to here.*
3. *You have met another player. You grab a virtual coffee and chat a little.*
4. *You chat with a merchant in his boutique.*

The description was followed up by the set of 36 questions and the participants were asked to rank them on the same scale from 1 to 5 as in the previous experiment. There were 11 questions of first category, 9 of the second, 13 of the third and three other questions with no time information (i.e. lures).

Results

Figure 2 depicts the mean and standard deviation of ratings for the three categories of questions. One-way ANOVA test was run on the data and confirmed that the means of all categories are non-equal ($p < 0.001$). There is clear evidence that exactly defined questions are less convenient than vaguely specified questions. Nevertheless, we have discovered few surprising discrepancies. First, one question from the 1st category “Do you know when we exactly went to sleep after that marathon run last Saturday?” (mean = 3.1, SD = 0.95) was rated relatively highly, possibly thanks to the compelling context. Second, the personal questions were again rated lowly in comparison with other questions in the same category despite their vague timing, for instance, two questions from the 2nd category. “How long did you eat the lunch last Thursday?” (mean = 1.3, SD = 0.46) and the question “What did you do last Friday after breakfast?” (mean = 2.1, SD = 0.97). It is possibly the unimportant context that decreases the rating of the questions. Third, one question from the 2nd category scored over 4 on average, despite the given time information was: “Monday two weeks ago”. This may indicate that the vagueness of time information can include even such patterns.

The questions in the third category were rated consistently high.

We again asked participants about their experience with virtual worlds; however, we did not find notable dependence between the ratings and the experience level of respondents. The only notable discrepancy was the “marathon” question, which received good ratings from non-players (mean = 3.6) whilst

obtaining relatively low marks from “gamers” (mean = 2.6). See Appendix for complete overview of results.

As was mentioned above the assignment of questions into categories was a subjective process. Differences between categories can be best illustrated on some border cases. Consider question no. 4: “What were you doing Wednesday the previous week?”, that was assigned to category 2. If it had addressed Wednesday from this week, it would have fallen to category 3. On the other hand, question no. 7: “Are you doing something on Sunday between 8 p.m. and midnight? We can go for a beer.” was assigned to category 2 even though it contained exact time information, because the context made it clear that we were actually asking for the evening. A similar type is question no. 35: “Are you doing something Tuesdays between 17:15 and 18:45? We can play squash.” This question was also assigned to the second group despite the exact time information.

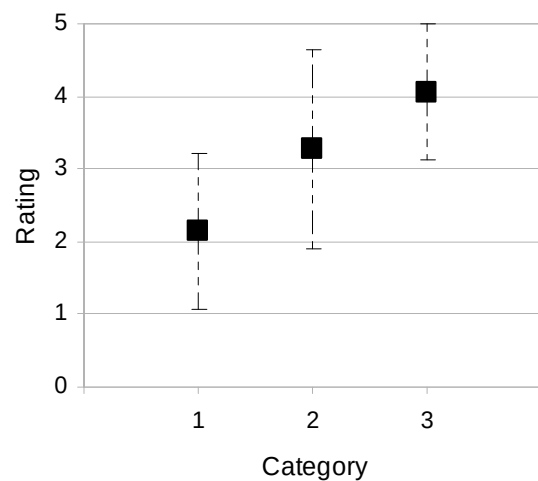


Figure 2. Mean category ratings. Whiskers indicate standard deviation, 1 indicates a weird question, 5 a valid question.

Discussion

The collected data is consistent with the proposed hypothesis that people prefer questions using only vague time specifications. The questions with exact time were rated lower, with an exception of “Squash question”, where the exact time is probably of high importance in everyday life and the “marathon run” question where the context probably influenced its rating. The vaguely specified questions from the 3rd category were rated highly.

Ultimately, the gathered data leads us to the conclusion that people would rather ask IVAs using vague time cues than exact ones. Now we know that IVAs should be capable of processing vaguely specified questions. Therefore they need more profound internal understanding of human-like time cues to be able to respond believably. Additionally, we can draw following points from the experiments. First, an IVA typically does not need to store its personal information in a great detail as the user’s interest lies usually in other domains. Second, the timing that belongs onto the scale of parts of a day (“morning”, “evening”) seems to be sufficient for most cases.

For future research, it would be interesting to investigate in detail ordinary human answers to these questions, the influence of context and relevance of questions to the described scenario, and how specific or vague place/object references are accepted in human dialogues. Answers to these issues can help with further constriction of requirements on episodic memory models for IVAs.

3 CONNECTIONIST EPISODIC MEMORY DESIGN

The purpose of this section is to review the timing component for the episodic memory for IVAs we have developed. For the space constraints, we cannot introduce all detail; these can be found in [17]. We will first review our agent architecture, second, the general episodic memory system, and third, the timing component. The chapter concludes by the summary of experiments made on the model.

3.1 Agent architecture

Our IVA's overall architecture is depicted in Figure 3. It is a reminiscence of a classical cognitive AI architecture, by which many virtual agents have been inspired. Our IVA is driven by *hierarchical reactive planning* with behaviour represented by *AND-OR trees*. The AND-OR tree metaphor works with abstract *goals* representing what shall be achieved, and *tasks* representing how to achieve the goals. Typically, every goal can be accomplished by several tasks, while every task can be achieved by adopting some sub-goals. The IVA needs to perform only one task to achieve a goal, provided there is no failure (hence, OR nodes); but to fulfil all sub-goals to solve a task (hence, AND nodes). The tasks that cannot be further decomposed are *atomic actions*, i.e. action primitives. Every task may need several resources to be performed, i.e. *objects*. Every top-level goal has its *activity level* based on drives, external events, and a schedule. The competition among the goals based on this level takes place within the *goal structure*, which also stores the AND-OR trees. The winning goal chooses the most appropriate task (e.g. "to eat" goal can chose "take something from the fridge") and passes its template to the *task field* of the *visual short term memory*. One goal can interrupt another, in which case the tasks of the interrupted top-level goal are remembered and can be resumed after the more important goal is achieved. From the AI standpoint, this mechanism capitalises on the BDI framework [23].

The visual short-term memory holds templates of objects seen that passed through a simple threshold-based *attention filter*. Every object is regarded as a tool for action, i.e. it is a set of "affordances" [24], meaning it possesses pointers to the tasks it can be used as a resource for. These pointers are perceived directly by the IVA when observing his environment. Objects in experiments we have been running are state-less for the sake of simplification, though our simulations allow the objects to have states as well. The *memory field* can also temporarily hold information about an object recalled from the long-term memory.

This architecture has several long-term memory appendages, such as spatial memory (LTSM) and episodic memory (LTEM). Here, we are concerned only with the episodic memory.

3.2 Long-term episodic memory

Recall that our present episodic memory model with the timing component extends the first generation of the model presented in [18]. While the old model was symbolic, the present one is connectionist. The present model is composed of a three-lobed neural network (Figure 4).

The core lobe is the *long-term episodic memory* (LTEM), which stores past episodes of the IVA's life, the flow of events. The memory contains a forest of AND-OR trees representing all possible tasks the IVA can perform (Figure 4 – middle). This structure mirrors IVA's behavioural representation. Note that this was the only part of the previous model [18]. In present version of the model, this core lobe is surrounded by other lobes responsible for timing.

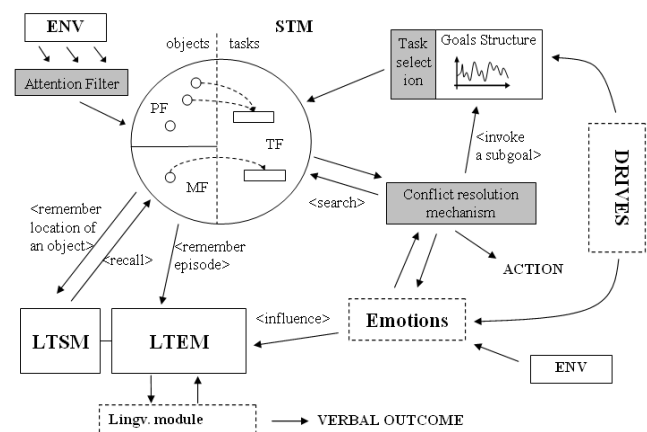


Figure 3. Outline of our IVA design. He receives inputs from the *environment* (ENV) which fills up the *perception field* (PF) of the *short-term memory* (STM). The decisions are made according to the activation of *goals* and *drives*. Tasks in progress are represented in the *task field* (TF). *Memory field* (MF) stores locations of resources obtained from the *long-term spatial memory* (LTSM) on request. Personal episodes are stored in the *long-term episodic memory* (LTEM). Forgetting in the LTEM is influenced by a simple valence-based *emotion model*.

The first surrounding lobe is a two-layered neural network. The first, input layer consists of *context nodes*, which describe the agent's internal or external state (i.e., context), and *Cartesian nodes*, which represent objective time (i.e., their activation changes regularly over time). Cartesian nodes can be also conceived as biorhythms. This layer is connected with the layer of daytime concepts. *Concept nodes* associate together particular context the IVA is situated in with some period of daytime, thus forming new time patterns. For example, if usual afternoon activities of our IVA comprise of studying for exams, some concept nodes will be linked with context nodes for studying and Cartesian nodes for time between 14.00 and 17.00 hrs. In the end, concept nodes should represent the desired vaguer human-like time patterns. Importantly, unlike the hard-coded layer of Cartesian and context nodes, the concept nodes are learned automatically during the course of simulation using Hebbian learning with subtractive normalization (see [17] for details).

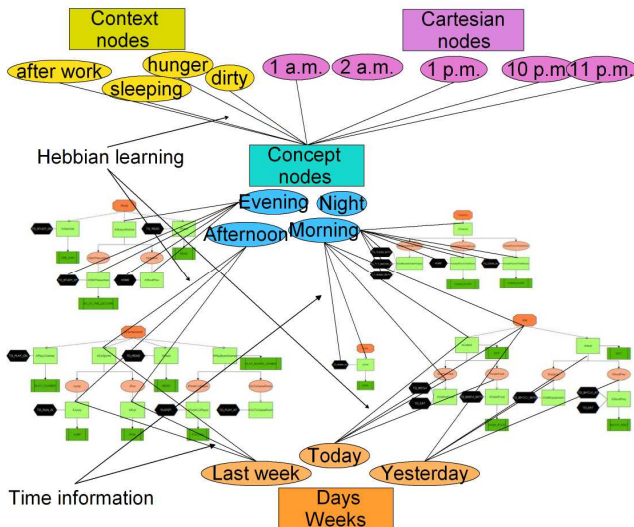


Figure 4. Episodic memory design outline. The context layer (top left) and Cartesian layer (top right) are connected with concept nodes (centre). Concept nodes are connected with AND-OR trees of goals/desires (middle bottom). These are also connected with the day/week nodes (at the bottom).

The second lobe contains *day nodes* (bottom at Fig. 4). These represent past days, such as “yesterday”, or abstractions for multiple-days and parts of weeks (such as “first two days of the week” or “last day of workdays” etc.). Note, that some psychological data suggest that humans use these abstractions [15]. These nodes are hard-coded.

All Cartesian, context, concept, and day nodes work as rate-coded neurons with a $\langle 0, m \rangle$ response function, where m is a tuning constant, which is typically from $\langle 2, 4 \rangle$ (see [17]).

The AND-OR trees are responsible for the actual episodic entry storage. During storage, the nodes of the AND-OR trees are interconnected with two kinds of nodes. First, with concept and day nodes, representing the time information, which can be later used during recall. Second, with nodes representing objects used in pursuing tasks and scenes where the tasks took place. The storage, i.e. building of links, happens continuously along the course of IVA’s actions. The links are weighted and the weights are built again on the Hebbian basis (see [17] for details).

Note that events the core lobe stores have various grain size, which allows for *gradual forgetting*: unimportant details of episodes can be “eaten away” from the bottom of the AND-OR trees. In the prototype implementation, there is a given threshold for each day which is decreasing over time. Every night, the network representing “today” is labelled as “yesterday” and analogically for the other days. Then the weights are decreased by an arbitrary coefficient. Weights which are weaker than an arbitrarily chosen threshold are discarded and their energy (weight) is distributed amongst remaining weights which promotes storing of important (stronger) connections. For example, one can originally remember that he was cooking a goulash yesterday morning, including all subtasks, but later forget the subtasks, keeping only the high-level information about cooking. To our knowledge, present episodic memory systems for IVAs do not have this feature. Moreover, it allows the IVA, to some extent, to blend similar episodes into one (see

Sec. 3.4). The details of the forgetting are described in [18] and other concept related to our memory systems in [5, 25].

3.3 Labelling of concept nodes and recall

So far, we have discussed only the structure of the episodic memory system, development of time concepts and storage. This section will cover the use of the built network for episodic recall. For recall, the memory structure is not conceived as a regular neural network, but as a spreading activation network. This means that we can inject activity into a node and propagate this activity to neighbouring nodes. The activity is propagated proportionally to the weight of each link. In the present version of the model, the activity is propagated only to direct neighbours (for the sake of analysing the experimental results).

Let us imagine that the IVA has already learned concept nodes (this takes approximately 10 days (of the time in the virtual world) with the current setting of parameters) and there is already stored some information in the network. Now, the user asks a question. Every question contains some cues – time, place, activity. Those cues are conceived as inputs for our memory system: basically, nodes corresponding to these cues are activated. The activity is propagated through the network from neurons representing question cues and the most active neurons of the demanded category are returned. For instance, question: “What were you doing in the evening two days ago?” infuses activity to the neurons for *evening* and *two days ago* and returns the most active activities; in example depicted on Figure 5 these are *swimming* and *dinner*.

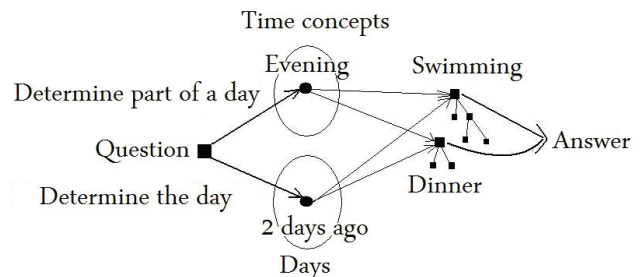


Figure 5. A query example. The arrows depict propagation of the activity. The activity is infused into the neurons for *evening* and *two days ago* and the most relevant answer based on the activity level is “*swimming and dinner*”.

With respect to questions on time information, there is one issue. As we have mentioned in the previous section, concept nodes are anonymous, internal representation of time concept clusters in the IVA’s mind. The recall returns which concept nodes are relevant to the question but it cannot provide their names. Therefore, we need a mechanism which would translate those internal time concepts into socially agreed time patterns. To this end, we use the links between Cartesian nodes and concept nodes. First of all, we define a set of six parts of a day – labels (morning, noon, afternoon, evening, late evening and night). Each label confines a fixed time interval. Then we measure the correspondence of concept nodes to one of those labels: the most appropriate label is assigned to each of the concept node. The measure of correspondence is obtained from weighted links between Cartesian nodes and concept nodes.

First, Cartesian nodes are assigned to human categories. For example, Cartesian nodes for time from 6 a.m. to 9 a.m. fall to “morning”. Then we activate all Cartesian nodes from a particular category and propagate their activity to the concept nodes, measuring the concept nodes’ activation. Each concept node is assigned the category that activated the node mostly. A crude natural analogy to this process is a child being taught by her parents to name various parts of day.

A notable feature of our mechanism is that the recall can be error prone. For instance, tasks and objects linked with the “after breakfast” node for “yesterday” can be merged with tasks and used objects that were linked with the “after breakfast” node the day “before yesterday”. Thus two episodes can be blended during the recall (when the recall is cued by time information).

3.4 Experiments

The design of the memory system was followed up by a successful implementation of the prototype [17] on the platform for fast prototyping of virtual agents Pogamut 2⁴ [26]. The implementation was tested by a set of experiments. The experiments included tests of memory accuracy, memory space demands, episode blending and the study of impact of various settings of context and Cartesian nodes on the memory performance. The simulation usually lasted, from the standpoint of the IVA, two to four weeks. The IVA lived according to several different lifestyles (student, millionaire, travel salesman).

Lifestyles usually contained few fixed, routine activities and few dynamic, occasional activities. The world of the IVA comprised of 6 rooms and 27 different objects and places and was situated in the 3D FPS game Unreal Tournament 2004. The agent had 10 different activities to perform. The student’s plan was filled with lectures during the week days mornings followed by various sport activities in the afternoons and cultural events in the evenings. The millionaire had a less structured life filled with entertaining activities. The travelling salesman worked mostly all the time with only occasional divergence in the plan. Each character had different time frame of the day (alarm clock settings).

The results we gained so far suggest that:

1. The network is able to learn and use the concept nodes properly and these concepts are qualitatively similar to those used by humans [15].
2. The memory is error-prone and blending can occur under some circumstances, e.g. an IVA who was watering a garden every evening cannot recall details of any particular watering episode when cued by time, but it can recall both that it was watering evenings last week as well as some detail of each watering when cued with time plus other details of the episode. However, this behaviour remains to be investigated on a rigour basis and the whole mechanism should be tuned appropriately; a future work.
3. The forgetting mechanism can ensure smooth forgetting of details and unimportant episodes.

4. The activation of Cartesian nodes should overlap to create a more compact time clusters represented by concept nodes.
5. Context nodes play an important role in the learning process. They support the diversification of the Cartesian nodes; more concept nodes are formed for periods in which activities change often (i.e. with different context nodes being activated) comparing to periods without changes of activities (e.g. a night). This is a desirable outcome.

To conclude, the conducted experiments revealed some key parameters of the learning process (i.e. the setup of Cartesian and context nodes) and interesting dependencies (i.e. the importance of context nodes). They also helped to verify the design of the memory system as they showed it is capable of recall on time-cued questions. The experiments also demonstrated the capacity of the memory to blend episodes and to forget details while keeping the important parts of memories.

4. CONCLUSION

The goal of this paper was twofold. First, the aim was to identify requirements for time representation underpinning episodic memory systems for IVAs, enabling conversations featuring believable time patterns. Second, the aim was to review our memory model adhering to theses requirements.

The data gathered from the two questionnaire rounds suggest that people would use rather socially established time patterns than exact time or date when questioning virtual characters. The time patterns can be parts of a day such as “morning”, or denotations of larger time intervals such as “last week”. Additionally, the data suggest that human users would not ask virtual characters for detailed depiction of personal concerns, unless they are directly relevant to the users’ interest.

We have designed, implemented and tested a novel model of episodic memory enhanced by a timing component. The results of experiments with this model suggest that the model enhance IVAs with ability to understand and work with human-like time patterns. Its connectionist basis allows IVAs to mimic some phenomena of human memory like episode blending, error proneness and gradual forgetting of details. In future, this episodic memory module can be used in many role-playing games and other applications featuring virtual characters inhabiting their worlds over larger periods of time.

APPENDIX

Appendix is available online on address:
<http://artemis.ms.mff.cuni.cz/main/papers/Timing-Appendix.pdf>.

ACKNOWLEDGMENTS

This work was partially supported by the research project MSM0021620838 of the Ministry of Education of the Czech Republic, and by the student research grant GA UK 21809 and by the grant 201/09/H057.

REFERENCES

⁴ <http://pogamut.cuni.cz/>

- [1]E. Tulving, W. Donaldson. *Organization of memory*. New York: Academic Press. (1972).
- [2]A. Baddeley, M. J. Conway. Aggelton. *Episodic Memory: New directions in research*, Oxford Uni. Press. (2001).
- [3]M. Wooldridge. *Reasoning about rational agents*, MIT press. (2000).
- [4]B. Loyall. Believable Agents: Building Interactive Personalities. Tech report CMU-CS-97-123. Ph.D. dissertation, Carnegie Mellon University. (1997).
- [5]C. Brom, J. Lukavský. Towards virtual characters with episodic memory II: Episodic memory strikes back. In: *Proc. Empathic Agents, AAMAS workshop*. 1 - 9. (2009).
- [6]W. C. Ho, S. Watson. Autobiographic knowledge for believable virtual characters. In: *Proc. of Intelligent Virtual Agents*, LNCS 4133. 383-394. Berlin, Springer-Verlag. (2006).
- [7]G. Castellano, R. Aylett, K. Dautenhahn, A. Paiva, P. W. McOwan, S. Ho, Long-term affect sensitive and socially interactive companions. In: *4th Int. Workshop on Human-Computer Conversation*. (2008).
- [8]I. Doherty, C. O'Riordan. Toward More Humanlike NPCs for First-/Third-Person Shooter Games. In: *AI Game Programming Wisdom IV*. Charles River Media. 499 – 512. (2008).
- [9]A. Nuxoll. Enhancing Intelligent Agents with Episodic Memory. PhD thesis, The University of Michigan. (2007).
- [10]W. Dodd. The design of procedural, semantic, and episodic memory systems for a cognitive robot. Master thesis. Vanderbilt University, Nashville, Tennessee. (2005).
- [11]W. Ho, K. Dautenhahn, C. Nehaniv. Computational Memory Architectures for Autobiographic Agents Interacting in a Complex Virtual Environment. In: *Connection Science*. (2008).
- [12]J. Rickel, W. L. Johnson. Animated Agents for Procedural Training in Virtual Reality: Perception, Cognition, and Motor Control. *App. Artificial Intelligence* 13 (4-5): 343-382. (1999).
- [13]J. Dias, W. C. Ho, T. Vogt, N. Beeckman, A. Paiva, E. Andre. I Know What I Did Last Summer: Autobiographic Memory in Synthetic Characters. In: *Proc. of ACII*. 606–617. Berlin, Springer-Verlag. (2007).
- [14]W. J. Friedman. Memory for the Time of Past Events. In: *Psychol Bull* 113(1): 44 – 66. (1993).
- [15]S. F. Larsen, C. P. Thompson, T. Hansen. Time in Autobiographical Memory. In: *Remembering Our Past: Studies in Autobiographical Memory*, New York: Cambridge University Press, 129-156. (1995).
- [16]W. A. Wagenaar. My memory: A study of autobiographical memory over six years. In: *Cognitive Psychology*. Vol 18(2), 225-252. (1986).
- [17]O. Burkert. Connectionist Model of Episodic Memory for Virtual Humans. Master thesis. Charles University in Prague. (2009).
- [18]C. Brom, K. Pešková, J. Lukavský. What does your actor remember? Towards characters with a full episodic memory. In: *Proc. ICVS*, LNCS 4871, Springer-Verlag. (2007).
- [19]R. A. Chechile. Memory hazard functions: A vehicle for theory development and test. In: *Psychological Review*. Vol 113(1), 31-56. (2006).
- [20]P. H. Bahrick, E. Phelps: Retention of Spanish vocabulary over 8 years. In: *Journal of Experimental Psychology: Learning, Memory, and Cognition*. Vol 13(2), 344-349. (1987).
- [21]J. R. Anderson, R. Milson. Human memory: An adaptive perspective. In: *Psychological Review*. Vol 96(4), 703-719. (1989).
- [22]O. Furman, N. Dorfman, U. Hasson, L. Davachi, Y. Dudai. They saw a movie: Long-term memory for an extended audiovisual narrative. In: *Learning & Memory* 14, 457–467. (2007).
- [23]M. E. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press: Cambridge, MA. (1987).
- [24]J. J. Gibson. The Theory of Affordances. In: *Perceiving, Acting, and Knowing*, Eds. Robert Shaw and John Bransford. (1977).
- [25]Brom, J. Lukavský. Episodic Memory for Human-like Agents and Human-like Agents for Episodic Memory. In: *Technical Reports FS-08-04, Papers from the AAAI Fall Symposium Biologically Inspired Cognitive Architectures*. Westin Arlington, VA, USA, pp. 42-47, AAAI Press, (2008).
- [26]O. Burkert, R. Kadlec, J. Gemrot, M. Bída, J. Havlíček, M. Dörfler, C. Brom. Towards fast prototyping of IVAs behavior: Pogamut 2 In: *Proceedings of 7th International Conference on Intelligent Virtual Humans*, LNCS Vol. 4722. Paris, France. Springer-Verlag, Berlin. (2007).

An Object-Based Memory for Supporting Attentive Virtual Agents

Christopher Peters¹

Abstract. This paper presents an overview of an object-based memory system for supporting attentive behaviours of a virtual agent to its surroundings. The memory model is multi-stage, consisting of successive stages of filtering of objects from the scene database based on the agent's synthetic vision of the environment. Stored information may be out of date and may not necessarily match the actual states of objects in the environment, thus providing the basis for the agent to have its own perspective of the environment. A brief description details the importance of an object-based memory system for supporting inhibition of return in visual attention systems. These allow mobile virtual agent's to continue to move their focus of attention around the scene, thus creating a loop between them and the environment.

1 Introduction

Internal storage, or memory, is crucial for virtual agents expected to act in a plausible manner within their environment. If an agent has no storage mechanism and interfaces directly with the scene database, it will always have perfect information about the state of the environment and, by definition, cannot have a unique perspective of its world. Such access ultimately reduces the credibility of the agent's behaviour with respect to viewers, particularly if it is intended to be simulating a real-world entity that is constrained by senses or must operate based on imperfect or uncertain environment information.

