Hybrid Event Memory as a Case Base for State Estimation in Cognitive Agents

David H. Ménager[1] and Dongkyu Choi[2]

[1] Parallax Advanced Research, Beavercreek OH 45431, USA
`david.menager@parallaxresearch.org`
[2] Institute of High Performance Computing (IHPC)
Agency for Science, Technology and Research (A*STAR)
1 Fusionopolis Way, #16-16 Connexis, Singapore 138632, Republic of Singapore
`Choi_Dongkyu@ihpc.a-star.edu.sg`

Abstract. State estimation is a fundamental problem in artificial intelligence that involves inferring the state of a system based on available measurements. We investigate the role an integrated event memory can play in tackling state estimation of complex domains. Our approach uses the hybrid event memory developed in our previous work as a case base within a cognitive architecture, which enables agents to incrementally learn and represent large joint probability distributions over states as Bayesian networks. To facilitate near real-time execution of this process for agents, we extended the event memory system with a rule-based representation for encoding large probability distributions in its networks. After a review of our cognitive architecture and its event memory, we describe the representational extension before presenting experimental results demonstrating our system's ability to scale to large state estimation problems under various partial observability conditions in the Minecraft domain.

Keywords: hybrid event memory, cognitive architecture, Bayesian networks, state estimation

1 Introduction

State estimation is one of the fundamental problems in artificial intelligence that involves inferring the underlying state of a system from a set of available observations. It is essential in many applications, such as robotics [3], machine learning [12, 39], and decision-making [15]. The state estimation problem has been widely studied in the literature [10, 26, 34–36], and various methods have been proposed to address it. However, this problem still poses significant challenges in complex domains as the number of states and observations can be prohibitively large. One promising approach for solving this is to use Bayesian networks [31], which provide a powerful framework for modeling complex systems and incorporating uncertainty into the estimation process [17, 27].

We believe Bayesian networks provide a powerful and flexible framework suitable for representing *cases*, but their usage in this manner have been sparse in the case-based reasoning (CBR) literature [32], despite the growing interests in integrating data-driven techniques with CBR methodologies [21]. Instead, Bayesian networks have typically been used to enhance inference engines supporting existing CBR systems by capturing general background knowledge about a problem. In [1, 29], for example, Bayesian networks are used to improve similarity assessment by encoding general domain knowledge like relationships about cooking processes. In [16], the authors present a Bayesian clustering algorithm that uses prototype exemplars in salient feature subspaces to describe clusters. Clustering is done in batch using a Bayesian technique, but the clustered objects are the data samples, from which the prototypes are sourced. Additionally, in [7], the authors report a context-aware Bayesian CBR system for constructing dynamic checklists. To form a case, the system augments observed data with the outputs of naïve Bayes inference for estimating answer probabilities.

In contrast, our present work represents cases themselves as Bayesian networks in our event memory that acts as a case base. Furthermore, we explore ways to integrate such CBR systems within a unified theory of cognition, which is an understudied area of research. Our work makes progress along both of these fronts by extending our hybrid event memory system introduced in prior work [24, 25], which is integrated into a cognitive architecture, ICARUS [4]. As we will show later in this paper, the current extension gives our event memory-enabled agent the near real-time capability to learn probability distributions of the state space as Bayesian networks, making it possible to estimate the states continuously during execution.

In Section 2 below, we will briefly review the ICARUS cognitive architecture and its hybrid event memory. We will start by discussing the architecture's knowledge structures that support conceptual inference and skill execution. We will then describe how its long-term event memory represents cases and operates over the stored cases for state estimation. After that, in Section 3, we will describe two extensions to our event memory system that enable faster and more efficient storage and retrieval of cases, which is essential for near real-time state estimation in agent settings. In Section 4, we introduce a popular video game, Minecraft, as our evaluation domain and present experimental results demonstrating the effectiveness of our approach for state estimation. Finally, we will review related literature in Section 5 before closing the paper with discussions of our plans for future work and drawing conclusions in Section 6.

2 Review of ICARUS and its Hybrid Event Memory

Cognitive architectures provide computational infrastructure for modeling general intelligence. One such architecture, ICARUS , makes specific commitments to the representation of knowledge, their organization of memories, and various processes that work over these memories. ICARUS shares some of these with other architectures like ACT-R [2], SOAR [19, 20], and CLARION [37]. The common features include the distinction between long-term and short-term memories, its use of relational pattern matching to access long-term contents, and the cognitive processes that occur through recognize-act cycles, to name a few. But Icarus also makes more distinctive assumptions like the hierarchical organization of long-term knowledge, its grounding of cognition on perception and action, and the structural distinction of categories and procedures. Some of these appeared elsewhere in the cognitive systems literature over time, but only the Icarus architecture makes a continuous commitment to this set of features and integrates them in a unified manner.

