Minds and Machines
https://doi.org/10.1007/s11023-021-09578-3

GENERAL ARTICLE

A Hybrid Theory of Event Memory

David H. Ménager[1] · Dongkyu Choi[2] · Sarah K. Robins[3]

Received: 20 January 2021 / Accepted: 1 October 2021

Abstract
Amongst philosophers, there is ongoing debate about what successful event remembering requires. Causal theorists argue that it requires a causal connection to the past event. Simulation theorists argue, in contrast, that successful remembering requires only production by a reliable memory system. Both views must contend with the fact that people can remember past events they have experienced with varying degrees of accuracy. The debate between them thus concerns not only the account of successful remembering, but how each account explains the various forms of memory error as well. Advancing the debate therefore must include exploration of the cognitive architecture implicated by each view and whether that architecture is capable of producing the range of event representations seen in human remembering. Our paper begins by exploring these architectures, framing casual theories as best suited to the storage of event instances and simulation theories as best suited to store schemas. While each approach has its advantages, neither can account for the full range of our event remembering abilities. We then propose a novel hybrid theory that combines both instance and schematic elements in the event memory. In addition, we provide an implementation of our theory in the context of a cognitive architecture. We also discuss an agent we developed using this system and its ability to remember events in the blocks world domain.

Keywords Event memory · Episodic memory · Hybrid theory of memory · Cognitive architecture

✉ David H. Ménager
dhmenager@gmail.com

[1] Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS 66045, USA

[2] Department of Social and Cognitive Computing, Institute of High Performance Computing, Agency for Science, Technology and Research, Singapore 138632, Singapore

[3] Department of Philosophy, University of Kansas, Lawrence, KS 66045, USA

Published online: 02 November 2021

1 Introduction

Humans have the ability to recall individual past events (Tulving, 1983; Rubin & Umanath, 2015). This appears to include both the successful retrieval of past events that one has experienced and reconstructions of such events from schematic information, the latter of which are often but not always accurate. There are two prominent philosophical approaches to event memory—causal theories (Martin& Deutscher, 1966) and simulation theories (Michaelian, 2016b). Causal theories focus, as the name suggests, on the importance of a causal connection to the remembered event, while simulation theories focus on the use of a reliable memory system to construct event representations. While each approach attempts to capture the entire range of event memory within its framework, neither is successful. We suggest that this difficulty stems from the reliance on a single representational format as well as the absence of detailed implementations of these theories. In this paper, we propose a hybrid event memory that unifies the theoretical postulates of each theory and their event representations. We further argue that this novel hybrid theory allows us to explain the full range of event memory abilities. Importantly, we introduce an implemented system that embodies our theoretical proposal and demonstrate its event memory capabilities in a simulated environment. We describe this implementation in detail and discuss the initial results from our experiments.

In the next sections, we briefly review the two main philosophical theories of human event memory in the literature and introduce our novel hybrid theory. Then we describe our implementation of this theory in detail and discuss some experiments carried out in blocks world to evaluate the system's ability to remember events. Finally, we conclude after a discussion of related work.

2 Existing Theories of Event Memory

Theorizing about memory is a focus of research throughout cognitive sciences. Our work here focuses on philosophical accounts of remembering, which aim to unify the concept of memory and empirical evidence about remembering. This focus allows us to bracket, at least for the time being, numerous empirical accounts of event memory. Accounts of event memory in psychology and neuroscience focus primarily on characterizing the activity of our memory system(s), explaining how they work. Researchers in these areas have played a critical role in identifying and demonstrating the phenomenon of false memory—where event memories fail to be accurate. Their focus is on documenting these false memories, the conditions under which they occur and how participants experience and incorporate these states into the rest of their thoughts and actions. There is little attention given to accounts of what successful remembering requires or theoretical distinctions between kinds of false memory.

Philosophical accounts of remembering, in contrast, aim to articulate what remembering requires, and then situate various memory activities as meeting or

failing those requirements. Theoretical approaches to event memory in the philosophical literature are often sorted into causal and post-causal theories (Michaelian & Robins, 2018), where simulationism is the most prominent and detailed post-causal account. This labeling of the two approaches reflect the fact that the causal theory is the better-established, traditional view. Simulationism and other post-causal accounts have emerged more recently, as part of an effort to address inadequacies in the causal theory.

These theories offer possible frameworks for organizing the myriad forms event remembering well-documented by memory science. Causal theories characterize success in terms of the maintenance of a causal connection to the past event. Simulation theorists instead emphasize the production by a memory system that is reliable. Proponents of each view have done their best to articulate the key feature of their respective account—what the appropriate causal connection involves, how the reliability of memory systems should be understood. Moving the debate forward requires exploration of how each view manages to explain the full range of human event remembering—not only success, but the various forms of memory error. This requires exploring how successful remembering is generated, and which parts of that process break down in the production of different forms of error. In other words, it requires understanding the implementation of each theory. But these philosophical theories do not tend to include an implementation. They do not often describe how the memory system is structured or specify the representational format of what is stored. These views do, however, have implications for implementation. That is, the theoretical commitments they involve place constraints on the underlying cognitive architecture of the memory system and representational format of the stored memories. Our interest is in exploring these theories to understand their architectural commitments—the kinds of cognitive structures and representations they are committed to—and whether and how they could be improved upon to better account for the range of event memory phenomena. Below we offer a brief review of each theory.

2.1 Causal Theory

The causal theory of memory is built up from the intuitive idea that remembering is a causal process. The ability to recall something now is due to what was previously learned and stored. For event memory, the cause of our recall must be a representation of that previous event, which derives from the event itself. This is the basic idea Martin and Deutscher (1966) laid out in their paper introducing the causal theory. They argued that remembering requires an accurate representation of the past event and that the person who is remembering be the person who experienced the previous event. What distinguished their account—and made it distinctly causal—was the argument for an additional requirement: that there be a causal link between the past event and the subsequent remembering. Much of their paper was then focused on spelling out exactly what kind of causal link was needed. After all, there are many ways that the two events could be causally connected that would not be instances of remembering. They emphasize that the right kind of causal link for event memory will be one that is sustained by a representation of that event. The representation

of the past event serves as the memory trace for that event with its content being roughly equivalent to what occurred during the event.

What then constitutes the right representation for the required causal link? There are debates amongst causal theorists with regards to the content of these representations (e.g., Bernecker (2010) proposes more substantive content, while Werning (2020) advocates for minimal content). As Robins (2016) has argued, the causal theory is committed to retaining a connection to particular past events. This is the only way to fulfill the causal theory's requirement on the unique causal history of the remembered event. Compilation into a schema would sever that connection, and with it, the possibility of remembering for the causal theorist. The view thus requires the storage of individual episodes, although it allows that those episodes could be either localized or distributed. In this way, causal theorists are committed to the existence of memory traces as representations of the past event being remembered.

In short, whatever representation that is used in the causal view must be able to store and preserve those unique features of an event. This still allows for a causal theory to accommodate reconstructed recollections, but this is handled in the details of how the representations—the memory traces—are structured. As Debus (2018) explains in a recent defense of the view: "it is necessary that a relevant event of information acquisition has left some 'trace', a trace which has been preserved and is causally relevant for the occurrence of the mental state or event which should count as a memory" (p. 68).

Causal theorists have worked to update their understanding of the stored representation and retrieval process so as to accommodate the host of empirical evidence (Wells, 1982; Schacter & Addis, 2007) that event memory is highly reconstructive—often recombining elements in ways that can alter or distort memory contents. For example, Debus (2007) claims that event memory representations can be systematically modified at the time of encoding while explaining perspective switches in observer memories, and Hintzman (1986) even argues for a theory that reconstructs events on the fly by retrieving a number of experiences from memory that match a retrieval cue and averaging them together. As a result, most causal theorists now accept that neither stored episodes nor produced recollections need to faithfully retain all of the information from the initial event; loss of information is possible and consistent with the retained ability to remember (Bernecker, 2008). Most causal theorists, however, continue to resist adding content to the episode. Some researchers also propose changing the representational format of the trace—from a localized representation to a distributed one (Bernecker, 2010; Michaelian, 2011).

These modifications help the causal theorists to explain some instances of reconstructive remembering. However, they fail to address many of the central forms of reconstruction. Reconstructive remembering, for example, often involves the incorporation of information from other events (Suddendorf et al. 2009). These additions often influence the accuracy of a memory, and they violate the causal theorists' constraint on episodic representations containing only information less than or equal to the information in the original event. In addition, the causal theorists cannot account for the possibility that reconstruction could lead to an accurate representation of the past event without any connection to the episodic memory trace. For these reasons, many theorists have shifted their efforts from revising the causal theory to looking

for alternatives. We discuss the most prominent post-causal alternative, simulation theory, in the next section.

2.2 Simulation Theory

Simulation theory (Michaelian, 2016b; Michaelian & Robins, 2018) is a theory of remembering, built around an endorsement of the evidence about reconstructive remembering, and a rejection of the causal condition on remembering and the causal theory more broadly. Simulation theorists view the act of remembering as a constructive process of simulation, where one builds event representations from a wide network of available information about past events. To illustrate the nature of this simulation, Michaelian (2016b) emphasizes that remembering should be seen as a form of imagination.

Like the causal theorists, simulation theorists continue to defend the idea that remembering requires accuracy. In order for one's memory of receiving a yellow bike for their eighth birthday to count as an instance of event memory, it must be the case that he or she received such a bike for that birthday. What is not necessary, however, is an episodic representation of that birthday stored in memory since the occurrence of the event. Instead, one's overall knowledge of past events can be used to reconstruct a plausible representation of the past event. Without reliance on an episodic representation of the event itself, Michaelian argues that "the only factor that distinguishes remembering an episode from merely imagining it is that the relevant representation is produced by a properly functioning episodic construction system" (Michaelian, 2016b, p. 97).

