Memory Based Goal Schema Recognition

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 propose a memory-based approach to the problem of goal-schema recognition. We use a generic episodic memory module to perform incremental goal schema recognition and to build the plan library. Unlike other case-based plan recognizers it does not require complete knowledge of the planning domain or the ability to record intermediate planning states. Similarity of plans is computed incrementally using a semantic matcher that considers the type and parameters of the observed actions. We evaluate this approach on two datasets and show that it is able to achieve similar or better performance compared to a statistical approach, but offers important advantages: plan library is acquired incrementally and the memory structure it builds is multi-functional and can be used for other tasks such as plan generation or classification.

Introduction

A large class of AI applications relies on recognizing complex events: language understanding and response generation (Allen and Perrault 1986; Perrault and Allen 1980), user interfaces (Goodman and Litman 1992), help systems (Mayfield 1992), collaborative problem solving (Lesh, Rich, and Sidner 1999), and threat detection.

The problem of recognizing such complex events and ascribing goals, intentions and future actions to an actor based on its observed actions is known as *plan recognition* (Schmidt, Sridharan, and Goodson 1978). A subproblem of plan recognition is *goal-schema recognition* where only the type of plan being executed is recognized.

The need for domain-independent plan recognition systems has long been recognized. (Huwer, Smart, and Cairns 1995) identifies several requirements for such plan recognition algorithms: efficiency, robustness, use of domain-independent representation of actions and plans, ability to detect and incorporate new plans into the plan library, ways of limiting the search through the plan library, and ability to deal with noise (i.e. erroneous actions). Domain-specific knowledge might improve performance but has to be balanced against the recognizer's purpose of being domain-independent.

Plan recognition algorithms should generate incremental predictions, after each action is observed, thus offering early predictions and be fast at this task (e.g. faster then the next action can take place so that the user of the recognizer system can take counter-action).

We propose a memory-based approach to the problem of goal schema recognition as a subtask of the *keyhole plan recognition*, a type of plan recognition problem where there is no cooperation between the agent and the recognizer (Albrecht et al. 1997). We use a generic episodic memory module (Tecuci 2007) and its recognition mechanism to store and retrieve plans. Our approach is domain-independent and addresses all the requirements listed above: the action and plan representation are generic in nature, plan library (i.e. the plan memory) is grown at the same time as recognition is attempted, and search is limited to the most similar prior plans through indexing. The memory module employs a flexible matching algorithm for sequences of actions that can deal with noise, while exhibiting desirable qualities of a memory-based application: efficiency, retrieval of relevant items and content-addressability.

Memory-Based Plan Recognition

A Generic Memory Module for Events

Remembering past experiences is an essential characteristic of any intelligent system. Such experiences enable the system to solve similar problems - by adapting previous solutions - and to avoid unwanted behavior - by detecting potential problems and trying to side-step them.

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 propose to separate the memory functionality from the system and build a *generic* memory module that can be attached to a variety of applications in order to provide episodic 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.

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.

These requirements largely overlap with those of plan recognition algorithms.

Episode 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 that a generic episode have three dimensions: **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 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).

The idea of having different indices for episodes based on the different kinds of information they encode is not new - Chef (Hammond 1986) indexed plans both by their goals and by their failures. We extend this idea by defining three *generic dimensions* for episodes and show that retrieval along one or more of these dimensions 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 a 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 is the Knowledge Machine (KM), an expressive frame-based representation and reasoning mechanism (Clark and Porter 2001).

Memory Organization and Retrieval

Episodes are stored in memory unchanged (i.e. no generalization) and are indexed for retrieval. We have adopted a multi-layer indexing scheme similar to MAC/FAC (Forbus, Gentner, and Law 1995), (Börner 1994) 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)).

Incremental Retrieval

The fact that data is presented incrementally seems to increase retrieval time as memory needs to do retrieval after the presentation of each new stimulus. However, incremental data reduces the size of each query. Humans are good at dealing with continuous streams of stimuli and employing expectations to focus attention and guide recognition. The question we address here is: can we devise such an algorithm for an episodic memory?

