A Robust Implementation of Episodic Memory for a Cognitive Architecture

David Henri Ménager (dhmenager@ku.edu)
Department of Electrical Engineering and Computer Science
1520 W. 15th Street, Lawrence, KS 66045 USA

Dongkyu Choi (dongkyuc@ku.edu)
Department of Aerospace Engineering
1530 W. 15th Street, Lawrence, KS 66045 USA

Abstract

The ability to remember events plays an important role in human life. People can replay past events in their heads and make decisions based on that information. In this paper, we describe a novel extension to a cognitive architecture, ICARUS, that enables it to store, organize, generalize, and retrieve episodic traces that can help the agent in a variety of manners. After discussing previous work on the related topic, we review ICARUS and explain the new extension to the architecture in detail. Then we discuss four architectural implications of the new capability and list some future work before we conclude.

Keywords: episodic memory; cognitive architectures; virtual sensing; expectations; impasse resolution

Introduction

Episodic memory is one of the cornerstones of human cognitive ability (Tulving, 2002). It is responsible for enabling one to remember the events of his or her life. This remembering, however, is not simply a recollection of personal facts. Rather, it is a *relived experience* made possible by three necessary tenents: a subjective sense of time; a sense of self; and autonoetic consciousness (Tulving, 1985, 2002). These allow humans to go back in time in one's head using the subjective sense of time without confusing the events as happening right now. This ability impacts many different aspects of human cognitive capabilities.

Despite the fundamental importance of the episodic memory, however, computational models of episodic memory in the context of cognitive architectures are not discussed very frequently aside from some recent work (Nuxoll & Laird, 2007; Faltersack, Burns, Nuxoll, & Crenshaw, 2011; Bölöni, 2011). These systems have demonstrated how episodic memory aids in problem solving, reinforcement learning, narration, and so on. In our work, we aim to build a system that provides these and other capabilities in a single implementation. We built a psychologically plausible episodic memory module and integrated it within a cognitive architecture, ICARUS (Langley & Choi, 2006). The initial implementation quickly resulted in three new or improved capabilities in our system that we believe are important.

In the sections below, we first describe the background in the literature that serves as basis for our work. After a brief review of ICARUS that follows, we provide a detailed description of how an episodic memory has been implemented in the architecture. We also discuss some architectural implications of the new extension. Then, we conclude after a discussion on future work.

Background

Researchers of memory have held that any biological or computational model of episodic memory must support encoding and retrieval of experience (Tulving, 1983). Encoding is the process of recording and organizing experiences into the episodic memory, and retrieval is the process of using a retrieval cue to find episodes in this memory. These two functionalities have been the subject of much discussion amongst the experts of psychology because it is rather difficult to characterize the processes that govern the interactions between episodic and semantic memories.

Researchers realized that the nuances of memory between implicit and explicit memory and the nature of knowing and remembering are related to the relationship between semantic and episodic memories (Schachter, 1987; Tulving, 1985). We believe that a psychologically plausible model of episodic memory should have an account for these nuances.

As for the details of how the episodic memory works, there is psychological evidence suggesting that it is an index-based long-term memory that supports cue-based retrieval (Hellerstedt, 2015; Tulving, 1983). Researchers also studied the three technicalities of episodic memory to define the notion of index-based memory, episodes, and cues, as summarized below.

Index-based Memory Schiller et al. (2015) argue that episodic memory receives many of its characteristics from the hippocampus, one of which is the ability to create cognitive maps, or a hierarchical network of memories. As memories come and go, the hippocampus is believed to dynamically change the structure of this network in order to preserve the similarity relationship between connected memories. With this structural representation, the index of an episode may be seen as the path from one of the top-level episodes in the network to the episodes of interest.

Episodes Previous research suggests that there exists an episodic buffer that interfaces between the episodic memory and the central executive of the working memory (Baddeley, 2000). This buffer is responsible for accepting diverse sets of data from the agent's sensors and creating a common representation of the data. This representation is what is used to create episodes. Hellerstedt (2015) further states that the creation of these episodes occurs in an on-line fashion.