Here, a virtual agent is endowed with a synthetic vision capability for sensing its environment, a 3D virtual environment. A memory-model filters visual information, limiting the amount of environmental information that must be processed and stored by the agent. The model described in Section 3 is based on the popular, but simplified, multi-stage model of memory inspired by Atkinson and Shiffrin [1] (see Figure 1), incorporating modifications from the literature that suggest the existence of a specialised visual short-term memory and allowing latitude [11] from the strict serialisation of information flow between memory suggested in the original model. Visual attention is discussed in Section 4, not only as a method for ensuring a sound filtering strategy, but also in relation to coupling with memory for generating attentive behaviours.

2 Background

While memory capabilities of some form are evident in most autonomous agent systems. While in many cases explicit memory models are not defined, some research has considered their implementation for a variety of purposes. Recently, Lim and colleagues [10] have considered memory for virtual and robot companions, particularly for allowing agents to migrate between embodiments and for enabling long-term interactions and relationships with humans. In contrast to work on autobiographical memory systems, other models have been concerned with memory for virtual agents embedded in virtual environments, based on sensory capabilities for assisting *active vision* [2], research sharing similarities with that undertaken in robotics [15]. Kuffner and Latombe [9], for example, presented a flat memory system for objects, which also facilitated forgetting processes. Evers and Musse [5] use a memory model with STM and LTM to study the impact of memory on behaviour using hierarchical finite state machines. The agents in Chopra-Khullar's [3] work store object-related memory thresholds are used to provide a measure of uncertainty for driving the allocation of attentive monitoring behaviours.

Peters and O'Sullivan previously presented a model based on synthetic vision and a staged memory model [13], later applied to visual attention and perspective-taking for theory of mind [12]. Here, further details are provided of how memory and attention mechanisms may be coupled to support attentive behaviours for autonomous virtual agents.

3 Memory Model

Stage theory posits that information progresses through a number of successive, discreet memory stages: A short-term sensory storage (STSS), a short-term memory (STM) and a long-term memory (LTM), as outlined in Section 3.1 and illustrated in Figure 1. An underlying notion in the model is that memories reaching later stages will tend to persist for longer. It represents a great simplification of the way in which information is thought to be stored and associated in humans. Nonetheless, it provides a very useful paradigm for real-time autonomous agents; it suggests a general filtering and storage scheme, free of the computational overhead and technical implications of more elaborate associative models (but also subject to the implied inherent restrictions).

¹ Interactive Worlds Applied Research Group, Coventry University, United Kingdom, email: Christopher.Peters@coventry.ac.uk

3.1 Memory Structure

Each memory stage consists of a list of memory entries (Section 3.1.1). In order to generalise the description of the stages in the model, here we describe stages parameterised according to *capacity*, *threshold*, *duration* and *rehearsal*. By varying these parameters, the different memory stages elaborated on in the stage theory of memory may be modelled, and new stages or sub-stages defined if necessary. *Capacity* refers to the maximum number of memory entries that a memory stage can store at any one time. The *threshold* parameter is used to decide when memory entries should be removed from a memory stage i.e. forgotten. While the trace strength of a memory entry is greater than this threshold, the item remains in memory and the probability of its recall is presumed to be 1. Memory entries are removed from a memory stage when their trace strength falls below the threshold value. *Duration* is an indication of the amount of time that a memory entry will persist in a memory stage without being rehearsed in any way. *Rehearsal* is another threshold, but is used to decide when an entry should be moved to the next memory stage. Rehearsal may be increased by attending to an object.

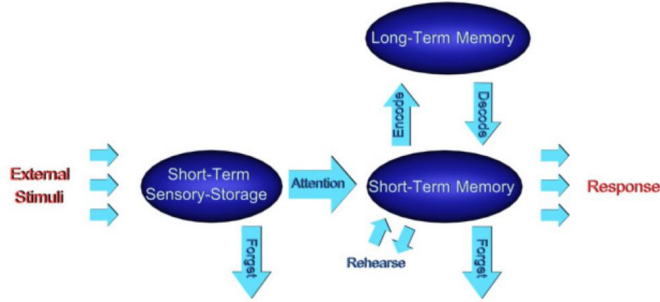


Figure 1. The basic stage theory of memory, a multi-stage model inspired by Atkinson and Shiffrin [1].

3.1.1 Memory Entries

Memories are considered to be composed of discrete constructs called *memory entries*. Memory entries contain information regarding what has been remembered and other information pertinent to the management of the memory system (see Table 1), allowing a recording of how recently the object was perceived and when it should be removed from memory. In the current implementation, there are two different types of memory entry: *proximal stimuli* and *observations*. Proximal stimuli do not contain resolved database object information, but merely information extracted from inspection, such as stimulus false colour (described shortly), retinotopic bounding box, size in pixels and average colour. Observations are elaborations of proximal stimuli, representing a snapshot of object state from the scene database.

Because observations persist in memory, they may often correspond to out-of-date information that is no longer accurate. An agent therefore has an internal representation of the environment that, in a similar manner to humans, may be imperfect. Only by continually sensing and attending to the environment can the agent maintain updates to the internal model.

Basic Memory Entry Structure

Member	Meaning
<i>trace strength</i>	The strength of the memory
<i>decay factor</i>	How quickly the memory decays
<i>activation</i>	Amount of time since the memory was rehearsed
<i>rehearsal</i>	Number of times that a memory has been rehearsed
<i>proximal stimulus</i>	Perceived stimulus
<i>observation</i>	Observation corresponding to the perceived stimulus

Table 1. The memory entry structure contains information for managing memory contents.

3.1.2 Synthetic Vision

The synthetic vision system takes snapshot renderings of the virtual environment from the agent’s viewpoint. Unlike normal renderings, here objects are *false-coloured*; they are attributed unique colours and rendered in flat-shaded mode, without textures. Reading the colour values back from synthetic vision false-coloured renderings provides details of those objects that are within the agents field of view and visual range, that are not occluded by other objects. More details are available in [13].

3.1.3 Short-Term Sensory Storage

The short-term sensory storage, or *STSS*, is the first storage stage for stimuli that have passed through initial visual filtering. A memory entry is created for each object that is read back from the false-coloured image, to create a list of *proximal stimuli*. These do not persist for very long between updates of the visual system. Therefore, entries in the STSS could be thought of as corresponding to the 2D optical images deemed to fall on the *retina* of the agent, without containing semantic details. The STSS is updated with each refresh of the agent’s viewpoint rendering: When a proximal stimulus is encoded in the STSS, a time-stamp is updated to the current time and the trace-strength is set to the maximum value for the STSS memory stage. The capacity of the STSS is considered large enough to allow storage of all stimuli falling within the agent’s view, subject to the process detailed in Section 3.1.2. When a proximal stimulus is encoded in the STSS, a time-stamp is updated to the current time and the trace-strength is set to the maximum value for the STSS memory stage. Memory entries in the STSS are regarded as having a very fast rate of decay. Because of this, entries in the STSS correspond to objects that are currently in view, or were in the agent’s view very recently.

3.1.4 Short-Term Memory

Short-term memory is more limited than the STSS in terms of the number of memory entries that can be remembered. However, memory entries generally persist for a longer mean duration. Generally, memory entries are removed, or forgotten, from the STM under two conditions: they are displaced by newer memories when the STM is full and they may also decay over time. Rehearsal describes how entries may remain

and persist for longer periods in the STM. As described later, these low-level values are intended to be controlled by other mechanisms, such as attention (Section 4).

3.1.5 Long-Term Memory

Long-term memory, or *LTM*, provides storage for information over a long period of time. For the purposes of the current model, memories that can no longer be retrieved due to lack of sufficient cues are considered to be the same as decayed entries. In LTM, as in other stages, such items decay over time and are completely removed when they fall below the threshold trace-strength.

Long-term memory is considered to have a *back and forth* relationship with short-term memory; items may be moved into short-term memory for processing and then moved back into long-term memory for storage. Information from short-term memory is transferred to long-term memory by means of rehearsal, related to attention mechanisms. In this model, repeated rehearsal of a stimulus will result in it being encoded in long-term memory. Unlike the other memory stores, there are no strict upper bounds on trace-strength in the LTM. If an item is frequently rehearsed in the STM, then it can remain easily accessible in the LTM.

3.2 Memory Dynamics

Given the memory structures described, memory dynamics deals with the movement of information between stages and when information should be removed from memory i.e. forgotten.

Eight different cases for memory updates are possible given the three stage model. Rules are based on the idea that objects in the STSS are, or were recently, visible, that objects in the STM are being attended to and objects in the LTM have previously been attended to and are currently in long-term storage. These rules are as follows:

1. If there is no memory entry for an object in the STSS, STM or the LTM, then it is presumed that the agent has no internal representation of the object whatsoever.
2. A memory entry that exists only in the STSS signifies that an object was recently within the agent's view field, but is not being attended to or rehearsed.
3. When a memory entry is only present in the STM, the agent is attending to it, even though it is no longer in the agent's view.
4. When an item is present both in the STSS and in the STM, then it is or was recently in the view of the agent and is also being attended to by the agent.
5. Memory entries that exist only in the LTM are subject to decay according to the decay function. The rehearsal parameter does not change since the agent is neither receiving new information about the object from the STSS, nor paying attention to it through the STM.
6. Memories that are present in both the STSS and LTM result in normal STSS decay. In the current model, LTM decay is paused while corresponding entries exist in the STSS. This presumes some sort of connection between the STSS and LTM that bypasses the STM. Although such a connection was not part of the original version of the

model proposed in [1], the abandonment of strict serial information flow between each memory component is seen as a necessary modification to support experimental evidence (see [11]).

7. If a memory is present in both the STM and the LTM, then trace strength decay takes place in the STM, but not in the LTM. Rehearsal of the entry takes place since the entry is being attended to.
8. When memories exist for an object in all three stages, then the entry trace strength and rehearsal is increased. Object uncertainty is also reduced. This case represents a scenario where the agent is or has recently looked at a stimulus, is paying attention to it and has also previously rehearsed the stimulus and paid attention to it.

These rules form the basis of all updates that take place in the memory system. When used in conjunction with decay thresholds and rehearsal thresholds, they provide a way for items to move between memory stages so that only the most important environmental stimuli are retained in memory.

3.3 Forgetting

Forgetting is of vital importance to a computational model of memory for agents. Items that are considered to be remembered by the agent must be stored in the physical memory of the computer system, which is finite. Operationally, forgetting is implemented as a scheme for the removal of memory entries that are no longer of importance or relevance. In the current system, a simplified model of forgetting is implemented based on the *decay hypothesis* of retention [4]. Power functions illustrate the decline in the ability to recall information from memory as dependent on a *retention interval*, the amount of time since a memory was last entered or rehearsed. In this work, it is assumed that items in memory always have a probability of recall of 1. Trace decay is used only to determine when a memory entry should be forgotten, as oppose to how successfully information is recalled and to what extent. Memory entries are presumed to decay according to an exponential decay function, as used in the decline function in the short-term store of Atkinson and Shiffrin's buffer model [1]. Memory entries are removed from a stage when their trace strengths are reduced below the threshold trace value defined for the respective memory stage.

4 Visual Attention

Attention refers to the way in which some stimuli may receive enhanced cognitive processing, while others may not be processed to the same degree. As a mechanism, a sound attention strategy enables an entity with limited resources to survive in a dynamic and complicated environment by prioritising the consideration of information that is potentially important and relevant, inherently and with respect to current task. The basic visual attention system in use is a bottom-up model that deems areas of high basic visual feature contrast to be of inherent potential relevance to an entity. It has been described elsewhere [14] and is based on [8]. It would be complemented by the integration of a top-down model. Nonetheless, it serves to illustrate important interrelationships between memory and attention, the focus of this paper. In this

case, the use of the object-based memory system is described for enabling the focus of attention to move around a scene, even when the viewer is mobile in the environment.

4.1 Inhibition of Return

Maxima in the saliency map, the primary output of the visual attention algorithm, correspond to those areas deemed by the model to grab the focus of attention. One may envisage an undesirable situation where a single location continually maintains the maximal level and therefore holds attention indefinitely. Inhibition of return, or IOR, is therefore usually implemented. It refers to the inhibiting of winning locations in the saliency map in order to ensure that the focus of attention may continue to move around and visit other locations of high value.

Since the saliency map is a spatial representation of a scene, IOR is usually also calculated based on 2D spatial coordinates [8]. In this case, an important problem arises that the spatial coordinates only remain valid if the viewpoint does not change e.g. the scene remains still. In the situation where the viewpoint moves, for example the agent walks forward, IOR locations stored during previous iterations can not be used. In this situation, the object-based memory system may instead be utilised to implement IOR. Rather than keeping track of attended-to *locations* i.e. 2D spatial coordinates, in this model, the synthetic vision system (see Section 3.1.2) resolves such locations to determine attended-to *objects*: a measure of the attention paid to an object is therefore considered, rather than spatial coordinates, for guiding the focus of attention around the scene.

4.2 Uncertainty and Scene Exploration

While inhibition of return operates over small time scales, the method can be extended in order to support more general exploratory behaviours. For example, an uncertainty value is stored with each object. This is a single metric stored with the memory entry describing the degree to which an agent has attended to it and could be thought of as representing the completeness and correctness of the agent's representation of the object in memory with respect to the actual object. High uncertainty values may be used to modulate the output of the visual attention mechanism, thus creating a loop between agent and environment. For example, an object with a high uncertainty value will attract the attention of the agent, causing the agent to orient towards it. This will result in the object being impressed into memory. As it remains there, the uncertainty level will decrease. Eventually, another object will win control of attention, the agent's gaze will be redirected to it and the process can continue. Thus the agent will have acquired the ability to explore the scene. Furthermore, scene uncertainty metrics, considering the mean uncertainty level over all visible objects, can be used to vary the arousal of the agent. This could be used to control, for example, the tendency of the agent towards environment exploration rather than other tasks. Highly uncertain environments, for example, may invoke more looking behaviours than environments that the agent has been familiarised with.

5 Conclusions

An outline of a model for storing incoming sensory information from the environment has been presented. Links between memory and attention have been briefly described: these mechanisms seem complimentary for a variety of functional reasons. Such a coupling allows IOR to be based on stored memory values for attended-to objects. Conversely, memory requires an attention mechanism for defining a sensible strategy for filtering and transferring entries between successive stages. In this respect, attention represents one possible candidate for controlling the values described in Section 3.1.1. While the attention mechanism described here is perceptual in nature and operates only in a bottom-up manner, the examples provided are only a few of the many possibilities that may be achieved. For example, a valid argument that has been made against independent memory systems for virtual agents is the cost of storage of multiple copies of the world database [6]. Robust filtering and forgetting mechanisms, based on a sound visual attention strategy, could greatly alleviate such problems, an important future issue for crowd simulation [7].

REFERENCES

- [1] R.C. Atkinson and R.M. Shiffrin, 'Human memory: a proposed system and its control processes', *The psychology of learning and motivation: advances in research and theory*, **2**, 89–195, (1968).
- [2] D. H. Ballard, 'Reference frames for animate vision', in *IJCAI'89: Proceedings of the 11th international joint conference on Artificial intelligence*, pp. 1635–1641, San Francisco, CA, USA, (1989). Morgan Kaufmann Publishers Inc.
- [3] S. Chopra-Khullar, *Where to look? Automating certain visual attention behaviors of human characters*, PhD dissertation, University of Pennsylvania, January 1999.
- [4] H. Ebbinghaus. *Über das Gedächtnis*, 1913.
- [5] T. F. Evers and S. R. Musse, 'Building artificial memory to autonomous agents using dynamic and hierarchical finite state machine', in *Proc. of Computer Animation 2002*, pp. 164–169, (June 2002).
- [6] M. Gillies, *Practical behavioural animation based on vision and attention*, PhD dissertation, University of Cambridge Computer Laboratory, 2001.
- [7] H. Grillon and D. Thalmann, 'Simulating gaze attention behaviors for crowds', *Computer Animation and Virtual Worlds*, **20**(2/3), 111–119, (2009).
- [8] L. Itti and C. Koch, 'Computational modelling of visual attention', *Nature Reviews Neuroscience*, **2**, 1–10, (2001).
- [9] J. Kuffner and J.-C. Latombe, 'Fast synthetic vision, memory, and learning for virtual humans', in *Proc. of Computer Animation*, (May 1999).
- [10] M.Y. Lim, R. Aylett, W.C. Ho, S. Enz, and P. A. Vargas, 'A socially-aware memory for companion agents', *The 9th International Conference on Intelligent Virtual Agents*, (2009).
- [11] *The psychology of attention*, ed., H. E. Pashler, MIT Press, Cambridge, Massachusetts, 1998.
- [12] C. Peters, 'Designing synthetic memory systems for supporting autonomous embodied agent behaviour', *Proceedings of the 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 14–19, (2006).
- [13] C. Peters and C. O'Sullivan, 'Synthetic vision and memory for autonomous virtual humans', *Computer Graphics Forum*, **21**, 743–752, (2002).
- [14] C. Peters and C. O'Sullivan, 'Bottom-up visual attention for virtual human animation', in *Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA)*, pp. 111–117. IEEE Computer Society, (2003).
- [15] C. Siagian and L. Itti, 'Biologically inspired mobile robot vision localization', *IEEE Transactions on Robotics*, **25**(4), 861–873, (July 2009).

Comparing Forgetting Algorithms for Artificial Episodic Memory Systems

Andrew Nuxoll¹, Dan Tecuci², Wan Ching Ho³ and Ningxuan Wang¹

Abstract. Episodic memory is essential for human cognition and has also proven necessary for some intelligent agents. The size of an episodic store grows over time unless some forgetting mechanism is in place to keep it in check. In this research, we investigate the effect that different forgetting mechanisms have on the episodic memory performance. We compare three different forgetting algorithms using three distinct episodic memory architectures in a domain that is designed to highlight the differences between them. Our results show that the choice of forgetting algorithm has a significant impact on overall agent performance.

1 INTRODUCTION

Any intelligent agent must have a memory that it uses to store knowledge that it applies to a domain. At the most fundamental level this contains basic procedural knowledge of how to perform its function. Most agents also possess semantic knowledge, or knowledge of facts about the agent's domain.

Recently, there has been a growing interest in episodic knowledge or memory of specific past events [4, 5, 8, 13, 16, and 17]. Episodic memory is a history, or chronological record of specific events, called episodes. The ability to remember past events enables a system to improve its performance as well as its competence.

In this research, we examine and compare three simple forgetting algorithms for episodic memory:

- Forgetting a randomly selected memory in the episodic store.
- Forgetting the oldest memory in the episodic store.
- Forgetting the memory with a lowest activation value that is calculated based upon the frequency and recency of use.

We conducted our research using three distinct, general-purpose artificial episodic memory systems that have already been applied to a variety of different problems [8, 13, and 17]. Each of the nine unique combinations of system and algorithm has been applied to a domain that we have specifically designed to highlight the effectiveness of the agent.

Our results show that while the random and oldest forgetting algorithms yield similar behaviour, an activation-based forgetting algorithm can be more effective at selecting memories with lower utility and, thus, improving the agent's performance.

Our results also show that agent performance does not decline linearly as the size of the episodic store decreases. Instead, there is a sudden decrease in performance when the size drops below a particular threshold.

Because we have performed our experiment on three different systems, we expect the results of the research to be indicative of the overall effectiveness of the forgetting algorithms. Furthermore, the test bed we have designed is well suited for future research into forgetting for episodic memory.

2 EPISODIC MEMORY

Our goal in building artificial episodic memory modules is not to mimic the human episodic memory, but rather to endow artificial agents with such episodic storage and recall strategies and to explore the additional capabilities they allow. Research in human episodic memory plays a guiding role in our endeavors.

An artificial episodic memory functions by storing events (or episodes) from the agent's past. Retrieval is performed based on a cue; the system presents the agent with episodes that are most similar to a given cue. These recalled episodes can help the agent make better decisions faster. For example, an agent might be able solve problems faster by adapting previous solutions. Additional tasks, such as avoiding unwanted behavior by detecting potential problems, monitoring long-term goals by remembering what subgoals have been achieved, and reflection on past actions, become feasible.

The size of the episodic store grows over time as the agent acquires more and more experiences. A large episodic memory raises a number of problems: it increases the search space of the retrieval algorithm thus negatively impacting retrieval time. An agent that indiscriminately stores all its past experiences might also run into external constraints related to the physical memory available to it.

While opportunities for using data compression to reduce the size of the episodic store exist, ultimately it will continue to grow. The only technique for keeping episodic store below a fixed size is for the agent to regularly forget portions of this knowledge. As a result, we anticipate that forgetting must be an essential part of maintaining a long-term episodic store for an intelligent agent.

A successful forgetting algorithm must fulfil two goals. First, the algorithm must remove knowledge that is least likely to be useful in the future. Logically, an agent that always forgets knowledge that will never be needed again performs equally well to an agent that does not forget anything. Second, the algorithm must be efficient. An algorithm that takes too long to select content to remove will affect the performance of the entire agent.

¹ Dept. of Electrical Engineering and Computer Science, Univ. of Portland, Portland, Oregon USA. Email: {nuxoll, wangn10}@up.edu.

² Dept. of Computer Science, Univ. of Texas at Austin, Austin, TX, USA. Email: tecuci@cs.utexas.edu.

³ School of Computer Science, Univ. of Hertfordshire, AL10 9AB, UK. Email: w.c.ho@herts.ac.uk.

Forgetting is particularly important for general-purpose episodic memory systems. These systems do not have domain-specific knowledge that they can use to identify what is important about a given episode. Without this knowledge, the entire content of the episode is potentially important and must be stored by default. As a result, the episodic store grows very quickly.

3 RELATED WORK

The distinction between procedural, semantic and episodic memory is well established in human psychology. Endel Tulving first described the distinction between episodic and semantic memory [19]. Further research in cognitive neuroscience has attempted to define the distinction between the two memory types [3]. One thing is clear; the behaviour of human amnesiacs demonstrates that a lack of episodic memory is a severe handicap to cognition.

It is readily apparent that most humans routinely forget details of their experiences [6]. Whether this is a result of actual loss of information or inability to retrieve it is uncertain. Empirical studies show that human memory performance declines with time or intervening events. This decay is well described by a power function - called the 'power-law of forgetting' [2, 20, 21, and 22]. The decay of memory performance can be attributed to two factors: decay of unused information or interference of new and old information. Tulving [19] argues that newer memories prevent the recall of older ones. Evidence suggests that human memory actively constructs and stores only abstract and general representations (also called schemata) of these experiences [1], which may be the source of human forgetting.

Previous research demonstrates that in the past, forgetting mechanisms were required in order to create effective intelligent agents. To date, we are aware of no prior work comparing forgetting algorithms.

The instance-base learning technique used in machine learning is a simple form of episodic memory system. In one implementation, Ram and Santamaría [15] forced the system to maintain a finite memory size, by forgetting the most redundant memory in their store.

McCallum [14] used a short-term episodic memory to distinguish between states in a partially observable environment. This agent would learn a proper maximum size for memory store and, thus, might decide to increase the memory store's size as needed to retain critical memories. When McCallum's implementation did forget, it would discard its oldest memory.

Kennedy and DeJong [10] used a utility-based metric for deciding which memory to discard. Their research also showed that forgetting can actually improve the performance of an agent since retrieval will be faster when searching a smaller episodic store.

Brom et al. [4] have conducted research to create human-like agents for virtual worlds. Because of their desire for realism, their system is based human cognition. Its episodic memory contains the most complex forgetting mechanism that we are aware of. This mechanism selects memories to remove based upon their agent and a measure of their "emotional saliency," an approach that is similar to the activation-based forgetting we use in this research.

While forgetting has already been established as a necessary part of some memory systems, our research aims to compare

multiple forgetting mechanisms for their relative merit outside of the context of a specific architecture.

4 EPISODIC MEMORY SYSTEMS

In order to mitigate the architecture-dependent effects in our data, this research was conducted simultaneously on three different episodic memory systems. These systems are described below in brief.

4.1 Generic Memory for Events

Overview: Tecuci [17] investigates building a "generic memory module for events" – an episodic memory that functions separately from the system that is using it. In other words, it encapsulates the episodic functionality as a way to reduce the complexity of the overall system. It focuses on the generic aspects of memory organization and retrieval in isolation of a specific environment and task. This system defines a representation of generic episodes. It uses a multi-layer indexing scheme and has a generic API to external systems.

Application: This memory module has been successfully applied to tasks such as planning, plan recognition and question answering.

Forgetting: The system leaves the decision of when to forget an episode up to the external system.

4.2 Computational Autobiographic Memory

Overview: Ho et al [8] have aims at modelling computationally the psychological concept of autobiographic memory (which is a specific kind of episodic memory). The design of the memory system focuses on identifying the significance of events through measuring the impact they have on agent's synthetic states (e.g. emotions). Therefore, agents with autobiographic memory can remember and recall significant events that originate in their own experiences in order to adapt to the environment and smooth the interaction with a human user in software applications.

Application: Computational autobiographic memory has been integrated into various agent architectures for different research domains, including Artificial Life and synthetic narrative agents [10].

Forgetting: To date, these agents remember every single episode in their memory and no forgetting algorithm has been introduced to the architecture to improve the efficiency of memory storage and retrieval.

4.2 Soar Episodic Memory

Overview: The episodic system created by Nuxoll and Laird [13] is integrated into the Soar cognitive architecture [11]. This system creates episodes by recording a series of snapshots of the current state of the architectures working memory. A new episode is recorded whenever an agent takes an action in its environment. Episodes are retrieved when the agent defines a memory cue in its working memory.