2.1 Conceptual and Procedural Memories

Icarus distinguishes between long-term knowledge and short-term contents, and between categorical and procedural knowledge. The architecture organizes these using distinct representations and stores them in separate memories. Its long-term conceptual memory houses definitions of categories, or *concepts*, that are similar to Horn clauses [11] and describe different relational situations, while a long-term skill memory stores definitions of procedures, or *skills*, that can be considered as hierarchical versions of STRIPS operators [5] and specify ways to achieve certain situations.[3]

Table 1 shows some sample concepts for the Minecraft domain. The first concept, `carrying`, describes a situation where the agent has a non-zero amount of an object type in its possession. This concept refers to a perceived object, `hotbar`, and its attributes, but it does not rely on any other concept, making it a *primitive* concept. The second concept, however, is a *non-primitive* one, which refers to other concept instances like `resource`, `right_of`, `left_of`, and `carrying` in addition to a perceived object, `self`. This concept depicts a situation where the agent and a resource object is vertically aligned and the agent is not in possession of the object.

Similarly, Table 2 shows two sample skills. The first skill, `make_torch`, describes the procedure of making a torch out of a `stick` and a `coal`, simply by executing a direct action `*make` in the world. This primitive definition includes conditions for its execution, the action to take, and its effects, but it does not involve any other skill instances. But the second skill, `craft_torch`, defines a complex procedure that involves gathering necessary resource before attempting to make a torch using the first skill, making it non-primitive.

Icarus places the instances of its concepts and skills in respective short-term memories. Its belief memory stores inferred concept instances, or *beliefs*, that the agent believes to be true at any given time. A short-term goal memory houses instantiated top-level goals and subgoals, along with the corresponding skill instances, or *intentions* of the agent for the goals.

[3] In addition, there is another storage, a long-term goal memory, for the Icarus agent's top-level goals and their relevance conditions, which are described using generalized concepts. But this memory and the goal reasoning process that works over it is not relevant to the current discussion.

Table 1: Sample Icarus concepts for the Minecraft domain.

```
((carrying ?o1 ^type ?type ^location ?loc ^size ?size)
 :elements ((hotbar ?o1 type ?type location ?loc size ?size))
 :tests ((> ?size 0)))

((on_vertical_axis ?o1 ?self)
 :elements ((self ?self))
 :conditions ((resource ?o1)
              (not (right_of ?o1 ?self))
              (not (left_of ?o1 ?self))
              (not (carrying ?o1))))
```

Table 2: Sample Icarus skills for the Minecraft domain.

```
((make_torch)
 :conditions ((carrying ?o1 ^type ?t1)
              (carrying ?o2 ^type ?t2))
 :tests ((eq ?t1 'stick)
         (eq ?t2 'coal))
 :actions ((*make "torch"))
 :effects ((carrying ?torch ^type torch ^location ?l ^size ?s)))

((craft_torch)
 :conditions ((resource ?o2 ^type coal)
              (resource ?o3 ^type stick))
 :subskills ((gather_resource ?o3 stick)
             (gather_resource ?o2 coal)
             (make_torch))
 :effects ((carrying ?torch ^type torch ^location ?l ^size ?s)))
```

2.2 Inference and Execution

The Icarus architecture operates in recognize-act cycles. As shown in Figure 1, the system first receives sensory data from its environment in its perceptual buffer and subsequently infers concepts that are true in the current situation. This inference process applies to primitive concepts first, finding all instances of these concepts that hold based on perceived objects and their attributes. Then the system infers non-primitive concept instances that refer to objects and other concept instances.

Once the architecture finishes inferring all beliefs for the current situation, it retrieves skills with matched conditions that are known to achieve its goals. Upon selecting one of the relevant skill instances as its current intention, the agent executes this skill instance in the world, thus changing the environment and its subsequent perceptions. This cyclic operation continues until the agent achieves all its goals.