Cues To retrieve episodic memories, semantic patterns, or cues, are used. These patterns can match against elements in episodic memory even if there is some missing information. Assuming that episodes are organized by similarity, the retrieval process is about finding the episode that contains the most similar situation to the cue. While retrieval is in progress, semantic memory is responsible for generating, specifying, and storing cues (Hellerstedt, 2015). A cue may return more than one episode, in which case the system must use some conflict resolution strategies to determine which episode to present.

Based on our understanding of episodic memory as described above, we implemented our extension to ICARUS. Before we describe the details of our implementation, however, it will be useful to review the architecture briefly to facilitate our discussions.

ICARUS Review

As a cognitive architecture, ICARUS provides a framework for modeling human cognition and programming intelligent agents. The architecture makes commitments to its representation of knowledge and structures, the memories that store these contents, and the processes that work over them. ICARUS shares some of these commitments with other architectures like Soar (Laird, 2012) and ACT-R (Anderson & Lebiere, 1998), but it also has distinct characteristics like the architectural commitment to hierarchical knowledge structures, teleoreactive execution, and goal reasoning capabilities (Choi, 2011). In this section, we provide a brief review of the architecture to facilitate our discussion on the new episodic memory module afterwards.

Representation and Memories

ICARUS distinguishes two main types of knowledge. One is its *concepts* that describe certain aspects of the situation in the environment. They resemble Horn clauses (Horn, 1951), complete with a predicate as the head, perceptual matching conditions, tests against matched variables, and references to any sub-relations. For example, the first two in Table 1 are concept definitions. The first one, `(on ?o1 ?o2)`, describes a *primitive* situation where a block is on top of another block, using only perceptual matching conditions for two blocks and tests against the matched objects and their attributes. The second one, `(clear ?block)`, depicts a *complex* situation where there is nothing on top of a block, using another concept as a sub-relation in addition to perceptual matching conditions for a block.

The other type of knowledge in ICARUS is its *skills* that describe procedures to achieve certain concept instances in the environment. These are essentially hierarchical versions of STRIPS operators (Fikes & Nilsson, 1971) with a named head, perceptual matching conditions, preconditions that need to be true to execute, direct actions to perform in the world or any sub-skills, and the intended effects of the execution. For instance, the last entry in Table 1 shows a *primitive*

Table 1: Two sample ICARUS concepts and a skill for the modified blocks world.

```
((on ?o1 ?o2)
 :elements (?o1 is (block ?o1 ^x ?x1 ^y ?y1 ^len ?len1)
            ?o2 is (block ?o2 ^x ?x2 ^y ?y2 ^len ?len2
                              ^height ?height2))
 :tests ((*overlapping ?x1 ?len1 ?x2 ?len2)
         (= ?y1 (+ ?y2 ?height2))))

((clear ?block)
 :elements (?block is (block ?block))
 :conditions ((not (on ?another ?block))))

((look-right ?robot)
 :elements (?robot is (robot ?robot ^looking ?looking
                                     ^holding ?holding))
 :conditions ((not (eq ?looking 'right)))
 :actions ((*look-right ?robot))
 :effects (?robot is (robot ?robot ^looking right
                                    ^holding ?holding)))
```

skill definition, `(look-right ?robot)`, that describes a procedure to get the robot to look right. The skill has a named head, perceptual matching against a robot and its attributes, the precondition of the robot not already looking right, the action to perform in the world, and the intended effect. *Complex* skills have a similar syntax, except that they include sub-skills instead of direct actions in their body.

To store these knowledge and other structures, ICARUS employs a handful of distinct memories. The concept and skill definitions are stored in conceptual and procedural long-term memories, respectively. The short-term instances of these definitions are stored in a belief memory and a goal / intention memory. The former maintains the current state of the world, while the latter houses the agent's current goals and intentions for execution.

Inference and Execution

The ICARUS architecture operates in cycles (see Figure 1). At the beginning of each cycle, the system receives sensory input from the environment as a list of objects with their attribute–value pairs. Based on this information, the architecture infers all the concept instances that are true in the current state by matching its concept definitions to perceived objects and other concept instances.

