An Episodic Based Approach to Complex Event Processing

Dan Tecuci and Bruce Porter
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, USA
{tecuci,porter}@cs.utexas.edu

Abstract

We present our work on building a generic episodic memory module. Such a memory module is intended to provide episodic memory functionality like storage and retrieval for different applications and tasks. We developed a generic representation, storage and retrieval mechanisms for such episodes. An episode consists of a sequence of events, along with the context in which they occurred and their outcome. We then discuss how such a memory module can be used for complex event processing tasks like: recognition, prediction, pattern mining for events, etc.

Introduction

Complex Event Processing (CEP) is concerned with developing tools and techniques for analyzing and controlling the complex series of interrelated events, the kind that are more and more frequently found in modern distributed information systems (Luckham 2002). CEP employs techniques such as detection of complex patterns of many events, event correlation and abstraction, event hierarchies, and relationships between events such as causality, membership, and timing, and event-driven processes.

There has been renewed interest recently in AI applications to enhance intelligent agents with a memory of their past functioning. The inspiration for such work is the human **episodic memory**, a functionally distinct subsystem of human memory that is concerned with storing and remembering specific sequences of events pertaining to a person's ongoing perceptions, experiences, decisions and actions (Tulving 1983).

We believe there is a lot of overlap between research in building an episodic memory for an intelligent system and research in complex event processing. This paper briefly presents some background in human episodic memory research and then describes our approach to building a generic episodic memory for an intelligent system. We then discuss how the results of our work on episodic memory can be applied in the domain of complex event processing.

Human Episodic Memory

This ability of humans to rapidly acquire **episodic memories** has been the focus of considerable research in psychology and neuroscience, and there is a broad consensus that this form of memory is distinct both in its functional properties and in its neural basis from other forms of memories involving common sense knowledge, perceptual-motor skills, priming, and simple classical conditioning (Shastri 2001).

Episodic Memory refers to a memory that maintains a record of 'events' pertaining to a person's ongoing perceptions, experiences, decisions and actions (Tulving 1983). Episodic memory is concerned with unique, concrete, personal experience dated in the rememberer's past (e.g. our last trip to the mall), while **semantic memory** (Feigenbaum 1995) refers to a person's abstract, timeless knowledge of the world that he/she shares with others (e.g. the color of the sky). Episodic and semantic memory subsystems are thought to be functionally distinct but closely interacting memory systems (Tulving 1983).

The episodic and semantic memory subsystems have a number of similarities: they deal with *knowledge acquisition* mostly through senses, this knowledge is *retained* in a passive and automatic way, requiring no effort on the part of the subject, and they both *use this knowledge*, retrieval being triggered by stimuli, subjects are not being aware of it, but only of its results.

Both episodic and semantic memories are thought to be propositional in nature - they can be contemplated introspectively, can be communicated to others in some symbolic form and questions about their veridicality can be asked. These characteristics contrast with those of the **procedural memory** (Winograd 1975), a memory subsystem concerned with the acquisition and utilization of procedures and skills, which is **non-propositional**.

Although they have a number of similarities, episodic and semantic memories differ significantly in other respects. These differences can be categorized along three dimensions: *the kind of information* they handle (specific vs. abstract) and their *operation* (acquired through one-shot learning, provide limited inferential capabilities, susceptible to change due to retrieval vs. gradually acquired, having rich inferential capabilities and relatively stable).

A memory system is widely believed to have three high-level activities: **encoding**, **storage** and **retrieval**. Each of

these activities achieves a particular set of functions (Tulving 1983).

Encoding is the process that converts a perceived event into a memory representation. It consists of **activation** (the process of determining *when* a new episode needs to be recorded), **salient feature selection** (deciding *what information* will be stored), and *cue selection* (deciding *what features* of the state will cue the memory). From a computational point of view, another aspect of encoding is the particular *representation of episodes* and their *organization in memory*.

Storage deals with how stored episodes are maintained over time (*storage medium*). *Forgetting* means preventing recall of episodes (or portions of episodes).

Retrieval is the process by which encoded episodes become available again to the agent. Retrieval is triggered by the agent's state and is based on *cues*, especially salient or significant parts of the retrieval information. *Cue construction* is the process of constructing the data used to retrieve a memory. *Matching* uses these cues in order to search for similar episodic memories. *Recall* means retrieving the memory from storage and placing it into working memory. After this process completes, the memory becomes available for the agent to use.