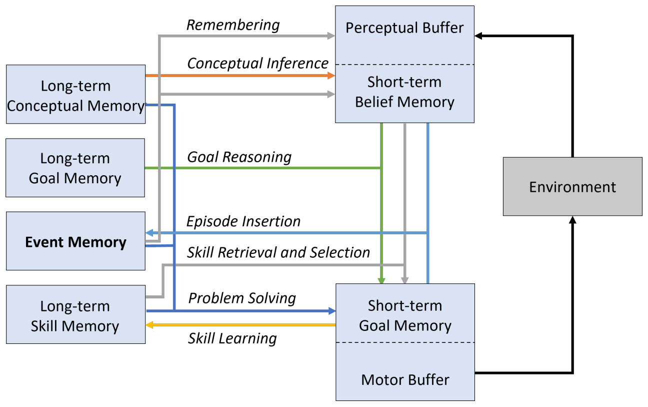


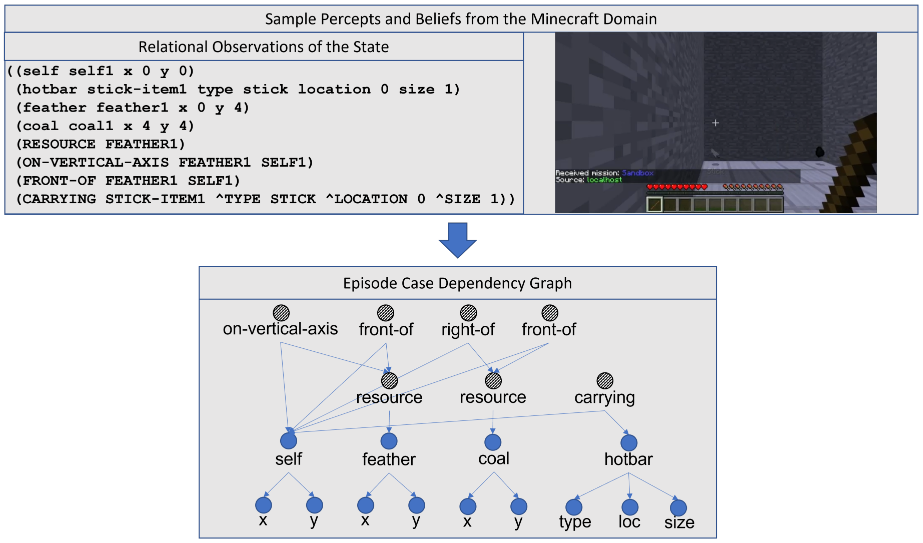
Fig. 1: Block diagram of the extended Icarus architecture featuring an integrated long-term event memory. The memory system forms episodes by combining the agent's perceptions, beliefs, and actions and supports cue-based retrieval into working memory.

2.3 Hybrid Event Memory

The Icarus architecture includes a long-term event memory [24]. This memory serves as a case base for the agent's experiences, storing propositional representation of states that occurred during execution as *episodic* cases. But the memory system also adds generalized *schematic* cases over those at the same time, which serve as propositional templates that aggregates cases in a probabilistic manner.

This memory system is a hybrid, because it represents a middle ground between the two predominant philosophical perspectives on event memory: the causal theory [23] and the simulation theory [28]. The causal theory supports memory systems that represent and maintain discrete, or episodic, events, and thus act like an archive of specific cases. Conversely, the simulationist perspective relies principally on a generalized schema to generate recollections through a reconstructive process. Similarly, our hybrid event memory system stores both episodic and schematic cases in a hierarchy, unifying causal and simulationist views into an elegant theory that exhibits the advantages of both while avoiding their shortcomings.

The long-term event memory forms cases by translating perceived objects and inferred beliefs from a relational description to a propositional one. Figure 2 shows how the architecture encodes a state where the agent is holding a stick and a piece of coal and a feather are visible in the scene. The current state includes some perceived objects like `self`, `hotbar`, `feather`, and `coal`, as well as some beliefs, shown in capital letters, that the agent inferred from the observed objects and their attributes. This state is then represented in a *dependency graph* shown below, which depicts the contents of the episodic case. Relational predicates are shown in grey, while perceived objects and their attributes are shown in blue. Each episodic case is a directed acyclic dependency graph where the nodes in the


Sample Percepts and Beliefs from the Minecraft Domain
Relational Observations of the State
Received mission: Sandbox
Source: localhost
Episode Case Dependency Graph
on-vertical-axis
front-of
right-of
front-of
resource
resource
carrying
self
feather
coal
hotbar
x
y
x
y
x
y
type
loc
size

```
((self self1 x 0 y 0)
 (hotbar stick-item1 type stick location 0 size 1)
 (feather feather1 x 0 y 4)
 (coal coal1 x 4 y 4)
 (RESOURCE FEATHER1)
 (ON-VERTICAL-AXIS FEATHER1 SELF1)
 (FRONT-OF FEATHER1 SELF1)
 (CARRYING STICK-ITEM1 ^TYPE STICK ^LOCATION 0 ^SIZE 1))
```

Fig. 2: Representing a relational observations as a dependency graph. For expositional simplicity, this figure does not show existing edges from the belief nodes to perceptual attributes.

graph contain their observed values from the environment. This graph structure is an episode in the system that represents a single observed state of affairs.

In contrast, schemas are *contextually scoped* cases that aggregate multiple episodes and other schemas together in a probabilistic manner. They encode Bayesian networks that specify the joint probability distribution of a set of correlated variables, X. This joint distribution can be written as:

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

where Pa_{x_t} are the parent nodes of x_t. The advantage of using Bayesian networks to encode schematic cases comes 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.

2.4 Event Memory Processes

As shown above, Icarus ' long-term event memory stores and maintains episodic and schematic cases in a probabilistic taxonomic hierarchy such that similar elements are grouped together separate from dissimilar ones. Episodic cases exist

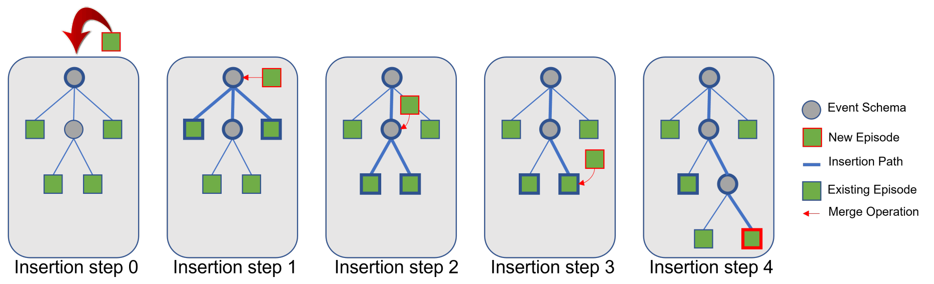


Fig. 3: Top-down insertion procedure for adding a new episode to the event memory. Each insertion step involves performing structure mapping to assess the similarity between the new episode and existing elements, then recursing down the lowest cost branch.