Once all the beliefs are inferred, the system finds all the relevant skill definitions for the current goal(s) that are executable in the current belief state. ICARUS then chooses one or more of them and execute them in the world. The architecture will continue its cycles in this manner until all of its goals are achieved or its operations are terminated for any other reasons. With this brief review, we now continue our main discussion on the new episodic module in ICARUS.

Episodic Module in ICARUS

In the context of episodic memory, ICARUS shares some architectural assumptions with Soar (Laird, 2012). More specifically, both the architectures assume that episodic memory is a long-term, cue-based system that maintains cues in the agent's working memory, and the agent deliberately encodes and retrieves episodes. However, there are also significant differences. Each time a Soar agent takes action(s), the architecture records a snapshot of its current state as an episode, which is stored statically in the episodic memory without any generalization (Nuxoll & Laird, 2007). An individual episode in Soar, therefore, does not capture the notion of a relived experience. Rather, it is necessary to compile multiple episodes to describe an event. Furthermore, the Soar agent does not remember events where it did not perform any action and stayed as an observer. In contrast, ICARUS has these unique characteristics in its episodic memory:

- Episodic memory is organized as a compound structure composed of an episodic cache, an episodic generalization tree, and a concept frequency tree.
- Episodic generalization tree is organized by similarity.
- Episodes represent durative experiences of variable length.
- Episodic memory is a dynamic structure that supports generalization among similar episodes.

In this section, we describe the new episodic memory module in ICARUS in detail. We start with the representation of episodes, and then explain the encoding, retrieval, and generalization processes.

Episodic Representation

The episodic generalization tree is the main data structure that organizes and stores episodes. The episodic cache acts as a storage for the agent's unprocessed experience. Since our agents do not run for days or years just yet, we assume that the agent has sufficient memory to store the complete state–action sequence. The concept frequency tree records the number of times ICARUS has seen each concept instance. The episodic cache and the concept frequency tree provide a mechanism for recognizing and explaining significant events.

Episodic Processes

On every cycle, ICARUS records the current state and executed actions into the episodic cache and updates the concept frequency tree, as indicated by the arrows going into the episodic memory in Figure 1. When the agent perceives or infers one or more *significant* events, it begins to encode a new episode. The architecture tries to explain each significant event by analyzing information stored in the episodic cache and finding a logical process that causes the event to happen. If the explanation attempt is successful, the state–action sequence that explains the significant event will be stored as the new episode. Otherwise, the significant event(s) and the time when the event(s) occurred are stored as the new episode. This is to make it possible for the agent to return to the particular episode and try to explain again after it accumulates more knowledge about the world.

The episodic memory supports episodic generalization, and the resulting hierarchy is stored in the episodic generalization tree. The root (top-level) node of this tree is the most general episode and is allowed to have an arbitrary number of children. Each child is a k-ary tree where $k \in \mathbf{N}$. Episodes become more specific at each decreasing level of the tree toward the leaf nodes, and there are fully instantiated episodes at the bottom of the tree. This structural organization reflects our understanding of Schiller et al. (2015), and the notion of episodes representing durative events is consistent with all the literature on episodic memory we are aware of. Next, we explain the processes for encoding, retrieving, and generalizing episodes in ICARUS.

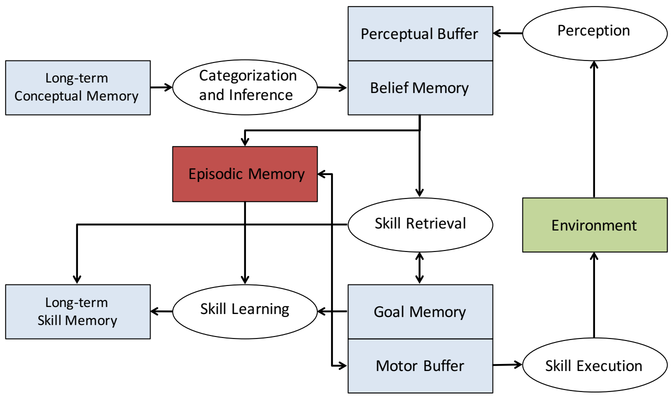

Figure 1: A block diagram that illustrates ICARUS's processes including the new episodic module marked in red. Arrows represent the direction of information flow.