Application: Previous research with Soar's episodic memory system has focused on demonstrating a variety of cognitive

capabilities that are granted to an agent by virtue of possessing an episodic memory. These demonstrations consist of performing a specific task that cannot be performed without a given capability.

Forgetting: Up until now, this system has lacked any sort of forgetting mechanism.

These three systems are different implementations of the concept of episodic memory as they were created with strikingly different research goals in mind. However, they all have an important feature in common: they were designed as general-purpose, domain-independent systems. As a result, they are well suited for comparing the effects of various forgetting mechanisms in a mutual domain.

5 THE WAREHOUSE DOMAIN

The domain used for this research was carefully selected to meet the following criteria:

- An environment with a virtually unlimited number of unique possible states. If the number of possible states is limited, then it would be possible for an agent with a large enough episodic memory store to have an exact match for every state. As a result, the agent's performance would be capped in the long term.
- A domain that allows for virtually unlimited degrees of success as opposed to just success/fail. This allows us to compare degrees of success between agents that used different forgetting algorithms.
- A task for which an optimal solution is very difficult to find but that a better-than-random solution is easy to find. This is intended to verify that all our agents can learn to improve their behaviour in the domain but still be better or worse than other agents.
- An environment that is highly configurable so that it can be fine-tuned it to make differences between agents maximally apparent.
- Because we are connecting this domain to multiple different episodic memory systems, we need an environment with a simple interface that will not need to be changed if we adjust its configuration. Specifically, we want a domain with small number of commands and a simple, extensible input format.
- A domain that can run for an unlimited amount of time. In effect, it must have no goal state and yet always present a challenge to the agent.
- Finally, we want a domain with an obvious connection to a real-world problem and not a "toy" task.

The domain we designed in order to meet these criteria is called the Warehouse domain and is depicted graphically in Figure 1. In this domain, the agent is the manager of a warehouse full of various goods that can be shipped to different cities. Every day one truck is sent to each of these cities one at a time. At any point there is only one truck at the warehouse to be loaded.

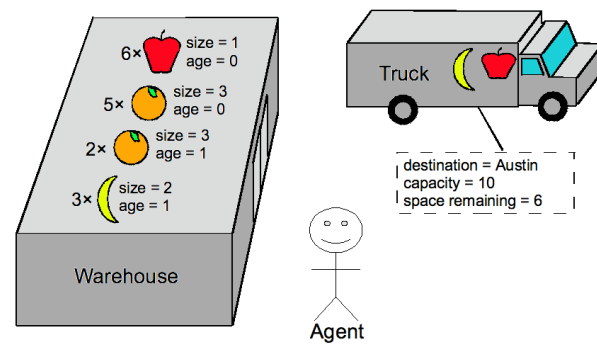


Figure 1. The Warehouse Domain

Each truck has a fixed capacity and therefore is able to hold more of some types of goods than of others. The agent's task is to decide what goods to put on each truck. A given truck could contain multiple different types of goods.

Whenever a truck is shipped to a city, the agent receives some amount of profit for the delivery. The amount of profit that the agent receives will vary based upon a subset of these factors:

- The base value of the goods. For example, gold is more valuable than manure.
- The amount of time that the goods sat in the warehouse before being shipped. In effect, each good has a perish rate that causes its value to change over time. For example, fresh fruit will rapidly lose value but non-perishable items will remain stable in value.
- The combination of goods that are shipped in each truck. Each unique, ordered pair of goods can have a good-to-good relationship that affects the value of one of the goods. For example, a shipment of fine china is worth a lot less if it is shipped in the same truck as livestock.
- The relative worth of the goods in the destination city. Each unique combination of good and city can have a city-to-good relationship that describes how that city affects the value of the good. For example, snowshoes are of little value in Rio De Janeiro but would be invaluable in Reykjavik.

The simulation reaches the end of a day when one truck has been shipped to each city. At this point, new randomly selected goods are added to the warehouse's inventory for the next day. In this way, the warehouse never runs out of goods but the combination of goods is relatively unique.

This domain meets all of the requirements specified at the beginning of this section. It provides the agent with a constant stream of unique states for which an optimal action is difficult to select but a better-than-random action is easy to select. Furthermore, the interface, which described in detail in the next section, is simple yet highly configurable.

While we have attached a meaningful veneer to this domain, it is essentially a combination of the knapsack problem with a scheduling problem [7]. These are both NP-Hard problems that regularly appear in real world software applications.

6 WAREHOUSE CONFIGURATIONS

This section describes in detail how we interfaced the warehouse domain with our episodic memory systems and how we configured the warehouse domain for our experiments.

6.1 Input

The following information is provided the agent at each time step:

- A complete list of all the goods that are in the warehouse. The following properties of each good are visible to the agent: name, quantity, age, size and a unique identifier.
- The truck's overall capacity, how much space is currently available on the truck and the truck's destination city.
- A complete list of all the goods that are in the truck. For each good in the truck, the same info is provided that is provided for the goods that are still in the warehouse.
- If the last action the agent took was to ship the truck then the agent is provided with a reward signal indicating the overall profit from that shipment.

It is important to note that no information is provided to the agent about the value of these goods and how they change based on various circumstances. Furthermore, the agent does not know the significance of the properties of the truck and the goods aside from the fact that it cannot load a good onto a truck for which there is insufficient space. In other words, no heuristic knowledge is being provided to the agent about the domain.

6.2 Output

The agent can perform one of the following actions at each cycle:

- *Load one good onto the truck.* Illegal load commands (i.e., unknown good or insufficient space on the truck) are ignored.
- *Ship the truck.* This sends the truck to its destination city. At this point the agent receives a reward signal and a new, empty truck arrives that has the next destination city.

6.3 Relationships

For our experiment we began with a particular configuration of the warehouse domain that allowed an episodic memory system to consistently provide a significant improvement over random behaviour but could not come close to optimal behaviour. As a result, there was plenty of room for the agent's performance to improve or degrade as we varied the configuration of our systems.

Specifically, we began with three goods (apple, orange, and banana) and three cities (Austin, Portland, and Shanghai). The names of these goods and cities are irrelevant to the task but helpful for differentiating the results we present later in this paper.

The city-to-good relationships were defined as shown in Table 1. In this table, a given entry indicates the effect that the city in the column has upon the good in the row. Specifically, the value of the good was multiplied by the given ratio.

	Apple	Orange	Banana
Portland	1.0	4.0	3.0
Austin	6.0	6.0	1.0
Shanghai	2.0	1.0	4.0

Table 1. City-to-Good Relationships

The good-to-good relationships were defined as shown in Table 2. Each entry indicates the effect that the good in the row has upon the good in the column. As with city-to-good relationships, the value of the good was multiplied by the given ratio.

	Apple	Orange	Banana
Apple	1.0	3.0	3.0
Orange	3.0	1.0	1.0
Banana	1.0	1.0	1.0

Table 2. Good-to-Good Relationships

We measured the performance of each agent by computing the difference between each reward received and the optimal reward for an agent that maximizes its immediate profit.

7 AN EPISODIC MEMORY AGENT FOR THE WAREHOUSE DOMAIN

To compare our forgetting mechanisms we built an agent using each episodic memory system. All agents used the following algorithm:

1. The agent creates a set of all possible actions it can take in the current state. For example, the agent in Figure 1 could load one apple, load one orange or either age=0 or age=1, load one banana or ship the truck as it is currently loaded.
2. Since the agent does not have sufficient semantic knowledge to immediately decide which action to take, it evaluates each possible action using its episodic memory. It does this as follows:
 - a. The agent creates a memory cue based upon its current state and the to-be-evaluated action that it is taking in that state. The specific content of the cue varies from system to system but the intent is the same: The agent is cueing for an episode wherein it took the candidate action in an identical state.
 - b. An exact match to the cue usually does not exist, so a closest partial match is retrieved by the episodic memory system. It is notable that the three systems we used for this research each have different algorithms for determining which episode is the closest match.
 - c. The agent examines the outcome of retrieved memory. In particular, it looks for the reward it received following that action. This reward is used as the evaluation score of the action it is considering in the present.
3. Once all the possible actions have been examined, the agent takes the action that has been assigned the highest evaluation score.
4. In the episodic memory systems created by Tecuci [17] and Ho, et al. [8] episodes span the entire loading of the truck. As a result, the best matching memory selected in step 3 is used to make all subsequent decisions by the agent until the truck has been shipped. In the episodic memory system created by Nuxoll and Laird [13] the episodes are recorded

each time the agent take any action (e.g., load or ship), as a result, steps 1-3 are repeated for each agent action. The impact of this difference is apparent in our experimental results.

8 EXPERIMENTS

We collected our data by performing a series of runs that persisted for 1000 virtual days in the warehouse environment. The agent began each run with an empty episodic memory and gathered episodic knowledge as the run progressed.

We varied three variables for each run:

- **Episodic memory system** – This was one of the three systems described in section 4.
- **Memory limit** – This was the maximum size of the episodic store. The episodic memory limits were selected such that they were spread evenly across a range from less than 1% of amount of data that could be recorded up to two-thirds of the maximum amount of data that could be recorded. The Tecuci [17] and Ho, et al. [8] systems recorded exactly 3000 episodes over the course of 1000 days, or one per shipment. For these systems, episodic memory limits were set at 10, 100, 200, 600, 1000, and 2000. The Soar episodic memory system records episodes more frequently, so its limits were prorated accordingly. In both cases, the limits were sufficient for the agent to completely remember the events over approximately 3, 33, 67, 200, 333 and 667 days respectively.
- **forgetting mechanism** – This was one of three forgetting algorithms that was used to decide what episode to remove from the episodic store when necessary:
 - *Random* - a random memory was selected to remove.
 - *Oldest* - the oldest memory in the store was removed.
 - *Activation* - each memory had an activation value that reflected the frequency and recency with which it had been retrieved by the agent. When it is time to remove an episode, the one with the smallest activation value was removed.

Note: The activation is calculated using the same algorithm as working memory activation in the Soar and ACT-R cognitive architectures [12]. When an episode is created and each time it is retrieved, its activation level is increased by a fixed amount. Over time, however, the activation level decreases exponentially. Specifically, the overall activation level is calculated using this formula:

$$A = \ln \left(\sum_{j=1}^n t_j^{-d} \right)$$

t is a vector of the times when the episode was created or retrieved measured as the number of cycles that have elapsed since that time. d is a constant that determines the decay rate of the activation level. For our research, this value was fixed at 0.8 which is the default value when this formula is used for memory decay [12].

For each particular configuration of the three variables, we performed a total of ten runs and averaged them. A typical result is shown in Figure 3 below. The x-axis of the figure measures the elapsed time as measured in virtual days. The y-axis of the figure is the difference from the optimal reward that was achieved by the agent. Since the performance is measured

relative to optimal behaviour, smaller values are better. A trend line has been added to the graph to show that the agent's performance is improving slightly.

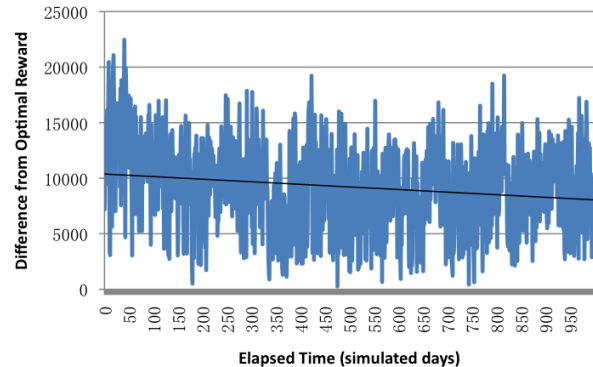


Figure 3. A typical result from one configuration

In this example, the improvement is very modest due to a small episodic memory size. This episodic learning was evident in all the data we collected and became more pronounced as the maximum size of the episodic store was increased.

Throughout this experiment, we gathered a total of 189 results like the one depicted in Figure 3 (or 1890 data runs). To keep the results presented here concise, we have chosen to focus on the results for just one city in our simulation (Austin). The results from the other cities are very comparable.

Figure 4 depicts a compilation of all the results for the computational autobiographical memory created by Ho, et.al. [8]. The horizontal (category) axis depicts the memory limits that were set. The height of each column indicates the average performance of the agent over the course of 10 distinct runs. As before, lower performance numbers are better. The three colors correspond to the three different forgetting mechanisms that were used.

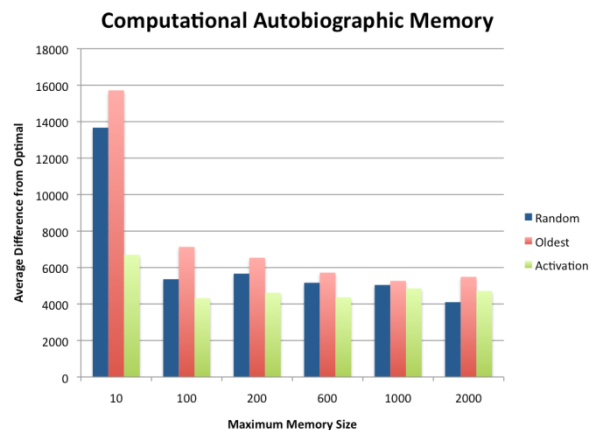


Figure 4. Average agent performance in the city of Austin for each selected memory limit using the computational autobiographical memory system created by Ho, et.al [8].

As the graph depicts, overall performance improves as the memory limit increases. For the random and oldest episode, performance declines sharply at the smallest memory limit.

However, activation-based forgetting maintains much better performance.

Figure 5 depicts the results of the same experiment performed on the episodic memory system created by Tecuci [17]. As shown in the figure, these results are generally comparable to those in Figure 4. The only difference is for a memory size of 10, when activation forgetting performs worse than the other two. Further investigation revealed that this is not due to the forgetting algorithm but the way the Generic Memory for Events performs retrieval. It uses a technique called surface similarity in which the most similar N episodes are retrieved and the best one is used to generate a new load decision. We used $N=10$, which means all episodes in memory are retrieved at every step. For activation forgetting that means they are all activated. This prevents any new episode (other than the original 10) from being stored in memory. When this N is much smaller than the maximum memory size, learning takes place and activation forgetting performs at least as well as random and oldest forgetting.

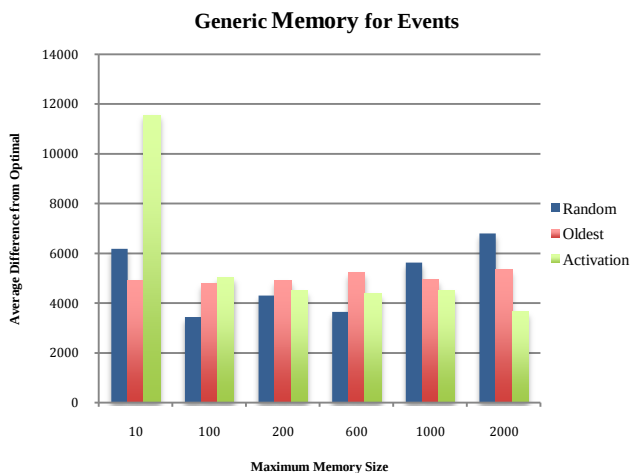


Figure 5. Average agent performance in the city of Austin for each selected memory limit using the generic memory for events created by Tecuci [17].

The results Soar episodic memory system created by Nuxoll and Laird [13] are depicted in Figure 6. These results are somewhat different due to the fact that episodes are recorded more frequently and only one decision is made per episode. However, the general result is the same. As the maximum size of the episodic store decreases, the activation-based forgetting algorithm yields an agent that maintains its performance better than the other two algorithms. Noticeably with the Soar episodic system this effect happens early (when memory size equals 3000) in comparing with the computational autobiographic memory (Figure 4) where it happens when memory size equals 10.

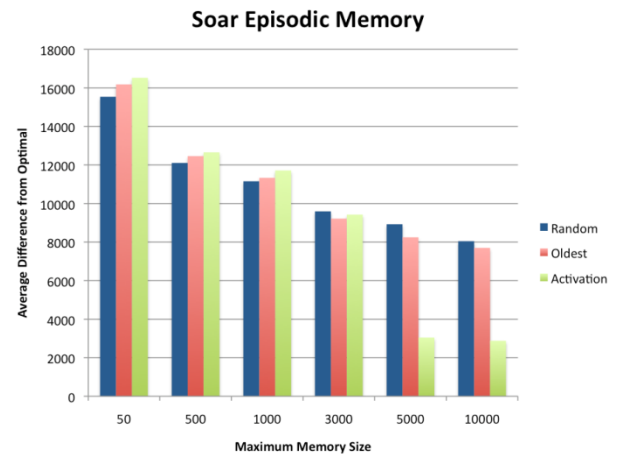


Figure 6. Average agent performance in the city of Austin for each selected memory limit using the Soar episodic memory system created by Nuxoll and Laird [13].

9 CONCLUSIONS AND FUTURE WORK

Current research suggests that forgetting will remain an important component of any episodic memory system that expects to operate over an extended period of time. Without it, the memory store will continue to grow until it exceeds the available storage space.

This paper presents the results of an experiment to study the effects of three forgetting algorithms on agent performance. To make our results as general as possible, the experiment was performed simultaneously on three different, general episodic memory systems. The experiment was also performed using a domain designed to highlight differences between agents.

Some important observations can be gleaned from our results. First, the effects of the memory limits are apparent in all three systems despite significant differences between them. This reinforces the notion that forgetting is an important component of any episodic memory system.

Second, when a forgetting mechanism is put in place the performance of the agent does not decline linearly with the size of the memory store. Instead, performance drops exponentially once the memory limit's size exceeds some threshold. This suggests that the memory limit setting for an episodic memory system is of particular importance. Since the performance of our agents was directly dependent upon the quality of the retrieved episodes, this also suggests that a given domain has a certain minimum number of episodes required to inform effective performance.

Third, the threshold at which performance makes its precipitous drop is partially dependent upon the forgetting algorithm. This suggests that seeking more effective forgetting algorithms is fertile ground for future research.

One factor to consider when studying forgetting mechanisms is the trade-off between the computational cost associated with forgetting and the system's task performance. Selecting the least activated episode to forget yields improved performance, but this comes at the cost of computing that activation. On the other hand, random or oldest forgetting are computationally cheap, but

do not offer the same task performance benefits. We plan on investigating this trade-off in the future.

For the future, some approaches to forgetting that show promise are:

- Removing redundant episodes – The forgetting subsystem would locate the two episodes that are most similar and remove one of them. Ram & Santamaría [15] used an approach similar to this in their continuous case-based reasoning work. The time required to locate the redundant memories may be prohibitive, although this might be combined with retrieval.
- Combining redundant episodes – As above, the forgetting subsystem would identify redundant memories. However, instead of removing them it would separate out the redundant portion and save only one copy of that portion. In effect, each episode would consist of a base episode and a small set of changes. This approach bears some similarity to the creation of schemata described by Alba and Hasher [1].
- Memory Decay – Rather than remove entire episodes, the forgetting subsystem would remove individual features of the episodes. To do this effectively, an algorithm would still be required to select features to remove. An extension of the activation mechanism used for this research appears to be a good candidate for this but may be intractable. In addition, any episodic memory system with a memory decay mechanism would require the agents that used it to be able to process incomplete memories.

REFERENCES

- [1] J.W. Alba and L. Hasher, Is memory schematic? *Psychological Bulletin* 93, 203–231 (1983).
- [2] J. R. Anderson. *Cognitive Psychology and Its Implications*. Worth Publishing, New York, Fifth edition, (2000).
- [3] A. Baddeley. The Concept of Episodic Memory. *Episodic Memory: New Directions in Research*, Oxford University Press, Oxford, UK., (2002)
- [4] C. Brom and Jiří Lukavský., Towards More Human-Like Episodic Memory for More Human-Like Agents. In: *Proceedings of the 9th International Conference on Intelligent Virtual Agents*, Amsterdam, The Netherlands (2009).
- [5] W. Dodd. The Design of Procedural, Semantic, and Episodic Memory Systems for a Cognitive Robot. Master's thesis, Vanderbilt University, (2005).
- [6] H. Ebbinghaus. *Memory: A Contribution to Experimental Psychology*, New York: Teachers College, Columbia University, (1885/1913)
- [7] M.R. Garey and D.S. Johnson, Computer and Intractability: A guide to the Theory of NP-Completeness, W.H. Freeman and Co., New York, USA pp. 5, 241 (1979).
- [8] W.C. Ho, K. Dautenhahn, and C.L. Nehaniv, Computational memory architectures for autobiographic agents interacting in a complex virtual environment: a working model, *Connection Science*, 20:1, 21 – 65 (2008).
- [9] W.C. Ho and K. Dautenhahn, Towards a Narrative Mind: Creating Coherent Life Stories for Believable Agents, *Proceeding of Intelligent Virtual Agent (IVA) 2008*, 59-72 (2008)
- [10] W.G. Kennedy and K.A. De Jong. Characteristics of Long-Term Learning in Soar and its Application to the Utility Problem, In: *Proceedings of the Twentieth International Conference on Machine Learning*, pp 337-344., AAAI Press, Washington, DC, USA (2003).
- [11] A. Newell. *Unified Theories of Cognition*, Harvard University Press, Cambridge, Mass, USA (1990).
- [12] A. M. Nuxoll, J.E. Laird and M. R. James, Comprehensive Working Memory Activation in Soar. In: *Proceedings of the International Conference on Cognitive Modeling (ICCM-04)*. Pittsburgh, PA, USA (2004).
- [13] A. M. Nuxoll and J. E. Laird, Extending Cognitive Architecture with Episodic Memory, In: *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-07)*, Vancouver, B.C., Canada (2009).
- [14] A.R. McCallum, Instance-Based State Identification for Reinforcement Learning, *Advances in Neural Information Processing Systems (NIPS 7)*. (1995).
- [15] A. Ram & J.C. Santamaría. Continuous Case-Based Reasoning. *Artificial Intelligence*, 90(1-2):25-77. (1997).
- [16] M. Sánchez-Marrè, U. Cortés, M. Martínez, J. Comas, and I. Rodríguez-Roda. An Approach for Temporal Case-Based Reasoning: Episode-Based Reasoning. In H. Munoz-Avila and F. Ricci, editors, *ICCBR*, volume 3620 of Lecture Notes in Computer Science, pages 465–476. Springer, (2005).
- [17] D. Tecuci. A Generic Memory Module for Events. *PhD Dissertation*, The University of Texas at Austin, (2007)
- [18] D. Tecuci, B. Porter. "A Generic Memory Module for Events". In *Proceedings to The Twentieth International FLAIRS Conference*, (2007).
- [19] E. Tulving. *Elements of Episodic Memory*. Clarendon Press, Oxford, UK., (1983).
- [20] J. T. Wixted. Analyzing the empirical course of forgetting. *Journal of Experimental Psychology: Learning, Memory & Cognition*, 16:927–935, (1990).
- [21] J. T. Wixted and E. B. Ebbesen. On the Form of Forgetting. *Psychological Science*:2:409–415, (1991).
- [22] J. T. Wixted and E. B. Ebbesen. Genuine Power Curves in Forgetting: A Quantitative Analysis of Individual Subject Forgetting Functions. *Memory & Cognition*, 25:731–739, (1997).

Short- and Long-term Adaptation of Visual Place Memories for Mobile Robots

Feras Dayoub and Tom Duckett and Grzegorz Cielniak¹

Abstract. This paper presents a robotic implementation of a human-inspired memory model for long-term adaptation of spatial maps for navigation in changing environments. The robot uses an appearance-based representation of its workplace as a map, where the current view and the map are used to estimate the robots current position in the environment. Due to the nature of real-world environments such as houses and offices, where the appearance keeps changing, the map may become out of date after some time. To solve this problem the robot needs to adapt the map continually in response to the changing appearance of the environment. In this work we use local features extracted from panoramic images to represent the appearance of the environment. Adopting concepts of short-term and long-term memory, our method updates the group of feature points for the image representation of a particular place. Experiments using robot sensor data collected over a period of 2 months show that the implemented model is able to adapt successfully to changes.

1 INTRODUCTION

Robotic helpers and companions have become a major topic for current research. This trend has been facilitated by the increased performance of modern processors and the emergence of low-cost, high performance sensors such as digital cameras, together with recent advances in academic disciplines including computer vision, artificial intelligence and mobile robotics. Simple robots such as vacuum cleaners and lawn mowers are already available to consumers, while many predict that personal service robots will become as ubiquitous in the home and office of the future as the PCs and mobile phones of today. Common to all such robots is that they will share physical spaces with humans, and will thus need to deal with a dynamic and ever-changing world.

A mobile service robot's first tour of a new environment will probably be guided by a human, for example, to explain the layout and function of different places, just as a human employee would be introduced to a new workplace. One problem for the robot is to build a sufficiently detailed map to begin working in the new environment. However, the initial learning of an environment will only be a short moment in the lifetime of a service robot that is expected to operate for many years.

Most of the existing work on localization and mapping assumes that the environment where the robot works is static. However, this assumption does not hold for many real environments. For example, the appearance of a room in a house is not static over time: new objects are sometimes added, existing objects like pictures or carpets may be changed or moved, and old objects may be removed. Some of

these changes occur very often, such as moving chairs and cushions, etc. These transient variations need to be excluded from the long-term representation of the appearance of the environment.

In this paper, we propose a method to update the reference views of a topological map for a changing environment, which is represented as an adjacency graph where the nodes of the graph represent distinctive places and the edges represent the transitions between places. Adopting concepts of short-term and long-term memory inspired by an information processing model (see Fig. 1) based on the multi-store model of human memory proposed by Atkinson and Shiffrin [2], our method updates a group of feature points for the reference view of a particular place. Thus the reference views are adapted to represent the information about the new appearance of the location.