at the leaf nodes in the hierarchy. On top of these are progressively more general schematic cases that summarize their children in a probabilistic manner. The top-level case in the hierarchy encompasses all the agent's experiences and is the most diffuse. In this way, the event memory system maintains a general-to-specific taxonomy. This taxonomy, however, not only serves to index episodes, but also to store a full-fledged probabilistic model in each intermediate schema that supports Bayesian inference. Unlike other CBR systems that use Bayesian networks, schematic cases in our system have a locally defined context via descendant paths in the hierarchy and therefore are not an amalgamation of general domain knowledge.

Within cases stored in the memory system, the boundary between problem and solution is not defined a priori, and this instead depends on the inputs passed to event memory processes for episodic insertion and cue-based retrieval. The former, depicted in Figure 3, occurs in an incremental manner, incorporating new episodes as they are encountered. On each cycle, the Icarus agent generates an episode including its perceptions and beliefs. The system then uses the best-first search to insert the episode top-down through the event hierarchy. At each insertion step, Icarus assesses similarity between the new episode and the existing event memory elements. By doing this for each element along the insertion path of the new episodic case, the system makes analogical comparisons between the new case and existing event memory elements.

Once the system identifies the best-matching event memory element at one level, it merges the new episodic case into the element, according to the correspondences found during the similarity computation, and updates the probability distribution in the matched element. Then, insertion continues down the subtree. The insertion procedure completes whenever the new episode becomes a child of the current case at the root of the subtree, or if the new episode merges with a pre-exisitng case in the hierarchy. The end result of this insertion process is an updated event memory hierarchy with the new example incorporated.

The retrieval process sorts a retrieval cue through the hierarchy employing the same best-first search mechanism as insertion. Instead of adding the retrieval cue to the memory at the end of the search, however, it returns the case that matches best to the cue. Once the system obtains the best-matching event memory element (typically a schematic case), it can use that to perform state estimation through probabilistic inference.

Given the retrieved case, the event memory system takes the observed values from the retrieval cue and sets them as observed variables in the corresponding schema. This creates a problem-solution pair suitable for state estimation such that the problem part corresponds to the observed variables, and the solution part corresponds to the remaining hidden variables, which have not been observed, whose values must be inferred based on the given observations. The probabilistic inference engine in the agent's long-term event memory system is a sum-product message passing algorithm that operates over an approximation of the network, known as a Bethe cluster graph [18]. The system outputs the posterior marginal distribution of each variable given the supplied evidence.

3 Extended Event Memory for State Estimation

In prior work, our event memory system used table-based representations for encoding conditional probability distributions (CPDs) contained in the cases. This was done to ensure efficient retrieval of information by providing random access to the data. This look-up time efficiency, however, came with the disadvantage of imposing high costs on the space requirements of the system. As the dimensionality of the tables increased linearly, the space requirements to encode their distributions grew *exponentially*. Additionally, the large number of parameters in table CPDs burdened the probabilistic inference steps required for performing state estimation. In this section, we describe our latest extensions to the memory system that address these problems.

3.1 Rule-Based Conditional Probability Distributions

The need to have both space- and time-efficient event memory processing for storage and estimation motivated a switch from table-based representations to rule-based ones. Intuitively, a rule in a rule-based conditional probability distribution is a tuple whose left hand side is an assignment to some of the variables in the distribution, denoted as $Scope[\rho]$, while the right hand side specifies the probability for that variable assignment. Rules may be understood as slices of the conditional probability distribution having equal probability. Two rules are *compatible* if the intersection of their left-hand side variables share the same assignments. From this notion of rules, we can formally define rule-based conditional probability distributions below.

Definition 1. *A* rule-based *CPD $P(X|Pa_X)$ is a set of rules $\mathcal{R}$ such that:*

- *For each $\rho \in \mathcal{R}$, we have that $Scope[\rho] \subseteq \{X\} \cup Pa_X$.*

- *For each assignment* $(x, \mathbf{u})$ *to* $\{X\} \cup Pa_X$*, we have one rule* $\langle \mathbf{c}; p, i\rangle$ *such that* $\mathbf{c}$ *is compatible with* $(x, \mathbf{u})$*. In this case we say that* $P(X = x|Pa_x = \mathbf{u}) = p$ *and has occurred* i *times.*
- *The resulting CPD* $P(X|\mathbf{U})$ *is a legal CPD in that*

$$\sum_x P(x|\mathbf{u}) = 1.$$

Definition 1 is taken from [18] with the modification that we add count information to the rule. When interpreting the assertions from Definition 1, it is useful to imagine rules serving as *local coverings* of the table-based CPD. The first item from the definition states that rules must be defined over a subset of variables in the CPD. Next, the second item asserts that each rule represents a slice of the distribution, defined by variables in the left-hand side of the rule, such that each slice covers a region in the table-based CPD. Then, the third statement claims that the sum of each row in the CPD must equal to 1.