This idea has been put forth before (Schmidt, Sridharan, and Goodson 1978; Schank 1982) and has been applied in areas like dialogue processing (Grosz and Sidner 1986; Litman and Allen 1987) and plan recognition (Schmidt, Sridharan, and Goodson 1978). The sequential structure of events helps constrain the type of expectations a system might form to just the next event(s) (its type and possibly its description).

Algorithm 1 Incremental-retrieve algorithm

```
candidates $\leftarrow$ []
observed-actions $\leftarrow$ []
while there are observed-actions left do
  curr-action $\leftarrow$ pop (observed-actions)
  new-candidates $\leftarrow$ retrieve(current-action)
  for all episode $\in$ new-candidates do
    // if it is not in candidate-episodes yet
    if episode $\notin$ candidates then
      // compare it to stimuli seen so far
      synchronize-candidate(episode, prior-actions)
    end if
  end for
  for all candidate $\in$ candidates do
    candidate-match $\leftarrow$ match(curr-action, candidate)
    // record whether it matched or not
    candidates $\leftarrow$ update-candidate(candidate-match)
  end for
  // re-rank candidate-episodes
  candidates $\leftarrow$ sort(candidates)
  // incremental predictions are now available
  prior-actions $\leftarrow$ prior-actions $\cup$ current-action
end while
result $\leftarrow$ sort(candidates)
// return final predictions
return first-n (*MAX-RETRIEVED*, result)
```

Our incremental retrieval algorithm 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 first level 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). Newly generated hypotheses need to by 'synchronized' (i.e. matched) with the stimuli seen so far, which is done using the synchronize-candidate routine. Hypotheses are then semantically matched to the new stimulus and then 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. At this point, incremental predictions can be generated based on the current set of hypotheses.

Most measures of semantic similarity compute similarity between two objects with respect to one of them. This does not deal well with noise, as the superfluous actions in the reference episode decrease the overall similarity score. The similarity metric used in this paper 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 $E_1 \sim E_2$ represents the isomorphic mapping from E_1 to E_2 (computed by the semantic matcher); t_i represents the isomorphic relation between a vertex in the E_1 graph and its corresponding counterpart in E_2; $score(t_i)$ measures how well the two vertices match and is a number between 0 and 1 provided by the matcher, and $|E_2|$ is the size of E_2 (i.e. number of triples).

Retrieval Complexity 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 remindings explored for a new stimulus (N). This situation is encountered when an identical episode is found in memory, therefore all the actions of the new episode match perfectly against that in memory. The worst case complexity is $O(Ns^3)$, when, at every step, memory explores the maximum number of episodes allowed (N) and each of them does not align with the observed action, so all their events are matched against. Our experiments showed that this case rarely occurs 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.

Experimental Evaluation

We evaluated out approach on a plan recognition task on two corpora: the Linux Plan Corpus (Blaylock and Allen 2004) and the Monroe Plan Corpus (Blaylock and Allen 2005a) and compared it to a statistical approach (Blaylock and Allen 2005b).

The Plan Corpora

The Linux plan corpus (Blaylock and Allen 2004) was gathered from human Linux users from the University of Rochester, Department of Computer Science. Users were given a goal like `find a file with 'exe' extension` and were instructed to achieve it using simple Linux commands (no pipes, no `awk`, etc.). 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[1], involving 19 goal schemas and 48 action schemas (i.e. Linux commands).

[1]Because some users were not able to judge this correctly, there are still a number of failed sessions and, therefore, data is noisy.

The Monroe corpus (Blaylock and Allen 2005a) consists of stochastically generated plans in the domain of emergency response. The plans have been randomly 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 schemas and 30 action schemas.

Experimental Setup

We performed a 10-fold cross-validation on each of the two corpora by dividing the set of plan sessions into 10 equal-sized subsets, and using 9 of them for training and the 10th for testing. The set of test plan sessions was presented to memory one action at a time, predictions being generated after each action from the top 4 most similar prior episodes retrieved by memory.