Encoding Encoding is the process that consumes a raw state–action sequence stored in the episodic cache and inserts a fully specified episode into the episodic generalization tree. This process is triggered by the noticing of one or more significant events. Currently, ICARUS considers the following four cases as significant:

1. A rarely seen concept predicate
2. A rarely seen partial set of bindings for a concept predicate
3. A rarely seen full set of bindings for a concept predicate
4. The absence of a concept instance that the agent expects

Once a new episode has been created in this manner, the architecture inserts it to the episodic generalization tree using a slightly modified level order search. In level order search, a node's children are added to the search queue after having visited that node. Rather than always doing this, a similarity test is done first to ensure that the episode already in the tree unifies to the episode to insert if and only if the two episodes are structurally similar. Figure 2 shows this procedure graphically. The numbers in curly braces represent the order in which the insertion happens. After the similarity test is passed, then and only then are the similar episode's children added to the search queue. If the episode to insert cannot be unified with any of the episodes on a given level, then that episode becomes a sibling of those episodes. A new episode has successfully been encoded into the episodic memory. If the episode to insert is a copy of one of the leaf episodes, then the counter for the number of times that particular episode has been observed goes up by one and the episode to insert is not inserted. All episodes are guaranteed to be instantiated by the general episode at the root node.

Retrieval ICARUS supports cue-based retrieval of the episodic memory. Given a cue, ICARUS performs the level order search described in the encoding process except that, instead of the similarity test, the system checks to see if the episode contains a matching fully specified copy of the cue or if the episode contains a generalized version of the cue that unifies with the cue. If so, the episode that matches with the cue is returned. Figure 3 shows the retrieval process. More than one episode can match the cue, so retrieving from a cue returns a forest of trees. One major advantage of this is mechanism, is that the agent only has to search the episodes directly under the root to discover if a situation is completely new. The reason is that, if the cue matches, the cue will either match directly, or at least one of the root's children will contain some generalized version of the cue and the generalization match test will pass. If all match tests fail then there does not exist a matching episode in the tree. This is known, in the worst case, at level two of the tree.

Generalization The ability to generalize knowledge is a key cognitive ability. One of the ways ICARUS supports this is by performing generalization in the episodic tree at encoding. The state–action trace in the episode represents a demonstration of how a significant event came to be. Therefore it is possible to learn from that trace. For example, if person *x* drops a glass on the ground and it breaks, and person *y* drops a glass on the ground and it breaks as well, ICARUS will generalize that knowledge to say that if anyone drops a glass on the ground, it will break (assuming $x \neq y$). This may not be true in general, but the number of times ICARUS makes such generalizations forms the conditional probability, or the confidence with which ICARUS believes a glass breaks when someone drops it. The ability to gain knowledge in this way seems quite central to general intelligence. Sibling generalization is implemented by iterating through the newly inserted episode's siblings. ICARUS makes a generalized episode, and for each sibling it checks to see if it can unify the sibling and the new entry by variablizing the bindings where conflicts are found. If a consistent variablization has been made, ICARUS tests to see if the generalized episode is still more specific than the parent of the newly inserted episode. If so, then the the generalized episode's parent becomes the freshly inserted episode's parent and the generalized episode's sub-episodes becomes the freshly inserted episode and the sibling episode that generalized with it. The count for the generalized episode is the summation of the count of its children.

Architectural Implications

The modified blocks world we use in this work is a partially observable world with two tables. When the agent is looking at one table it cannot see the contents of the other table. Both tables contain three boxes, a red, a blue, and a green, and one box on each table has a block inside of it. Though the contents of the tables are the same, the tables themselves are labelled differently and the positioning and names of the boxes are different. Specifically, one table is labeled *Rainbow* and the other is named *Cloud*, and the boxes are named `Box1` through `Box6`. This is done so that the agent knows that there are two distinct environments in this domain.

We aim to demonstrate many interesting applications from our work, but for the moment we use the modified blocks world domain to supply examples for learning from observation, impasse resolution and making expectations.