With this in mind, it becomes clear that the rule-based representation is a powerful abstraction for achieving *lossless* compression of table-based CPDs by exploiting their local structure. To illustrate this point, consider the example shown in Table 3 depicting the notional conditional probability distribution $P(A|B, C, D)$. Table 3a shows the table CPD for this distribution contains $2^4 = 16$ parameters. By encoding this distribution as a set of rules, as shown in Table 3b, we reduce the space requirements to store the table in half, generating only eight rules. Key to this improvement lies in the fact that one rule can cover multiple slots in the table. For example, for Rule 0, the value of D is ignored allowing it to cover the cases when $D = 0$ and $D = 1$. Additionally, the rule set also contains rules whose counts equal zero. Because these are rules for which outcomes were never observed, we can safely drop them from the rule set to obtain the minimal number of rules, as shown in Table 3c. If there is a query about one of the dropped rules, we can recover the appropriate value from the default distribution, $[1, 0]$[4].

The ability to only store rules for which outcomes have been observed is a boon for building Bayesian networks, captured inside cases, in incremental fashion. As examples are encountered, the architecture can efficiently flesh out the CPDs. In contrast, a table-based representation must reserve space even for outcomes which have not, and may never occur during an agent's operation. This, in combination with exponential growth of table CPDs makes it apparent that they are not appropriate for supporting agents in complex structured environments, and instead rules should be preferred.

3.2 Similarity via Analogical Reasoning for Insertion and Retrieval

The insertion and retrieval processes rely on structure mapping [9] to guide search through the event memory hierarchy. To efficiently solve this problem,

[4] In Table 3a, rows with distribution $[1, 0]$ have not been observed. For the purpose of exposition, such rows were chosen arbitrarily.

Table 3: Encoding table and rule-based CPDs

(a) Table CPD

$P(A|B,C,D)$

			A	
D	C	B	a^0	a^1
d^0	c^0	b^0	1	0
d^0	c^0	b^1	1	0
d^0	c^1	b^0	1/3	2/3
d^0	c^1	b^1	1	0
d^1	c^0	b^0	1	0
d^1	c^0	b^1	1	0
d^1	c^1	b^0	1/3	2/3
d^1	c^1	b^1	0	1

(b) Rule-based CPD

$P(A|B,C,D)$

0	$\langle a^1 b^0 c^1; 2/3, 3\rangle$
1	$\langle a^0 b^0 c^1; 1/3, 3\rangle$
2	$\langle a^1 b^1 c^1 d^1; 1, 1\rangle$
3	$\langle a^0 b^1 c^1 d^1; 0, 1\rangle$
4	$\langle a^1 b^1 c^1 d^0; 0, 0\rangle$
5	$\langle a^0 b^1 c^1 d^0; 1, 0\rangle$
6	$\langle a^1 c^0; 0, 0\rangle$
7	$\langle a^0 c^0; 1, 0\rangle$

(c) Minimal Rules

$P(A|B,C,D)$

0	$\langle a^1 b^0 c^1; 2/3, 3\rangle$
1	$\langle a^0 b^0 c^1; 1/3, 3\rangle$
2	$\langle a^1 b^1 c^1 d^1; 1, 1\rangle$
3	$\langle a^0 b^1 c^1 d^1; 0, 1\rangle$

Icarus converts it to a local optimization problem and applies simulated annealing to obtain matches in polynomial time. Simulated annealing returns a solution by iteratively making random matches between nodes in the new episode and existing memory element nodes. On each iteration, the event memory generates a partial solution and scores it using the Bayesian Information Criterion [33].

Additionally, the architecture also places extra constraints on the structure mapping process to enable it to quickly find solutions. These are:

1. Structure mapping is performed in top-down manner;
2. Matched nodes must share the same type; and
3. Candidate match pairs must have the same matched parents.

The structure mapping procedure terminates whenever the simulated annealing temperature arrives at zero. At that point, the architecture returns the best solution to the upstream process, which is either insertion or retrieval.

4 Empirical Evaluations

To evaluate our system's ability to build its case base from experience and use it for state estimation, we programmed an Icarus agent for Minecraft, a popular sandbox-style video game developed by Mojang Studios[5]. This game presents a unique and challenging environment for artificial intelligence research. It allows players to explore an expansive world and manipulate their surroundings through the use of blocks. The game's open-world environment, coupled with its complexity, requires artificial agents to possess high-level decision-making skills and faculties for handling partial observability to navigate and accomplish goals.

To facilitate the agent's efficient operation, we assumed a flat world where the agent only perceives objects on the floor and limited the agent's inventory slots

[5] We use Minecraft (https://www.minecraft.net/) with its agent interface, Malmo [13]

to three. Nonetheless, the agent could infer spatial relations such as `(left_of ?a ?b)` and `(right_of ?a ?b)` based on these simplified perceptions, maintaining the generality of our experiments. In addition, we simplified the programming of our agent by having it issue only discrete movement commands through the interface. Our agent can choose to move forward, backward, and strafe by one step, as well as turn left or right by 90 degrees.