We limited the maximum number of explored remindings generated after each action is observed to 5 for both domains. All observed plans have been stored in memory, no deletion policy being implemented.

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.

An important characteristic of incremental recognizers is the **convergence point (CP)** - how soon they start making the *same* correct prediction in the unfolding of a plan. 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.

We compare these results against a statistical approach using a bi-gram model (Blaylock and Allen 2005b) that used the same corpora.

Besides accuracy, we are also interested in how the memory mechanism performs in terms of efficiency of retrieval as the memory size grows. We measured the retrieval effort for each prediction, both in terms of number of actions matched, as well as a percentage of the total number of stored episodes.

Experimental Results

Experimental results are reported in Figure 1. The precision and recall 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.) This is in contrast to (Blaylock and Allen 2005b) where predictions are made only if confidence is above a certain threshold.

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)). Precision 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.

Although we compute the similarity between the observed plan and some prior plans and use it to rank predictions, we chose not to implement confidence thresholds, thus generating predictions at each recognition step. The variability of the plans in our corpora makes the similarity measure not a good candidate for the confidence thresholds.

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.

Memory Performance Figure 3 plots the number of explored actions per recognition session versus the number of total episodes observed (which is also the number of stored episodes since all episodes are stored in memory). This number grows fast as memory develops but at a much slower pace after memory has matured. Please note that we measure the number of events (not of episodes) matched, since the retrieval is incremental.

Although the theoretical complexity of the retrieval algorithm is $O(s^3)$, in practice only a fraction of that are explored. After all training data has been observed in the Linux domain memory explores about 120 events, which represents under 11% of the number of episodes predicted by the theoretical worst case. The corresponding number for the Monroe domain is 558 episode, about 13% of the worst case prediction.

Extensions to the Current Approach

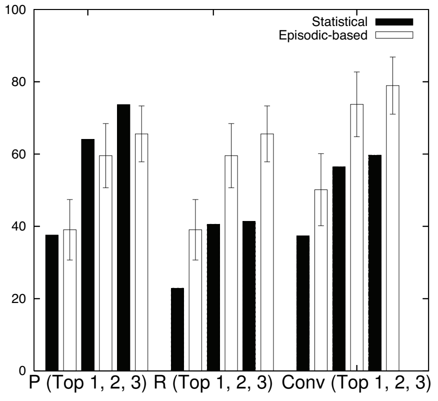
A logical next step is to extend our algorithm to predict also goal schema parameters, thus achieving full plan recognition. Mappings between parameters of observed actions and prior plans are already provided by memory as a result of the matching process, but an adaptation step is needed to ensure consistency. Preliminary results with parameter recognition are encouraging (Tecuci 2007).

A mature memory should be selective in what it stores. A good deletion policy should reduce memory size, without negatively affecting performance. Such a policy could be based on the similarity of a new episodes with past ones, as well as on the success or failure when using a retrieved episode.

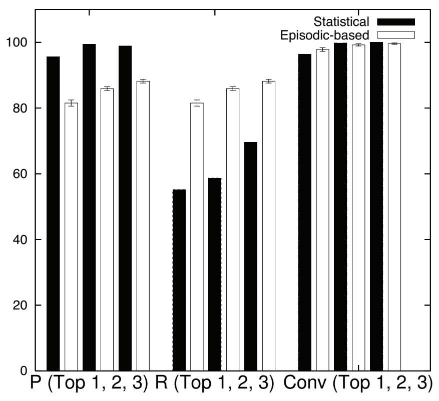
In complex domains it is likely that an agent will carry on multiple plans at the same time, interleaving their actions. A

	Linux				Monroe			
N-best	1	2	3	4	1	2	3	4
P, R (%)	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.