A Generic Episodic Memory Module

Our research addresses a long recognized need for *general* tools to aid the development of knowledge-based systems (van Melle, Shortliffe, and Buchanan 1984). We attempt to enhance an intelligent system with capabilities to capture, organize and reuse the experience it acquires during the course of its functioning. Although we seek inspiration from the human episodic memory, we do not attempt to build a model of it, but merely to replicate some of its functionality such that it can be used by an intelligent system.

We propose to separate the memory functionality from the system itself and to build a *generic* memory module that can be attached to a variety of applications in order to provide memory functionality (Tecuci 2007).

Encapsulating the complexity of such a memory into a separate subsystem should reduce the complexity of other parts of the overall system, allowing us to focus on the generic aspects of memory organization and retrieval and its interaction with the external application. Each application will use the retrieved memories differently, depending on their task. We do not propose complete solutions for problem solving in these domains as this would require domain specific knowledge (e.g. for adapting prior plans); rather, the episodic memory will have a supporting role.

As the system needs to store its individual experiences, we chose to implement *episodic* memory functionality for our module as opposed to *semantic* (i.e. a memory for generic knowledge). Additionally, the system's experiences can easily be described in terms of its *actions* and the order in which they were taken (e.g. sequence of application of operators in a search-space problem).

Using such a memory module, an intelligent system can store, recall and reuse experience of virtually every reasoning action taken. This is a different kind of knowledge that captures the result and also the context in which a reasoning action was taken. Availability of such knowledge can improve both the correctness and the performance of a reasoning system by focusing on actions that are more likely to produce desirable results, given the history of their past applications.

The functionality of such a memory system in terms of its ability to store, organize and retrieve sequences of events makes it easily applicable to event processing tasks like complex event recognition, prediction, discovery of similar events, etc.

General Memory Requirements

A generic memory module should be: *accurate* (it should return memories relevant for the situation at hand), *scalable* (it should be able to accommodate a large number of episodes without a significant decrease in performance), *efficient* (it should provide efficient storage and recall), *content addressable* (memory items should be addressable by their content), and should provide *flexible matching* (the appropriate previous episodes should be recalled even if they only partially match the current context).

From a software application perspective, a generic memory module for events needs to provide: a *generic representation of events* that can be used with different types of events; a *flexible interface* that allows various types of queries to be formulated and provides feedback to the application on how these queries were matched against memory, and *domain-independent organization and retrieval techniques* that efficiently index events.

Representation

The episode is the basic unit of information that memory operates on. The decision as to what constitutes a meaningful episode is domain dependent and left to the external application to make. In general, an episode is a sequence of actions with a common goal, which cannot be inferred from the individual actions taken in isolation.

Episodes are dynamic in nature, changing the state of the world in complex ways. Besides the sequence of actions that make up the episode, the context in which the episode happens, as well as its effect on the world, are important. We propose to represent a generic episode as a triple: **context**, **contents** and **outcome**. **Context** is the general setting in which an episode happened; for some applications (e.g. planning) this might be the initial state and the goal of the episode (the desired state of the world after the episode is executed). **Contents** is the ordered set of events/actions that make up the episode; in the case of a planner, this would be the plan itself. The **outcome** of an episode is an evaluation of the episode's effect (e.g. if a plan was successful or not, what failures it avoided).

Episodes will be indexed along each of these three dimensions. This allows *the same memory structure* to be used for various tasks that require reasoning about actions. For example, a memory of plan goals, their corresponding plans and whether or not they were achieved by a given plan can be used for tasks such as:

planning - given an initial state and a goal, devise a plan to accomplish the goal. In terms of our representation this corresponds to memory retrieval using episode context (i.e. initial state and goal of a planning problem) and adapting the contents of the retrieved episodes (i.e. their plans).

prediction - given an initial state and sequence of actions, predict their outcome. This corresponds to retrieval based on episode context and using the outcome of the retrieved episodes.

explanation - given a set of observations (including actions) find the best explanation for it. An example of this is plan recognition, where the explanation is the plan being executed. This corresponds to retrieval based on episode contents (i.e. observed actions) and adapting the context of retrieved episodes.

The semantics of individual actions (i.e. their applicability conditions and the goals they achieve), as well as knowledge about the state of the world is represented using our knowledge base - a library of about 700 general concepts such as `Transport`, `Communicate`, `Enter` and 80 semantic relations like `agent`, `object`, `causes`, `size` (Barker, Porter, and Clark 2001). The underlying representation language and reasoning engine is KM (Clark and Porter 2001).