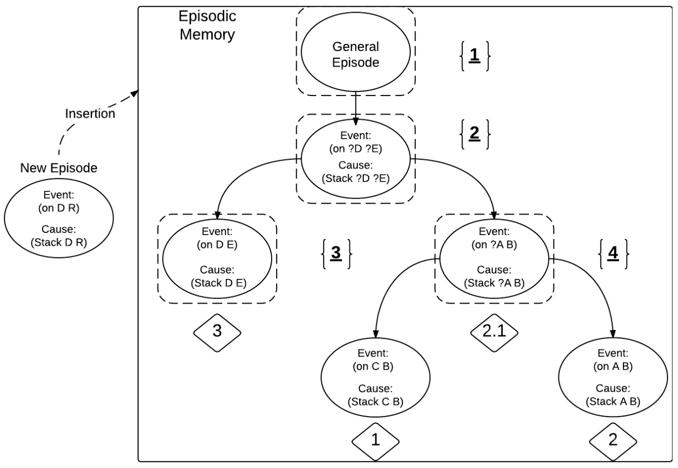
Episodic Memory
General Episode
{1}
{2}
{3}
{4}
Insertion
New Episode
Event: (on D R) Cause: (Stack D R)
Event: (on ?D ?E) Cause: (Stack ?D ?E)
Event: (on D E) Cause: (Stack D E)
Event: (on ?A B) Cause: (Stack ?A B)
Event: (on C B) Cause: (Stack C B)
Event: (on A B) Cause: (Stack A B)
3
2.1
1
2

Figure 2: The insertion into the episodic generalization tree. The numbers in the diamonds represent the order in which the existing episodes were experienced.

Learning from Observations

Despite episodic memory being the memory for events, constantly reflecting on passed experiences may cause the agent to disproportionally slow down as a result of the amount of resources required to search the episodic memory for an episode that answers or responds to a query. When an agent enters a new environment it may not know how to characterize it, and as a result may be unsure of what to do and relies on its episodic memory. As it collects more experiences it may come to knowledge of specific patterns about the environment and formally characterize them in terms of rules.

Since episodes in the episodic memory have a count associated with them, we can capture this ability in ICARUS. If the agent experiences an episode a sufficient number of times ICARUS will try to formalize it into a rule. In our domain, if the agent experiences 10 times that an extra box appears when it stacks two boxes on top of each other, it would have a sufficient amount of evidence to logically relate a stacking two boxes with producing a new box. This mechanism also ties into remembering and knowing because even if ICARUS forgets the stacking experience it would still maintain the rule. Of course there is the possibility that the agent overfits rules

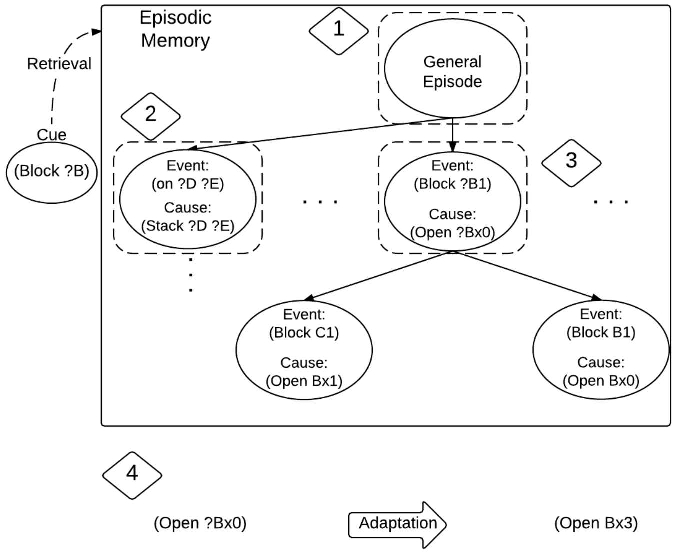
Episodic Memory
Retrieval
Cue
(Block ?B)
1
General Episode
2
Event: (on ?D ?E) Cause: (Stack ?D ?E)
3
Event: (Block ?B1) Cause: (Open ?Bx0)
Event: (Block C1) Cause: (Open Bx1)
Event: (Block B1) Cause: (Open Bx0)
4
(Open ?Bx0)
Adaptation
(Open Bx3)

Figure 3: Retrieval and impasse resolution in ICARUS.

so a mechanism for modifying rules needs to exist. Also, since ICARUS keeps a complete state–action sequence it can always review old experiences to learn something new.