The reminder of this paper is structured as follows. After dis-

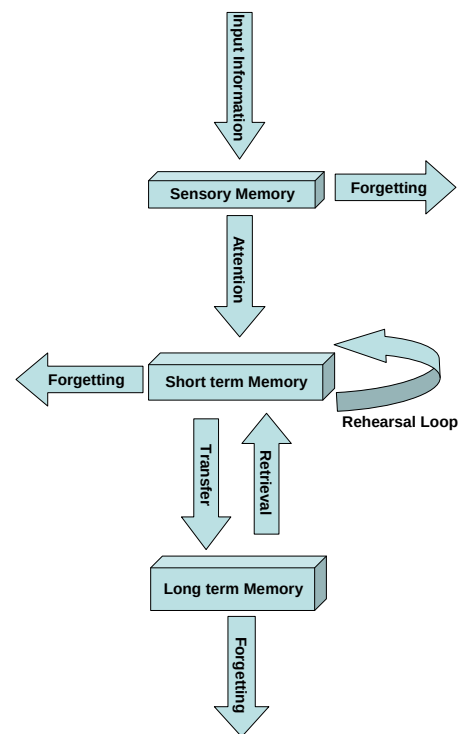


Figure 1. The Information Processing Model.

¹ University of Lincoln, United Kingdom, email: fdayoub,tduckett,gcieniak@lincoln.ac.uk

cluding related work in Section 3, we introduce our memory model and the adaptation mechanism in Section 4. Section 5 presents the experiments and results obtained. Finally we draw conclusions and discuss future work.

2 ADAPTABILITY AND THE HUMAN MEMORY

From a biological point of view, an organism is adaptable if it can survive significant changes in its environment, spread to new habitats, and come up with novel solutions to its surroundings². For humans, adaptability is a key factor for survival and depends heavily on accurate evaluation of environmental changes. This accuracy is a product of our personal experience which is in turn collected and stored in our memory.

When talking about mobile robots which can work among us and inside our everyday environments, the biologically inspired approach sounds very appealing. This is because these robots will work in the same context in which the biological systems themselves operate and will face the same challenges. One of the important problems which need to be solved before we can see these robots sharing our rooms and offices is the ability to adapt to changes in the working space, from the appearance of a room to the behaviour of people.

Nature has had millions of years to evolve and has lead to very successful systems making life able to survive under very hard conditions. Based on this observation one can suggest that we should copy nature's design directly and implement these systems. But anyone with a biological background will know that each biological system has a very long inheritance of tinkering which has produced very complex structures and implementing this is not the right way to go. Instead we should be inspired by biology by discovering the general principles on which these systems are built and then applying such themes in the heart of the solution.

Human memory is not yet fully understood and many of the ideas about how it works are still controversial. Among the ideas which is widely accepted is to look at the memory in terms of multiple stores of information and a set of processes that act on these stores. In this work, we adopt and bring this multi-store theory to the domain of mobile robot mapping and localization, where we use it to enable the robot to adapt its internal representation of the world over time in response to the changing of the appearance of its surroundings.

3 RELATED WORK

Most previous approaches to mapping in dynamic environments assume that the underlying structure of the environment is static, and try to separate moving objects from the stationary parts. The dynamic effects are often treated merely as measurement outliers. Alternatively moving objects are detected and tracked separately using multi-target tracking techniques [15]. Another recent approach tries to classify landmarks as moving or stationary, and incorporates reversible data association within a sliding window of recent observations, to allow moving objects to be included into the SLAM estimate [4]. In general, while these approaches mitigate some problems of classical SLAM algorithms, they cannot handle long-term changes to the structure of an environment.

Several authors have investigated mapping systems that incorporate simple forgetting mechanisms based on recency weighting. For example, Yamauchi [16] developed a system that learns and updates

the topology of a map during runtime based on successful and unsuccessful attempts to traverse a given link. Andrade-Cetto [1] developed an EKF-based mapping system that is able to forget landmarks that have disappeared, where an existence state associated with each landmark measures how often it has been seen. However, none of these methods are general enough to handle environmental changes occurring at different rates, nor has the long-term robustness of these approach been demonstrated in real world environments.

This paper tackles the problem of working in a changing environment by using an appearance based approach to represent the workplace of the robot. An early approach for appearance-based mapping and localization was published in [13] where the operational area of the robot is represented as a graph. The nodes in this graph represent distinctive places and the edges represent the transitions between places. This approach consists of two stages. In the first stage the robot is driven through its operational area to learn a model of the environment, by taking a sequence of images in certain places and then creating a topological map from these images. In the second stage the robot uses the map to find its current position, i.e. the node which is most similar to the current view.

Two different approaches to measure the similarity between images have been presented in the literature: global and local methods. The global methods capture global properties of the image using approaches based on colour histograms [7], principal component analysis (PCA) [14], Fourier transform [10], etc. The local methods extract local properties from the image and produce a group of landmark features. Local feature including SIFT [8], SURF [3], MSER [9], etc., are used to find the similarity between images. The local methods have been shown to be more reliable and robust to illumination and viewpoint changes, thanks to the feature descriptors that are built using a local region around selected feature points. Each feature is described by a high-dimensional vector, which has high invariance to image translation, scaling and rotation, and partial invariance to illumination changes and affine projection.

Using local image feature descriptors, the similarity between two images can be measured by finding the correspondences between the features in the two images. This can be done by finding the closest feature in the feature descriptor space. However, the method can be time consuming if the number of images in the map is large and the search is done linearly. Using a text retrieval approach, Sivic and Zisserman [12] presented a very fast retrieval system using a visual vocabulary. Nister and Stewenius [11] extended the ability of the system by using hierarchical K-means clustering to improve the performance, so that Fraundorfer et al. [5] were able to implement global localisation in real-time.

4 LONG-TERM ADAPTATION USING MEMORY STORES

For any mobile robot using a topological map to represent its working space, the map could become out-of-date after some time in a changing environment. A naive solution to this problem would be simply to replace the image representation for each node in the topological map from time to time, in order to reflect the changed appearance of the corresponding location. Provided that the robot is correctly localized, this approach would enable the robot to remove out-of-date information from the map. However, it could also remove useful features due to temporary occlusions, and could lead to catastrophic results in the case of localization errors. A better solution would be to update the image representation of a node incrementally, by gradually adding information about new stable features in the en-

² <http://humanorigins.si.edu/aop/adaptability.htm>

vironment, while removing information about features that no longer exist. In our approach, each node is represented by a group of features (using SURF in these experiments, though other features could be used). This group of features is updated over time by adding persistent new features and removing older ones that are no longer used. In order to achieve this we adopted short-term and long-term memory concepts based on the multi-store model of human memory proposed by Atkinson and Shiffrin [2]. This model, which forms the basis of modern memory theories, divides human memory into three stores:

- sensory memory (SM),
- short-term memory (STM),
- long-term memory (LTM).

4.1 Recall, Rehearsal, Transfer

The sensory memory contains information perceived by the senses, and selective attention determines what information moves from sensory memory to short-term memory. Through the process of rehearsal, information in STM can be committed to LTM to be retained for longer periods of time. In return, the knowledge stored in LTM affects our perception of the world, and influences what information we attend to in the environment.

Applying these concepts to our approach for robot mapping, the sensory memory will contain the features extracted from the current image. Then an attentional mechanism selects which information to move to STM, which is used as an intermediate store where new observations are kept for a short time. Over this time the system uses a rehearsal mechanism to select features that are more stable for transfer to LTM. In order to limit the overall storage requirements and adapt to changes in the environment, the system also contains a recall mechanism that forgets unused feature points in LTM by removing these features from the node. LTM is used in turn by the attentional mechanism for selecting the new sensory information to update the map.

We model the world as a set of discrete places. In our experiments, omni-directional vision is used to provide the features for localization and mapping. We assume that an initial map of the whole environment has already been created by the robot, e.g. using an existing algorithm for topological mapping of static environments. (In this work we selected the places by hand.)

Each place has two memory stores: STM, which is a temporary stage, and LTM, which provides the reference views in the map used for self-localization. We assume that the robot is able to self-localize by matching features extracted from the current view to the stored reference views. The purpose of our algorithm presented here is to maintain up-to-date reference views for the nodes in the map, using recall and rehearsal concepts inspired by human memory. To initialise the map, the image data from the robot's first tour of the

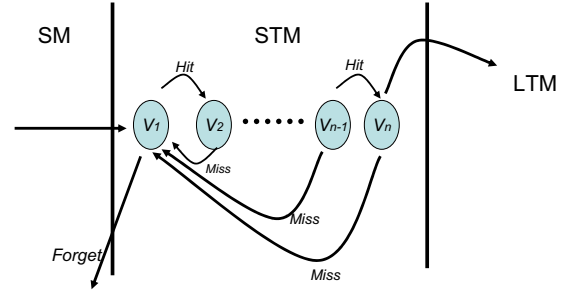


Figure 2. The rehearsal stage in the STM.

Algorithm 2 The rehearsal stage in the STM.

```

for (every feature in CrrSTM) {
  if (feature in newFP) {
    Move the feature to the next state.
    if (feature state > STMIng) {
      Move the feature to the CrrNode.
      Remove the feature from CrrSTM.
    }
  } else if (feature in the first state) {
    Remove the feature from CrrSTM.
  } else {
    Reset the feature to the first state.
  }
}
for (every feature in newFP) {
  if (feature was not in CrrSTM) {
    Add the feature to CrrSTM in the first state.
  }
}

```

environment is used. One image is selected to represent each node in the map. For each node, local features are extracted and used directly to initialise LTM, while STM for each node is initially assigned to be empty.

Thereafter, every time the robot visits an existing node, the following steps are carried out. Feature points are extracted from the current view. Self-localization is carried out by comparing the current features to the reference features of each node (LTM) to estimate the current node. In our case, we apply global localization by place recognition using a simple winner-take-all strategy, although any appropriate self-localization algorithm could be applied, e.g. Markov localization. After localization, the current features are used in the recall stage for updating the LTM of the current node. Only new features which do not match any feature in LTM are used in the rehearsal stage. Algorithm 1 describes the two main stages; (1) recall, where the difference in appearance between the reference and current views is computed, and (2) rehearsal, where this difference is used to update STM and commit persistent new features of the location to LTM. Algorithm 2 shows the rehearsal process for a stored feature in STM, which is also represented as a finite state machine in Fig. 2. This stage represents what Atkinson and Schiffrin called rehearsal in their memory model (Fig. 1), i.e. the process of continually recalling information into the STM in order to memorise it. In order to

Algorithm 1 Update the reference view.

Definitions:

CrrNode: The reference view of the current node.
 CrrView: The current view for the current node.
 CrrSTM: The current STM for the current node.
 STMIng: The maximum number of states in the STM.
 LTMIng: The maximum number of states in the LTM.
 newFP: The difference between the CrrView and CrrNode.

```

for (every visit to the node) {
  newFP = recall( CrrNode, CrrView, LTMIng )
  rehearse( CrrSTM, newFP, STMIng )
}

```

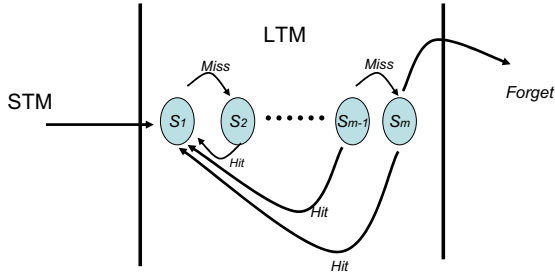


Figure 3. The recall stage in the LTM.

Algorithm 3 The recall stage in the LTM.

```

newFP = []
for (every feature in CrrNode){
    if (feature in CrrView){
        Rest the feature to the first state.
    }
    else
        Move the feature to the next state.
}
if (feature state > LTMIng){
    Remove the feature from CrrNode.
}
}
for (every feature in CrrView ){
    if (feature not in CrrNode){
        Add the feature to newFP.
    }
}
return newFP

```

transfer a feature point from STM to LTM the feature has to be seen frequently in that node. Features enter STM from sensory memory and must progress through several intermediate states (V_1 to V_n) before transfer to LTM. Every time the robot visits the node and finds the feature (“hit”), the state of the feature is moved closer to LTM. However if the feature is missing from the current view (“miss”), it is returned to the first state (V_1) or forgotten if it is already there. This policy means that spurious features should be quickly forgotten, while persistent features will be transferred to LTM.

Algorithm 3 shows the recall process for a stored feature in LTM, which is also represented as finite state machine in Fig. 3. This process first involves updating the LTM by matching the reference view to the current view. In order to remain in the LTM, a feature has to be seen occasionally in that node. In contrast to rehearsal, features enter LTM from STM and must progress through several intermediate states (S_1 to S_m) before being forgotten. Stored features which have been seen in the current view are reset to the first state (S_1), while the state of features which have not been seen is progressed, and a feature point that passes through all states without a “hit” is forgotten. Finally, recall returns the list of new features that were not already present in the LTM.(i.e the difference in appearance between the current and reference views).

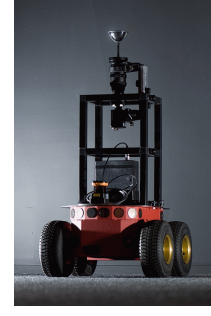


Figure 4. The experimental platform. An ActivMedia P3-AT robot equipped with an omnidirectional vision system.

5 EXPERIMENTS AND RESULTS

To investigate our method of updating the reference views in a topological map, we conducted two experiments. In the first experiment we tested the system for a single node represented by a view of an office room. In the second experiment we used an image data set recorded over a period of approximately two months from a real changing environment.

Our experimental platform was an ActivMedia P3-AT robot equipped with a GigE progressive camera (Jai TMC-4100GE, 4.2 megapixels) with a curved mirror from 0-360.com (see Fig. 4).

For local feature extraction we use the SURF algorithm. This algorithm extracts local features from the scale-space of the image based on the Hessian matrix and approximates the second order Gaussian derivatives with box filters. A fast non-maximum suppression algorithm is also used. The resulting algorithm has a good performance in the extraction process and a high accuracy. For more details, see [3].

5.1 Testing the system using data from a manually changing environment

For the first experiment, the robot first takes a reference view in the middle of a room. Then the robot is moved back and forth covering approximately 1 m on each side of the place where the reference view was recorded, while the appearance of the room is changed manually. The changes involve the arrangement of the objects inside the room, including adding new objects like boxes and posters, removing existing objects individually and also covering them with movable office dividers for certain periods. Fig. 6 shows two sample views from the room where the experiment was conducted.

While the robot was moving, a set of 251 images was recorded to capture the changing appearance of the room. After capturing each image, SURF image features were extracted and saved to a file. Using these files, the memory model was then tested offline.

The average number of features extracted from each image was 600.0 ± 56.1 . The mean number of matched points between the images of the dataset and the static reference view was 45.2 ± 25.4 points whereas it was 98.8 ± 18.7 when the dynamic view was used, indicating that the adaptive reference view represents the current state of the environment more faithfully than the static view.

Fig. 5 shows how the number of feature points changes inside the LTM and STM during the experiment, using 8 stages for the LTM and 3 stages for the STM. The size of the STM changes rapidly where the features enter and stay for a short time whereas the change on the LTM size is slow as the features stay longer.



Figure 6. Views of the room during the experiment. Various objects were added, removed or changed manually to test the adaptive capabilities of the system.

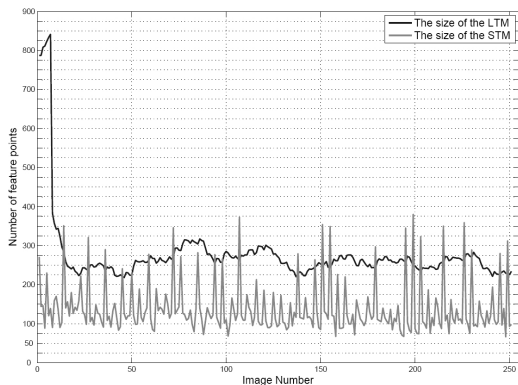


Figure 5. Number of feature points into the LTM (with 8 stages) and the STM (with 3 stages) during the experiment.

5.2 Long-term experiment using data from a real changing environment

This experiment was carried out in a real changing environment of an office floor at the University of Lincoln. This area is used for student and staff activities.

Over a period of approximately two months, we took the robot on tours between the offices while recording a set of images. The result was 36 tours (one tour per day) generating 6161 images with approximately 35cm between consecutive images. We used the first tour to create a topological map by choosing 21 omni-directional images from different places to form the reference views for the nodes. Fig. 7 shows the ground truth positions of the recorded images. Each colour represents the group of images which were assigned to the same node. The crosses represent the reference view for each node. In order to obtain the ground truth data we used the GMapping library [6]. The GMapping algorithm provides a Simultaneous Localization and Mapping (SLAM) solution based on a Rao-Blackwellized particle filter. The output of the algorithm is an estimate of the robot trajectory along with an occupancy grid map of the environment. The first tour is used for creating the topological map. For each subsequent tour, we estimated the robot's true trajectory through the environment then we aligned the output map with the first tour map in order to transform the ground truth navigation trajectories into a common reference frame. We emphasize that our method does not build or require a metric map of the environment. We only use this information for ground-truth validation.

We tested our method for adapting the reference views in the map using the collected images. We used a global localization method based on place recognition (winner-takes-all). We carried out the

tests using the memory model with 8 stages in the LTM and 2 stages in the STM.

The results show that the failure rate of the global localization was around 1.5% when the adaptive approach was used whereas the failure rate for the static one was 3.5%.

Fig. 8 shows the cumulative percentage of matched points used for the global localization. The red line illustrates the results when the static reference views were used for the map whereas the green line corresponds to the adaptive views. The adaptive approach provides better matching scores leading to better localization performance and also better navigation performance. For example, 43% of the time the number of matched points between the reference view and the current image from the robot was below 35 points for the static views, whereas this happened only 24% of the time with the adaptive views.

6 CONCLUSIONS AND FUTURE WORK

This paper introduced an adaptation mechanism for topological localization methods based on features extracted from images that represent the appearance of the environment. Our method updates the reference views of the nodes in the map and keeps track of the changes in the appearance of the environment, while the robot is

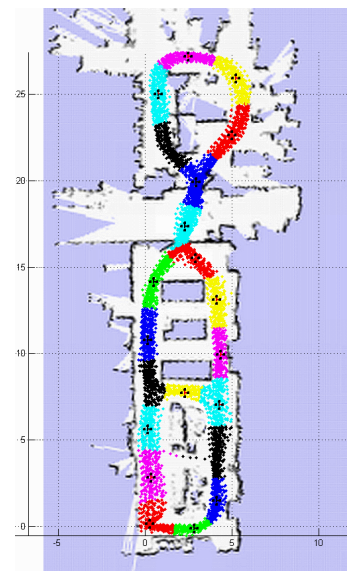


Figure 7. Ground truth positions of the recorded images obtained from the laser-corrected odometry. The constructed map consists of 21 nodes, each colour represents the group of images which were assigned to the same node. The crosses represent reference views for each node.

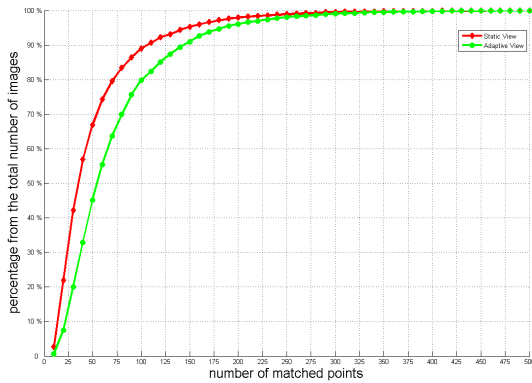


Figure 8. The cumulative percentage of matched points used for the global localization. The red line illustrates the results when the static reference views were used for the map whereas the green line corresponds to the adaptive views.

working over long periods of time. To achieve this we adopted short-term and long-term memory concepts to adapt the reference views of the nodes in response to the dynamics of the environment. To test our method, we conducted two experiments. The first one was in an office room where we manually changed the appearance of the place. For the second experiment we used data recorded from a real dynamic environment over two months. In both experiments the method gave improved results over static mapping.

In this work, the number of the stages in LTM and STM were determined empirically based on the recorded sensor data. As a future work, the number of the stages could be learned depending on the dynamics of the real environment. The attention mechanism could be improved by adding real-time tracking of features in the scene to filter out spurious features due to noise or temporary occlusion. The adaptive capability of the map could be further extended to the topological level, by making the robot able to add or remove nodes and links from the map.

7 DISCUSSION

Memory is a process by which we can store and retrieve pieces of information. In this work, the pieces of information are the image features. We used these features to recognize a place in a topological map by finding the correspondences between the features in the LTM and the current view, and also adapting the group of features in the LTM in response to the changing appearance of the node in the map through the recall and rehearsal processes. These image features have very interesting properties. The descriptor of each feature is a signature making the matching process to find these features invariant to scale, orientation, affine distortion and partially invariant to illumination changes. These properties can open the door to use these features in a higher level of representation for the environment. The topological map, as it has been presented, is a coarse level of representation for the environment which limits the possibilities to interact with the robot. For example, we can ask the robot to go from A to B but we cannot ask the robot to search for a specific object in the map. For these kind of tasks we need what is called a semantic map, where the map will contain, in addition to the spatial information, information about the objects which share the space. In our case, the semantic level can be built using the same image features. If we extract image

features from an omnidirectional image taken from a room inside a home environment, we can notice that the features are not distributed uniformly over the space and certain areas have more density than others due to how the objects are arranged inside the room. Each area could contain one real object or multiple parts of objects which generate the features. In order to describe a place in the environment, the robot can use these groups as what we will call abstract objects (AO). These AO's can be tracked over time using the memory model. Each time an abstract object reaches the LTM memory, the robot can ask a human to associate it with a semantic label if the features of the AO came from a real object or ignore the AO if the features do not correspond to a single object. In this way the robot will start to build a map of the objects over time and also, thanks to the memory model, be able to forget objects which have been disappeared from view.

REFERENCES

- [1] J. Andrade-Cetto and A. Sanfeliu, 'Concurrent map building and localization in indoor dynamic environments', *Int. J. Pattern Recognition and Artificial Intelligence*, **16**(3), 361–374, (2002).
- [2] R.C. Atkinson and R.M. Shiffrin, 'Human memory: A proposed system and its control processes', in *K.W. Spence & J.T. Spence (Eds.), The Psychology of Learning and Motivation*, **2**, 89–195, (1968).
- [3] H. Bay, T. Tuytelaars, and L. Van Gool, 'SURF: Speeded up robust features', in *Proc. European Conference on Computer Vision (ECCV)*, (2006).
- [4] C. Bibby and I. Reid, 'Simultaneous localisation and mapping in dynamic environments (SLAMIDE) with reversible data association', in *Proc. Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, (June 2007).
- [5] C. Fraundorfer, F. Engels and D. Nister, 'Topological mapping, localization and navigation using image collections', in *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)*, (2007).
- [6] G. Grisetti, C. Stachniss, and W. Burgard, 'Improved techniques for grid mapping with Rao-Blackwellized particle filters', *IEEE Transactions on Robotics*, **23**(1), 34–46, (2007).
- [7] H.M. Gross, A. Koenig, C. Schroeter, and H.J. Boehme, 'Omnivision-based probabilistic self-localization for a mobile shopping assistant continued', in *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)*, (2003).
- [8] D.G. Lowe, 'Object recognition from local scale-invariant features', in *Proc. IEEE International Conference on Computer Vision (ICCV)*, (1999).
- [9] J. Matas, O. Chum, M. Urban, and T. Pajdla, 'Robust wide-baseline stereo from maximally stable extremal regions', *Image and Vision Computing*, **22**(10), 761–767, (2004).
- [10] E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro, 'Image-based Monte Carlo localisation with omnidirectional images', *Robotics and Autonomous Systems*, **48**(1), 17–30, (2004).
- [11] D. Nister and H. Stewenius, 'Scalable recognition with a vocabulary tree', in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2006).
- [12] J. Sivic and A. Zisserman, 'Video Google: a text retrieval approach to object matching in videos', in *Proc. IEEE International Conference on Computer Vision (ICCV)*, (2003).
- [13] I. Ulrich and I. Nourbakhsh, 'Appearance-based place recognition for topological localization', in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, (2000).
- [14] N. Vlassis, B. Terwijn, and B. Krose, 'Auxiliary particle filter robot localization from high-dimensional sensor observations', in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, (2002).
- [15] C.-W. Wang, C. Thorpe, and S. Thrun, 'Online SLAM with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas', in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 842–849, Taipei, Taiwan, (2003).
- [16] Brian Yamauchi and Randall Beer, 'Spatial learning for navigation in dynamic environments', *IEEE Trans. Systems, Man and Cybernetics*, **26**(3), 496–505, (1996).

I can (almost) remember what you are doing: from actions to tasks

Rudolf Kadlec¹, Cyril Brom¹

Abstract. Recently there has been an increased attention to the episodic memory research in the community studying intelligent virtual agents. An episodic memory can exploit a hierarchical representation, in which higher layers present increasingly more abstract tasks. During storage, an agent with episodic memory can directly access these abstractions from its own behavioral representations or from behavioural representations of agents being observed, provided the representations in question are hierarchical. However, the abstractions are lacking when observing behaviour of human users. In this paper we propose a way in which our already existing computational model of episodic memory can be extended by an ability to incorporate knowledge about abstract tasks of human users. We have tested ID3 classification algorithm and hidden Markov model for reconstructing these tasks from observations. Our approach was tested in a simple scenario with one agent situated in a 3D environment of the game Unreal Tournament that is being observed and its tasks are then inferred.