Moving away from episodic representations of particulars allows the simulation theory to capture a range of event memory phenomena that were left unexplained by the causal theory. Simulation theorists can explain how instances of remembering include more information than was present in the original event and also cover instances where the information derives from a range of distinct sources. Cases like one's memory of the yellow bike at the birthday party, however, remain problematic. In such cases, Michaelian insists that remembering is possible even without the episodic representation (Michaelian, 2016a, p. 118). But how is this possible? It seems highly unlikely that one could construct a memory that just happens to be accurate about details of a particular past event without storing information from that event itself. Generalized information about birthday parties could help a remembering subject to build this memory, but such information would presumably provide features of the event that are common to many birthday parties—things like cake, gifts, and balloons. While it is not unheard of to receive a yellow bike, it is far from expected of a birthday party.

Even a properly functioning reconstructive process must rely on the input memory representations to generate a recollection. This implies that the yellow bike should come as an input to the reconstructive system, embedded in an event representation of some sort, if it is to be included in the recollection output. Of course, it is possible, by chance, to have an available event schema that includes a yellow bike sourced from another context, but there is no particular reason why this chance

inclusion should always happen for the yellow bike when remembering the particular birthday party. Unless a representation of this past event in memory explicitly records the existence of the yellow bike, we cannot expect to remember the yellow bike reliably when recalling the event of the eighth birthday party from the reconstructive process. In other words, a properly functioning reconstructive process alone is not sufficient for remembering particulars of a past event. What is also necessary is a representation that is a reliable model for the target event as an input to this process, which, in our view, has to be an episode as a record of the specific event.

While the causal and simulation theories do not directly mention implementational details, they do have implementational consequences. More specifically, these views have implications for the representational features of the underlying system. We hope that our coverage of causal and simulation theories helps clarify the implicit representational constraints of each view. The causal theory requires the memory system to be able to store and preserve information about specific events. The simulation theory, in contrast, denies that such features are required. Instead, this view implies that schemas alone should be able to sufficient for remembering using a reconstructive process. We believe that, by requiring an explicit implementation of these theories, we can frame the debate and suggest ways to move forward. We also observe that both of these theories face challenges from the features of human event remembering, and therefore a new framework may be necessary.

3 Hybrid Theory of Event Memory

As discussed above, both the causal and the simulation theories are unable to capture the full range of human event memory performance, which includes both the ability to remember particular past events and the construction of possible past events on the basis of schematic information. It is critical to develop a theory that can accommodate all these aspects of human memory. We suspect that the limitations faced by causal and simulation theorists, while different, stem from the same basic problem—reliance on a single representational form to explain all of event memory. In response, we propose a new theory that retains the virtues of both the simulation and the causal theories. The core argument of this hybrid theory is that the human memory for events stores both episodic and schematic representations. More specifically, our theory posits that:

Event Memory is a Long-Term Memory that Stores Both Episodes and Schemas. This commitment to both representational forms illustrates the hybrid nature of the theory, incorporating the elements of causal and simulationist approaches. The event memory in our framework is a long-term memory that stores records of events experienced by the remembering agent. The contents of this event memory are stable, but incremental changes occur over time to schematic representations as new events are encountered.

Episodes are Propositional Representations of Specific Events. The contents of which are the agent's internal and external state descriptions, including the agent's perceptions, beliefs, goals, and intentions. Further, episodic representations are causally dependent on past events in the sense that a specific past event was operative in producing the episodic representation. In this way, they reflect the causal theory aspect of our hybrid theory.

Schemas Are First-Order Propositional Templates with Probabilistic Annotations. They summarize episodes by embedding probability distributions associated with the summarized episodic content. In so doing, schemas employ probability to represent a range of possible events. They can also summarize similar event schemas in the same manner. In this way, they reflect the simulationist aspect of our hybrid theory.

Event Memory Elements are Organized in Hierarchies. At the lowest level, event memory records episodes. Over this, it stores schemas that summarize the elements of the episodes at the lower level in a probabilistic manner. Event memory elements are connected by ISA links from a child node to its parent. This indicates that the child is a specialization of its parent. Hence, the event memory elements gradually become more specific at progressively lower levels in the hierarchy.

Retrieval Cues Play a Central Role in Remembering. Our theory characterizes remembering as a response to a retrieval cue. Cues are often a subset of the agent's current observations of the world. When presented with such a cue, we claim that the goal of the event memory system is to remember an event that is at least consistent with information contained in the cue.

Remembering an Event Involves Performing Probabilistic Inference. Given a retrieval cue, the system searches for a memory element that best matches it. Sometimes the best match will be an episode and other times the best match will be a schema. When the found match is an episode rather than a schema, it is returned as a remembered event as it provides a deterministic description of the past event. In contrast, if the match is a schema, the system performs probabilistic inference to output the most likely instance of the schema conditioned on the retrieval cue provided. This reconstructive process is inherently approximate, allowing for inaccuracies in the representation that is returned.

Our novel theory as described above unifies aspects of causal and simulation theories by employing hybrid event representation. It includes both specific episodes and generalized schemas as event memory elements. The generalization hierarchy they form can afford remembering that is consistent with the full range of human memory for events. For the time being, we assume events that describe states of affairs and ignore the temporal aspects of lived experience. Hence, our theory does not make any specific claims about how to determine where one event ends and another begins, and it stays agnostic to views like event segmentation (Zacks & Swallow, 2007; Kurby & Zacks, 2008). In the next section, we describe our implementation of this new theory, beginning with the hybrid representation and continuing on to the processes that work over the event memory elements.

4 Hybrid Event Memory System

The core commitments described above dictate the computational implementation of our theory. We believe that the hierarchical organization of event memory elements, which includes specific episodes at the bottommost level and partially generalized schemas at higher levels, results in an elegant combination of causal and simulation theoretic aspects of memory storage and retrieval. In this section, we describe our implemented system in detail and discuss its implications, starting with the representation and continuing to the processes that work over it.

4.1 Event Representation and Generalization Structure

We begin our discussion of representation with some definitions that will set the foundation for the event memory structures we will discuss. In our framework, the world is composed of a potentially infinite set of *objects*, and this set can be partitioned using object classes. Such classes impose representational constraints over its members, allowing objects in a class to be described with the same set of attributes. Hence, object classes are templates for generating grounded representations, or instances, of objects. We represent objects with lists that include the object type, a unique name, and attribute-value pairs.

These objects and their attributes satisfy certain hierarchical *relations* in the world, which we define as predicates. Relations may also be partitioned into a set of relational classes. For example, we can consider an `on` relation defined with a block stacked on top of another block and a second `on` relation defined with a triangle stacked on top of a block as members of a larger class, $\texttt{on}_{\texttt{generic}}$. We represent instantiated relations, or beliefs, as lists that include the relational class and a subset of component object identifiers as arguments.

Then an *episode* in our framework is a set of grounded objects as well as the corresponding beliefs derived from them. Since beliefs are inferred from hierarchically defined relations like `on` and `next-to` based on perceived objects like `blocks` and their attributes, episodes form a *dependency graph* composed of objects, their attributes, and beliefs. Figure 1 shows a sample state in blocks world and the corresponding dependency graph. The system represents a perceived object like `block1` as a tree of height 1 with the object class as the root. There are directed edges from this root to the nodes that correspond to the attributes of the object like `width`, `height`, `x`, and `y`. Each attribute node stores its perceived value in it. Based on such objects, we define relations like `on` by adding a node to the dependency graph. The outgoing edges from this node link to the component objects and their attributes over which the relation is defined. Similarly, for higher-level relations, the outgoing edges link to the relevant lower-level relations as well as component objects and their attributes. In addition, we represent relational classes as parent nodes of relations.

In an episode, the dependency graph describes the event that actually occurred, by storing a set of value assignments for relation, object, and attribute nodes. But when multiple episodes are aggregated into a schema, the resultant dependency

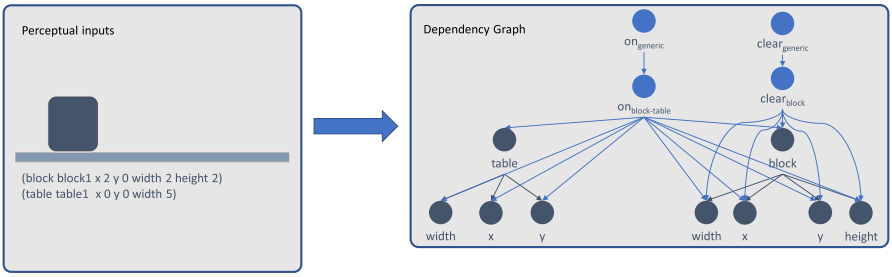


Fig. 1 Representing an episode as a dependency graph

graph stores probability distributions in these nodes that specify the joint probability of correlated variables. More formally, let the nodes in the graph be a set of m random variables, $\{x_1, x_2, ..., x_m\}$, with the set of valid assignments as their domains. If we assume that these variables are organized hierarchically from top to bottom, and impose conditional independence on them, namely, for any three variables, x_i, x_j, and x_k, we assume $p(x_i, x_j|x_k) = p(x_i|x_k)p(x_j|x_k)$, then their joint distribution is:

$$\begin{aligned} p(x_1, x_2, ..., x_m) &= p(x_1|x_2, x_3, ..., x_m)p(x_2|x_3, x_4, ..., x_m)...p(x_m) \\ &= p(x_m)\prod_{t=1}^{m-1} p(x_t|\mathbf{x}_{pa(t)}), \end{aligned} \tag{1}$$

where $\mathbf{x}_{pa(t)}$ are the parent nodes of x_t. This equation specifies a structure known as a *Bayesian network*, where $\mathbf{x}$ is the collection of state variables. The advantage of using Bayesian networks to encode event schemas stems from their systematic reliance on conditional independence assumptions. They are the key to compactly representing complex joint distributions since they reduce the number of parameters necessary to encode the full joint probability distribution. In the case of hierarchical graph structures, it ensures that the probability of a variable taking on a specific value is determined only by the variables that came immediately prior, namely, its parents. These conditional independence assumptions are made manifest by the directed edges between nodes in the network. If the m nodes of a network have $O(F)$ children which can take K possible values, then the number of parameters needed in the model is $O(mK^F)$, which is much less than $O(K^m)$ that would be needed if no conditional independence assumptions were made. Although this is a drastic improvement, $O(mK^F)$ is still exponential, and therefore using large Bayesian networks can be problematic. We address this issue later in Sect. 4.2.2 when we discuss inference in this setting.