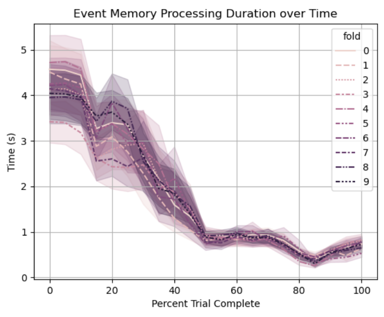
In our experiment, the agent operated in a 7 x 5 room with the objective of making a torch from component resources found in the room. A different resource existed at each corner in the room, and the agent had to walk to the appropriate ones to fashion the item that satisfied its goal. The recipe for making a torch required one item each of `stick` and `coal`. We generated 24 different maps by permuting the locations of the resources in the room. We also gave the agent two skills each for making a torch, resulting in 48 different possible scenarios. In each scenario, the agent followed a recipe for crafting a torch. During the course of execution, we recorded the agent's observations and beliefs about the world, thus generating an execution trace. This enabled us to create a rich dataset of examples from which to build the agent's event memory.

Given this dataset, we split it into training and testing partitions, and performed 10-fold cross validation to build the agent's event memory by inserting the execution traces sequentially. Then in test, we compared our agent's ability to perform state estimation under various partial observability conditions. Under these conditions, the agent received a subset of the perceptual information sensed from the environment, then had to elaborate all missing perceptions, if any, and infer all beliefs since these were never directly perceived.

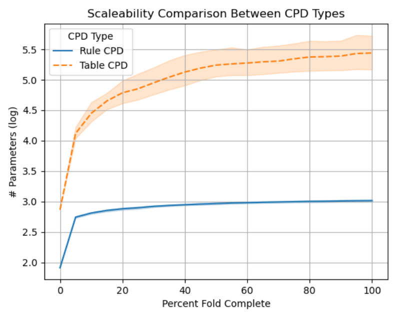
In Figure 4a, we present the average cycle duration across 10 folds as the ICARUS agent inserts the observation traces through its event memory. The figure shows that time it took to insert the examples moved from around three to five seconds initially and progresses down to less then one second at the end of the trace. We believe that because the initial state is similar across the observation traces, inserting them into the event hierarchy progressively expanded the same subtree. As a result, the insertion time for these early observations was comparatively higher than the later observations because the event memory system needed to complete more insertion steps. In contrast, later observations which had more distinguishing features, consequently, wound up in more shallow, less developed parts of the hierarchy resulting in faster insertion times.

Figure 4b compares the number of parameters needed to encode the case distributions at the top-level of the hierarchy by table- and rule-based CPDs. In the figure, the y-axis is the log transform of the number of parameters, while the x-axis shows the percentage of traces observed in the fold. The trends show an order of magnitude savings for rule-based CPDs meaning that this representation scales nicely to highly structured domains.

Next, we present training performance results for state estimation in Figure 5. Figure 5a measures the *local* likelihood of the model, meaning we measure the likelihood of each individual state variable in the model, then report the average. By comparison, Figure 5b measures the average likelihood of the *joint assign-*

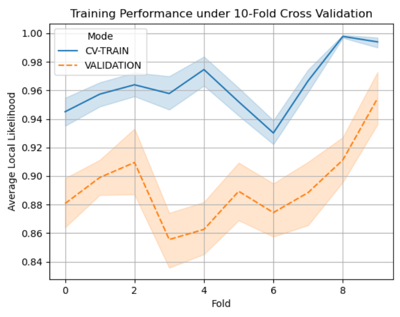


(a) Average time required to process state in ICARUS cognitive cycle.

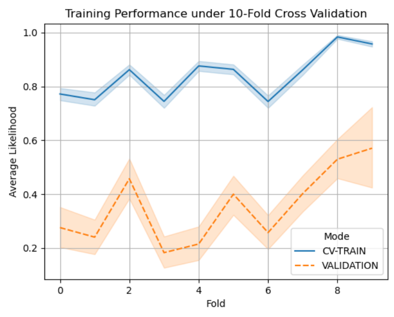


(b) Comparing space requirements of rule-based CPDs with table-based CPDs.

Fig. 4: Insertion performance results of the long-term event memory.



(a) Average *local* likelihood of reconstructing the ground truth from the retrieved model.



(b) Average likelihood of reconstructing the ground truth from the retrieved model.

Fig. 5: Training performance results.

ment of the state variables in the model. During training, the system obtained high marks, but differences in performance arose under the validation set. The local likelihood measure remained high, but a significant drop in performance is observed for the likelihood. This difference in performance occurred because the likelihood measure is conjunctive. This means that if the model assigned low probability to one of the ground truth state variables, it significantly decrease the likelihood score of the model because the low probability value is multiplied to the other probability assignments in the network. Despite this, the likelihood of recovering the ground truth state using the retrieved case significantly exceeds random chance because the space over which the Bayesian network is defined covers, on average, 3.07×10^{29} number of outcomes.