1 INTRODUCTION

Episodic memory [1] (EM) is a memory for personal history of an entity. It was proposed that it enhances believability of life-like intelligent virtual agents (IVAs) [2].

Compared to other published computational models of EM (see [2] for overview), our model [3] has some unique features capitalizing on the fact that IVAs' behaviour is often expressed in a hierarchical fashion. Today, models like hierarchical behaviour trees or hierarchical finite state automata are *de facto* standards in computer games industry [4]. The behavioural hierarchies present in the program controlling the IVA can be at the same time used for storing the IVA's history in EM, making the EM representation also hierarchical.

We are developing our model in an incremental way. First, we have implemented a central module for remembering history of one IVA [3] in a hierarchical tree-like structure. An IVA's behaviour is represented by a forest of AND-OR trees where each tree corresponds to one top-level task. The inner nodes of the tree represent tasks or sets of alternative tasks that the IVA has to complete in order to satisfy the top-level task. Leafs of the tree are atomic actions that can be executed in the environment. Figure 1 shows an example of such tree.

In the central module of our EM, the IVA's remembered history is represented as a sequence of executed tasks preserving their hierarchical structure. Upon this central module we have implemented gradual forgetting [2, 3] that is driven by emotional response to the remembered event and also by its age. Lower

levels of the hierarchy can be forgotten first, thus preserving at least an abstract summary of agent's actions. Other implemented extensions of the central module include abilities to use socially established time patterns [5] and more plausible spatial representation [2].

So far the model was able to store only an IVA's own actions. Theoretically this approach can be also used for other computer controlled agents because the behavioural hierarchy is represented in their "minds" and can be disclosed to an observing IVA. However, human users cannot be treated this way since their abstract tasks are not directly accessible by the program, an observer can see only a stream of atomic actions. The hierarchy has to be inferred from the available data – that is from that stream of atomic actions and from a state of the world. In other words, we have to reconstruct the AND-OR tree from observations to store it in the episodic memory.

In order to deduce the actual AND-OR tree, we have to: 1) Find a suitable representation of an IVA's state. 2) Find out the concrete task that most probably produced the observed state (we mean by a *concrete task* the tasks from the first layer upon the atomic actions – see Fig. 1 – and, for brevity, when the context makes it clear, we will use the term *task* for *concrete tasks*). 3) Then reveal more abstract levels in the tree based on already known lower level tasks (layers 2 and 3 in Fig. 1).

We have taken the first step towards this goal by inferring the concrete tasks from streams of atomic actions. To this end we have tested two machine learning algorithms. First of the selected approaches has taken a naïve assumption that only one observation of an agent's current state is needed in order to deduce the concrete task. ID3 [6] classifier was used in this stage. The other approach used hidden Markov model (HMM) [7] and operated with sequences of observations. ID3 was taken as a simple solution that served as a baseline for comparison with the HMM that is better suited for temporal inference.

Performance of the algorithms was tested in a scenario with an IVA whose behaviour was classified by both algorithms. The IVA was used for generating training data that consisted of atomic actions, state of the agent's surrounding and the agent's current concrete task. Then, testing data, only atomic actions and the agent's surrounding, were generated by the same agent. The agent's program was not deterministic. Behaviour of human users was not classified yet. Both the agent and the classification mechanism were implemented in Java. Agent was coded using Pogamut [8], which is a middleware for interfacing with the virtual environment of Unreal Tournament 2004 [9].

The rest of the paper continues as follows: Sect. 2 presents possible alternative solutions, Sect. 3 describes our IVA's architecture, Sect. 4 formalizes the problem, Sect. 5 presents our scenario, Sect. 6 describes the experiments, and Sect. 7 discusses the results and possible future directions. The last section concludes the paper.

¹ Dept. of Software and Computing Science Education, Charles University in Prague, Ke Karlovu 3, 12118, Prague, Czech Republic.

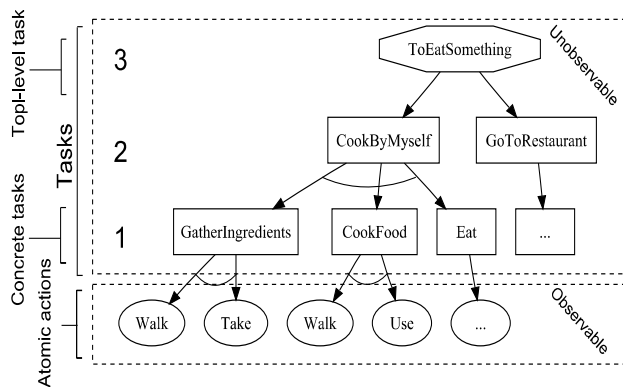


Figure 1. An example of a tree representing behaviour of an IVA. The top-level task *ToEatSomething* on the highest level can be realized by the task *CookByMyself* or by the task *GoToRestaurant*. Each of these tasks is further decomposed into several concrete tasks. Atomic actions, shown on the lowest level, are the only output that can be observed when behaviour is generated by a human user’s avatar.

2 POSSIBLE ALTERNATIVES

In general the problem of determining an agent’s motivation, based on the observed state is referred to as a “keyhole plan recognition problem”. There exist many approaches to solving this issue; here is a non exhaustive list of these methods.

Case based plan recognition (CBPR) used for example in [10] originates from the case based reasoning (CBR) paradigm. CBPR works with a library of cases/plans, each case is a sequence of observed states. Another possibility is to use mental state abduction [11]. This technique supposes that the decision mechanism can be described by a set of implications. Switching antecedents and consequents then makes it possible to reason what rules may have caused current state of the IVA. Both the abduction and the CBPR are symbolic approaches and it is questionable how they would deal with noise and uncertainty.

The other branch of techniques follows the probabilistic approach. Majority of these algorithms use a form of HMM, for example [12]. Algorithm for hierarchical probabilistic plan recognition is proposed in [13]. For HMM’s ability to cope well with uncertainty, we decided to employ this approach to our purpose as well. To our knowledge, HMM was not used previously for the problem of high level task recognition in domain of computer games.

3 OUR AGENT’S ARCHITECTURE

Our IVA’s architecture is a reminiscence of a classical cognitive AI architecture, by which many virtual agents have been inspired. Our IVA is driven by *hierarchical reactive planning* with behaviour represented by *AND-OR trees*. The AND-OR tree metaphor works with abstract *goals* representing what shall be achieved, and *tasks* representing how to achieve the goals. Typically, every goal can be accomplished by several tasks, while every task can be achieved by adopting some sub-goals. In Figure 1, goals were intentionally omitted for parsimony. As said

in Introduction, in this paper, we are interested only in concrete tasks; thus, the higher layers of the hierarchy, including goals, are of no importance here.

The tasks that cannot be decomposed are *atomic actions*, i.e. action primitives. Every task may need several resources to be performed, i.e. *objects*. Every top-level goal has its *activity level* based on drives, external events, and a schedule. The goals compete among themselves and the winning goal chooses the most appropriate task (e.g. “to eat” goal can chose “take something from the fridge”) and passes its template to the *task field* of the *visual short term memory*. From the AI standpoint, this mechanism capitalises on the BDI framework [14].

The visual short-term memory holds templates of objects seen that passed through a simple threshold-based *attention filter*. Every object is regarded as a tool for action, i.e. it is a set of “affordances” [15], meaning it possesses pointers to the tasks it can be used as a resource for. These pointers are perceived directly by the IVA when observing his environment. Objects in experiments we have been running are state-less for the sake of simplification, though our simulations allow the objects to have states as well.

4 FORMALISATION

In this section, we will formalize previously presented agent architecture in the context of task recognition problem.

Suppose that the IVA is controlled by a decision making system (DMS), which can be formalised by function $DMS: P \times S \rightarrow A \times S$ where P is the IVA’s perception, A actions that can be performed in the environment and perceived by other IVAs and S is the agent’s internal state. A part of S is also the IVA’s *motivation* M . M provides an explanation of the agent’s actions; it can be for example currently running procedure or a plan speaking in terms of BDI [14] or a trace in AND-OR tree going from last issued atomic action to the root of the active tree or an emotion state. For purposes of our experiments, M will contain only concrete tasks. In case of IVAs, both A and S can usually be disclosed for observing agents (with some exceptions such as an agent going to raid a bank). In case of human users, S and thus M is usually unobservable.

The representation of the set P can be fairly complicated. Considering domain of computer games, there can be positions and states of all observed objects, information about geometry of the level and virtually any feature of the simulation engine that can be accessed by the *DMS*. To reduce the dimensionality of P we have mapped it to a new space $P' = \{0, 1\}^n$ where each vector component means presence of a salient feature in an agent’s surrounding. Detected features have to be selected by a designer with respect to the agent’s *DMS*. P' also has to contain only features that can be observed by an external observer because it will be used for external task inference.

With this notation, the classification of IVA’s motivation, i.e. the inference of concrete tasks, is a job of constructing the sequence $m_1, m_2, \dots, m_T$; $m_t \in M$ given the sequence $x_1, x_2, \dots, x_T$; $x_t \in P' \times A$. This means inferring motivation based only on the external observation of actions issued by the agent and on current state of the environment.

5 PROBLEM DOMAIN

In our example scenario the IVA have 5 top-level tasks: *Cooking, Training, GoToToilet, PlayComputerGames, Hygiene*.. Those were decomposed into 20 concrete tasks. All the concrete tasks and their associated top-level tasks are listed in Table 1. As can be seen there are generally two types of concrete tasks: tasks that usually control the IVA's interaction with some world objects (e.g. *WashHands*) and tasks executed when the IVA is looking for some objects/places with which/where he can execute another desired tasks (eg. *Search_PlaceToWashHands*).

Concrete task names	Top-level intention the concrete task belongs to
<i>Idle</i>	<i>NONE</i>
<i>Search_InPocket</i>	All top-level tasks
<i>Search_PlaceToWashUp</i>	<i>Cooking, GoToToilet, Hygiene</i>
<i>WashHands</i>	<i>Cooking, GoToToilet,</i>
<i>Search_Ingredients, GatherIngredients, Search_ToCookIn, Cook, Search_PlaceToEat, EatTheFood</i>	<i>Cooking</i>
<i>Search_Toilet, UseToilet</i>	<i>GoToToilet</i>
<i>Search_InternetAccess, ConfirmTraining, Search_SportGround, Play</i>	<i>Training</i>
<i>Search_Computer, PlayVideogames</i>	<i>PlayComputerGames</i>
<i>BrushTeeth, WashFace</i>	<i>Hygiene</i>

Table 1. List of all concrete tasks and their top-level tasks.

In our experiments, M was a set of all concrete tasks from Table 1. Set A consisted of atomic actions $\{USE, EAT, TAKE, IDLE, WALK\}$. In simplified perception P' , each component of the vector was associated with one affordance. If an object with this affordance was near the agent, the component was assigned value 1, otherwise it was set to 0. Affordances specified in the scenario were *COOK_IN, EAT_IN, STUDY_AT, WASH_UP, FOOD_STORE, PLAY_ON, HAS_INTERNET, GO_ON_TOILET, WORK_OUT_AT*. As you can see, P' has 9 dimensions. Actions were coded similarly as perception. Each atomic action has been associated with a dimension: if the agent was performing an action, the vector component corresponding to this action was assigned value 1; otherwise it was set to 0.

6 EXPERIMENTS

In this section, we describe two classification methods we have used for inferring motivations of IVA's in our scenario.

In the first instance we tested a naïve classification method that have no access to the observed history – ID3 tree classifier. Hidden Markov model that was tested in the second experiment was chosen because it is well suited for temporal reasoning.

In the training phase the IVA was executed for approximately 12 hours of the simulated time. This resulted in 1200 data samples. Then we tested the algorithms on 650 samples obtained from the second simulation run.

6.1 Reasoning about one observation - Tree based classification

Motivation. In our first experiment we have used tree based ID3 [6] classifier to serve us as a baseline in assessing performance of the HMM based classification. We choose ID3 because it is one of the well known algorithms suitable for classification of vectors with nominal attributes without many parameters that has to be fine tuned to get the best performance.

Method. Input to the ID3 was just the current snapshot of the agent's perception and its action ($P' \times A$), not the previous time steps. The target class was the current task. We used ID3 implementation from Weka 3.6 [16].

Results. Error of the classification was 47%. The classification was successful in cases where the perceivable context implied the task, as in the cooking example (error in all non *Search_** tasks was only 9%), but failed when the context was insufficient (error in *Search_** tasks was 78%). This happened because P' contained neither the current location of the IVA nor its past actions, all states when the agent was walking were perceived as the same. There are more ways of how to overcome this problem. For instance, we can aggregate several last states into one vector and run the ID3 on that vector, but this does not seem as a scalable solution. Instead, we have adopted a solution employing hidden Markov model.

6.2 Reasoning about sequence of observations - Hidden Markov model based classification

Motivation. HMM is well suited for reasoning under uncertainty that is why we decided to test it also in our domain. We hypothesize that HMM will outperform ID3 results from the previous experiment.

Method. HMM works with an assumption that the observed process is driven by some probabilistic finite automata whose current state $S_i \in S$ can be observed only indirectly via symbols from an observation alphabet $V = \{v_1, v_2, \dots, v_m\}$. The HMM λ is then defined by $\lambda = \langle A, B, \pi \rangle$ where A is a matrix of probabilities of transition between every two states; B is an observation symbol probability distribution, that is, probability of observing symbol v_j given the automata is in state S_i ; and π is an initial state probability.

If we have a sequence of observations $O = O_1 O_2 \dots O_t$, $O_i \in V$ we can compute the most probable sequence of inner states $Q = q_1 q_2 \dots q_t$, $q_i \in S$ that generated this observation. This sequence can be found efficiently by the Viterbi algorithm [7].

If we put S equal to set of all tasks and V equal to $P' \times A$ then we have translated our plan recognition problem into finding the most probable sequence Q given the set O and HMM λ . For practical reasons, the set of all observations V was simplified by k-means clustering to just 20 possible symbols. Observation space clustering is a standard technique used in HMM [7].

In the training phase, matrices A and B were estimated from the data logged through the IVA's execution, π was set to equal value for all states since we do not know which state will be the initial. When some conditional probability was 0 according to the data we set it to 0.001, this expresses the fact that even events that were not observed in the training data may happen.

In the testing phase, the Viterbi algorithm was executed with the whole testing sequence as an input. Implementation of the Viterbi algorithm was taken from the Jahmm³ library.

Results. Resulting error in prediction was 20%. Error in *Search_** tasks was 30%, error in non *Search_** tasks was only 7%. Compared to the ID3 the improvement was in sequences when the IVA was mostly walking between locations. The information that was missing in the context of the current state was compensated by context of previous states. Most of the remaining error in *Search_** tasks is caused by situations when the IVA was looking for some object *A* but then changed its mind and started looking for some other object *B*. When the *B* object was later used in some recognized task then the whole previous searching sequence was classified as *Search_B* instead of *Search_A* and then *Search_B*.

7 DISCUSSION AND FUTURE WORKS

A positive result of this paper is that the experiments with HMM from previous section suggest that this technique can be used for classification of concrete tasks. Thus the first level of the behavioural hierarchy upon atomic actions can be reconstructed and then stored in EM. However, the current model of task inference is only a proof of concept. We have used the same agent for both training and testing. The next step will be to extend the scenario and explain actions of one IVA by another IVA controlled by different tasks which in turn determine the HMM λ used for inference. This means that an IVA with different “vocabulary” of known tasks determined by its own behavioural representation will try to explain behaviour of other IVA. However, our major focus is on testing the performance of our model on explaining tasks of human players. We will have to obtain sequences of users’ actions in a virtual scenario of similar complexity to the current scenario. Then we will annotate these sequences by a hierarchy of tasks and finally test the HMM model on inferring this hierarchy.

The current experiments were focused on determining only the concrete tasks, i.e. the first level upon leaves of the behavioural hierarchy. To infer higher levels of the tree, we aim at trying to construct a second HMM that will have already discovered concrete tasks on its input and the top-level tasks will be its states.

8 CONCLUSION

In this paper we have presented our existing computational model of episodic memory, proposed how it can be extended to represent actions of other IVAs and humans whose internal state cannot be accessed by the EM module. Finally we have proposed, implemented and tested two solutions in a simple virtual scenario. One solution used the ID3 classifier and one used the HMM. As we have expected the HMM outperformed the ID3 in revealing the task hierarchy.

Acknowledgement. This work was partially supported by the student research grant GA UK 21809 and by the grant 201/09/H057 and by the research project MSM0021620838 of the Ministry of Education of the Czech Republic.

REFERENCES

- [1]E. Tulving, W. Donaldson. *Organization of memory*. New York: Academic Press. (1972).
- [2]C. Brom, J. Lukavský. Towards virtual characters with episodic memory II: Episodic memory strikes back. In: *Proc. Empathic Agents, AAMAS workshop*. 1 - 9. (2009).
- [3]C. Brom, K. Pešková, J. Lukavský. What does your actor remember? Towards characters with a full episodic memory. In: *Proc. ICVS, LNCS 4871*, Springer-Verlag. (2007).
- [4]D. Isla. Handling Complexity in the Halo 2 AI. (2004). URL: http://www.gamasutra.com/gdc2005/features/20050311/isla_01.shtml [21.1.2010]
- [5]O. Burkert. Connectionist Model of Episodic Memory for Virtual Humans. Master thesis. Charles University in Prague. (2009).
- [6]J. R. Quinlan. Induction of Decision Trees. In: *Machine Learning 1*, pp. 81-106. (1986).
- [7]L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*. pp. 257—286. (1989).
- [8]J. Gemrot, R. Kadlec, M. Bida, O. Burkert, R. Pibil, J. Havlicek, L. Zemcak, J. Simlovic, R. Vansa, M. Stolba, T. Plch, C. Brom. *Pogamut 3 Can Assist Developers in Building AI (Not Only) for Their Videogame Agents*. In: *Agents for Games and Simulations*, LNCS 5920, Springer, pp. 1-15. (2009).
- [9]Epic Games Inc. Unreal Tournament 2004. (2004). URL: <http://www.unrealtournament2004.com> [21.1.2010]
- [10]M. Fagan, P. Cunningham. Case-Based Plan Recognition in Computer Games. In: *Case-Based Reasoning Research and Development*. Springer Berlin. pp. 161-170. (2003).
- [11]M.P. Sindlar, M.M. Dastani, F. Dignum & J.-J.Ch. Meyer. Mental State Abduction of BDI-Based Agents. *Proceedings of DALT 2008*, Springer Verlag, pp. 161-178. (2008).
- [12]K. Han, M. Veloso. Automated robot behavior recognition applied to robotic soccer. In: *Robotics Research: the Ninth International Symposium*, John Hollerbach and Dan Koditschek, editors, pp. 199-204. Springer-Verlag, London. (2000).
- [13]N. Blaylock, J. Allen. Hierarchical instantiated goal recognition. In: *AAAI Workshop on Modeling Others from Observations*, Boston, (2006).
- [14]M. E. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press: Cambridge, MA. (1987).
- [15]J. J. Gibson. The Theory of Affordances. In: *Perceiving, Acting, and Knowing*, Eds. Robert Shaw and John Bransford. (1977).
- [16]M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten. The WEKA Data Mining Software: An Update. In: *SIGKDD Explorations*, Volume 11, Issue 1. (2009).

³ Jahmm: <http://code.google.com/p/jahmm/> [19.1.2010]

Extending Soar with Dissociated Symbolic Memories

Nate Derbinsky¹ and John E. Laird¹

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

1 INTRODUCTION

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

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

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

2 ARCHITECTURAL OVERVIEW

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

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

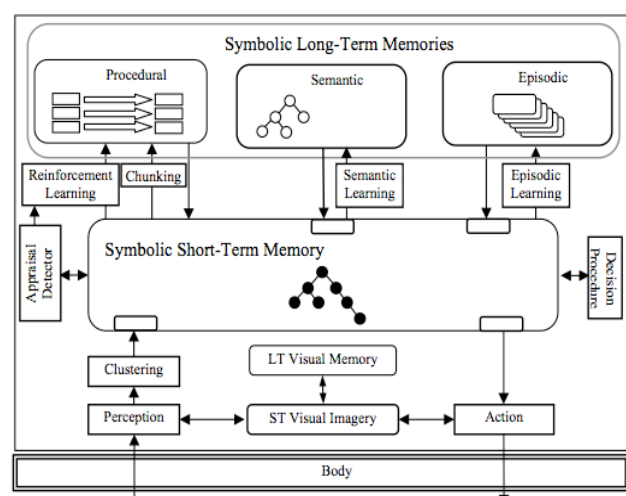


Figure 1. Soar 9: memories (rounded) and processes (box)

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

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

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

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

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

3 WORKING MEMORY

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

Soar's working memory entails no active processing, but instead serves as a common representation substrate for procedural reasoning, initiating external action, and cueing retrievals for other long-term memories. It is composed of an arbitrarily large set of *mental entities* and associated symbolic augmentations [22], represented in a single connected graph. A mental entity can represent an object in the world (present, past, or hypothetical) or a mental concept, such as an "idea," with no experienced physical parallel. Mental entities can be created from the external environment (either via perception or motor system feedback) or the internal memory systems (such as the firing of a production rule or an episodic/semantic retrieval) and the representation does not distinguish as to knowledge

provenance. Augmentations, implemented as an arbitrarily large set of symbolic attribute-value pairs for each mental entity, provide a fully general, relational description language. With these theoretical commitments, working memory serves functionally to allow the agent to symbolically represent arbitrary and novel combinations and compositions of experienced and hypothetical mental entities during its decision-making.

4 PROCEDURAL MEMORY

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

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

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

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

5 SEMANTIC MEMORY

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

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

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

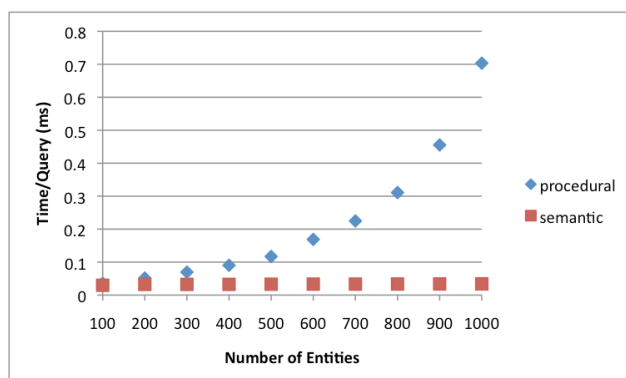


Figure 3. Semantic matching time vs. number of entities

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

6 EPISODIC MEMORY

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

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

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

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

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

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

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

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

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

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

7 SUMMARY OF MEMORY SYSTEMS

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

entities. The procedural memory represents knowledge about when and how to perform internal and external actions.

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

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

8 INTER-MEMORY OBJECT IDENTITY

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

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

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

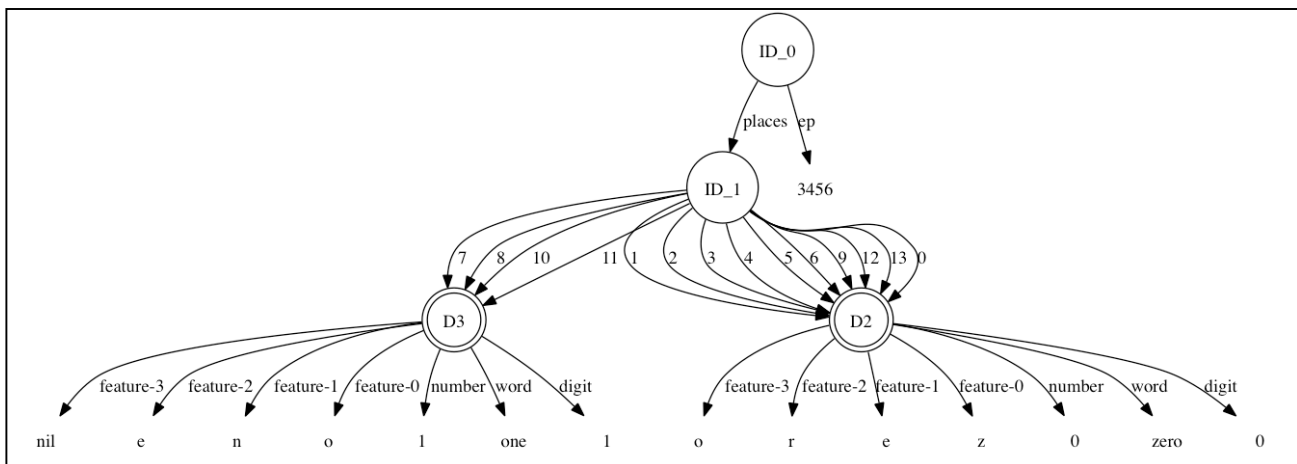


Figure 4. Generalized counting example

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

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

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

We have evaluated the performance implications of object identity in Soar’s episodic retrievals using a generalized counting domain. The task requires a sequence of retrievals, in which the agent relationally represents an incrementing counter in a particular numeric base. For instance, in Figure 4, the agent is representing the decimal number 3456 in base 2. Here, node D2 represents a zero (0), node D3 represents a one (1), and the numeric edges connecting node ID_1 to D2 and D3 indicate that

when 14 positional “places” are used to represent 3456 in binary (“00110110000000”), the 7th, 8th, 10th, and 11th digits (zero-based) will be one’s, and the rest zero’s. In our experimental setup, we can vary the number of episodes (the maximum number to which the agent counts), the number of positional digits used to represent the number, and the numeric base in which the value is relationally represented. Additionally, we can associate an arbitrary number of features with each digit object.