In our framework, event schemas as Bayesian networks consist of trees that represent objects and relations. Object trees contain two different types of nodes. The root node r denotes an object class and follows a categorical distribution parameterized by a k-dimensional vector encoding the probability that r instantiates one of its members. Attribute nodes express conditional probabilities of taking on a specific value, given the object, r_k. In other words, every attribute node a_i with domain $\{v_1, v_2, ..., v_K\}$ specifies $p(a_i = v_j|r = r_k)$. This is a useful result, since:

$$p(r_k|a_1, ..., a_F) \propto p(a_1|r_k)p(a_2|r_k)...p(a_F|r_k)p(r_k)$$
$$\propto p(r_k) \prod_{i=1}^{F} p(a_i|r_k).$$

This means that every object the agent perceives is represented as a Naïve Bayes classifier in the dependency graph, which will capture the class-conditional correlations between object attributes and their associated objects. Importantly, the attribute nodes of Naive Bayes classifiers are assumed to be independent from each other. This, however, may not be true in the physical world. Object attributes may, in fact, be correlated with each other, but the influence may not be unidirectional like in Bayesian Networks. For example, two attributes may be correlated with each other via an undirected edge, but we do not capture these potential correlations among the variables in our current representation. This, however, is usually not an issue because Naïve Bayes classifiers tend to perform well in practice.

We represent relations as trees of height 1, where the root node, z, represents the relational class parameterized by a k-dimensional probability vector $\boldsymbol{w}$ following a categorical distribution. Each child node in the tree corresponds to a specific disjunction of the relation, and $\boldsymbol{w}$ describes the probability distribution of these definitions. Hence, the variable z, once inferred, acts as a selector for which definition to generate in the current state. When z is known, an object or attribute, x, given z follows a categorical distribution parameterized by an n-dimensional probability vector $\mathbf{p}_z$, which is the distribution over the values of x where z-th disjunction is considered.

In our implementation, episodes and schemas employ conditional probability distribution (CPD) tables. Table 1 shows a notional CPD tabulating the conditional probability distribution of different state elements. The distribution specifies $p(x|y, z)$ defined over categorical variables x, y, and z. The two left-most columns enumerate the range of the independent variables, while the range of the dependent variable, x, is in the second part of the top row. The joint probability of variable assignments are displayed in the center part of the table. For episodes, these probabilities deterministically describe a single event, but for schemas, the distributions capture a range of possible events.

Finally, our system stores episodes and schemas in a generalization tree. As Fig. 2 shows, the root node is a schema that summarizes all the episodes the agent has experienced. The leaf nodes in this tree are the individual episodes, and there are layers of event schemas on top of these. In this hierarchy, higher-level elements are probabilistic summaries of their children, and an edge from a node to its parent indicates that the child belongs to the kind of its parent. The representational choices we outlined so far allows our system to capture both specific episodes and their generalized schemas. In the next section, we describe various processes that work over this representation to store new events in memory and retrieve past events for remembering.

Table 1 Notional conditional probability distribution

		x	
y	z	A	B
A	D	1/2	1/2
A	E	0	1
B	D	1/3	2/3
B	E	1	0
C	D	1	0
C	E	2/5	3/5

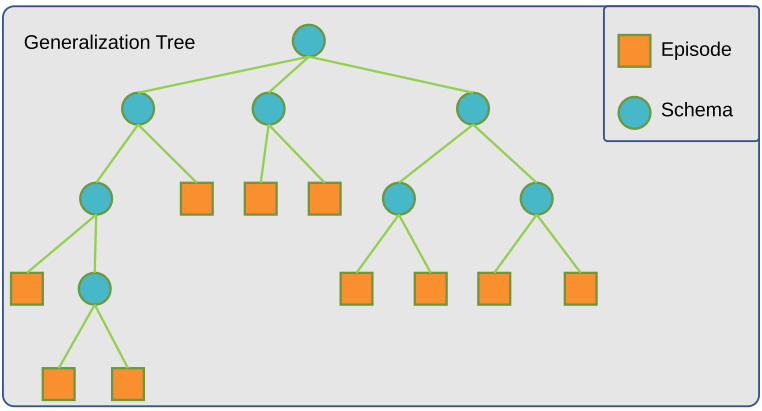


Fig. 2 Notional generalization tree

4.2 Event Memory Processes

Agents with event memory routinely store their experiences and retrieve the records during their operation. In our framework, we define two multi-step processes, episodic insertion and event schematization, and retrieval and remembering, for this purpose. The insertion process introduces new episodes into the event memory. As instances are placed into the event memory, the system updates existing contents and maintains the hierarchical organization of schemas over episodes through schematization. When the system needs to remember an episode, it invokes its retrieval process to produce a remembered event. In this section, we discuss details of these processes that work over event memory contents.

4.2.1 Episodic Insertion and Event Schematization

Insertion occurs in an online fashion, incorporating episodes as they are encountered. When the agent experiences a new event, the system attempts to find the most similar element in the generalization hierarchy. It begins from the root node and works recursively down toward the leaf nodes, as it updates probability distributions

Table 2 Procedure for inserting an episode into the event memory

```
1:  procedure INSERT-EPISODE(eltm, ep)
2:      (eltm, cost-of-merging-p) ← EP-MERGE(eltm, ep)
3:      if HAS-BRANCHES(eltm) then
4:          mappings ← ∅
5:          costs ← ∅
6:          best-child-cost ← ∞
7:          for each child branch in eltm do
8:              for each factor, P, in ep
9:                      and each factor, Q, in branch do
10:                 ⟨sol, cost⟩ ← STRUCTURE-MAP(P, Q)
11:                 costs ← APPEND(cost, costs)
12:                 mappings ← APPEND(sol, mappings)
13:             if SUM(costs) < best-child-cost then
14:                 best-child-cost ← SUM(costs)
15:                 best-child ← branch
16:     if eltm is empty then
17:         eltm ← LIST(ep)
18:     else if ep = ROOT(eltm) then return eltm with increased count
19:     else if cost-of-merging-p < best-child-cost then
20:         eltm ← ADDCHILD(eltm, ep)
21:     else INSERT-EPISODE(best-child, ep)
```

along the way. Once the system identifies the best matching element, it inserts the new episode as a child of this element. By doing this, the system incrementally clusters and classifies the episodes into event schemas that summarize them.

Table 2 shows this process in detail. The system sorts a new episode using a recursive level-order traversal through the hierarchy. Initially, the system matches the new episode with the root node in its event memory. If the root node has any children, the system sequentially checks each child to find the locally best matching node to the current episode (Line 7–15). If the root node is a better match to the episode than its children (Line 19), the new episode is inserted as a child of the root (Line 20). If not, one of the children nodes is a better match than the root, and the level-order traversal continues by recursively running the procedure on the best child (Line 21). In a degenerate case where the new episode and the root node of the tree are identical, the episode is absorbed into the root without continuing the traversal (Line 18).

During insertion, the system attempts to match a new episode against an existing event memory element (Line 10) using structure mapping (Gentner, 1983). This allows the system to determine the similarity of the two structures and measure the quality of match between them. The system attempts to map every node in the new episode's dependency graph to a node in the dependency graph of the memory element. During this process, we impose an additional constraint that requires a matching pair of nodes to be of the same class and to have CPDs that contain matching dependent variables, but the independent variables may vary.

For example, a valid match for a node from a new episode that has the CPD, $p(x|y, z)$, can only match a node from the existing element that has the CPD, $p(x|a, b)$. This not only ensures that the entity types match but also allows different relations to be defined over these entities.

The structure mapping procedure returns the lowest-cost mapping between an episode and an event memory element. In our framework, we define cost to be the probability that the event memory element does not generate the episode, namely, $(1 - score_{BIC}(Q : P))$, where P and Q are the dependency graphs of the new episode and the event memory element, respectively, and $score_{BIC}$ is the Bayesian Information Criterion (BIC) specified as:

$$score_{BIC}(Q : P) = \ell(\hat{\theta}_Q : P) - \frac{logM}{2} Dim[Q] \tag{2}$$

Here, $\ell(\hat{\theta}_Q : P)$ is the likelihood of P under Q, $\hat{\theta}_Q$ are the parameters of the conditional probability distribution in Q, M is the number of episodes summarized by the event memory element, and $Dim[Q]$ is the number of free parameters in Q that are not in P. Simply put, this score computes a trade-off between how predictive the event memory element is and how structurally complex it is. When M is small, $score_{BIC}$ favors more predictive event memory elements, but as M grows, it prefers more parsimonious event memory elements. This is a built-in regularization that avoids overfitting while maximizing likelihood, especially as M grows. This effectively applies Occam's Razor (Jefferys & Berger, 1992) to selecting a good match for the new episode, since event memory elements with high $score_{BIC}$ are the structurally simplest elements with the highest probability of generating the new episodes.