Memory API

The memory module provides two basic functions: **store** and **retrieve**. **Store** takes a **new Episode** represented as a triple [context, contents, outcome] and stores it in memory, indexing it along one or more dimensions; **retrieve** takes a **stimulus** (i.e. a partially specified Episode) and a **dimension** and retrieves the most similar prior episodes along that dimension. Memory retrieval provides also information on how a stimulus matched the retrieved episodes (e.g. shared structure, differences, mappings). This information is intended to be used by the external application that works in connection with the memory module and helps it better utilize the returned episodes for its purpose (e.g. adaptation).

Memory Organization and Retrieval

Episodes are indexed using a multi-layer indexing scheme similar to MAC/FAC (Forbus, Gentner, and Law 1995) and Protos (Porter, Bareiss, and Holte 1990): a *shallow indexing* in which each episode is indexed by all its features taken in isolation and a *deep indexing* in which episodes are linked together by how they differ *structurally* from one another.

During retrieval, shallow indexing will select a set of episodes based on the number of common features between them and the stimulus. Starting from these candidate episodes, a hill-climbing algorithm using semantic-matching will find the episode(s) that best match the external stimulus. A robust memory needs to employ a flexible matching algorithm, so that old situations are still recognized under new trappings. The semantic matcher we use (Yeh, Porter, and Barker 2003) employs taxonomic knowledge, subgraph isomorphism and transformation rules in order to resolve mismatches between two representations.

It is the *organization of memory* given by this indexing mechanism and the *search-based retrieval* that sets our approach apart from those employing a flat memory structure that is searched serially (e.g. (Nuxoll and Laird 2004; Forbus, Gentner, and Law 1995)).

The ability to deal with streams of events is important in a lot of applications (e.g. surveillance), which typically require making a decision/prediction after each observation. Incremental availability of data seems to increase retrieval time as memory needs to perform retrieval after the presentation of each new stimulus. However, incremental data reduces the size of each query, while the context of the previously observed actions reduces the search space. (Schmidt, Sridharan, and Goodson 1978; Schank 1982; Litman and Allen 1987).

Our incremental retrieval algorithm (see Algorithm 1) functions as follows: after an action is observed, revise the current set of hypotheses so that they account for the last seen stimulus. This is done by trying to match the stimulus against current hypotheses or by generating new ones. New hypotheses are generated using the shallow indexing mechanism that retrieves episodes that are superficially similar to the given query. We limit the number of such new hypotheses to the most likely N (we have experimented with N=5 and N=10). Current hypotheses are then semantically matched to the new stimulus and, based on the result, they are re-ranked according to their similarity to the plan observed so far. Mismatches between an observed action and the action of a prior episode are allowed. Memory treats both as possibly superfluous actions.

Algorithm 1 Incremental-retrieve algorithm

```
candidates ← []
observed-actions ← []
while there are observed-actions left do
   curr-action ← pop (observed-actions)
   new-candidates ← retrieve(current-action)
   for all episode ∈ new-candidates do
      if episode ∉ candidates then
         synchronize-candidate(episode, prior-actions)
      end if
   end for
   for all candidate ∈ candidates do
      candidate-match ← match(curr-action, candidate)
      if candidate-match ≠ [] then
         candidates ← update-candidate(candidate-match)
      else
         candidates ← update-candidate(candidate-match)
      end if
   end for
   candidates ← sort(candidates)
   prior-actions ← prior-actions ∪ current-action
end while
result ← sort(candidates)
return first-n (*MAX-RETRIEVED*, result)
```

The complexity of this incremental retrieval algorithm in the best case is linear in the number of actions observed

(s) and the maximum number of reminded episodes explored for a new stimulus (N). The worst case complexity is $O(Ns^3)$, but rarely happens in practice (Tecuci 2007). Unlike statistical approaches, the complexity is not a function of the number of goal schemas, but only of the number of observed actions.

Episodic Memory and Complex Event Processing

This section explores the ways in which research on building and using such a generic episodic memory can contribute to the area of complex event processing. In what follows, episodes and complex events will be used interchangeably.

Episode Similarity

The episode representation proposed in the previous section uses an ontology to describe individual event types and their parameters (e.g. the domain objects it employs). Adding semantics to individual events allows the processing of simple and complex events at a semantic rather than syntactic level. For example, the similarity of two individual events can be better judged even when they are not of the same, by examining how their respective types are taxonomically related. For example, when computing the similarity between two Linux commands like `Copy-File` and `Move-File`, one can use the fact that both are a kind of file creation.