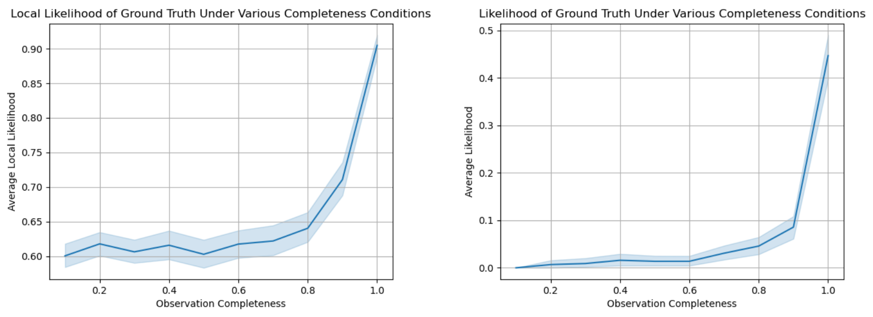


Fig. 6: Likelihood measures under various partial observability conditions.

During test, the ICARUS agent performed state estimation under various partial observability conditions. When the agent had complete observations of the perceived objects both likelihood measures obtained their maximum results, with the local measure achieving greater than 90% performance and the joint measure recovering the ground truth state with greater than 40% likelihood. The performance fell sharply when the agent could not perceive 10% of the perceptual environment, but gracefully degraded beyond that point. For the local likelihood, performance did not drop below 60% on average, which suggests that the state estimation was largely correct with a moderate amount of uncertainty. This uncertainty, however, proved costly for the joint likelihood because it reduced the likelihood of the model generating the ground truth state down to 0% by the time the agent could only observe 10% of the perceptual environment.

5 Related Work

In addition to the previous work from the case-based reasoning literature that we discussed earlier in Section 1, there are other groups of work that our system shares insights with. Most importantly, our event memory system descends intellectually from incremental concept formation systems like COBWEB, TRESTLE and LABYRINTH [6, 8, 22, 38]. Such systems gradually acquire knowledge about a problem domain by clustering examples, as they are encountered, into probabilistic hierarchies. Incremental concept formation emphasizes the compositional nature of knowledge, continual learning, and improvement in predictive performance over time. The event memory system in ICARUS differs from these mainly in that nodes in the hierarchy are Bayesian networks, and it is capable of scaling to a broader range of inference tasks.

Event memory has been modeled as declarative episodic memory in another cognitive architecture, SOAR [19]. SOAR's episodic memory stores contents of the agent's working memory in a flat storage container. It supports insertion and cue-based retrieval of content to aid in other cognitive tasks such as problem solving

[30] and anticipatory thinking [14]. Unlike Icarus , episodes in Soar's episodic memory cannot serve as predictive models of the world. ACT-R and CLARION have declarative memories, but do not make clear distinctions between semantic and episodic content.

6 Future Work and Conclusions

Future research extending this work could include inference over continuous- and discrete-valued variables, insertion strategies that efficiently assess the similarity between related event memory elements, and learning approaches for acquiring conceptual knowledge that enable the agent to describe and categorize states. Pushing the research in these directions can significantly enhance the efficiency and accuracy of state estimation in event memory-enabled agents across a wide array of domains, and we hope to report our results in a near future.

State estimation is a crucial problem in various fields of AI, and the extended Icarus cognitive architecture offers a powerful approach to it, which relies on its event memory system storing and maintaining Bayesian network representations. Studies have shown the benefits and utility of Bayesian networks, but challenges still remain when dealing with complex structured domains. Our work represents a step towards addressing these challenges by integrating our Hybrid Event Memory System into the Icarus cognitive architecture to create an event memory-enabled agent capable of learning probability distributions of the state space as Bayesian networks in an online manner. This research opens up new avenues for exploring the integration of data-driven techniques with Case-Based Reasoning and for unifying theories of cognition.

Acknowledgments

This research is supported in part by Agency for Science, Technology and Research (A*STAR) under its Human-Robot Collaborative AI for Advanced Manufacturing and Engineering (Award A18A2b0046). Any opinions, findings and conclusions, or recommendations expressed in this material are those of the authors and may not necessarily reflect the views of the agency. No official endorsement should be inferred.

References

1. Aamodt, A., Langseth, H.: Integrating bayesian networks into knowledge-intensive cbr. In: AAAI Workshop on Case-Based Reasoning Integrations. pp. 1–6 (1998)
2. Anderson, J.R., Matessa, M., Lebiere, C.: ACT-R: A theory of higher level cognition and its relation to visual attention. Human–Computer Interaction 12(4), 439–462 (1997)
3. Barfoot, T.D.: State estimation for robotics. Cambridge University Press (2017)
4. Choi, D., Langley, P.: Evolution of the ICARUS cognitive architecture. Cognitive Systems Research 48, 25–38 (2018)