Moreover, in Bayesian networks, the BIC score decomposes to a series of local computations over the CPDs themselves (Koller & Friedman, 2009). This means that we can compute the BIC score of the entire network during the structure mapping process. Using the decomposed BIC score, we can reformulate our cost function as:

$$\sum_{i=1}^{N} height(p_i) \cdot (1 - score_{BIC}(q_i : p_i)), \tag{3}$$

where N is the number of nodes in the episode's dependency graph, p_i is the i-th node in that graph, $height(p_i)$ is the height of the i-th node in the episode's dependency graph, and q_i is the corresponding match to p_i in the event memory element. Considering $height(p_i)$ in the cost function forces the structure mapping procedure to prioritize higher-level matches in the dependency graph and prefer to match more general schemas given all the other conditions are equal. When p_i is matched, the $score_{BIC}$ dominates the cost. In contrast, when p_i does not have any corresponding node in Q, then the height of the unmatched node determines the cost of match. As a result, our system prefers to match states that are qualitatively similar at a higher level. For example, the system will prefer to match an episode that involves two blocks, A and B in an `on` relation to another episode with two blocks, C and D with the same relation, instead of matching to a third episode with two blocks A

and B in a `next-to` relation. Although the first and the third episodes contain the same blocks, their higher-level relations are different. Therefore, the cost of matching these two episodes will be higher than matching the first episode to the second.

Our system interleaves the insertion process we described so far with event schematization, which uses the results of structure mapping to merge two event memory elements into a schema. The system performs normalized factor addition by filtering the nodes from the episodic dependency graph with their associated matches. When the matching element from the event memory is also an episode, the merging operation will yield a new schema. In cases where the existing event memory element is already a schema, the merging operation will update the distributions of that schema. It is important to note that the CPDs from the new episode and their counterparts in the matched element are not necessarily defined over the same domains. For example, the episode might contain a new predicate like `(tower C D)` not previously modeled in the matching schema. Therefore, before the merging operation takes place, the system temporarily renames the variables in the episode CPDs with the names of their corresponding matches. This enables the system to easily identify which variables to update and to what extent. Once the CPD variables from both sides are made directly comparable in this manner, the system proceeds to update the domains and the variable assignments.

Then the system uses a filtering procedure that passes an episode CPD through a schema CPD. Figure 3 shows an example, where the system updates the schema distribution for $p(height|on, block)$ with evidence contained in a new episode. Note that the `block` variables in the two CPDs have different domains. Namely, the episode describes a block, `D`, while the schema describes another block, `B`. In order to merge these two CPDs properly, the system expands the domain of the `block` variable in the schema and puts placeholders for the additional value, `D`, for this variable, while it similarly adds placeholders for `B` in the episode. Then, the merging operation simply performs element-wise addition over these expanded structures to update the distribution for $p(height|on, block)$.

Table 3 shows a pseudocode for this element-wise addition process. The system constructs a new CPD ψ with the expanded domain (Line 2–3). To determine the merged CPD's variable assignments, it takes the sum of the counts of each variable assignment in the schema and episode (Line 6–11). Then, the system calculates the total counts (Line 13) and the probabilities (Line 14) of the variable assignments. This update procedure ensures that the system maintains the correct empirical probability distributions for CPDs because it preserves the count information. In cases when a variable from a dependency graph does not have a corresponding match, it inherits the domains of the parent node variables because those variables may have matched counterparts. Doing this ensures that the variable domains remain consistent throughout the episode or schema.

As outlined above, the system inserts new episodes into the best matching location in its memory through structure mapping and uses its merging process to incrementally schematize the event memory elements. This results in a spectrum ranging from individual episodes at the lowest level to progressively more general schemas at higher levels. Our system retrieves elements from this hierarchical structure and

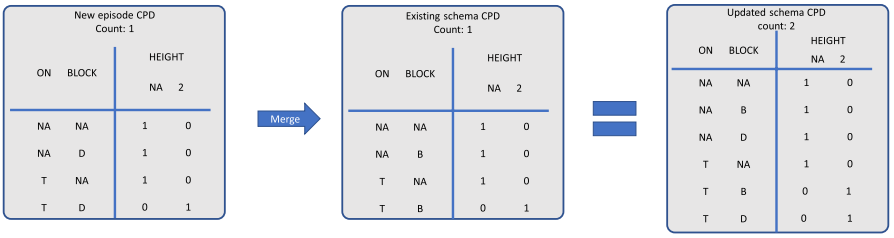


Fig. 3 Updating event schema with merge operation

Table 3 Procedure for filtering one CPD with another comparable CPD

procedure FACTOR-FILTER(ϕ_1, ϕ_2, op)
  $\psi \leftarrow$ MAKE-CPD(ϕ_1, ϕ_2)
  ψ-assignments $\leftarrow$ MAKE-ARRAY(num-assignments)
  j $\leftarrow$ 0
  k $\leftarrow$ 0
  for i from 0 to num-assignments - 1 **do**
    if op = "+" **then**
      ψ-assignments[i] $\leftarrow$ (CPD-ASSIGNMENTS(ϕ1)[j] $*$ CPD-COUNT(ϕ1)) + (CPD-ASSIGNMENTS(ϕ2)[k] $*$ CPD-COUNT(ϕ2))
    else if op = "*" **then**
      ψ-assignments[i] $\leftarrow$ CPD-ASSIGNMENTS(ϕ1)[j] $*$ CPD-ASSIGNMENTS(ϕ2)[k]
    (j, k) $\leftarrow$ ADVANCE-INDICIES(ϕ_1, ϕ_2, j, k)
  if op = "+" **then**
    count $\leftarrow$ CPD-COUNT(ϕ1) + CPD-COUNT(ϕ2)
    normalized $\leftarrow$ {val/count : val in ψ-assignments}
  else if op = "*" **then**
    count $\leftarrow$ CPD-COUNT(ϕ1)
    normalized $\leftarrow$ NORMALIZE(ψ-assignments)
  cpd-assignments(ψ) $\leftarrow$ normalized
  return ψ

produces remembered events in response to retrieval cues. In the next section, we describe these processes in detail.

4.2.2 Retrieval and Remembering

When an agent with event memory operates in the world, it stores experienced episodes and forms the episodic generalization hierarchy as described in the previous section. At any given point, the agent might need to remember a past event, and the system invokes a retrieval process to produce a remembered event. Our event memory system performs retrieval in a level-order fashion using a retrieval cue. Depending on the situation, a retrieval cue may be a fully observed state, or only a partial description of an event. The system tries to find an episode or a schema in its memory that best matches the given retrieval cue. This search involves the same machinery as what we use for episodic insertion, except that the system does not update the probability distributions of the existing event memory elements using the contents of the retrieval cue.

If the search yields an episode as the best match to the retrieval cue, the event memory system returns that episode immediately, since the episode is a deterministic representation of a specific event to be remembered. In contrast, if the system finds an event schema as the best match, it needs to further process the schema to return the most likely instantiation as the remembered event. The process involves performing probabilistic inference over the probability distributions contained in the schema. The inference step discovers the values of the unobserved state elements of the schema using the retrieval cue as a set of observed state elements. We describe this process more formally as in:

$$P(\mathbf{x}_h|\mathbf{x}_v, \boldsymbol{\theta}) = \frac{P(\mathbf{x}_h, \mathbf{x}_v|\boldsymbol{\theta})}{P(\mathbf{x}_v|\boldsymbol{\theta})} \tag{4}$$

where $\mathbf{x}_v$ are the observed state elements, $\mathbf{x}_h$ are the unobserved state elements, and $\boldsymbol{\theta}$ are the parameters of the retrieved schema. This equation shows that the system can divide the joint distribution of all the schema variables by the probability of the retrieval cue to infer the posterior distribution of unobserved variables. But computing the exact posterior distribution is not trivial, and computing the probability of the retrieval cue can involve intractable integrals if some of the schema variables are continuous. Even in cases where all the variables are discrete, computing the posterior distribution takes an exponential time to the tree-width of the Bayesian network (Murphy, 2012).

To remedy this, we turned to an approximate class of inference strategies called *variational inference* (Murphy, 2012). This kind of inference strategy is appropriate whenever the true joint distribution, $p^*(\mathbf{x})$, of variables is complex, but can be approximated by a simpler, variational family of distributions, $q(\mathbf{x})$, which can be made close to $p^*(\mathbf{x})$ by minimizing the KL divergence between the two. Hence, using a variational inference strategy turns the inference problem into an optimization problem, for which many efficient strategies exist. In our context, $p^*(\mathbf{x})$ is the distribution captured in the schema, and we chose the variational distribution, $q(\mathbf{x})$, to be the distribution represented by a Bethe cluster graph (Koller & Friedman, 2009). As shown in Fig. 4, a Bethe cluster graph is a bipartite graph that has the CPDs of the Bayesian network as its first set of nodes, and the individual variables in the Bayesian network as the second set of nodes. Each node in the Bethe cluster graph is initially assigned a *factor*, ϕ_i. For the nodes in the first set, their factors are the corresponding CPDs, while the nodes in the second set have the initial factors of all 1's. This ensures that all of the variable's outcomes are equally likely at the outset. But the variables corresponding to the observations given in a retrieval cure are set to the observed values. Finally, an edge exists between a node i from the set of variables, to a node j from the set of CPDs, only when the variable i participates in the CPD j.