Many tasks involving complex events (e.g. recognition tasks) require computing the similarity between two *sequences of events*. Most measures of semantic similarity compute similarity between two such objects with respect to one of them. This does not deal well with noise, as the superfluous actions in the reference episode artificially decrease the overall similarity score. One similarity metric that we used is defined as the product of the two individual similarity scores when each object is in turn the reference one.

$$sim(E_1, E_2) = semsim(E_1, E_2) * semsim(E_2, E_1)$$

where $semsim(E_1, E_2)$ is the similarity between the episodes E_1 and E_2 with respect to E_2:

$$semsim(E_1, E_2) = \sum_{t_i \in E_1 \sim E_2} match - score(t_i)/|E_2|$$

where E_1 and E_2 are episodes (i.e. sequences of events), $E_1 \sim E_2$ represents the isomorphic mapping from E_1 to E_2; t_i represents the isomorphic relation between a vertex in the E_1 graph and its corresponding counterpart in E_2; and $match - score(t_i)$ measures how well the two vertices match and is a number between 0 and 1 provided by the matcher.

Complex Event Recognition

A large class of established AI applications relies on recognizing complex ongoing events: language understanding and response generation (Allen and Perrault 1986; Perrault and Allen 1980), user interfaces (Goodman and Litman 1992), help systems (Mayfield 1992), and collaborative problem solving (Lesh, Rich, and Sidner 1999). There are also emerging applications that could benefit from this: security threat detection, business activity monitoring, etc.

We believe that memory recognition is applicable to this class of problems. To test this, we applied the episodic-based approach to the problem of *goal schema recognition* (i.e. given a sequence of observed actions, predict the plan that is being executed) (Tecuci 2007).

A memory-based approach to this problem has the advantage that it does not require all recognizable plans (or plan schemas) to be known in advance, but is able to dynamically grow the plan library. However, due to the instance-based nature of such an approach, it has to provide a scalable way of plan storage and retrieval.

We performed an evaluation on the goal schema recognition task on two datasets (Linux (Blaylock and Allen 2004) and Monroe (Blaylock and Allen 2005a)). The Linux plan corpus was gathered from Linux users from the University of Rochester, Department of Computer Science, who were given goals like 'find a file with `exe` extension' and were instructed to achieve it using simple Linux commands. All user commands along with their results were recorded. For each goal, users were also asked to assess whether they accomplished it. The users judged 457 sessions to be successful involving 19 goal types and 48 action types (i.e. Linux commands). Because some users were not able to judge this correctly, the dataset is noisy (i.e. there are still a number of failed sessions marked correct).

The Monroe corpus consists of stochastically generated plans in the domain of emergency response. The plans have been generated by allowing a planner to make nondeterministic decisions and therefore generating a diverse set of plans (in terms or ordering of their actions) for the goal. It contains 5000 plans with an average of 9.5 actions per plan, a total of 10 goal types and 30 action types.

Our experiments showed that the episodic-based approach achieves similar performance to a statistical approach (Blaylock and Allen 2005b) and that memory retrieval is scalable (retrieval effort does not grow at the same rate as memory size).

We measured the accuracy of the recognizer in terms of **Precision (P)** and **Recall (R)**. Precision is the number of correct predictions divided by the total number of predictions (i.e. the number of times the recognizer chooses to make a prediction), while recall is the number of correct predictions divided by the number of predictions opportunities (i.e. the number of observed actions). They measure the *overall accuracy* of the recognizer as it includes predictions made after each *new* observed action.

A measure of how many plans were *eventually* recognized is denoted by **convergence (Conv)**, which is the number of correct predictions after the last plan action was observed. A recognition session is said to have converged if its last prediction was correct.

Experimental results are reported in Figure 1. The precision, recall and F-measure are the same because the memory-based approach makes predictions after each action (e.g. the number of prediction opportunities is the same as the number of predictions made.)

Compared to the statistical approach, EM converges on more sessions for the Linux domain (see Figure 2(a)) and on a similar number for the Monroe domain (Figure 2(b)). Pre-

cision is slightly lower on both domains (see Figure 2(a) and 2(b)), although probably not significantly different. (Blaylock and Allen 2005b) does not report variance for their data. However, recall is much higher for the memory-based approach on both domains. An increase in precision at the expense of recall is expected given that the statistical recognizer only makes predictions when a certain confidence threshold was achieved.

Event-stream processing

Algorithms that implement recognition of complex events should be able to generate incremental predictions, after each action is observed, thus offering early predictions. They also need to be fast at this incremental recognition task (e.g. faster then the next action can take place so that the user of the recognizer system can take counter-action).