Impasse Resolution

ICARUS encounters an impasse when it does not know how to reach its goal in the current state. One way to overcome an impasse is to problem solve through means-ends analysis. Note, however, that a plan generated by this method is dependent on the agent's conceptual knowledge of how the world works. So, if the agent does not know how to logically relate the goal state to skills or concepts the agent fails. In such cases it would be advantageous to recall a similar experience to the current one and repeat the actions that produced the goal in the previous experience. Figure 3 shows the order in which ICARUS matches against the cue. Note that it is possible to match against generalized episodes. In stage 4, the environment's current bindings are applied to the retrieved solution. Another interesting aspect of impasse resolution is that it is one of the means for which ICARUS gathers support to formalize experiences into a rules as discussed above.

ICARUS spends some time at the Rainbow table exploring the world. During this time it tries a number of different actions and eventually it opens all the boxes and realizes that one of them has a block inside of it. The agent goes to the Cloud table and sees a new set of boxes on the table. The task is to find the block inside the box. Even though ICARUS knows that opening a box might reveal an item contained inside it, there is no logical reason why the box has to contain a block so the problem solver cannot reason properly about what to do in this case. When ICARUS uses `(block ?b1)` as a cue to the episodic memory, it remembers that it found a block at Rainbow by, say, opening `Box0`. This solution is adapted to the current situation and the agent decides to open all the boxes because `Box0` is of type `box`.

Expectations

Our system uses the concept frequency tree to create expectations of what beliefs should be true relative to a given environment. As an agent collects more experiences it collects information about how often it sees certain beliefs over the total number of times it has been in a certain environment. Over time these conditional probabilities give the agent an idea of what to expect when it enters an environment.

In our example ICARUS spends several cycles at the Rainbow table. As it is exploring and acting in the environment, the beliefs change. At a later time when the robot thinks about being at the Rainbow table it may realize things like "Box Box0 is always on the table" and "There was a block in the box for about half the time I was there". In subsequent interactions, unless proven otherwise, ICARUS would assume that `Box0` is on the Rainbow table. This *virtual sensing* is a process for discovering hidden or unseen facts about a given environment. Incorporating this ability into ICARUS is important because often times, agents act in partially observable worlds. So being able to recall information that pieces together a more complete view of the world may allow an agent to operate in a more natural and efficient manner.

Remembering and Knowing

Many psychologists and students of memory have discussed the nature of *remembering* and *knowing*. That is, once someone recognizes an item, is that item recognized because the person had a recollective experience of the item, or did he or she somehow know about the item without having any recollective experience? This nuance has encouraged researchers to characterize knowing and remembering responses and the protocols that govern the interaction of the two.

It seems that remembering is based on episodic memory while knowing responses are based on semantic memory (Gardiner, 1988; Rajaram, 1993; Knowlton & Squire, 1995). Specifically, in a remembering situation, responses are heavily influenced by the conditions present at the time of the encoded episode and by the amount of resources available to spend on remembering. For example, the more distracted a person is while performing a task, the less likely the person is able to recollect on what happened. On the other hand, knowing responses are automatic and influenced neither by the conditions at encoding nor the amount of resources available (Gardiner, 1988; Jacoby, 1991). While remembering involves searching through the episodic memory, knowing simply involves the state of the semantic memory, thus giving rise to the automatic property of know responses.

In our work, we take this research into account and aim to provide a computational theory that is consistent with these results. For instance, remembering responses in ICARUS utilize the episodic memory retrieval mechanism and are thus susceptible to encoding conditions as suggested in the literature. Knowing in ICARUS is facilitated by the semantic memory and does not involve extensive search, thus giving rise to seemingly automatic performance.

Discussions

We added an episodic memory to our theory of cognition in order to better capture the full range of human cognitive abilities for intelligent computational agents. One of the novel features of this episodic implementation is the ability to generalize knowledge at encoding. This serves a two-fold purpose. It provides an ordering to the episodes that lends itself to efficient search and secondly it reduces the demand during retrieval time to adapt a previous solution to the current situation because solutions can be arbitrarily specific. Since ICARUS learns from observations at the level of episodes, the ability to generalize knowledge at encoding also implies that the agent will be able to learn generalized models of how the world works.