Given the initialized Bethe cluster graph, we use a message passing procedure, loopy belief propagation (Koller & Friedman, 2009), to perform probabilistic inference over this structure. Table 4 outlines this process. It begins on Line 2 by setting the values of observed variables in the cluster graph. The body of the procedure contains a loop, in which messages are passed from cluster to cluster. The loop

Table 4 Procedure for performing probabilistic inference

```
1:  procedure CALIBRATE-FACTOR-GRAPH(factors, op, edges, evidence)
2:      messages ← INITIALIZE-GRAPH(edges, evidence)
3:      repeat
4:          calibrated ← true
5:          for i from 0 to LENGTH(edges) - 1 do
6:              j ← edges[0]
7:              k ← edges[1]
8:              sepset ← CPD-IDENTIFIERS(factors[j]) ∪ CPD-
    IDENTIFIERS(factors[k])
9:              current-message ← messages[j][k]
10:             new-message ← SEND-MESSAGE(j, k, factors, op, edges, mes-
    sage, sepset)
11:             messages[j][k] ← new-message
12:             if new-message ≠ current-messgae then
13:                 calibrated ← nil
14:     until calibrated
15:     if op = "+" then
16:         for i from 0 to LENGTH(factors) - 1 do
17:             collect COMPUTE-BELIEF(i, factors, edges, messages)
18:     else if op = "max" then
19:         constraints ← nil
20:         for i from 0 to LENGTH(factors) do
21:             constraint ← COMPUTE-BELIEF(i, factors, edges, messages)
22:             constraint ← THRESHOLD-ON-MAX(constraint)
23:             constraints ← ADD-CONSTRAINT(constraints, constraint)
24:         CONSTRAINT-SATISFACTION-PROBLEM-SOLVER(constraints)
```

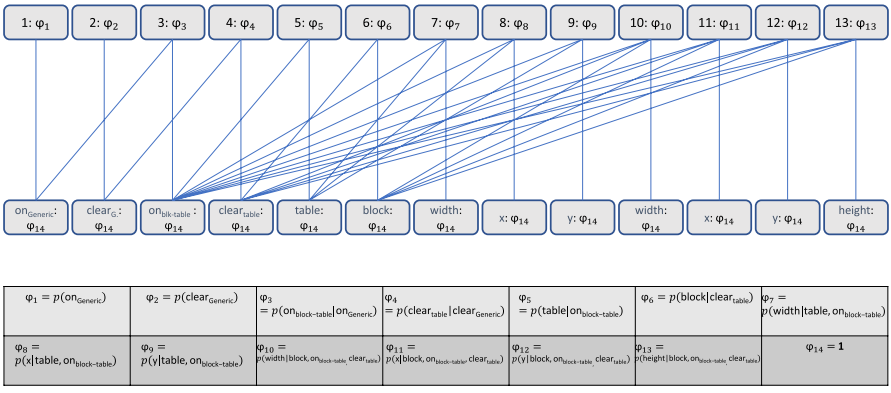


Fig. 4 Bethe cluster graph for Bayesian network in Fig. 1

continues until the messages converge. At that point, two adjacent clusters will contain the same marginal distribution for variables they have in common, so the loop exits. Following this, the system can either return a list of the marginal probabilities of all variables as shown in Line 17 or return the most probable state based on the posterior marginals as shown on Line 21. We are interested in remembering which corresponds to the latter. In this case, the inference procedure produces a list of

constraint rules that are fed into a constraint satisfaction problem solver to uncover the specific values of the variables.

Running probabilistic inference procedures over Bethe cluster graphs provides us a more efficient inference process than those running directly over Bayesian networks. But this advantage comes at a price. This is due to the fact that messages from one node in the cluster graph to another can only carry information about one variable. For example, for *node1* to send a message to *node2* in Fig. 4, it must marginalize out information about every variable except for the variable in *node2*. For this reason, the joint behavior of variables is not considered during the inference procedure. This can lead to errors when our event memory system attempts to remember events. This is an unavoidable consequence of approximate inference. We accept this consequence, since this strategy makes inference over Bayesian networks tractable.

We have now covered our system in detail, describing its theoretical commitments, its representations, and processes for storing and remembering events. In the next section, we will turn our attention to testing and evaluating it in a simulated domain. We describe our experimental objectives and present the application domain. Then, we describe our experimental procedure and present our results.

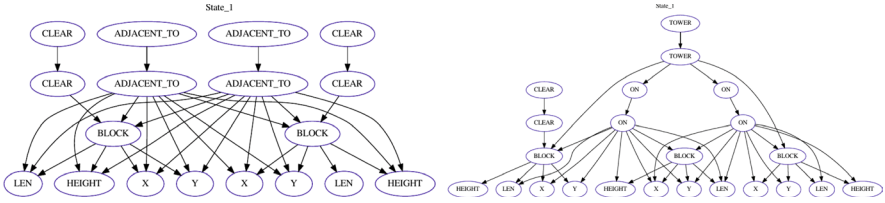
5 Experimental Analysis

As described so far, our novel event memory system uses hierarchically organized elements annotated with probability distributions, to store both specific episodes and event schemas in its generalization tree. The system uses event memory processes that work over this representation to insert and schematize new episodes, and retrieve episodes from the event memory when needed. We argue that the hierarchy in our system affords a level of representational flexibility that neither the causal nor simulation theoretic perspectives capture. We hypothesize that: (1) our system stores and categorizes events in qualitatively distinct schemas; (2) the memory supports event remembering using both episodic and schematic representation; and (3) the retrieval cue dictates which representation the system will use for remembering.

To verify our three hypotheses, we designed two experiments in a version of Blocks World.[1] In the first experiment, the agent observes situations with several blocks of different sizes, infers hierarchical relations based on the sensory input, and store the inferred states as events. Then, in the second experiment, we took a typical event generalization hierarchy from the first experiment and asked the system to retrieve stored events from this hierarchy using partial states as retrieval cues.

More specifically, we first generated 50 sequences of 50 states, randomly sampled from a distribution of two state classes, `adjacent` and `tower`, with 50% probability for each class. As Fig. 5 shows, the two classes represent two distinct kinds

[1] Here, our goal is to evaluate the hybrid memory system on its ability to unify aspects of causal and simulation theories, and this makes Blocks World an ideal domain for evaluation. Our future work will adopt a more complex domain and address the time complexity of our event memory system.



(a) Dependency graph for states that belong to `adjacent` class.

(b) Dependency graph for states that belong to `tower` class.

Fig. 5 Dependency graph for the two event classes used for our experiment

of situations in the world. The former describes states where two blocks are next to each other, and it has a relatively flat dependency structure that includes `clear` and `adjacent_to` relations over perceived objects and their attributes. In contrast, the latter includes states where three blocks are stacked vertically, and it has a taller structure that includes `clear`, `on`, and `tower` relations. In both cases, the blocks vary in their lengths, heights, and positions in x and y directions. For the first experiment, we presented each of these 50 state sequences to the system, which incrementally built its event generalization hierarchy from its observations. We then inspected the 50 hierarchies our system built and analyzed their structure and contents.

Out of these 50 generalization hierarchies, we chose one typical example for our second experiment. We gave this to the system and re-supplied the original 50 states used to build this hierarchy as retrieval cues at different levels of completeness, to see if the system can successfully retrieve and remember the original events. Our experimental procedure consisted of ten epochs, with the completeness of the retrieval cues gradually decreasing from 100 to 10%. Namely, the first epoch uses each of the full states as a retrieval cue, the second epoch uses 90% of each state, and the percentage goes all the way down to 10% of each state for the last epoch. Every epoch included one trial for each set of the original 50 states, in which we presented 20 different partial states of the original state as retrieval cues for remembering. We used partial states that we generated by randomly removing the amount of nodes from the original states that corresponds to the epoch. We recorded several different performance measures during the 10,000 runs. These measures include: the percentage of original state elements provided as retrieval cues, the total number of nodes in the retrieval cues, whether the system retrieved an episode or a schema for remembering, the match distance between the retrieval cue and the retrieved element, and so forth. In the following sections, we analyze our results from the two experiments to verify each of our hypotheses presented above.

5.1 Storing and Schematizing Episodes

Our first hypothesis is that our system can store and categorize events in qualitatively distinct schemas. Since it is not practical to study the qualitative meaning of every single schema from all 50 cases, we instead checked the overall structures of the generalization trees and compared the probability distributions of siblings at a

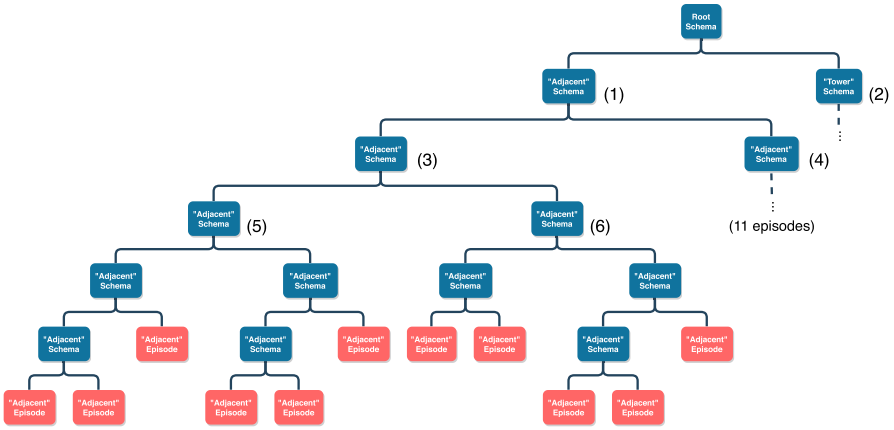


Fig. 6 A typical generalization hierarchy after observing a mixed sequence of `adjacent` and `tower` classes totaling to 50 states

few different levels. Each of the 50 randomly generated sequences of events resulted in a generalization hierarchy like the one shown in Fig. 6 that includes observed episodes at the leaf nodes (red boxes) and several layers of event schemas (blue boxes) over them. The two highest-level schemas under the root node are a generic schema for `adjacent` class of episodes (schema (1) in the figure) and another (2) for `tower` class of episodes, respectively. Further, we found a binary sub-tree under the former, which represents different sub-classes of `adjacent` events. The sub-tree includes three to four levels of schemas and then episodes at the leaf nodes.