In the previous section we presented our incremental retrieval algorithm. We evaluated its performance on the goal schema recognition task in the Linux and Monroe domains. Specifically, we measured the **convergence point** - how soon in the unfolding of a plan it starts making the *same correct prediction*. This was measured both in terms of observed actions as well as in terms of percentage with respect to the average number of actions of converged sessions.

In terms of convergence point, EM converges with approximately the same speed as the statistical approach (after seeing 63%, 57% and 51% of actions in sessions that converged, compared to 59%, 55% and 57%), but the length of converged session is lower (4.30, 4.14, 4.48 compared to 5.9, 7.2 and 7.2). This might be due to the fact that the statistical approach only makes predictions when above a certainty level, for which it needs to see more actions.

Other Applications

Our episode representation also stores the context and outcome of an episode, which can be used as retrieval cues. This enables **contextual reasoning** tasks to be performed by an external system using such a memory module. For example, in a planning application, given a goal and a current state of the world, the system can answer questions like 'what is the action most likely to lead to the accomplishment of the goal?'. Such a task can be implemented by querying memory for previous episodes with similar goals and similar partial states.

The complex event recognition algorithm enables **other predictions tasks**, besides the type of plan being observed. Using the results of the recognition algorithm, an external application can predict the outcome of complex events (by extrapolating from a similar event that happened in the past), the goals of the agent executing the actions observed, or the next action to be observed.

The episodes stored by memory are particular instances of more generic complex events in the world and can provide useful data for mining algorithms that can abstract away various event types or relationships in order to build **complex event patterns**.

In complex domains it is likely that an agent will carry on multiple plans at the same time, interleaving their actions. A recognizer will have to be able to deal with these different plans unfolding at the same time and still perform recognition on such data. We believe that the proposed recognition algorithm lends itself easily to this task given that episodic-based recognition is already able to entertain multiple hypotheses at the same time.

Conclusions

This papers argues that complex event recognition can benefit from research into enhancing intelligent agents with episodic memories.

We presented our work on building a generic episodic memory that is separate from the application itself and whose purpose is to provide episodic storage and retrieval. The proposed representation scheme stores not only sequences of events but also their contexts and outcomes, described in terms of an underlying knowledge base. Such episode structures are multi-functional in that they can be used for various tasks like planning, prediction and explanation. Adding semantics to individual actions enables a more accurate similarity assessment of both simple and complex events.

We also described an incremental recognition algorithm for complex events based on such similarity. An evaluation on a goal schema recognition task on two datasets showed this algorithm is as accurate as a state-of-the-art statistical approach and quite scalable. The same recognition algorithm can be used to generate other kinds of predictions.

References

Allen, J., and Perrault, C. R. 1986. Analyzing Intention in Utterances. In *Readings in natural language processing*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. 441–458.

Barker, K.; Porter, B.; and Clark, P. 2001. A Library of Generic Concepts for Composing Knowledge Bases. In *K-CAP 2001: Proceedings of the International Conference on Knowledge Capture*, 14–21. ACM Press.

Blaylock, N., and Allen, J. 2004. Statistical goal parameter recognition. In *The 14th International Conference on Automated Planning and Scheduling (ICAPS'04)*.

Blaylock, N., and Allen, J. 2005a. Generating Artificial Corpora for Plan Recognition. In Ardissono, L.; Brna, P.; and Mitrovic, A., eds., *Lecture Notes in Artificial Intelligence - User Modeling 2005*, volume 3538. Springer.

Blaylock, N., and Allen, J. 2005b. Recognizing Instantiated Goals Using Statistical Methods. In Kaminka, G., ed., *IJCAI Workshop on Modeling Others from Observations (MOO-2005)*, 79–86.

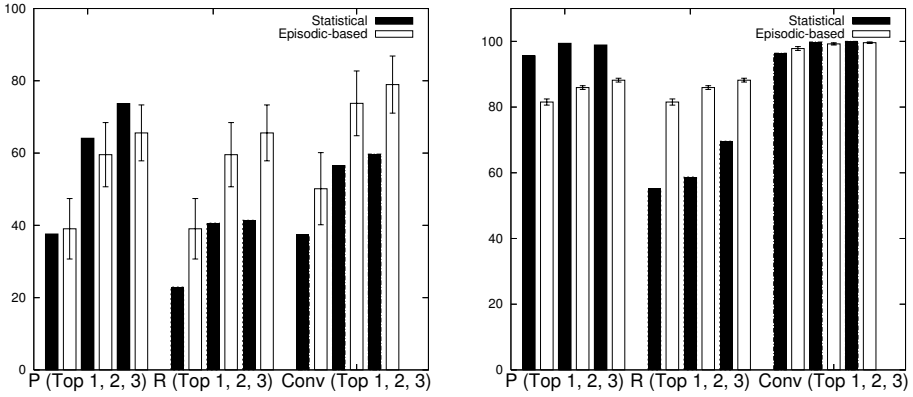
Clark, P. E., and Porter, B. W. 2001. *KM v2.0 - The Knowledge Machine: User's Manual and Situations Manual*. University of Texas at Austin.