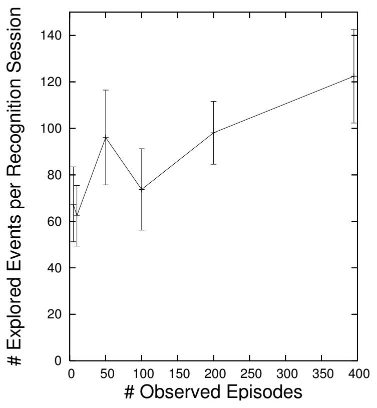


(a) Linux corpus.

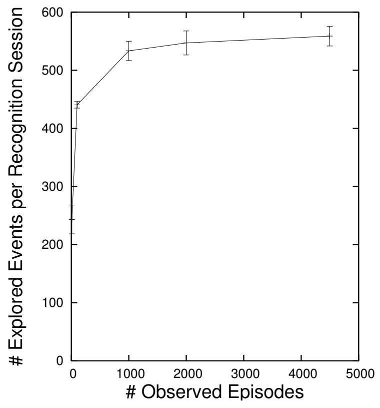


(b) Monroe corpus.

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



(a) Linux corpus.



(b) Monroe corpus.

Figure 3: Number of explored actions per recognition session.

recognizer will have to be able to deal with these different plans unfolding at the same time and still perform recognition on such data. The proposed memory based plan recognition lends itself easily to this task. Episodic-based plan recognition is already able to entertain multiple hypotheses at the same time.

Related Work

Related approaches include the case based plan recognition of (Cox and Kerkez 2006) which is based on state indexing. To alleviate the complexity explosion of state space, the observed actions are used to compute an abstract planning state, which is used as an index into plan memory. Unlike traditional plan recognizers, both this and our approach can deal with incomplete plan libraries, growing these libraries as recognition proceeds, effectively learning from observations. They both employ multi-level indexing schemes.

(Cox and Kerkez 2006) require that the observer has the ability to record intermediate planning states and need a complete model of the planning domain including consequences of actions. Our approach does not require complete domain knowledge and its matching algorithm can take advantage of as much or as little domain knowledge is available.

Another important difference compared to case-based reasoning approaches is that memory builds a multi-functional plan library that can be used for tasks other than plan recognition. For example, plans can be generated for a given goal, goals can be classified into solvable or not, and users proficiency at a task can be assessed by observing them execute that task.

Conclusions

In this paper we have proposed a memory-based approach to goal schema recognition that uses a generic memory module to store and retrieve plans. We showed that its incremental retrieval algorithm is efficient and scalable.

We evaluated it against a statistical approach on two domains and found that it achieves similar performance, but offers some important advantages.

Due to the generic nature of the memory module and of the retrieval algorithm, this approach should be easily portable to new domains. The memory structure built by the plan recognizer as well as its retrieval algorithm are multi-functional and can be used for other purposes like plan generation or prediction of a plan outcome.

References

Albrecht, D. W.; Zukerman, I.; Nicholson, A. E.; and Bud, A. 1997. Towards a Bayesian Model for Keyhole Plan Recognition in Large Domains. In Jameson, A.; Paris, C.; and Tasso, C., eds., *Proceedings of the Sixth International Conference on User Modeling (UM97)*, 365–376.

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.

Börner, K. 1994. Structural Similarity as Guidance in Case-Based Design. In *EWCBR '93: Selected papers from the First European Workshop on Topics in Case-Based Reasoning*, 197–208. London, UK: Springer-Verlag.

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.

Cox, M. T., and Kerkez, B. 2006. Case-based plan recognition with novel input. *Control and Intelligent Systems* 34(2):96–104.

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

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.

Grosz, B. J., and Sidner, C. L. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics* 3(12):175–204.

Hammond, K. J. 1986. CHEF: A Model of Case-Based Planning. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, 267–271.

Huwer, S.; Smart, B.; and Cairns, A. 1995. Domain Independent Plan Recognition. http://citeseer.ist.psu.edu/huwer94domain.html.

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.

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.

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

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.

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.