Given such a structure, an important question then is whether or not the schemas our system generates group episodes and lower-level schemas properly according to the observed state's qualitative aspects. For this, we chose some schemas from some sample sub-trees and inspected them in detail. We found that there exist a few dominant variables that distinguish the sibling schemas at each level. Figure 7a shows how the two schemas, (5) and (6), differ in their distributions. The probabilities for the first schema's variables are plotted in blue, whereas those for the second schema's variables are plotted in red. The result shows that most of the variables have purple-colored plots, implying that the two schemas have very similar distributions. However, a small number of variables, including x_0, x_1, and $height_0$, shows noticeable red or blue colors where the two schemas differ from each other. This means that these two schemas, (5) and (6), are distinguishable by the x locations of `block0` and `block1` and the height of `block0`. Namely, one schema represents cases where the blocks are located closer to the origin and the first block is shorter, compared to the other schema where the blocks are farther out on the x axis and the first block is taller. The two siblings schemas under schema (4) also exhibited similar characteristics, being distinguishable by the x locations, heights, and lengths of `block0` and `block1` as shown in Fig. 7b. This shows that our system indeed categorizes events in qualitatively distinct, hierarchical schemas while storing them in its event memory.

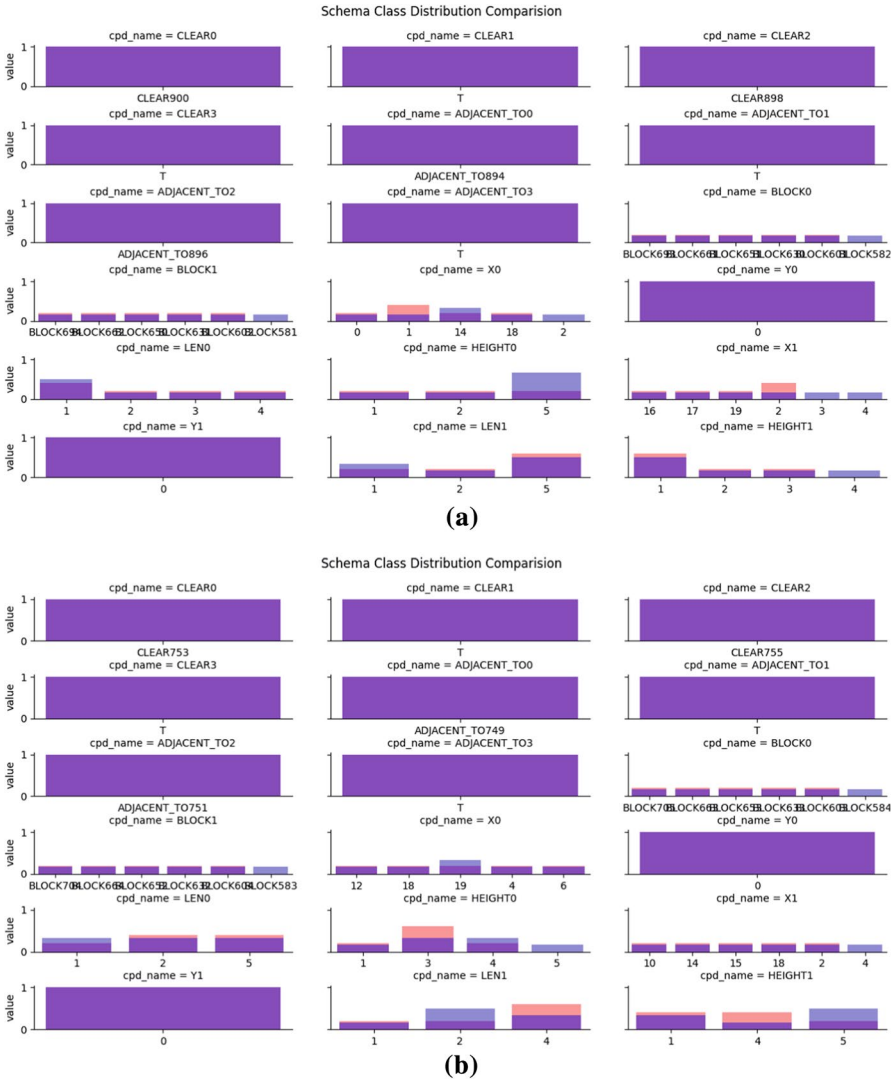
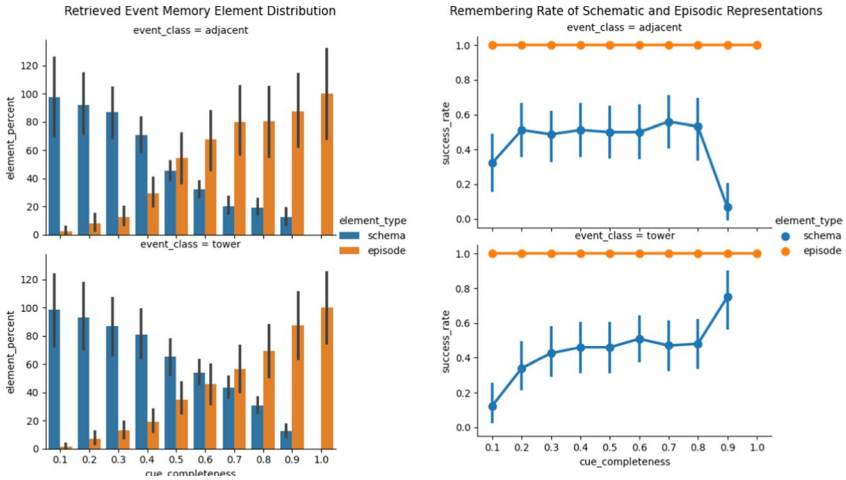


Fig. 7 Comparison of marginal probabilities of two second-level sub-schemas of `adjacent` class

5.2 Remembering Using Episodes and Schemas

Based on the evidence that our system stores and categorizes events into schemas properly, we then proceed to verify the two remaining hypotheses related to remembering. We believe that our system can remember events using either episodic or schematic elements in memory depending on situations. Further, we suspect that there is a parameter that dictates which of these representation the system will retrieve for remembering, and argue that some aspects of the retrieval cue play an important role for this.



(a) Retrieved event memory element distribution.

(b) Remembering success rate of retrieved element type.

Fig. 8 The system remembers events using both episodic and schematic representations

To verify these hypotheses, we first studied the type of event memory elements that our system uses during the second stage of our experiments. Figure 8a shows the percentages of episode usage and schema usage in each of the epochs during our remembering experiment. The plot at the top depicts the percentages of episodes (orange) and schemas (blue) used to remember `adjacent` events, whereas the plot at the bottom shows those used to remember `tower` events. In the epochs where the completeness of retrieval cue is low, the system relies mainly on event schemas to produce remembered events, but it uses more episodes than schemas as the retrieval cues become more complete. In other words, the system prefers schemas when the cue provides only a small amount of information, but it switches to episodes when the cue gives more information. For `adjacent` class of events, this switch happens near 50% cue completeness, while it occurs around 63% for the `tower` class of events. We believe that the switch from schemas to episodes for more complete cues is the right strategy to remember properly. On one hand, it is reasonable to fill in the details from the aggregated previous experience reflected in event schemas, when the retrieval cue gives very little information. On the other hand, it makes sense to retrieve an episode that matches situational details, when the cue provides more complete picture of the state to be remembered.

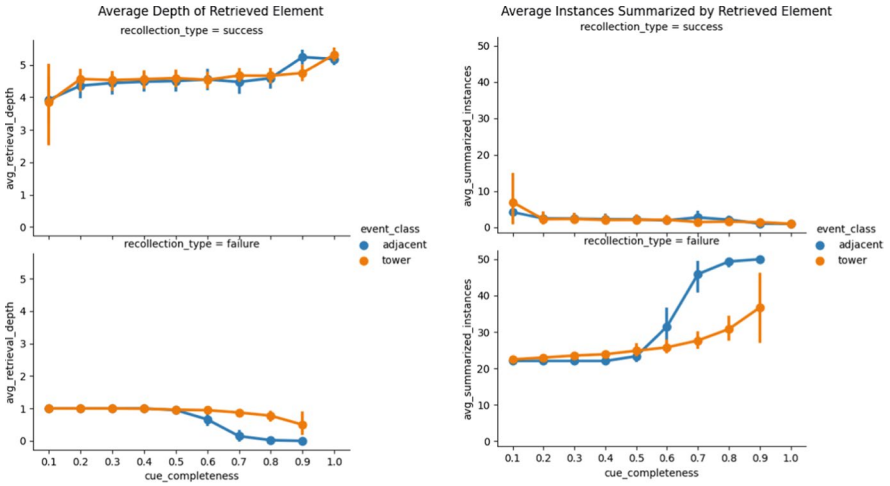
It is important to note that retrieval of an element does not automatically mean that the system is able to remember. In case of episodic retrieval, an element returned from event memory is indeed what the system returns as a remembered event. However, in schematic retrieval, the system needs to generate a specific instance from this schema to produce a remembered event. For this reason, we also measured the rates of successful remembering when the system is using episodes and schemas, respectively. As expected, Fig. 8b shows that the system is always able to produce

a remembered event when it retrieves an episode (orange) for both `adjacent` and `tower` classes. The episodes stored in event memory are full descriptions of events and the system can simply return them when its retrieval process deems these to be the best match.