Feigenbaum, E. A. 1995. The Simulation of Verbal Learning Behavior. In *Computers & Thought*. Cambridge, MA, USA: MIT Press. 297–309.

Forbus, K.; Gentner, D.; and Law, K. 1995. MAC/FAC: A Model of Similarity-Based Retrieval. *Cognitive Science* 19(2):141–205.

	Linux				Monroe			
N-best	1	2	3	4	1	2	3	4
P, R, F (%)	39.05	59.55	65.58	69.13	81.53	85.94	88.19	89.66
Conv (%)	50.14	73.76	78.95	82.57	97.82	99.24	99.60	99.80
CP/AvgLen	2.7/4.30	2.37/4.14	2.28/4.48	2.16/4.60	3.04/9.44	2.56/9.49	2.29/9.49	2.11/9.54
CP/AvgLen (%)	62.79	57.24	50.89	46.95	32.2	26.97	24.13	22.21

Figure 1: Experimental results for the memory-based approach on the goal schema recognition task.



(a) Linux corpus. (b) Monroe corpus.

Figure 2: Comparison between memory-based and statistical approaches on a goal schema recognition recognition

Goodman, B. A., and Litman, D. J. 1992. On the Interaction between Plan Recognition and Intelligent Interfaces. *User Modeling and User-Adapted Interaction* 2(1-2):83–115.

Lesh, N.; Rich, C.; and Sidner, C. 1999. Using Plan Recognition in Human-Computer Collaboration. In *Proceedings of the Seventh International Conference on User Modeling*, 23–32.

Litman, D., and Allen, J. 1987. A plan recognition model for subdialogues in conversation. *Cognitive Science* 11:163–200.

Luckham, D. 2002. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley.

Mayfield, J. 1992. Controlling Inference in Plan Recognition. *User Modeling and User-Adapted Interaction* 2(1-2):83–115.

Nuxoll, A., and Laird, J. E. 2004. A Cognitive Model of Episodic Memory Integrated With a General Cognitive Architecture. In *Proceedings of the Sixth International Conference on Cognitive Modeling*, 220–225. Mahwah, NJ: Lawrence Earlbaum.

Perrault, C. R., and Allen, J. F. 1980. A Plan-Based Analysis of Indirect Speech Acts. *American Journal of Computational Linguistics* 6(3-4):167–182.

Porter, B. W.; Bareiss, R.; and Holte, R. C. 1990. Concept Learning and Heuristic Classification in Weak-Theory Domains. *Artificial Intelligence* 45:229–263.

Schank, R. C. 1982. *Dynamic Memory. A Theory of Reminding and Learning in Computers and People*. Cambridge University Press.

Schmidt, C. F.; Sridharan, N. S.; and Goodson, J. L. 1978. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence* 11:45–83.

Shastri, L. 2001. Episodic memory trace formation in the hippocampal system: a model of cortico-hippocampal interaction. Technical Report TR-01-004, International Computer Science Institute, Berkeley, CA.

Tecuci, D. 2007. *A Generic Memory Module for Events*. Ph.D. Dissertation, The University of Texas at Austin.

Tulving, E. 1983. *Elements of Episodic Memory*. Oxford: Clarendon Press.

van Melle, W.; Shortliffe, E.; and Buchanan, B. 1984. EMYCIN: A Knowledge Engineer's Tool for Constructing Rule-Based Expert Systems. In Buchanan, B., and Shortliffe, E., eds., *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley. chapter 15.

Winograd, T. 1975. Frame representations and the procedural - declarative controversy. In Bobrow, D., and Collins, A., eds., *Representation and Understanding*. Academic Press. 185–210.

Yeh, P. Z.; Porter, B.; and Barker, K. 2003. Using Transformations to Improve Semantic Matching. In *K-CAP 2003: Proceedings of the Second International Conference on Knowledge Capture*, 180–189.