5. Fikes, R., Nilsson, N.: STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence 2, 189–208 (1971)
6. Fisher, D.H.: Knowledge acquisition via incremental conceptual clustering. Machine Learning 2(2), 139–172 (1987)
7. Flogard, E.L., Mengshoel, O.J., Bach, K.: Creating dynamic checklists via bayesian case-based reasoning: Towards decent working conditions for all (2022)
8. Gennari, J.H., Langley, P., Fisher, D.: Models of incremental concept formation. Artificial Intelligence 40(1-3), 11–61 (1989)
9. Gentner, D.: Structure-mapping: A theoretical framework for analogy. Cognitive Science 7(2), 155–170 (1983)
10. Hausknecht, M., Stone, P.: Deep recurrent q-learning for partially observable MDPs. In: 2015 AAAI Fall Symposium Series (2015)
11. Horn, A.: On sentences which are true of direct unions of algebras. Journal of Symbolic Logic 16(1), 14–21 (1951)
12. Hu, X., Li, S.E., Yang, Y.: Advanced machine learning approach for lithium-ion battery state estimation in electric vehicles. IEEE Transactions on Transportation Electrification 2(2), 140–149 (2015)
13. Johnson, M., Hofmann, K., Hutton, T., Bignell, D.: The Malmo platform for artificial intelligence experimentation. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence. pp. 4246–4247 (2016)
14. Jones, S., Laird, J.: Anticipatory thinking in cognitive architectures with event cognition mechanisms. In: Cognitive Systems for Anticipatory Thinking at the AAAI Fall Symposium (2021)
15. Kebriaei, H., Rahimi-Kian, A., Ahmadabadi, M.N.: Model-based and learning-based decision making in incomplete information cournot games: a state estimation approach. IEEE Transactions on Systems, Man, and Cybernetics: Part A Systems and Humans 45(4), 713–718 (2014)
16. Kim, Been an Rudin, C., Shah, J.A.: The bayesian case model: A generative approach for case-based reasoning and prototype classification. Advances in Neural Information Processing Systems 27 (2014)
17. Kim, D., Park, M., Park, Y.L.: Probabilistic modeling and bayesian filtering for improved state estimation for soft robots. IEEE Transactions on Robotics 37(5), 1728–1741 (2021)
18. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
19. Laird, J.E.: The Soar Cognitive Architecture. MIT Press, Cambridge, MA (2012)
20. Laird, J.E., Newell, A., Rosenbloom, P.S.: Soar: An architecture for general intelligence. Artificial Intelligence 33(1), 1–64 (1987)
21. Leake, D., Crandall, D.: On bringing case-based reasoning methodology to deep learning. In: Watson, I., Weber, R. (eds.) Case-Based Reasoning Research and Development. pp. 343–348. Springer International Publishing, Cham (2020)
22. MacLellan, C.J., Harpstead, E., Aleven, V., Koedinger, K.R.: Trestle: Incremental learning in structured domains using partial matching and categorization. In: Proceedings of the Third Annual Conference on Advances in Cognitive Systems (2015)
23. Martin, C.B., Deutscher, M.: Remembering. The Philosophical Review 75(2), 161–196 (1966)
24. Ménager, D.H., Choi, D., Robins, S.K.: A hybrid theory of event memory. Minds and Machines pp. 1–30 (2021)
25. Ménager, D.H., Choi, D., Robins, S.K.: Modeling human memory phenomena in a hybrid event memory system. Cognitive Systems Research (2022)

26. Ménager, D.H., Choi, D., Floyd, M.W., Task, C., Aha, D.W.: Dynamic goal recognition using windowed action sequences. In: Workshops at the Thirty-First AAAI Conference on Artificial Intelligence (2017)
27. Mengshoel, O.J., Darwiche, A., Uckun, S.: Sensor validation using bayesian networks. International Symposium on Artificial Intelligence, Robotics, and Automation in Space (2008)
28. Michaelian, K.: Mental Time Travel: Episodic Memory and Our Knowledge of the Personal Past. MIT Press (2016)
29. Nikpour, H., Aamodt, A.: Inference and reasoning in a bayesian knowledge-intensive cbr system. Progress in Artificial Intelligence 10, 49–63 (2021)
30. Nuxoll, A.M., Laird, J.E.: Extending cognitive architecture with episodic memory. In: Proceedings of the Twenty-Second National Conference on Artificial Intelligence. pp. 1560–1565 (2007)
31. Pearl, J.: Fusion, propagation, and structuring in belief networks. Artificial Intelligence 29(3), 241–288 (1986)
32. Richter, M.M., Weber, R.O.: Case-based reasoning. Springer (2016)
33. Schwarz, G., et al.: Estimating the dimension of a model. The Annals of Statistics 6(2), 461–464 (1978)
34. Seo, T., Bayen, A.M., Kusakabe, T., Asakura, Y.: Traffic state estimation on highway: A comprehensive survey. Annual Reviews in Control 43, 128–151 (2017)
35. Shivakumar, N., Jain, A.: A review of power system dynamic state estimation techniques. In: 2008 Joint International Conference on Power System Technology and IEEE Power India Conference. pp. 1–6. IEEE (2008)
36. Sukthankar, G., Geib, C., Bui, H.H., Pynadath, D., Goldman, R.P.: Plan, Activity, and Intent Recognition: Theory and Practice. Newnes (2014)
37. Sun, R.: Anatomy of the mind: exploring psychological mechanisms and processes with the Clarion cognitive architecture. Oxford University Press (2016)
38. Thompson, K., Langley, P.: Concept formation in structured domains. In: Concept Formation, pp. 127–161. Elsevier (1991)
39. Zamzam, A.S., Sidiropoulos, N.D.: Physics-aware neural networks for distribution system state estimation. IEEE Transactions on Power Systems 35(6), 4347–4356 (2020)