In contrast, schemas require probabilistic inference based on the retrieval cue to produce a remembered event, and the rate of successful remembering is less than 100%. As shown with blue color in the figure, the system is able to remember an event with a retrieved schema about half of the time, except for cases when the retrieval cues are too vague or too specific. Indeed, the system behavior is very stable in the range from 20 to 80% cue completeness. This implies that the system can remember events in a reliable manner not only when using episodes, but also when using schemas. Further, we believe that the rates of successful remembering in the latter case are about 50%, not higher, because we enforce a complete constraint satisfaction during remembering. In this default setup, we require the system to return failure when the probability distributions in event schemas do not yield a consistent set of assignments for the unobserved variables, namely, the variables not mentioned in the retrieval cue. Later, when we relax this condition, we expect the system to be able to remember events more often. This will be at the expense of the system sometimes producing remembered events that are not exact and fully consistent, but we know that humans often do the same and, in fact, this is central to our planned approach for modeling human memory errors.

In the extreme cases where the completeness of retrieval cue is very low, we observed that the rate of successful remembering when using a schema harbors lower at around 33% for the `adjacent` class and around 12% for the `tower` class. This is reasonable because the retrieval cues provide very little information in this region and the system needs to produce a large, consistent set of variable assignments from probability distributions which is less likely to be successful due to the high uncertainty. The episodic retrieval is still successful all the time in this region, but this is not very meaningful since the system rarely uses episodes here anyway. In the other extreme cases where the completeness of retrieval cue is very high, we found that the two classes yield opposite system behavior. At 90% completeness, the rate of successful schematic remembering for the `adjacent` class is much lower than the norm, whereas that for the `tower` class is higher. We are still investigating this interesting behavior, but we suspect that the more vertical dependency graph of the latter class might be providing more useful information for successful remembering than the wider dependency graph of the former. Finally, at 100% cue completeness, the system depends entirely on episodes and does not use schemas for remembering as we suspected.

In addition, we also looked at the average depth in the hierarchy where the system's retrieval occurs for remembering and the average number of instances those retrieved elements summarize. Figure 9a shows the average depth from which the system retrieved event memory elements when the system succeeds in producing a remembered event through probabilistic inference (top graph) and when it fails to do so (bottom graph). For both `tower` and `adjacent` classes, the system was successful in remembering when it used schemas close to the episodes at the leaf nodes (with depths around 4 to 5), whereas the system mostly failed when



(a) Average depth of retrieved elements

(b) Summarized instances of retrieved elements.

Fig. 9 Characteristics of retrieved elements for remembering

it chose schemas near the top of the hierarchy. Figure 9b also shows a consistent result, where the smaller number of instances summarized by the retrieved schemas in successful remembering cases (top graph) and the higher number of instances in failed cases (bottom graph). As before, we believe that the strict constraint satisfaction we required in this setup played a part in this phenomenon. Furthermore, we attribute the lack of intermediate-level schemas retrieved for remembering to the two event classes being mostly distinct and not sharing many component objects and relations.

In summary, we found evidence that verifies our three hypotheses about the hybrid event memory system we developed. The system is capable of storing events in memory as specific episodes and generalized schemas at the same time, and it can retrieve either of these two to produce a remembered event. We also found that the completeness of retrieval cues modulates the system's behavior as to which representation to use for remembering. These results suggest that our system is a reasonable implementation of the hybrid event memory theory we described earlier, and that it provides a powerful computational framework for us to model various human memory phenomena. In the next section, we review some of the previous work in related directions to position our work in the proper context.

6 Related Work

As discussed in the sections above, we believe that our hybrid theory of event memory provides a novel and elegant unification between two main philosophical theories of human memory. The system we developed based on this hybrid theory can

serve as a computational framework to explain various human event memory phenomena, as we hinted in our earlier review of existing theories. We trust, for the moment, that the review adequately positioned our current work in the philosophical literature. We plan to explore in a separate paper how our system relates to human memory phenomena, as extensively studied in cognitive psychology and neuroscience. There, it will be of particular interest to explore how our implementation would relate to current accounts of human event memory, for example (Eichenbaum, 2017; Yonelinas et al., 2019; Moscovitch et al., 2016). In this section, we instead focus on previous work that are directly related to the technical aspects of our computational theory.

Our system's ability to generate schemas that aggregate numerous episodes is similar in spirit to research on concept formation, where scientists use a wide array of techniques to acquire knowledge from examples. In machine learning, there are methods to learn new concepts from examples included in a dataset (e.g., Murphy, 2012). But these typically separate the learning task from the performance task. In other words, the task of learning the domain knowledge is separated from making predictions using the learned knowledge. This is because all the examples used for forming concepts must be present at the outset of learning. Hence, this methodology can be problematic especially for cognitive systems that must learn and operate over extended periods of time. This suggests that incremental methods for concept formation, which gradually form concepts while observing new examples by interleaving the performance and learning tasks, are more appropriate for cognitive systems.

Some of the earliest incremental concept formation systems include CYRUS (Kolodner, 1983), COBWEB (Fisher, 1987), and UNIMEM (Lebowitz, 1987). These systems represent events as fixed sets of attribute-value pairs. They organize examples into hierarchies with specific instances at the bottom and generalized events at the top. Another early concept formation system, MERGE (Wasserman, 1985), extends the representation of events and decomposes them using a *fundamental* relation. Our system is similar to these systems in organizing events in a hierarchical manner, but it represents episodes as a set of predicates unlike the first three. Further, in contrast to MERGE, our system does not impose a limit on the number of relations present in the state and naturally handles cases where no fundamental relation is identified.

Among these early systems, COBWEB stands out due to its use of probabilistic concepts, which our system also features. COBWEB uses category utility (Gluck & Corter, 1985) as a heuristic for guiding search in the concept hierarchy. This enables the system to construct concepts whose member instances have high intra-class similarity. But it also causes problems in model fit due to the fact that the heuristic has a natural tendency to form categories that capture spurious relationships in the data. In contrast, our system avoids this problem by using the Bayesian Information Criterion as its heuristic evaluation function. As discussed in Sect. 4.2.1, our system considers the predictiveness and complexity of the model simultaneously. The former measures the goodness of fit of the given instance, while the latter acts as a built-in regularization to avoid overfitting.

There are a number of systems built using the COBWEB infrastructure. CLASSIT (Gennari et al., 1989) substitutes categorical attributes with real-valued ones

and generalizes the category utility heuristic into continuous domains, although it still suffers the problem of overfitting like COBWEB. Our system is not currently capable of handling continuous probability distributions, but we plan to extend it to use hybrid Bayesian networks (Koller & Friedman, 2009) and allow inference with both continuous and categorical variables.

Another system using COBWEB as its basis is LABYRINTH (Thompson & Langley, 1991). While most of the early systems could not handle structured information, this system is capable of doing so by extending its representational language for concepts. The system uses hierarchical concepts over primitive, ordered set of attribute-value pairs to describe complex relations in the world. LABYRINTH follows MERGE in the use of a fundamental relation that is believed to naturally decompose events in many domains. Our system uses a different representation where objects, their attributes, and relations are all included as nodes in a Bayesian network. It also uses a top-down matching of events unlike LABYRINTH's bottom-up search, ensuring more efficient insertion and retrieval.

The most recent COBWEB-based system is TRESTLE (MacLellan et al., 2015). It makes many improvements to COBWEB's representational capabilities. TRESTLE can learn clusters containing numerical, categorical, relational, and component information, and it supports partial matching, allowing the system to handle partially observed elements and predict missing elements. But its concepts are flat and does not contain components, whereas our system supports hierarchical concepts. Further, TRESTLE still uses the category utility heuristic to evaluate the quality of match, and it suffers from the same overfitting issues as the original COBWEB.

A recent work that takes a different approach than the COBWEB family of systems for discovering new knowledge in structured domains is SUBDUE (Jonyer et al., 2001). This system incrementally discovers substructures from input data via a lossy iterative compression procedure. To support incremental concept formation, the authors extended their system to form a *concept lattice* of substructures. Interestingly, the heuristic SUBDUE uses, namely, minimum description length, is the negation of the Bayesian Information Criterion we use in our system, which, according to (Koller & Friedman, 2009), we can interpret as the number of bits needed to encode both the model and the data given the model. But the two systems are different in other aspects, like SUBDUE's use of a directed multi-graph versus our use of a tree-based hierarchy, as well as SUBDUE's omission of specific instances in its memory versus our system's storage of both episodes and generalized schemas.

Another recent work related to our system is the Nearest-Merge algorithm (Liang & Forbus, 2014). It constructs hierarchical, probabilistic concepts via analogical generalization. The system stores both a set of generalizations and a set of unassimilated examples. Nearest-Merge extends an analogical generalization system, SAGE (McLure et al., 2010), that implements structure-mapping theory (Gentner, 1983; Falkenhainer et al., 1989). Our system uses a matching algorithm that is similar in spirit, although it is different from Nearest-Merge's two-stage mapping using MAC/FAC processes (Forbus et al., 1995).

In the area of continual learning, Lopez-Paz and Ranzato (2017) designed a gradient episodic memory system capable of learning a diverse set of recognition tasks

as cases stream in, all while avoiding catastrophic forgetting. The system accomplishes this by keeping a dedicated storage for each task that can store up to m examples. Deep neural nets with two hidden layers are then learned for each storage. Our system also learns multiple schemas over instance representations, but unlike the gradient episodic memory, one instance is summarized by many schemas in the hierarchy and can also be retrieved whenever appropriate.

Hintzman 's MINERVA (Hintzman & Ludlam, 1980; Hintzman, 1984, 1986; Collins et al., 2020) is another influential work on modelling human event memory capabilities. The system stores a trace for each experience it encounters in an archive and forms recollections in response to a retrieval cue. Unlike most archival views, MINERVA utilizes a reconstructive retrieval process. When given a retrieval cue, the system matches against all elements in its memory in parallel and forms a schema by averaging elements with high activations from each stored example. Unlike in our work, the generated schema is not stored in memory. This distinction means that MINERVA will gradually lose its ability to recollect events as forgetting takes place since it only stores the examples. Our system, on the other hand, because it stores schemas, can lose some or all of the observed examples and still use the schematic information to remember.