We plan to build on this work in a number of different directions. We mention here three of them which, we believe, are most relevant. The strength of our system largely depends on ICARUS's ability to explain significant events. Towards that end, we would like to tightly integrate an explanation mechanism for the creation of episodes and to augment learning from observations within the context of the episodic memory. We will also expand the notion of expectations in ICARUS to use both the concept frequency tree and the primitive skills. This will enable the architecture to learn from different types of surprises. Another interesting direction is to implement a grammar for self-cueing. We expect that this will facilitate the agent's use of the episodic memory for a number of tasks like virtual sensing and other applications of case-based reasoning.

Conclusions

In this work, we detailed a computational model of episodic memory within the context of a cognitive architecture, ICARUS. We founded our approach on psychological evidence concerning the nature of episodic memory, the mechanisms of remembering and knowing, and the distinct features of implicit and explicit memory. We believe episodic memory is a fundamental component of human cognitive ability, and the extended architecture serves as an important basis for future research. We showed that the extension provides ICARUS with at least three new or improved cognitive functions. We plan to continue our research in this promising direction and hope to report in the near future the results of our evaluations using both qualitative and quantitative measures.

References

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Erlbaum.

Baddeley, A. (2000). The episodic buffer: a new component of working memory? *Trends in Cognitive Sciences*, *4*(11), 417–423.

Bölöni, L. (2011). An investigation into the utility of episodic memory for cognitive architectures. In *AAAI fall symposium: Advances in cognitive systems.*

Choi, D. (2011). Reactive goal management in a cognitive architecture. *Cognitive Systems Research*, *12*, 293–308.

Faltersack, Z., Burns, B., Nuxoll, A., & Crenshaw, T. L. (2011). Ziggurat: Steps toward a general episodic memory. In *AAAI fall symposium: Advances in cognitive systems.*

Fikes, R., & Nilsson, N. (1971). STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*, 189–208.

Gardiner, J. M. (1988). Functional aspects of recollective experience. *Memory & Cognition*, *16*(4), 309–313.

Hellerstedt, R. (2015). *From cue to recall: The temporal dynamics of long-term memory retrieval*. Unpublished doctoral dissertation, Lund University.

Horn, A. (1951). On sentences which are true of direct unions of algebras. *Journal of Symbolic Logic*, *16*(1), 14–21.

Jacoby, L. L. (1991). A process dissociation framework: Separating automatic from intentional uses of memory. *Journal of Memory and Language*, *30*(5), 513–541.

Knowlton, B. J., & Squire, L. R. (1995). Remembering and knowing: two different expressions of declarative memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *21*(3), 699.

Laird, J. E. (2012). *The soar cognitive architecture*. Cambridge, MA: MIT Press.

Langley, P., & Choi, D. (2006). A unified cognitive architecture for physical agents. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence.*

Nuxoll, A. M., & Laird, J. E. (2007). Extending cognitive architecture with episodic memory. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence* (pp. 1560–1565).

Rajaram, S. (1993). Remembering and knowing: Two means of access to the personal past. *Memory & Cognition*, *21*(1), 89–102.

Schachter, D. L. (1987). Implicit memory: History and current status. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *13*(3), 501–518.

Schiller, D., Eichenbaum, H., Buffalo, E. A., Davachi, L., Foster, D. J., Leutgeb, S., & Ranganath, C. (2015). Memory and space: towards an understanding of the cognitive map. *The Journal of Neuroscience*, *35*(41), 13904–13911.

Tulving, E. (1983). *Elements of episodic memory.* Oxford University Press.

Tulving, E. (1985). Memory and consciousness. *Canadian Psychology/Psychologie Canadienne*, *26*(1), 1.

Tulving, E. (2002). Episodic memory: From mind to brain. *Annual Review of Psychology*, *53*(1), 1–25.

Wallace, S. A., Dickinson, E., & Nuxoll, A. (2013). Hashing for lightweight episodic recall. In *AAAI spring symposium series.*