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

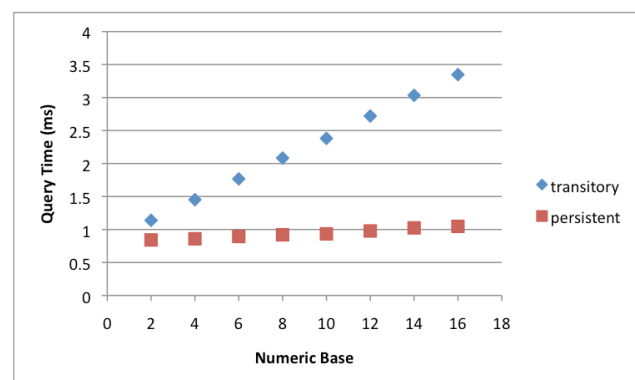


Figure 5. Time (ms) retrieving the most recent episode vs. numeric base

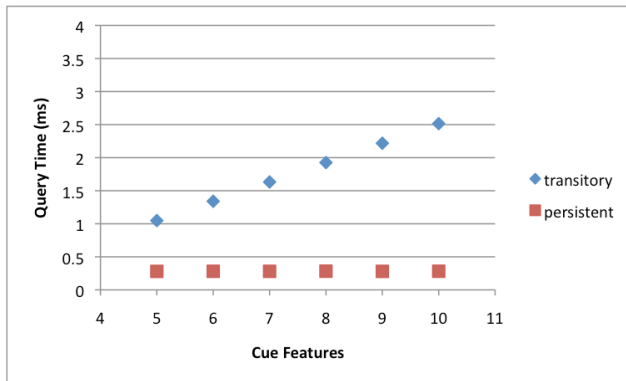


Figure 6. Time (ms) retrieving the most recent episode vs. cue size

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

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

9 CONCLUSIONS & FUTURE WORK

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

Many important integration issues remain for future investigation. As indicated, learning *when* to commit to persistent object identity is crucial for an effective balance of

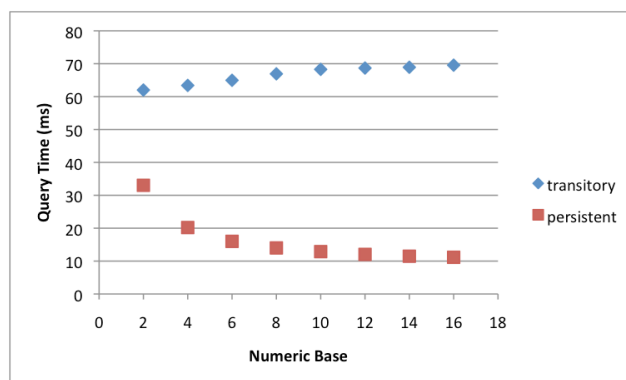


Figure 7. Time (ms) to traverse all candidate episodes

efficient knowledge access and learned knowledge transfer. Right now the agent performs deliberate storage. One possible route is to incorporate working memory activation or appraisal [14] meta-data as measure of entity importance, and commit when some threshold is exceeded.

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

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

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

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

REFERENCES

- [1] Anderson, J.R., Bothell, D.B., Michael D., Douglass, S., Lebiere, C., Qin, Y.: An Integrated Theory of the Mind. In: *Psychological Review*, Vol. 111 (4), pp. 1036-1060 (2004)
- [2] Derbinsky, N., Laird, J.E.: Efficiently Implementing Episodic Memory. In: *Proceedings of the 8th International Conference on Case-Based Reasoning (ICCBR)*, pp. 403-417 (2009)
- [3] Doorenbos, R.B.: *Production Matching for Large Learning Systems*. PhD Thesis, Carnegie Mellon (1995)
- [4] Douglass, S., Ball, J.: Large Declarative Memories in ACT-R. In: *Proceedings of the 9th International Conference of Cognitive Modeling (ICCM)*, (2009)
- [5] Forgy, C.L.: Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. In: *Artificial Intelligence*, Vol. 1, pp. 17-37 (1982)
- [6] Gorski, N.A., Laird, J.E.: Learning to Use Episodic Memory. In: *Proceedings of the 9th International Conference on Cognitive Modeling* (2009)
- [7] Khoshafian, S.N., Copeland, G.P.: Object Identity. In: *Proceedings of the 1986 Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, pp. 406-416 (1986)
- [8] Laird, J.E.: Extending the Soar Cognitive Architecture. In: *Proceedings of the 1st Conference on Artificial General Intelligence (AGI)*, pp. 224-235 (2008)
- [9] Laird, J.E., Derbinsky, N.: A Year of Episodic Memory. In: *Workshop on Grand Challenges for Reasoning from Experiences*, 21st IJCAI (2009)
- [10] Laird, J.E., Rosenbloom, P.: The Evolution of the Soar Cognitive Architecture. *Mind Matters: A Tribute to Allen Newell*, pp. 1-50 (1996)

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

Exploring the Space of Computational Memory Models

Nate Derbinsky¹ and Nicholas A. Gorski¹

Abstract. Over their lifetimes, intelligent agents gain knowledge that may be pertinent to their decisions about acting in the world. One goal of memory system research is to develop an optimal set of encoding, storage, and retrieval mechanisms that will harness these experiences to facilitate rational decisions. In this paper, we propose a direction of empirical, computational research that seeks to better understand the behavioural dynamics that arise when an agent endowed with long-term memory is situated in a task, by determining which properties of task and characteristics of memory systems in combination are responsible for which aspects of behaviour. We propose preliminary taxonomies for task and memory spaces, and also propose metrics for systematic evaluation of memory systems.

1 INTRODUCTION

Learning agents gain knowledge that may be pertinent to making intelligent decisions. To harness these experiences while remaining reactive to dynamic environments, cognitive architectures employ one or more *memory systems*: mechanisms that efficiently implement a fixed policy to encode, store, and retrieve agent knowledge [6]. Prior work provides significant psychological [10][14] and computational [3][8] evidence for dissociated memory systems [12][15][16], yet a significant challenge lies in understanding, for a particular task, the functionally optimal set.

One approach to this problem is to empirically explore the functional benefits of individual classes of memory systems in isolation. For instance, Nuxoll [7] has provided theoretical and empirical evidence that endowing an intelligent agent with a task-independent episodic memory affords it a multitude of cognitive capabilities that may be crucial to the efficacy of a robustly intelligent agent. While insightful and focussed, this approach has its limitations. For instance, implementing a memory system may prove to be quite challenging [2], limiting the speed with which one can study the space of design decisions over multiple tasks. As a compounding factor, a memory system's efficacy may also depend on how well it supports an agent's ability to learn to use it [4]: this encompasses not only how an agent might learn to condition behaviour in the environment based on knowledge that is retrieved from memory, but also how an agent learns to perform encoding, storage and retrieval actions to control the memory mechanisms themselves.

Comprehensive empirical study of a large design space has been done before. One example is in reinforcement learning (RL) [13], which has achieved broad and successful application in cognitive and computer science, partially due to its general formulation and dependency only on a scalar reward function.

While RL offers insight into adaptive control given this function, it does not specify the source or nature of reward. Much work, for instance [1], has shown the importance and difficulty of specifying an optimal reward signal. Therefore, recent work looks to empirical study over reward function space [11]. Similar approaches are being applied to studying the large design spaces of cognitive architecture [5] and empirical game modeling [9].

In this paper, we propose a similar direction of empirical, computational research over the space of memory models, such as to understand the connection between properties of task, characteristics of memory systems, and an agent's learned behaviour. Our hypothesis is that for different classes of problem spaces (distinguished along the dimensions of task that we propose below), different classes of memory models will result in qualitatively different behaviour. By empirically exploring the interactions that arise between different memory models and tasks, we will improve our understanding of which memory models are appropriate in which classes of problem spaces. While many of the dimensions of memory that we identify are inspired by research into the properties of human memory, we are also interested in exploring computational memory models that differ from human memory along certain dimensions; this may allow us to formulate principled functional arguments for why human memory mechanisms might have certain properties. The ultimate goals of our research, then, are (1) to understand how memory system requirements and efficacy change along with parameterized properties of task and (2) to develop computational structure and constraints for studies of memory system dissociation, ideally resulting in evidenced sets of generally useful memory systems.

To accomplish our goals, we propose a methodological framework of empirical study that will enable us to measure the behavioural implications of the interaction between properties of task and characteristics of architectural memory systems. As conceptually depicted in Figure 1, we assume an adaptive agent, A , possessing fixed initial procedural knowledge but endowed with the ability to learn additional control knowledge (over actions both in its environment and its internal memory mechanisms [4]). This agent is endowed with a set, M , of one or more memory systems selected from architectural memory system space (Section 2), and situated within a domain, T , selected from task space (Section 3). We posit that quantitative and qualitative analysis of the agent's learned behaviour (Section 4), while systematically varying M and T , will serve to objectively evaluate the efficacy of sets of memory systems under different task conditions. After describing these spaces and metrics, we propose an approach for exploring the space of possible memory models that we believe will yield insights into the relationships between memory and task (Section 5).

We propose that agents in our study use RL to modify their control knowledge over time. We are interested in adaptive agents, rather than those with fixed procedural knowledge, as it is difficult to predict the best strategies for memory usage across

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

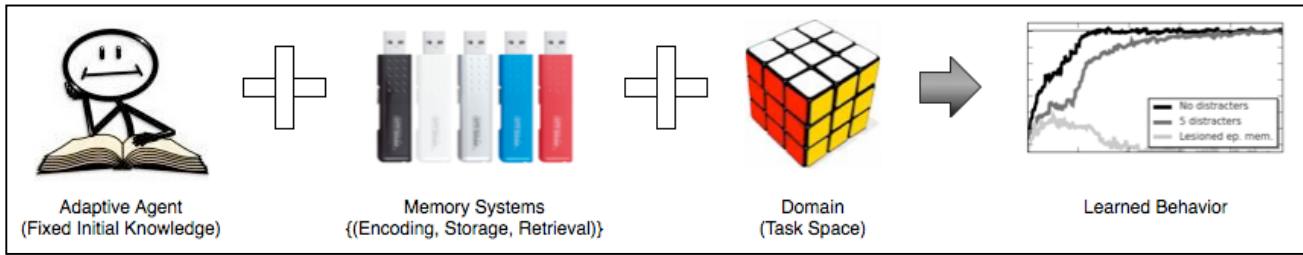


Figure 1. A conceptual depiction of the methodological framework that we will use in our investigations.

arbitrary combinations of memory models and problem spaces. RL offers a framework for optimizing agent control knowledge, without the need for an explicit model of the external constraints of the environment, nor the internal constraints of the agent architecture.

2 MEMORY MODEL SPACE

We have developed the following initial characterization of the space of memory systems, borrowing heavily from and generalizing Nuxoll's breakdown of the space of episodic memory systems [7]. These dimensions of memory mechanisms will be explored in combination with dimensions of task space (section 3) in order to better understand the dynamics that relate memory usage to task behaviour.

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.

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.

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.

We define a memory system implementation as a commitment to features from within the space defined by these dimensions, represented as (encoding, storage, retrieval).

When deliberate agent knowledge can affect architectural memory mechanism function, such as to initiate encoding or retrieval, the agent has a space of actions that it can execute in order to affect this change. These actions are modulated by the same action selection process that controls actions in the external environment, and may vary from the null set (where all encoding, storage and retrieval is architecturally fixed and not affected by the agent's central action selection loop) to where any access to memory (encoding, storage, or retrieval) must be deliberately selected by the agent's internal reasoning system.

3 TASK SPACE

We have identified a preliminary taxonomy of environmental characteristics. These characteristics can be quantitatively parameterized and are independent of each other. This allows for a principled empirical exploration of task space, where an agent with fixed memory models can be situated in a task that varies along one or more dimensions. While these characteristics are not comprehensive, they do offer a preliminary set with which we can begin to explore the circumstances in which particular classes of memory models afford advantages over others.

- *Temporal Distance to Salient Knowledge* – Consider a task in which the agent perceives a sign containing a single element of knowledge (for example, either the symbol "A" or "B"). Later, the agent encounters two doors, and must choose which door to open. If it chooses correctly, then it

receives positive reward; incorrectly, negative. Each door is identified by a symbol; the symbol on the correct door also appeared on the sign. The temporal distance between the salient knowledge is a parameter that can be varied, and although in the example only a single element of knowledge was considered, it can be varied across all elements of salient knowledge in a task. How this characteristic of task interacts with a particular memory mechanism will depend on the persistence of knowledge in memory, as well as whether the temporal distance will affect how easily knowledge can be retrieved from memory.

- *Categories of Salient Knowledge* – We refer to the number of distinct elements of knowledge that must be simultaneously maintained in memory while an agent acts as the number of categories of knowledge. If the example above were to be extended, such that there were three signs, each with a unique symbol corresponding to three sets of doors through which the agent must pass, then there would be three categories of salient knowledge vs. one category for the original example. In complex domains, there might be large amounts of knowledge that can be perceived in the domain, but relatively few categories of actual salient knowledge that must be brought to bear on reasoning in order to act well, requiring only very limited capacity in a memory model. Alternatively, there could be a very large number of categories, in which case an effective memory model would require large capacity.
- *Quantity of Salient Knowledge* – We refer to the number of possible values that each category can take on as the quantity of knowledge. This can vary from one to infinity for each category. Intuitively, this is the number of distinct symbols that could appear on a sign in the example task.
- *Quantity of Distracting Observations* – The number of observations that are irrelevant to acting in the world (i.e. are not salient) will have an effect on how easily an agent can retrieve salient knowledge from memory. These observations are distractors across all tasks in a study, not just a single task instance.
- *Sparseness of Reward* – As rewarding events become more sparsely distributed, a learning agent has more difficulty in determining which actions were beneficial and which were not. This will directly affect how well a learning agent is able to adapt to constraints of task and memory.
- *Relative Cost of Acting vs. Reasoning* – Actions that don't lead to immediately rewarding events may be associated with a penalty. The penalty for acting in the world vs. for accessing memory can vary.
- *Size of Action Space* – Particularly pertinent to adaptive agents, the size of the action space will determine how much exploration the agent must undertake, and will affect how easily the agent is able to learn to use memory (if the agent has adaptive control over internal memory actions).
- *Stochasticity of Actions* – As with the size of the action space, this characteristic is particularly relevant to adaptive agents

and will in part determine how much exploration in the environment is necessary, and thus how easily an agent can learn to control memory.

4 EVALUATION METRICS

Once the agent, endowed with its memories, has completed the task, we must evaluate its learned behaviour both quantitatively and qualitatively. As to the former, we could consider any/all of the following learning metrics:

- Average reward-per-step
- Maximum reward attained
- Speed of convergence

These measures can be tempered by the amount of memory system storage. We also introduce the concept of *knowledge coverage*. As depicted in Figure 2, at each decision, there is a certain amount of agent experience pertinent to making an optimally rational decision. A retrieval from any given memory system, such as an episodic or semantic memory, may achieve some subset of this knowledge (and multiple retrievals may well overlap). The concept of coverage attempts to get at the role of memory systems in making rational decisions. From a quantitative standpoint, an ideal memory system set on a task will learn quickly, converging to a relatively high reward-per-step, while obtaining maximal pertinent knowledge coverage using the least amount of storage.

Qualitative analysis is also crucial to understanding the role of memory systems, and we have two intended directions. First, our qualitative analysis will involve categorizing behaviours supported by memory in accordance with the cognitive capabilities that agents exhibit [7]. For example, in [4], the authors identified behaviour that was suboptimal but involved using an episodic memory mechanism as a stigmergic memory to condition behaviour. If some memory models support a subset of the cognitive capabilities afforded by other models, then an analysis of how the memories vary along the space of characteristics will improve our understanding of memory. Clustering algorithms applied to agent trajectories can assist in detecting classes of behaviour, which can then be examined by hand and categorized as to how memory was used by the agent.

Second, our qualitative analysis will also take the form of comparing agent actions in a single task across different points in memory space. By doing so, we can gain a conceptual understanding of the functional benefits and drawbacks of a

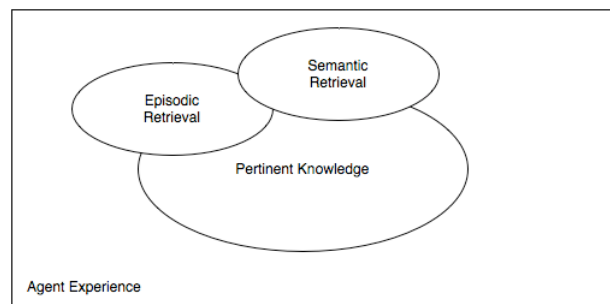


Figure 2. Knowledge coverage

particular memory system. We do not yet have a definitive understanding of how to do this systematically, especially since many points in memory system space may have no direct or even intuitive parallel in humans, but we are confident that a programmatic approach will yield satisfactory results.

5 PROPOSED EXPLORATION

Having discussed characteristics of memory, properties of task, and evaluation metrics, we now discuss our proposed direction for systematically exploring the space of computational memory models in order to understand the functional capabilities that they afford intelligent agents. As the possible spaces of memory models and tasks is infinitely large, our approach is to identify several specific points in the space and initially begin expanding the space of models that we consider starting from those specific points. The initial points in this space will be selected based on biological inspiration, as biological memories are existence proofs of significance, and similarity to existing computational memory models (for relevance).

We plan on pursuing two initial directions. The first, inspired by work performed by Gorski and Laird [4], is focused on exploring the functional characteristics of memory mechanisms situated in tasks but in the absence of adaptive control. In this direction, we will begin with one type of memory mechanism and vary it along several dimensions to better understand how architectural commitments impact the functional capabilities of a memory model.

For example, we have considered systematically sampling the space of episodic memory systems. Given Tulving's functional requirements for an episodic memory implementation, we could fix encoding initiation as automatic (at some architectural frequency), architectural (meaning deliberate agent knowledge cannot modify stored knowledge), and implement a deliberate, bit-vector retrieval policy, using nearest-neighbor (NN) symbolic match and recency as a tie-breaking bias. While holding task properties constant, we could systematically and efficiently vary encoding determination as a function of perception, activation, and other architectural state. Furthermore, we could introduce *forgetting* dynamics (meaning, architectural decay of stored bits) while sampling across a range of temporal distance to salient knowledge. All of these modifications are trivial to implement in a bit memory, and performance is not an issue. In a learning task, we would hypothesize that an agent whose forgetting mechanism decayed faster than the temporal distance would perform quantitatively no better than an agent without a long-term memory, and probably learn not to make use of deliberate memory retrievals.

The second direction focuses primarily on the interaction of learning to control memory. Here, we will begin with several initial memory models: Soar's episodic memory mechanism [2], Soar's semantic memory [3][6], and a simple computational bit memory. We will explore agents that learn to control these memories in several different tasks that vary along the dimensions described above so as to understand their initial functional limitations. When we identify differences in the tasks that agents can learn to use each memory model, we will begin modifying the memory models along dimensions that are most relevant to the functional differences that we have observed in order to determine which characteristics of memory have the biggest functional effects on which types of tasks.

From a computational standpoint, there are many unresolved issues with our approach. For instance, the memory and task taxonomies we have proposed are far from comprehensive and, even if they were, it is unclear how to computationally instantiate an element within these spaces. Given these instantiations, we are also far from realizing an intelligent agent in which we have confidence to optimally learn control over any set of memory systems. Finally, once we overcome these hurdles, the space of memory systems is truly vast, and any comprehensive search therein will be a significant computational and analytical challenge.

6 CONCLUSION

In this paper, we have proposed a principled approach to evaluating memory systems. While there is much analytical and computational work left to implement, we believe this direction of investigation will lead to a better understanding of the role of memory in intelligent agents situated within interesting tasks.

REFERENCES

- [1] Abbeel, P., Ng, A.Y.: Apprenticeship Learning via Inverse Reinforcement Learning. In: Proceedings of the Twenty-first International Conference on Machine Learning (ICML), (2004)
- [2] Derbinsky, N., Laird, J.E.: Efficiently Implementing Episodic Memory. In: Proceedings of the 8th International Conference on Case-Based Reasoning (ICCBR), pp. 403-417 (2009)
- [3] Derbinsky, N., Laird, J.E.: Extending Soar with Dissociated Symbolic Memories. In: Symposium on Remembering Who We Are – Human Memory for Artificial Agents, AISB (2010)
- [4] Gorski, N.A., Laird, J.E.: Learning to Use Episodic Memory. In: Proceedings of the 9th International Conference on Cognitive Modeling (2009)
- [5] Howes, A., Lewis, R.L., Vera, A.: Rational Adaptation Under Task and Processing Constraints: Implications for Testing Theories of Cognition and Action. In: Psychological Review, Vol. 116 (4), pp. 717-751 (2009)
- [6] Laird, J.E.: Extending the Soar Cognitive Architecture. In: Proceedings of the 1st Conference on Artificial General Intelligence (AGI), pp. 224-235 (2008)
- [7] Nuxoll, A.: Enhancing Intelligent Agents with Episodic Memory. PhD Dissertation, University of Michigan (2007)
- [8] O'Reilly, R.C.: Modeling Integration and Dissociation in Brain and Cognitive Development. In: Processes of Change in Brain and Cognitive Development: Attention and Performance, Munakata, Y. & Johnson, M.H. (Eds.), Vol. 21, pp. 375-402 (2006)
- [9] Schvartzman, J.L., Wellman, M.P.: Exploring Large Strategy Spaces in Empirical Game Modeling. In: AAMAS-09 Workshop on Agent-Mediated Electronic Commerce (2009)
- [10] Sherry, D.F., Shacter, D.L.: The Evolution of Multiple Memory Systems. In: Psychological Review, Vol. 94 (4), pp. 439-454 (1987)
- [11] Singh, S., Lewis, R.L., Barto, A.G.: Where Do Rewards Come From?. In: Proceedings of the 31st Annual Conference of the Cognitive Science Society, pp. 2601-2606 (2009)
- [12] Squire, L.R., Knowlton, B., Musen, G.: The Structure and Organization of Memory. In: Annual Review of Psychology, Vol. 44, pp. 453-495 (1993)
- [13] Sutton, R.S., Barto, A.J.: Reinforcement Learning: An Introduction (1998)
- [14] Tulving, E.: Elements of Episodic Memory (1983)
- [15] Tulving, E.: Multiple Memory Systems and Consciousness. In: Human Neurobiology, Vol. 6 (2), pp. 67-80 (1987)
- [16] Tulving, E.: Organization of Memory: Quo Vadis? In: The Cognitive Neurosciences, Gazzaniga, M.S. (Ed), pp. 839-853 (1995)

Forgetting and Generalisation in Memory Modelling for Robot Companions: a Data Mining Approach

Patricia A. Vargas¹, Alex A. Freitas², MeiYui Lim¹, Sibylle Enz³, Wan Ching Ho⁴ and Ruth Aylett¹

Abstract. The idea of a robot as a long-term companion has not been widely accepted and sometimes not even considered or imagined. Hence, several issues should be addressed in order to facilitate this long-term human-robot interaction. This work focuses on the memory modelling for artificial companions and its forgetting processes by proposing a novel approach based on hierarchical classification data mining techniques to implement memory generalisation mechanisms.

1 INTRODUCTION

Modelling a human-like memory has always fascinated AI researchers and has led to a variety of memory models contributing to the understanding of human cognition (Franklin, 1997). In the earlier years, modelling the major characteristics of these memories allowed intelligent programs to remember situations as cases and, most importantly, to extract reasoning rules from these cases stored in a database (see Kolodner (1993) for an overview).

Overall, the organisation of a human memory has a hierarchical structure in the sense that memories of single events are nested within larger cognitive structures. Moreover, most of the events particulars themselves are also organised in a hierarchical fashion (Neisser, 1986; Wright and Nunn, 2000). This intrinsic hierarchical structure seems to permit the storage of an enormous amount of information. In spite of that, if we were to record every bit of incoming information, we would have information overload, difficulty in organizing the information and in focusing on one piece of information at a time. Therefore, forgetting is essential and useful.

Recently there has been an increasing interest in modelling memory for artificial companions, either robots or graphical characters, which could potentially establish long-term relationship with the user (Companions, 2007; LIREC, 2008). The plethora of information that should be “processed” by the robot tends to be enormous each day. Therefore, the generalisation mechanisms involved in such processing is seen as crucial and should be addressed while modelling the memory of long-term robot companions.

Our objective in this work is twofold. First, to propose a means of tackling this generalisation process via a conceptual framework based on data mining techniques for hierarchically structured data, in particular, techniques from the area of Hierarchical Classification (HC) (Sun and Lim, 2001; Freitas and Carvalho, 2007). Second, to *prepare* the robot’s memory data (Witten and Frank, 2005) in order to exploit its inherent hierarchical structure. It should be noted that the data preparation is fundamental before applying any HC technique.