In the cognitive systems literature, there have been several systems with event memory capabilities. Stachowicz and Kruijff (2011) built a memory system with episodic capabilities for a cognitive robot. Nuxoll and Laird (2012) also built an episodic memory for a simulated robot using the Soar cognitive architecture (Laird, 2012). These memory systems were built with an eye toward real-time agents in physical environments, rather than focusing on providing a theoretical framework for event memory. Events are copied into an archive, and then later retrieved to aid the robot achieve a goal state or answer questions. One of the motivations behind our system also relates to its use in cognitive systems, and we previously reported on an earlier version of our memory system (Ménager, 2016) in the context of a cognitive architecture, Icarus (Choi and Langley 2018). We have not tested our latest system in a goal-driven execution agent beyond some trial runs, but we plan to further study our memory system in an agent setting and report our results at a later time.

7 Future Work and Conclusions

Event memory is a central component of human cognition, and our current work covers a small fraction of its functionality. Previous models of the mental structures and processes necessary for remembering events are unable to explain the full range of human uses of event memory. After reviewing the two most prominent theoretical approaches, namely, the causal and simulation theories, we introduced a novel hybrid theory of event memory. Our aim was to develop a hybrid theory that captures the full range of event memory phenomena. We offered a detailed description of our implementation of this hybrid theory, including the representation of episodes and event schemas and the processes that work on these representations.

We also reported results from our experiment to test the system's remembering capabilities in a modified blocks world. The results not only show that our system is able to store both specific episodes and generalized event schemas, but also validate our claim that the system can remember using hybrid representations. Our account thus provides a novel unification of existing theories, which can now be used to model various other features of human event memory. Our model tested relatively simple memories, featuring two items and their relation. Event memories can, of course, be far more complicated. A key feature of future work will be exploring how our framework accommodates more complex representations and relations.

Going forward, we plan to test our event memory system on other aspects of memory processing—most notably, forgetting. We hope to extend this investigation into a broader list of higher-level abilities enabled by event memory storage and retrieval, such as generating explanations about one's past and making predictions of the future from event memory contents. In addition, we hope that our model will prompt further discussion of how philosophical conceptions of event memory can be implemented in Artificial Intelligence systems. Testing our model further in that direction, we also plan to assess our event memory system in real-world situations, evaluating its run-time performance during increasingly complex scenarios. A fast and efficient event memory system is crucial for embodied agents and we will stay mindful of this aspect while we work to extend our event memory system in the context of the ICARUS cognitive architecture.

Acknowledgements This research is partially supported by A*STAR under its *Human–Robot Collaborative AI for Advanced Manufacturing and Engineering* (Award A18A2b0046). The first author was supported by Self Graduate Fellowship at the University of Kansas while performing this research. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the A*STAR or the University of Kansas.

References

Bernecker, S. (2008). *The Metaphysics of Memory*. Springer Science & Business Media.

Bernecker, S. (2010). *Memory: A philosophical study*. Oxford University Press.

Choi, D., & Langley, P. (2018). Evolution of the ICARUS cognitive architecture. *Cognitive Systems Research, 48*, 25–38.

Collins, R. N., Milliken, B., & Jamieson, R. K. (2020). Minerva-de: An instance model of the deficient processing theory. *Journal of Memory and Language, 115*, 1–13.

Debus, D. (2007). Perspectives on the past: A study of the spatial perspectival characteristics of recollective memories. *Mind & Language, 22*(2), 173–206.

Debus, D. (2018). Handle with care: Activity, passivity, and the epistemological role of recollective memories. In D. Debus, D. Perrin, & K. Michaelian (Eds.), *Recollective' memories* (pp. 119–136). New Directions in the Philosophy of Memory.

Eichenbaum, H. (2017). Prefrontal-hippocampal interactions in episodic memory. *Nature Reviews Neuroscience, 18*(9), 547–558.

Falkenhainer, B., Forbus, K. D., & Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence, 41*(1), 1–63.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning, 2*(2), 139–172.

Forbus, K. D., Gentner, D., & Law, K. (1995). MAC/FAC: A model of similarity-based retrieval. *Cognitive Science, 19*(2), 141–205.

Gennari, J. H., Langley, P., & Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence, 40*(1–3), 11–61.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science, 7*(2), 155–170.

Gluck, M.A., & Corter, J.E. (1985). Information, uncertainty and the utility of categories. In: Proceedings of the Seventh Annual Conference of the Cognitive Science Society, pp 283–287

Hintzman, D. L. (1984). Minerva 2: A simulation model of human memory. *Behavior Research Methods, Instruments, & Computers, 16*(2), 96–101.

Hintzman, D. L. (1986). Schema Abstraction in a multiple-trace memory model. *Psychological Review, 93*(4), 411.

Hintzman, D. L., & Ludlam, G. (1980). Differential forgetting of prototypes and old instances: Simulation by an exemplar-based classification model. *Memory & Cognition, 8*(4), 378–382.

Jefferys, W. H., & Berger, J. O. (1992). Ockham's razor and Bayesian analysis. *American Scientist, 80*(1), 64–72.

Jonyer, I., Cook, D. J., & Holder, L. B. (2001). Graph-based hierarchical conceptual clustering. *Journal of Machine Learning Research, 2*(10), 19–43.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. MIT Press.

Kolodner, J. L. (1983). Reconstructive memory: A computer model. *Cognitive Science, 7*(4), 281–328.

Kurby, C. A., & Zacks, J. M. (2008). Segmentation in the perception and memory of events. *Trends in Cognitive Sciences, 12*(2), 72–79.

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

Lebowitz, M. (1987). Experiments with incremental concept formation: Unimem. *Machine Learning, 2*(2), 103–138.

Liang, C., & Forbus, K. (2014). Constructing hierarchical concepts via analogical generalization. In: Proceedings of the Annual Meeting of the Cognitive Science Society

Lopez-Paz, D., & Ranzato, M. (2017). Gradient episodic memory for continual learning. In: Advances in Neural Information Processing Systems, pp 6467–6476

MacLellan, C.J., Harpstead, E., Aleven, V., Koedinger, K.R. (2015). Trestle: Incremental learning in structured domains using partial matching and categorization. In: Proceedings of the Third Annual Conference on Advances in Cognitive Systems

Martin, C. B., & Deutscher, M. (1966). Remembering. *The Philosophical Review, 75*(2), 161–196.

McLure, M.D., Friedman, S.E., Forbus, K.D. (2010) Learning concepts from sketches via analogical generalization and near-misses. In: Proceedings of the Annual Meeting of the Cognitive Science Society, vol 32

Ménager, D. (2016). Episodic memory in a cognitive model. In: Proceedings of the Twenty-Fourth International Conference on Case-Based Reasoning, Atlanta, GA, pp 267–271

Michaelian, K. (2011). Generative memory. *Philosophical Psychology, 24*(3), 323–342.

Michaelian, K. (2016). Confabulating, misremembering, relearning: The simulation theory of memory and unsuccessful remembering. *Frontiers in Psychology, 7*, 1857.

Michaelian, K. (2016). *Mental time travel: Episodic memory and our knowledge of the personal past*. MIT Press.

Michaelian, K., & Robins, S. K. (2018). The causal theory of memory. In D. Debus, D. Perrin, & K. Michaelian (Eds.), *Routledge handbook of philosophy of memory beyond the causal theory?* (pp. 13–32). New Directions in the Philosophy of Memory.

Moscovitch, M., Cabeza, R., Winocur, G., & Nadel, L. (2016). Episodic memory and beyond: The hippocampus and neocortex in transformation. *Annual Review of Psychology, 67,* 105–134.

Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. MIT Press.

Nuxoll, A. M., & Laird, J. E. (2012). Enhancing intelligent agents with episodic memory. *Cognitive Systems Research, 17*, 34–48.

Robins, S. (2016). Representing the past: Memory traces and the causal theory of memory. *Philosophical Studies, 173*(11), 2993–3013.

Rubin, D. C., & Umanath, S. (2015). Event memory: A theory of memory for laboratory, autobiographical, and fictional events. *Psychological Review, 122*(1), 1–23.

Schacter, D. L., & Addis, D. R. (2007). On the constructive episodic simulation of past and future events. *Behavioral and Brain Sciences, 30*(3), 331–332.

Stachowicz, D., & Kruijff, G. J. M. (2011). Episodic-like memory for cognitive robots. *IEEE Transactions on Autonomous Mental Development, 4*(1), 1–16.

Suddendorf, T., Addis, D. R., & Corballis, M. C. (2009). Mental time travel and the shaping of the human mind. *Philosophical Transactions of the Royal Society of London B, 364*(1521), 1317–1324.
Thompson, K., & Langley, P. (1991). Concept formation in structured domains. In: Concept Formation, Elsevier, pp 127–161
Tulving, E. (1983). *Elements of episodic memory*. Oxford University Press.
Wasserman, K. (1985). Unifying representation and generalization: Understanding hierarchically structured objects. PhD thesis, Columbia University
Wells, G. L. (1982). Attribution and reconstructive memory. *Journal of Experimental Social Psychology, 18*(5), 447–463.
Werning, M. (2020). Predicting the past from minimal traces: Episodic memory and its distinction from imagination and preservation. *Review of Philosophy and Psychology, 11*(2), 301–333.
Yonelinas, A. P., Ranganath, C., Ekstrom, A. D., & Wiltgen, B. J. (2019). A contextual binding theory of episodic memory: Systems consolidation reconsidered. *Nature Reviews Neuroscience, 20*(6), 364–375.
Zacks, J. M., & Swallow, K. M. (2007). Event segmentation. *Current Directions in Psychological Science, 16*(2), 80–84.