To the best of our knowledge, our proposed data mining approach has a novel aspect, as it has not yet been applied to the

memory modelling of robots as long-term companions. In general, HC has not yet been applied to any robotics-related problem in the literature. Another contribution of this paper is the proposal of high-level attributes of the memory’s data suitable for the generalisation based on HC techniques. These high-level attributes represent a relevant abstraction of the vast amount of low-level data captured about the robot’s daily activities and their identification requires expert knowledge about the robot’s activities.

Section 2 describes how to prepare the memory data for the memory model together with some relevant psychological background on forgetting and generalisation processes. Section 3 presents a brief account on HC methods. Section 4 describes the proposed conceptual data mining framework for the hierarchical memory and provides an example of the application of our conceptual framework to illustrate our proposal. Section 5 draws some conclusions and suggests future work.¹

2 TOWARDS A HIERARCHICAL DATA STRUCTURE OF THE ROBOT’S MEMORY

The human memory works on the basis of three different processes. First, information from the external and internal sensory system of the organism is encoded. Second, the information is stored either in the short-term memory (STM) or long-term memory (LTM). Finally, the information can be remembered or retrieved (if it has not been forgotten). Note that it is as yet unclear and controversial among scientists, how exactly human memory works (Wilson, 1994; De Zeeuw, 2005; Levenson, 2006).

According to one of the existing models of the human memory (Tulving, 1972), LTM can be divided into declarative and procedural memory. The declarative memory is formed of a semantic and an episodic memory. Autobiographic memory (AM), which is the focus of our research, is a specific kind of episodic memory that contains relevant and meaningful personal experiences for a human being (Conway, 1990; Nelson, 1993).

In our recent work (Ho et al., 2009; Lim et al., 2009) we have proposed an autobiographic memory model for a long-term

¹ Dept. of Computing Science, Heriot-Watt University, Edinburgh, Scotland, UK. Email: {p.a.vargas,myl,ruth}@hw.ac.uk

² Computer Science Department, University of Kent, Canterbury, UK. Email: a.a.freitas@kent.ac.uk

³ Otto-Friedrich Universität Bamberg, Germany. Email: sibylle.enz@uni-bamberg.de

⁴ Dept. of Computing Science, University of Hertfordshire, England, UK. Email: w.c.ho@herts.ac.uk

companion whose information stored in the long-term episodic memory component of the memory architecture was modelled as a collection of events. Nonetheless, there was no hierarchical structure reflected in the data stored.

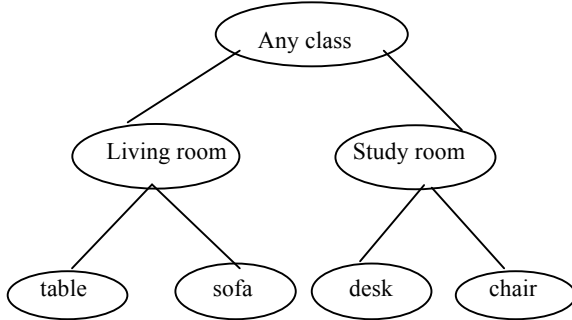


Fig. 1. Class hierarchy example for the memory’s data.

Therefore, one of our aims in the present work is to analyse the information to be stored in the robot’s memory and compose a novel hierarchical structure. Our enhanced memory model will enable the companion to remember events that are relevant or significant to itself or to the user. For other events with a lower long-term value, the memory model will support forgetting through the processes of generalisation and memory restructuring. The long-term memory module in particular, contains episodic events that are chronologically sequenced and derived from an agent’s interaction history both with the environment and the user. Meanwhile LTM also produces concepts as knowledge about the world in order to help in formulating and processing new goals.

We propose a novel structure taking into consideration its inherently hierarchical nature (Fig. 1). Basically, the autobiographic LTM will be composed of events/episodes (or data instances) and the information related to each event will be stored and organised as attributes. It is beyond the scope of this paper to describe how the data will be collected and stored following the proposed structure. Our focus here is on the proposed generalisation mechanisms to be implemented.

Table 1 enumerates the key attributes that should be part of the robot’s memory every time it records an event. Each event should be decomposed into the *subject* “who” or *person* of the *action*, the *object* “whom” or *person* directly involved in the *action* and the *action* “what” itself. The *time* “when” (has a hierarchical structure) and place or *location* “where” the *action* took place (has also a hierarchical structure) should also be part of the event particulars. Finally, the robot should store its “*internal state*” related to the event, e.g. its emotional status, and the subject’s internal state, e.g. her/his emotional status, named here “*external state*”.

Table 1. Attributes of each event to be stored.

ID	NAME	DESCRIPTION	TYPE
1	Who	Subject/Person	Alphabetic
2	Whom	Object/Person	Alphabetic
3	What	Action	Alphabetic
4	Where	Location	Alphabetic
5	When	Time	Numeric
6	Internal State	Robot’s emotional status	Alphabetic
7	External State	Whom’s emotional status	Alphabetic

3 HIERARCHICAL CLASSIFICATION (HC)

Classification is one of the most important problems in data mining (Witten and Frank, 2005). Basically, it can be defined as (Freitas and Carvalho, 2007): given a set of training instances (examples) composed of pairs $\{x_i, y_i\}$, find a function $f(x)$ that maps each attribute vector x_i to its associated class y_i , $i = 1, 2, \dots, n$, where n is the total number of training instances.

After training, the predictive accuracy of the classification function induced is evaluated by using it to classify a set of unlabeled (unknown-class) instances, unseen during training. This evaluation measures the generalisation ability (predictive accuracy) of the classification function induced.

In Hierarchical Classification (HC) problems (Sun and Lim, 2001; Freitas and Carvalho, 2007; Vens et al., 2008), the classes are disposed in a hierarchical structure, such as a tree or a direct acyclic graph. In these structures, the nodes represent classes, and the edges typically represent a “is-a” relationship between classes. If an instance is assigned to a given class c that instance should also be assigned to all of the ancestral classes of c .

We emphasise that in general what defines a HC task is the fact that the class attribute has hierarchically-structured values, regardless of whether the other attributes (the predictor attributes) have hierarchically-structured values or not. Our problem is even more complex than a conventional hierarchical problem for in our case not only the class attribute has a hierarchical structure but also some predictor attributes have hierarchically-structured values.

There are two major types of HC approaches, namely the “top-down” approach and the “big-bang” approach (Witten and Frank, 2005). In essence, in the top-down approach a set of classification models is built from the training set and their predictions are combined in a top-down fashion. The classifier predicts, for the current instance, one of classifier’s child classes at level 1. The instance is further passed down to the classifier whose child class was just predicted, and so on until the instance cannot be further classified (e.g. when a leaf class node is reached). An example of this top-down HC process will be discussed, in the context of the proposed memory’s data, in the next section.

In the big-bang approach, a single classification model is induced which is in principle capable of predicting any class, at any level of the hierarchy, for a test instance. HC algorithms following the big-bang approach in general are more difficult to design and implement than algorithms following the top-down approach, since in the latter each of the classifiers can be induced in a modular fashion, solving a simpler problem, therefore implementing a kind of “divide and conquer” approach.

In this work we consider initially the top-down approach for HC, given the complexity of the target problem and the lack of previous work in the area, it seems sensible to start with a relative simpler approach before trying more complicated ones.

4 THE “DREAMING PROCESS”

Memory forgetting is useful to improve efficiency, scalability and adaptability of cognitive systems operating in dynamic environments, such as a robot’s interaction environments. Forgetting could be viewed as a way of controlling the memory of the robot companion for it could be used to *regulate* the type

Table 2. Attributes of each event to be stored (a small subset of the entire dataset).

Event	Who	Whom	What	Where	When	Internal State	External State
1	Amy	Apple	Eat	LivingRoom.Table	2009.06.11.Morning	Happy	Happy
2	Claire	Game	Play	Living Room.Sofa	2009.06.11.Evening	Happy	Excited
3	Amy	Game	Play	Living Room.Sofa	2009.06.12.Evening	Happy	Excited
4	John	Book	Read	StudyRoom.Desk	2009.06.13.Afternoon	Default	Sleepy
5	John	Andy	Chat	StudyRoom.Chair	2009.06.13.Evening	Happy	Happy
6	John	TV	Watch	StudyRoom.Desk	2009.06.14.Evening	Concerned	Bored

and amount of data stored on the robots memory (Gold,1992), giving rise to a more reliable companion, in terms of data security and privacy (Vargas et al., 2009; Lim et al., 2009).

Neuroscientists and psychologists have developed a number of theories of forgetting including generalisation processes. Generalisation describes processes of abstraction in the human memory. Concrete episodes are continuously restructured losing their concrete features and being reduced to their core meaning, forming abstract categories that include a variety of episodes with different concrete features. This process serves the limited cognitive capacity of the human brain and allows for quick and easy categorization and representation of a complex, ever-changing environment. Generalisation serves the successful use of memory in daily life as it allows for abstraction from concrete and singular experiences to develop a model of the extremely complex and ever changing environment. Hence, here we propose a mechanism named “dreaming process” to implement a generalisation process in the robot companion’s long-term memory.

In our proposed long-term memory model the attributes (including the class attribute) of the events or episodes (data instances) are going to be recorded and stored using a hierarchical data structure, hence, particularly suitable for HC techniques (Section 3). Initially, we plan to use the dreaming process to “clean” the autobiographic long-term memory by means of generalisation. Nevertheless, it important to emphasize that this process will also be responsible for generating a knowledge base representation that will enable the companion to learn and make “predictions” if required.

Basically, the information that receives frequent attention will go through reconstruction processes before it is consolidated as LTM. By being able to notice and recall differences in experiences, the robot companion will be able to learn about its environment more effectively. General structures will help the companion in deciding what to pay attention to, and “reminding” forces it to make use of prior knowledge to form expectations. Nonetheless, care needs to be taken when generalising information to ensure that particular differences that may be valuable are not lost. This anticipated “dreaming process” is crucial for the maintenance of a meaningful autobiographic long-term memory in the sense that only relevant events will be kept stored in the memory database and upcoming information will be stored according to its relevance and pertinence to previously stored data.

4.1 Hierarchical Classification in Memory Modelling

Our proposed conceptual framework for HC is based on the top-down HC approach (Section 3). The proposed HC method will run during the “dreaming process” of the robot, which could take place every night, or at any predetermined time slot. In order to

define our target HC problem, one must first define the instances, the predictor attributes and the class attribute.

Table 2 depicts an example list of events or data instances (a small subset of the entire dataset) that are being stored by a robot. In this paper we are rather interested in the HC problem that arises when the attribute chosen as the class attribute is a hierarchically-structured one. Suppose that the attribute “where” of Table 2 is chosen as the class attribute. Assuming the robot is moving just inside a house, the first data instance shows that the event occurred in the general location *living room*, and more specifically in the location *table*. In this example scenario, the class hierarchy is show in Figure 1. In order to implement the proposed top-down HC approach, we would need to train a classifier for the root node that will discriminate between classes *living room* and *study room*, a classifier for the *living room* node will discriminate between subclasses *table* and *sofa*, and a classifier for the *study room* node will discriminate between subclasses *desk* and *chair*. In the testing phase, each test instance is first assigned to the root classifier.

We will show how the generic method can be implemented by using a simple Naïve Bayes classifier (Witten and Frank, 2005) as the base classifier. Naïve Bayes is a well-known classification algorithm. We are using it here for its conceptual simplicity.

We show here a subset of the probability calculations associated with the training of Naïve Bayes at the root node, a conceptually similar set of probability calculations would be performed for the Naïve Bayes trained at the class nodes *living room* and *study room*. Let “Where” be the class attribute to be predicted and let X_i , $i = 1, \dots, 6$, denote the predictor attributes Who, Whom, What, When, Internal State, External State, respectively. Naïve Bayes has to compute, for each class, the value of its posterior probability $P(\text{class}|\text{attributes}) = P(\text{attributes}|\text{class}) \times P(\text{class}) = \prod_i P(X_i|\text{class}) \times P(\text{class})$ (1) where $P(\text{class})$ is the class’s prior probability, and then it assigns to an instance to be classified the class with the largest posterior probability value.

Using the very small dataset in Table 2, the prior class probabilities for the classifier at the root node would be: $P(\text{living room}) = 3/6 = 1/2$, $P(\text{study room}) = 3/6 = 1/2$. The conditional probabilities of the form $P(X_i|\text{class})$, for example, would be computed as $P(\text{Who} = \text{Amy}|\text{living room}) = 2/3$, $P(\text{Who} = \text{Claire}|\text{living room}) = 1/3$, $P(\text{Who} = \text{John}|\text{living room}) = 0/3 = 0$, $P(\text{Who} = \text{Amy}|\text{study room}) = 0/3 = 0$, $P(\text{Who} = \text{Claire}|\text{study room}) = 0/3 = 0$, $P(\text{Who} = \text{John}|\text{study room}) = 3/3 = 1$. The conditional probabilities of the form $P(X_i|\text{class})$ for $i = 2, \dots, 6$ can be computed in an analogous way.

Suppose now that a given test instance is submitted to the Naïve Bayes classifier trained for the root node. After considering the values computed by Equation 1, for each class,

as a function of the attribute values observed in the test instance, the classifier would assign the class *living room* or *study room* to the instance. Note that, if the test instance has the value *John* for the attribute *Who*, this fact by itself would suggest (ignoring other attributes for now) that the most likely location for an event involving *John* (at the first level of the class hierarchy) would be the *study room*, since $P(Who = John | study\ room) = 1$ and $P(Who = John | living\ room) = 0$ and the two classes have the same prior probability (1/2). Of course, the prediction of the location of an event involving John will also consider all the other predictor attributes too (calculations not shown due to lack of space). So, if the user wants to predict the location of an event involving John, the user will have to provide all the contextual information for the event, such as whom John would be interacting with, what John would be doing, etc. A similar prediction process would be performed for the most specific level of the location class hierarchy, as per the above-described top-down approach.

5 DISCUSSION AND FUTURE WORK

In this work we propose a novel approach towards the goal of devising an enhanced memory model for a long-term robot companion including the use of HC data mining techniques to implement generalisation mechanisms. We believe that generalisation is useful to improve efficiency, scalability and adaptability of cognitive systems operating in dynamic environments, such as a robot's interaction environments. We plan to address all forgetting mechanisms in our robot's memory model (Vargas et al., 2009; Ho et al., 2009, Lim et al., 2009) and we see this present work as a step forward towards this endeavour.

We first propose to structure the memory data hierarchically for it is envisaged that this hierarchical data structure will facilitate the implementation of the aforementioned forgetting and generalisation mechanisms. We have also provided a simple example of how the proposed conceptual framework could be applied to the memory generalisation process using a HC method.

HC concepts and techniques are particular suitable to our memory model for the attributes (including class attributes) of the memory's events or episodes (represented as data instances) will be prepared to be recorded and stored using a hierarchical data structure. To the best of our knowledge, computational models of AM have neither so far adequately accounted for its hierarchical organisation and nested structure nor had fully exploited these characteristics (Ho et al., 2008).

We are currently working in the implementation of the proposed conceptual framework based on HC, initially using a Naïve Bayes algorithm as the base classifier, given its simplicity, computational efficiency and relatively good predictive performance, and we will also investigate more sophisticated kinds of base classifiers in the future.

Acknowledgments. This work was partially supported by European Community (EC) and is currently funded by the EU FP7 ICT- 215554 project LIREC (LIVING with Robots and Interactive Companions). The authors are solely responsible for the content of this publication. It does not represent the opinion of the EC, and the EC is not responsible for any use that might be made of data appearing therein.

REFERENCES

- Companions (2007). <http://www.companions-project.org/>
- Conway, M. A. (1990), *Autobiographical Memory: An Introduction*. Buckingham: Open Univ. Press.
- De Zeeuw, C. I. (2005). "Time and tide in cerebellar memory formation," *Current Opinion on Neurobiology*, vol. 15, pp. 667–674.
- Franklin, S. (1997). "Autonomous agents as embodied ai," *Cybernetics and Systems Special Issue on Epist. Aspects of Embodied AI*, vol. 28, no. 6, pp. 499–520.
- Freitas, A. A. and de Carvalho, A. (2007). A Tutorial on Hierarchical Classification with Applications in Bioinformatics. In: D. Taniar (Ed.) *Research and Trends in Data Mining Technologies and Applications*, pp. 175-208. Idea Group.
- Freud, S. (1937). *The Ego and the Mechanisms of Defence*. London: Hogarth Press and Institute of Psycho-Analysis, 1937.
- Gold, P. E. (1992), A proposed neurobiological basis for regulating memory storage for significant events. *Affect and Accuracy in Recall: Studies of 'Flashbulb' Memories*, 141-161.
- Ho, W. C., Dautenhahn, K. and Nehaniv, C. L. (2008) "Computational memory architectures for autobiographic agents interacting in a complex virtual environment: a working model", *Connection Science*, 20:1, 21 - 65.
- Ho, W. C., Lim, M., Vargas, P. A., Enz, S., Dautenhahn, K. and Aylett, R. (2009), "An Initial Memory Model for Virtual and Robot Companions Supporting Migration and Long-term Interaction", *RO-MAN'2009*, Japan.
- Kolodner, J. L. (1993) *Case-Based Reasoning*. San Mateo, CA: Morgan Kaufmann.
- Levenson, J. M. (2006). Epigenetic mechanisms: a common theme in vertebrate and invertebrate memory formation. *Cellular and Molecular Life Sciences*, 63:1009-1016.
- Lim, M., Ho, W. C., Vargas, P. A., Enz, S. and Aylett, R., (2009), "A Socially-Aware Memory for Companion Agents", 9th International Conference on Intelligent Virtual Agents (IVA'2009), Amsterdam.
- LIREC (2008). The LIREC Project, <http://www.lirec.org/>
- Neisser, U. (1986) Nested structure in autobiographical memory, in D. C. Rubin (ed.), *Autobiographical Memory*, Cambridge University Press, UK, pp. 71–88.
- Nelson, K. (1993). "The psychological and social origins of autobiographical memory," *Psych. Science*, vol. 4, pp. 7–14, 1993.
- Sun, A. and Lim, E. P. (2001). Hierarchical text classification and evaluation. In *Proceedings of the 1st IEEE International Conference on Data Mining*. IEEE Computer Society Press (pp. 521-528).
- Tulving, E. (1972). "Episodic and semantic memory," in *Organization of Memory*, E. Tulving and W. Donaldson, Eds. New York: Academic Press, pp. 89–101.
- Vargas, P. A., Ho, W. C., Lim, M., Enz, S. and Aylett, R., (2009), "To Forget or Not to Forget: Towards a Roboethical Memory Control", *Proceedings of the AISB-2009*, Edinburgh, UK, pp. 18-23.
- Vens, C., Struyf, J., Schietgat, L., Dzeroski, S. and Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning* 73(2), pp.185-214.
- Wilson, M. (1994). Reactivation of hippocampal ensemble memories during sleep. *Science*, 265.
- Witten, I. H. and Frank, E. (2005). *Data Mining, practical machine learning and techniques*. 2nd ed., Elsevier, Morgan Kaufmann.
- Wright, D. B., & Nunn, J. A. (2000). "Similarities within event clusters in autobiographical memory". *Applied Cognitive Psychology*, 14, 479-489.

Human-like Memory Systems for Interactive Robots: Desiderata and Two Case Studies utilizing Grounded Situation Models and Online Social Networking

Nikolaos Mavridis and Michalis Petychakis, Interactive Robots and Media Lab, UAEU

Abstract. Although rudimentary memory systems for autonomous agents have existed for quite a while, human-like memory systems which contain extensive episodic, social and affective components, and which can be utilized in order to maintain relationships with humans, have only recently started to appear. Interactive robots are a special case of an autonomous agent, which is physically situated, interacts with humans, and which has sensing and motor abilities. Furthermore, social robots might have regular encounters and interactions with other human or non-human entities, and might maintain relationships with them. In this paper, a set of desiderata for memory systems of companions will be presented, together with case studies of the memory subsystems of two existing robotic systems: first, Ripley the Robot, an interactive manipulator robot arm with vision, dialogue, and a sensorymotor-grounded situation model; and second, Sarah the FaceBot, a social mobile robot with face recognition, dialogue, as well as a social and interaction database which is bi-directionally connected to the FaceBook website. The case studies provide the basis for an interesting discussion involving physical situations, self-models, expectations, and most importantly episodic and social memories, human- or machine- authored, internal or external, private or shared.

1 INTRODUCTION

As autonomous agents mature and their use in the role of artificial companions becomes more widespread, it can be expected that the *importance of human-like memory systems* for such companions will increase. Such artificial companions could be embodied in physical or virtual bodies, and could also be migrating among many different embodiments [1][2]. In order for them to be able to establish and sustain meaningful relationships with humans, fluid and natural communication is one of the prerequisites: and thus *a sharable framework for representing situations and memories as well as communicating about them in natural language* is required. Furthermore, one could even conjecture that a stronger prerequisite might not only include the sharing of the framework for memories, but also of the actual contents. For example, in the FaceBots project [3][4], the central experimental hypothesis is that if a human and a robot have and talk about their *shared episodic memories* (experienced together or communicated), as well as about their *shared friends* and memories with them or about them, then more meaningful and sustainable long-term relationships between human and robot will be created. The more general question thus arises:

Q1: *What features* should a memory system for #artificial# companions have, and *how should it be utilized*, in order to help sustain relationships with humans?

Hard empirical evidence towards answering this two-part question for the case of human-virtual character and human-robot interaction is still at early stages of its development. However, if we temporarily forget about artificial memory systems, and concentrate on work on *human memory*, we enter a subject for which a long history of discussion as well as empirically-supported modelling exists within philosophy [5], as well as the cognitive and the social sciences. In AI, although numerous approaches towards creating memory systems exist, most existing approaches have covered partial aspects only of the numerous intricacies and complexities of human memory. Such approaches ([6-8]), often provide an overtly generalized framework, which however provides adequate specialization for only a small part of the wide range of aspects of human memory. Only recently, have larger-scale attempts towards human-like memory systems started ([9],[10]). Returning back to Q1, and in light of the large corpus of existing work in human memory, one possible answer to the first part of the above question could be:

A1: Let us try to make the *memory systems for artificial companions become human-like too*, so that they can possibly be better “aligned” (i.e. have a shared framework and can be communicated more easily) with those of their human partners.

Thus, the next set of questions follows:

Q2: What are the *main features* of a #human-like# memory system? What are different *ways that they are utilized*, and what might be the *implicit purposes* they might be serving?

In this three-part question we are on much stronger ground as compared to Q1, given the existing history of scholarship. Based on this question, a set of *desiderata* can be formed, towards creating artificial human-like memories, as will be done in the next section. After forming these desiderata, *two specific case studies* of memory systems will be described. First, Ripley: a manipulator arm with vision, language and *situation modelling* abilities. Ripley is equipped with a memory system supporting sensory expectations, and having an episodic memory which includes past states of three models (self, human, environment) and which enables simple mental imagery and awareness of uncertainty. Second, Sarah: a social robot with face recognition, dialogue, as well as a social and interaction database, which is also updated through the *FaceBook website*. Sarah is equipped with an extensive social memory, and a rudimentary on-board episodic memory. However, due to her connection to FaceBook, she effectively has access to a large external memory store, which is human-authored, and shared within a circle of friends. These case studies will be presented, and then discussed in light of the proposed desiderata. Then, we will proceed with a unifying discussion, and finish with a forward-looking conclusion.

2 DESIDERATA FOR HUMAN-LIKE MEMORY SYSTEMS FOR COMPANIONS

In this section, an set of desiderata for human-like memory systems for artificial companions will be proposed¹, which will later provide axis for discussing and comparing different existing attempts. A *concise list* will be first provided, followed by a *short description* of each.

- D1) Distinction between Sensory, Short Term, and Long Term
- D2) Distinction between Procedural, Semantic & Episodic
- D3) Support for awareness of vagueness and uncertainty
- D4) Support for mental imagery and perspective flipping
- D5) Support for expectation generation and testing
- D6) Encoding of physical and mental attributes of self and others
- D7) Support for social aspects of memory
- D8) Support for affective aspects of memory
- D9) Encoding of active and non-active goals
- D10) Human-like encoding, retrieval, consolidation & forgetting
- D11) Grounding of memory to external reality
- D12) Support for explicit as well as implicit memory phenomena
- D13) Distillation of episodic to declarative memory
- D14) Rich involvement of memory in behaviour and action
- D15) Sensory and contextual cueing
- D16) Support for interplay of internal with external memory

In more detail, before we utilize the D1-16 in our case studies:

D1) Distinction between Sensory, Short Term, and Long Term
We would like to have at *least three separate levels* of storage – essentially, some implementation inspired by the Atkinson-Shiffrin model [11] and its variants. It is worth noting here that sensory memory is usually *sensory modality-specific*, and might take place at *different distances* from the sensor itself in the processing chain, before fusion.

D2) Distinction between Procedural, Semantic & Episodic
Procedural memory accounts for “remembering how”: for example, sensorymotor skills such as picking up a ball. For a robotically embodied companion, this might for example have the form of learnable motor or sensorymotor routines. *Semantic memory* accounts for “remembering that”: for example, that Athens is the capital of modern Greece. Commonsense knowledge databases, such as the knowledge base part of [12], are an example of contents usually held in semantic memory. Also, semantic memory can arise out of observed regularities in events in episodic memory: for example “after the darkness turns to light, the sun will rise” (more on this in D13). *Episodic memory* usually accounts for “remembering something that was experienced”: the sunset yesterday night, my feelings when I first heard this music. Autobiographical memory could be construed as episodic memory augmented with semantic memory content relevant to the self [13]– and plays a central role in numerous philosophical theories regarding personal identity – see for example the memory criterion [14].

D3) Support for awareness of vagueness and uncertainty
A big proportion of beliefs held at any moment, might not be held with absolute certainty – and various *levels of confidence* weightings might be assigned to them. Also, different *granularities of precision* are inherent in different descriptions; a simple spatial example would be: “my cell phone is in the living

room” vs. “my cell phone is on the (living room’s) table”. These two reasons, among others, suffice for an illustration of the need for supporting and handling representations of vagueness and uncertainty in memory systems.

D4) Support for mental imagery and perspective flipping

Apart from the arguably pointless and ill-defined mental imagery debate in cognitive science [15], there are various phenomena associated with human memory, such as *viewpoint-taking* [16], that are quite useful to companions too.

D5) Support for expectation generation and testing

Expectations are beliefs about the future, which might or might not be fulfilled, i.e. a form of *prospective* (as contrasted to *retrospective*) memory. Expectations might be derived through predictions on the basis of communicated beliefs or past observations, promise speech acts by other agents etc.

D6) Encoding of physical and mental attributes of self and others
When moving further away from modality-specific sensory memory, usually (a partial exception being [17]) we arrive to some form of a *post-sensory-fusion situation model* (term used in the cognitive science sense of [18]). Within such a model, descriptions of both *passive objects* as well as *agentive entities* exist. The desideratum here is that, within the companion’s memory, the descriptions of agents will not only contain *physical* attributes (i.e. would not deteriorate to a passive object model), but will also contain *mental* attributes, such as estimated beliefs, intentions, and affect of agents. These descriptions could even extend from models of the mental content of agents to models of their cognitive processes². Such models could be created for other agents (through processes associated with having a *theory of mind* in the sense of [20]) or for the self (through metacognitive self-reflective processes).

D7) Support for social aspects of memory

Any companion which represents within its memory not only models of other agents, but also *social relationships* (friendship, acquaintance etc.) among others or among others and the self would qualify for partially achieving this desideratum. Essentially, social memory is an extension of other-models in order to account not only for the transient state and permanent characteristics of other agents, but also for their evolving relations to one another, and to the self.

D8) Support for affective aspects of memory

Emotion and memory are strongly connected [21]. Any memory system in which events would be associated with *affective significance* would qualify for this desideratum, and even more strongly if the associated significance influences in appropriate ways the further processing of these events or their involvement in other cognitive processes.

D9) Encoding of active and non-active goals

Any purposive agent which is involved in planning usually internally represents goals. These goals might often be numerous, and might change status over time: inactive, active, accomplished etc. Goals are also strongly intertwined with attentional effects in memory, resulting in increased availability to certain aspects as compared to others.

D10) Human-like encoding, retrieval, consolidation & forgetting
Human memory comes together with a set of fairly standard processes that operate on its contents; for example forgetting, for which models such as [22] have been proposed.

¹ This is an initial proposal for desiderata; no claim of completeness is being made. However, we believe it provides a strong foundation.

² For example, such models of cognitive processes of other agents could correspond to folk-psychology-inspired models [19], or could start by self-models.

D11) Grounding of memory to external reality

A companion's memory system usually does not (and should not) exist in a void. Its contents should be connected to sensory data, which in turn correlate for example to external physical reality. On the other hand, this connection could also be bi-directional, for example through the creation of sensory expectations (D5). Furthermore, this connection to external reality could be sensor-mediated, or second level, i.e. relying on communicated information from other agents – essentially a form of secondary, indirect grounding.

D12) Support for explicit as well as implicit memory phenomena
Apart from the cases where the act of remembering is comes together with conscious awareness of remembering, there is a large number of other phenomena during which a past experience could result to behavioural modification in the future, without conscious awareness of remembering. These could potentially fall under the category “implicit memory” [23]: for example, various forms of conditioning.

D13) Distillation of episodic to declarative memory

As commented upon in (D1), observed regularities in events stored in episodic memory, such as “after the darkness turns to light, the sun rises” can be turned into semantic knowledge, not tied to a single-shot particular time and place, but having a wider degree of generality. This can for example happen through some form of empirical induction, as the philosophical term would be.

D14) Rich involvement of memory in behaviour and action

Due to the relative simplicity of just creating one more field in a class prototype in an object oriented language, often artificial memory systems are rich in posited representations but quite poor in their utilization towards behavioural modifications or action selection. Memory for example is heavily involved in action selection, as well as inference.

D15) Sensory and contextual cueing

Memory recall is often improved considerably give suitable sensory and contextual cues. Thus, the desideratum here would be to have context-sensitive memory systems for companions, which increase availability of certain items depending on context, and which can furthermore exhibit priming on the basis of suitable sensory cues.

D16) Support for interplay of internal with external memory

Extending upon (D15), one can observe various forms of coupling between, on the one hand, internal representations of memory and processes of remembering, and the natural as well as social environment, on the other. The external part does not merely augment the internal, but in Clark's words [24] external media are “best seen as alien but complementary to the brain's style of storage and computation”.

3 RIPLEY THE ROBOT

Ripley [25][26] is a manipulator robot arm equipped with vision and natural language dialogue, and situated in an environment consisting of a table on which objects are placed, and where a human can sit by (figure 1). Ripley's purpose is to act as a conversational helping hand robotic assistant. The software architecture of the robot is based on the Grounded Situation Model (GSM) proposal, and consists of a number of modules intercommunicating with each other through a custom inter-process communication system.

At the center of the GSM architecture, there exists a *standardized representational hierarchy* for modelling

situations, together with a *set of standard processes* towards: updating the representations given incoming sensory or linguistic data, gradually reducing certainty in case of sensory inaccessibility, etc. Through the GSM, i.e. these representations and processes, not only object permanence is achieved, but also the situation-model-mediated bidirectional connection between language and the senses: sensory data can map to verbal descriptions, and verbal descriptions can create *sensory expectations* – that can be fulfilled or not fulfilled depending on future incoming sensory data..

Ripley's *dialogue system*, which is single-initiative (human), is able to service three major kinds of speech acts: *Request_For_Information(SA1)*, *Request_For_MotorAction(SA2)* and *Inform(SA3)*: “there is a small object at the right”)

A typical dialogue follows for illustration (for more extensive examples, look at [26] as well as the online videos [27]):

(Human places a green ball within the robot's field of view)

H1: Ripley, what size is the red one?

R1: No such object exists

H2: Ripley, where is the green one?

R2: At the left

H3: Ripley, look at the right!

(Robot moves, and now object is not visible, so slowly the positional certainty decreases through a diffusion process)

H4: Ripley, where is the green one?

R4: Most probably at the left

H5: Ripley, there is a blue object at the top

(Sensory expectation created; the robot can speak about the object, and resolve references to it, and awaits for sensory verification when the top area of the table becomes visible)

H6: How big is the blue one?

R6: I have no idea

H7: What color are the objects?

R7: One is red, and the other is blue

(Human places a small blue object at the top)

H8: Ripley, look at the blue object!

(Ripley moves and looks at the top area of the table, where he expects to see the blue object. The expectation is fulfilled, as there was a successful match between sensory expectation and sensory input. Thus, the instantiated token for the blue object is updates through the incoming sensory information)

H9: How big is the blue one?

R9: Small

H10: Hand me the green one

(Ripley moves to the left area of the table, where the most probable position for the green object is, grips the object, hands it towards the direction where the human face was last seen)

H11: Where was the blue one when your head started moving?

R11: Most probably at the top



Figure 1. Ripley the Robot, Human Partner, table and objects

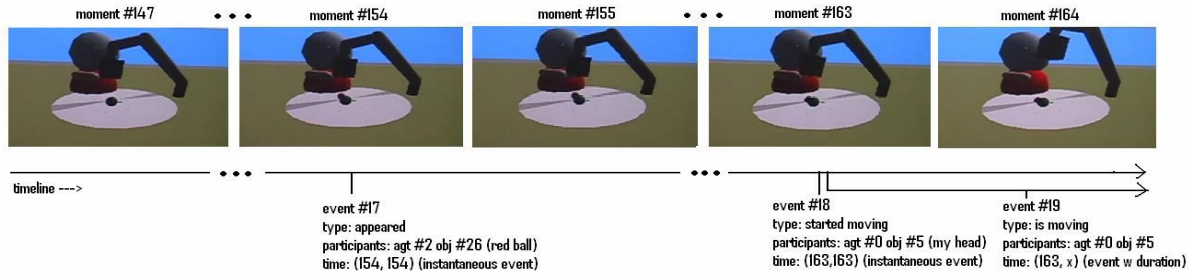


Figure 2. Moments and Events with times, types and participants: the episodic memory of Ripley the Robot

There are a number of points worth making, following this dialogue fragment example: First, notice the mix of all *three kinds of speech acts* mentioned above; H1 is an example of SA1, H10 of SA2, and H5 of SA3. Also, notice H11: an example of a Request_For_Info referring to the past; in which *two kinds of referring expressions* (RE) need to be resolved: Referring to objects (RTO): “the blue one” or Referring to events (RTE): “when your head started moving”. Furthermore, an RTO might either be adjectival / attributive (“the small object at the left”) or using personalpronoun-noun body part names (“your head”). Also, notice the fact that *uncertainty* needs to be taken into account in order to achieve such dialogic capabilities; for at least two kinds of reasons: *Non-continuous availability of sensory data* (when the robot looks away from a region), and *Lack of specificity in verbal descriptions* (“there is an object at the left” gives neither exact position nor size or color).

The required dialogic capabilities, necessitate:

- 1) Representing properties in a way that supports *sensory input/expectation* as well as *verbal input/output*, with flexible/learnable mappings between words and numerical values, and handling *uncertainty*,
- 2) Using Situation Models containing models for *objects*, as well as for the *robot* (the “self/me” for example in order to resolve “your head” RTOs), and the *human* (“you” - for example in order to know where to give an object when asked “hand me the...”)
- 3) Representing *current, past, or imagined* (verbally-instantiated: “there is a ...”) situations

Ripley’s GSM implementation is able to accomplish the above in the following manner (for details, see [25][26]):

- 1) Four *parallel representations* are used for each property, i.e. single-valued discrete, multi-valued discrete, single-valued continuous, and multi-valued continuous, in order to interface both with words (discrete categories) as well as with the senses (continuous measurements), and in order to handle uncertainty.
- 2) A representational hierarchy with the “situation” (moment) type at the top level is used. A situation is composed of a description of agents and agent relations (passive objects seen as potentially agentive), and each agent description is broken to a physical description, and a description of postulated mental content, etc.
- 3) Each current moment (situation) is stored, and the memories of all time instants of the past are also indexed through eight types of *detected primary events*, thus creating Ripley’s episodic memory³ (figure 2).

³ For an interesting information-theoretic approach towards sensorimotor-grounded interaction histories, and an actual implementation, the reader is referred to [28]

Each event consists of a frame with four fields:

UniqueID: #

Type: {create, destroy, appear(get in view), disappear(out of view), start_move, stop_move, start_touch, stop_touch}

Participants: (agt#, obj#), (agt#, obj#)

Time: (start, end)

If we consider both the contents of the current situation with the previous moments as well as events, we get the complete representational contents of Ripley’s GSM. Finally, it is worth noting that there is support for empirical learning of the grounded models of the meaning of adjectives and action / event verbs in Ripley’s architecture, through trainable classifiers [26].

4 SARAH THE FACEBOT

Sarah (figure 3) is a social mobile robot (Activmedia Peoplebot), equipped with a software architecture with modules for face recognition, dialogue, navigation, as well as an onboard social and interaction database, and a real-time connection to the FaceBook website. The purpose of Sarah is to create long-term relationships with human friends, through physical meetings, online meetings, or a mixture of both.



Figure 3. Sarah the FaceBot, FaceBook page & Friends graph

The main experimental hypothesis behind the original creation of FaceBots [3] is that if, during daily conversations between the robot and humans, there is reference to shared memories and to shared friends, then more sustainable as well as meaningful long-term relationships will arise. Sarah, apart from using information from the status and minifeeds of her FaceBook friends while speaking with them, also has her own page on FaceBook, containing her friend list, which also has automatically updated status when she is speaking with somebody. Numerous tangential side-gains were achieved by this interesting combination of interactive robots, face recognition and online social networking; for example, novel algorithms for enhancing face recognition in multi-face pictures given social information, as well as for estimating the friendship graph given massive collections of photos can be found in [4]

[29], and the wider issue of possible roles for artificial agents within human social networks is discussed in [30]. Sarah's software architecture is comprised from multiple modules. The main module we are concerned with here is the social db module – where both social as well as interaction information is located. Also, we will focus on the processes connecting it to natural language dialogue as well as to the FaceBook internet link. A typical dialogue follows: (robot-initiative, in contrast to Ripley).

//Greetings / Recognition (the robot roams around, and stops):

R1: Hello! My name is Sarah. Welcome to the interactive robots and media laboratory. Hi! Are you Panos Toulis?

H1: Yes

(Robot FaceBook status chng: "interacting with Panos Toulis")

//Query of overall state:

R2: Hey, Panos Toulis, are you doing well today?

H2: Yes, pretty well

//General news-based discussion:

R2: Let me tell you some interesting news that I've heard. The architect of the world's tallest building, Adrian Smith, has spoken to the BBC about the 828m high Dubai tower. Have you heard about this? (item taken from BBC News)

H2: Yes

R2b: That's great!

//Robot discusses Facebook status of human:

R4: I see you are chasing robots in the desert

//Robot gives minifeed news about a mutual friend

R5: Did you know that our friend Nikolaos Mavridis has posted a new photo on Facebook?

(information acquired from Facebook minifeed)

H5: No

R5b: I am sending you a message about this to check it out

//Robot remembers past encounter with mutual friend

R6: I saw Nikolaos yesterday evening. He was doing well. (information acquired from interaction database)

//Robot gives status news about a mutual friend

H6: Did you know that our friend Chandan is "waiting for Sarah to have that little sweet conversation with me"?

(information acquired through chandan's Facebook status)

R6: Yes, of course.

//Robot connects an online friend with the human

R7: One of our friends, Nikolaos Mavridis, is online – would you like me to send him a message?

H7: Yes ("Nikolaos, Panos says hello" is sent)

//End of interaction

R8: Hey, Panos, it was nice talking to you! See you later!

The dialogues of the robot follow this overall structure, and are carried out either physically or through facebook chat. As can be seen, they are comprised by a number of basic acts:

Greet, Query State, Tell News, Discuss Human Status, Discuss Minifeed info about Mutual Friend, Refer to old meeting with Mutual Friend, Discuss Status of Mutual Friend, Connect online to Friend, Refer to old meeting with Human, Goodbye

Currently, the order of these acts is fixed, while experimentation about randomization, optimal ordering, repetition etc. is taking place. The `find_mutual_friends()` method which is utilized in a number of the above acts, uses a query to the online social database, which is periodically updated through facebook. The `get_status()` and minifeed queries currently directly access FaceBook, as does `set_status()`, while the discuss old meeting acts utilize the onboard interaction database.

The database contains entries for identities, personal information, friendship relations, and interactions, plus a number of other entries under development. *Interactions* deserve a little more attention. They are effectively comprised of three fields:

F1) Participants: Who participated in the interaction

F2) Time Marks: Beginning and end time of interaction

F3) Description: A single cumulative string containing a verbal description which comprises the gist of the interaction, as contributed by the acts comprising it: "I saw Panos yesterday evening. He was doing well. We discussed about Michalis".

Finally, let us look at the FaceBook page structures, which together with the online status of the human and of mutual friends, are also utilized for a number of the acts taking place. There are many different types of posts in a minifeed: photo & video posts, announcements of friendships, comments to other posts, of liking another post etc. User-adjustable privacy settings in FaceBook restrict global access. Furthermore, each page contains photos, often with tagged faces, whose exploitation is discussed in [29]. Videos of Sarah are available at [30].

5 DISCUSSION

Let us now view the two robots in terms of the desiderata Di. Ripley's memory system contains separate working memory and long-term memory stores (D1). The parametrization of its motor routines, which are executed upon requests for action verbs ("pick up", "hand me"), for which we had demonstrated learning by examples, for a rudimentary procedural memory (D2). No semantic store per se exists; and an all-remembering episodic memory indexed by events is available. Furthermore, apart from simple retrieval (D10) mechanisms (fetch the most recent event matching the verbal description), no forgetting or consolidation processes are supported, and affect (D8) is neither represented within nor effects memory and behaviour. The real strengths of Ripley's system though exist, and are two-fold: first, the offered *bi-directional grounding* to the senses (D11) and second, the *rich capabilities of its working memory system* for: generation and verification of sensory expectations (D5), mental imagery and perspective flipping (D4), representation of agentive models of the self and others (D6 - achieved through "embedded situation models" and virtual allo-centric camera based simulation ToM [26]), and handling of uncertainty (D3) due to lack of specificity or decrease of confidence over time.

Sarah's memory system is quite simple regarding its internal episodic component (D2), and only contains minimal affective (D8) capabilities (i.e. remembering answers to the question: "are you doing well today"). However, it achieves two significant novelties: first, it has quite an *advanced social memory* (D7), *holding friendship relations as well as personal information*, and second, and most importantly, has the direct bidirectional connection to FaceBook, which effectively translates to an interesting complementation of the internal interaction and social databases, with external status-reports for friends and others, with a minifeed digest of their own episodic memories as published on FaceBook etc. Thus, Sarah's memories are effectively extended and blended within the partially-observable external (D16) pool of friendship data and episodic minifeeds of FaceBook. Furthermore, they are utilized effectively (D14) in order to create references to shared memories and friends towards creating long-term relationships. Table 1 provides a cross-comparison of the memory systems of the two robots.

	GSM(Ripley)	FaceBot(Sarah)
D1a) Sensory	~ (visual only)	N
D1b) Short Term	Y	N
D1c) Long Term	Y	Y
D2a) Procedural	~ (motor prog)	N
D2b) Semantic	N	N
D2c) Episodic	Y (moments & events)	Y (interaction memory, onboard&externalFbk)
D3) Vagueness/Uncertain	Y (2 types & processes)	N
D4) Mental Imagery	Y (ego/allo-centric vpt)	N
D5) Expectations	Y (sensory, from sens. detach & verb.descript.)	N
D6) Agentive Models	Y (self/other, beliefs etc)	~(via socialDB & Fbk)
D7) Social Memory	N	Y
D8) Affect	N	~ (storing answer of “are you doing well”)
D9) Active Goals	N	N
D10) Retr., Forg, Cons.	N (most recent item)	N (most recent item)
D11) Grounding	Y, bi-direction	~(Facedet,SpeechRec, Fbk reports)
D12) Implicit Memory	N	N
D13) Episodic to Declarat.	N	N
D14) Rich involvement of memory in Behavior	QuestionAnswering, present or past	Dialogue fed mainly by on-board or exter. Mem
D15) Sensor & Cxt Cues	N	~(socialctx-facerec[28])
D16) Interplay w External	N	Y (Fbk)

Table 1. Cross-comparison of the two memory systems

7 CONCLUSION

The central topic of this paper is the important problem of *designing memory systems suitable for artificial companions*, robotic or virtual, which aspire to create meaningful relationships with humans. We started by posing the question, of *what features* should such memory systems have, and how they should be utilized, towards solving our problem. Given the corpus of available knowledge, we moved towards a slightly modified question, focused on replicating selected features of *human memory* towards artificial systems. Then, we proposed a number of *desiderata* that such memory systems should fulfil, selecting from certain features of human memory, while omitting others. These desiderata, apart from serving as goals, provide a framework for analysis and comparison of existing systems.

Two such systems were presented as *case studies*: First, Ripley, a Grounded-Situation-Model equipped manipulator arm. Second, Sarah, a FaceBook-connected mobile social robot with onboard social and interaction memories. The memory systems of these two companions were later *cross-compared* on the basis of the proposed desiderata, and their strong points as well as shortcomings were discussed. Among the *strong points* of Ripley’s memory was bidirectional grounding, agentive models, and uncertainty handling. Sarah on the other hand, was equipped with a social memory as well as what effectively accounts for a partially private and partially public extended memory, which was collaboratively being created through direct experience or potentially multi-hop verbal reports. Of course, not all of the proposed desiderata were fulfilled by the discussed systems.

Thus, numerous *interesting next steps* lie ahead: cross-comparing more existing systems given the desiderata, which could further be refined; creating new memory systems with greater coverage of the required goals; and quite importantly, empirically testing their real effectiveness towards our main goal of meaningful relationships with artificial companions. And as artificial companions are continuously maturing and unleashing new possibilities, our worthy goal is getting constantly closer.

REFERENCES

- [1] K. Ogawa and T. Ono, “Itaco: Effects to interactions by relationships between humans and artifacts,” in *Proc. of Intelligent Virtual Agents* 2008, pp. 296–307.
- [2] K. L. Koay, D. S. Syrdal, M. L. Walters, and K. Dautenhahn, “A user study on visualization of agent migration between two companion robots,” in *HCI* 2009
- [3] N. Mavridis, C. Datta et al., “Facebots: Social robots utilizing and publishing social information in Facebook”, *IEEE HRI* 2009
- [4] N. Mavridis, W. Kazmi, P. Toulis, C. Ben-AbdelKader, “On the synergies between online social networking, Face Recognition, and Interactive Robotics”, *CaSoN* 2009
- [5] Entry on Memory in *Stanford Encyclopedia of Philosophy* <http://plato.stanford.edu/entries/memory> (last retrieved January 2010)
- [6] A. M. Collins, E. F. Loftus, “A spreading-activation theory of semantic processing”, *Psychological Review*. 1975, 82(6) 407-428
- [7] R. Schank and R. Abelson, *Scripts, Plans, Goals and Understanding*, Hillsdale, NJ: Erlbaum, 1977
- [8] J. Kolodner, *Case-Based Reasoning*. Morgan Kaufmann, 1993
- [9] European Union Project LIREC (Living with Robots and Interactive Companions), <http://www.lirec.org> (last retrieved January 2010)
- [10] W. C. Ho, M. Y. Lim et al., “An initial memory model for virtual and robot companions supporting migration and long-term interaction”, *RO-MAN* 2009
- [11] R. C. Atkinson, R. M. Shiffrin, “Human Memory: A proposed system and its control processes”, *Readings in Human Memory*, 1976, Methuen
- [12] H. Liu, P. Singh, “ConceptNet—a practical commonsense reasoning tool-kit”, *BT Technology Journal*, 2004, Springer
- [13] D. Rubin, R. Schrauf, et al., “Belief and recollection of autobiographical memories”, *Memory & Cognition*, 2003, 31 (6), 887-901
- [14] J. Locke, *An Essay Concerning Human Understanding*, 1690
- [15] S. M. Kosslyn, *Image and Brain: The Resolution of the Imagery Debate*, 1985, MIT Press, Cambridge, MA.
- [16] G. Nigro and U. Neisser, “Point of View in Personal Memories”, *Cognitive Psychology*, 15: 467-482, 1983
- [17] L. W. Barsalou, “Perceptual Symbol Systems”, *Behavioral and Brain Sciences* 22: 577-660, 1999
- [18] R. Zwaan and G. Randvansky, “Situation models in language comprehension and memory”, *Psychol. Bulletin*, 123(2):162–185, 1998
- [19] I. Ravenscroft, “Folk Psychology as a Theory”, *Stanford Encyclopedia of Philosophy*, 2004, <http://plato.stanford.edu/>
- [20] D. Premack and G. Woodruff, “Does the chimpanzee have a theory of mind?”, *Behavioral and Brain Sciences*, 1978 1 (4): 515
- [21] S. Hamann, “Cognitive and neural mechanisms of emotional memory”, *Trends in Cognitive Sciences*, 5, 394-400, 2001
- [22] E. Altmann and W. Gray, “Managing attention by preparing to forget”, *Cognitive Ergonomics*, vol. 1, no. 4, pp. 152-155, 2000
- [23] D. L. Schacter, “Implicit Memory: History and Current Status”, *Journal of Experimental Psychology*, Vol 13(3), Jul 1987, 501-518.
- [24] A. Clarke, *Being there: putting brain, body, and world together again*, 1998, MIT Press, Cambridge, MA
- [25] N. Mavridis and D. Roy, “Grounded situation models for robots: Where words and percepts meet”, in *Proc. of IEEE IROS* 2006
- [26] N. Mavridis, Grounded Situation Models for Situated Conversational Assistants, PhD thesis, Massachusetts Institute of Technology 2007, MIT DSpace, print version at: <http://www.dr-nikolaos-mavridis.com>
- [27] Interactive Robots and Media Laboratory online videos, <http://www.youtube.com/channel/irmluae> (last retrieved Jan 2010)
- [28] Grounded Sensorimotor Interaction Histories in an Information Theoretic Metric Space for Robot Ontogeny, *Adaptive Behaviour*, vol. 15, no. 2, pp. 167-187, 2007
- [29] N. Mavridis, W. Kazmi and P. Toulis, “Friends with Faces: How Social Networks Can Enhance Face Recognition and Vice Versa”, in book *Computational Social Network Analysis*, 2010, Springer Verlag
- [30] N. Mavridis, “Artificial Agents entering Social Networks”, in book *The Networked Self: Identity, Community and Culture on Social Network Sites*, 2010, Routledge, New York