

# Building Plans for Household Tasks from Distributed Knowledge

Chirag Shah\* and Rakesh Gupta

Honda Research Institute USA, Inc.

800 California Street, Suite 300

Mountain View, CA 94041

chirag@cs.umass.edu, rgupta@hra.com

## Abstract

To accomplish a household task, an autonomous system needs a plan with steps. It is desirable to derive this plan dynamically instead of pre-coding it in the system. In this paper, we find a plan by using common sense knowledge collected from volunteers over the web through distributed knowledge capture techniques. This knowledge consists of steps for executing common household tasks.

We first pre-process the data with part-of-speech (POS) tagging to identify the actions and objects in the steps in all available plans for the task. We then determine the order of the steps to accomplish the task using discriminative as well as generative models. For the discriminative approach, we cluster the plans using hierarchical agglomerative clustering and choose a plan from the biggest cluster. In the contrasting approach, we make use of generative Markov chain techniques. Using human judgments, we show that the generative model with the first order Markov chain has the best performance. We also show that environmental constraints can be incorporated in the generated plans.

## 1 Introduction

The long term goal of our work is to make indoor mobile robots that work in environments like homes and offices more intelligent through common sense. Mobile robots in homes and offices will be expected to perform tasks within their environment to satisfy the perceived desires and requests of their users. In order to meet these needs, we want to endow these robots with some knowledge to use as the basis to accomplish these tasks. Examples of household tasks include making coffee, washing clothes, and cleaning a spill.

Common sense does not require expert knowledge, and hence it may be gathered from non-specialist net citizens (netizens) in the same fashion as the projects associated with the rather successful *OpenMind Initiative* pioneered by David Stork [Stork, 1999]. In this paper, we focus on how we can use the knowledge collected about task steps in the OpenMind Indoor Common Sense project (OMICS) [Gupta and

Kochenderfer, 2004] to populate an initial robot knowledge base with default household task plans.

In the work reported here, we show two approaches to find a plan for a given task. Our first approach is *discriminative*, where we perform hierarchical agglomerative clustering of the plans before choosing one. Our second approach is *generative*, where we make use of the first order Markov chains. We start with all the plans from OMICS database for the selected tasks and capture the sequence information between steps using the first order Markov chains. This allows us to combine multiple plans with interleaved chains into a generative model from which an appropriate plan can be derived. We also experimented with capturing the globally optimal sequence of steps.

Figure 1 shows an outline of different techniques. Technique two is discriminatory, and techniques three, four and five are generative. All these techniques are compared to the baseline technique of selecting a random plan (technique one). We performed experiments with human subjects and used statistical significance tests to compare these techniques.

The rest of the paper is organized as follows. The following section discusses the related work. We then give examples of data in our knowledge base and describe preprocessing to extract action-object pairs. Section 4 discusses discriminative approach along with the clustering technique to find the largest cluster of plans reflecting consensus. We then describe the generative approach, for capturing the sequence information using the first order Markov chains to generate locally and globally optimal plans. We also show how the generative approach can be used to generate a plan with environmental constraints. The results and analysis are presented in section 6. We finally conclude our paper with some pointers to the future work.

## 2 Related Work

In the past, expert systems have been used to encode the steps for accomplishing a task algorithmically [Waterman, 1986]. A key component was the capture of human expert knowledge using a laborious manual process. However, not everything that humans learn is taught by the experts. Most of the activities of our day-to-day life are learned by observations and experience – by looking at other non-experts (e.g. tying shoe laces).

---

\*Currently at University of Massachusetts, Amherst, MA

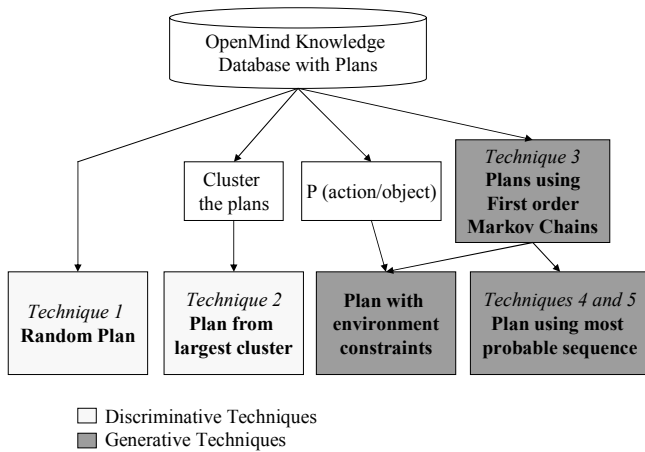


Figure 1: outline of proposed approaches.

Human actions can be analyzed on a variety of different levels. At lower level, execution of an action can be controlled by a motor response schema of sensory motor mapping [Schmidt, 1975]. For higher levels of action control, concepts such as scripts [Schank and Abelson, 1977] and memory Organization Packets (MOP) [Schank, 1982] have been proposed to represent the organization of well-learned activities such as going to a restaurant or visiting a doctor for surgery. When a MOP is activated, only one of the steps is generally carried out at a time, but steps can sometimes be combined with other activities (for example, one can read while waiting at doctor’s office).

Between these low and high level extremes lies a whole range of well-learned activities like making breakfast, cleaning one’s teeth, starting a car, dressing and so on. In these situations, the steps are represented as discrete units and their parallel execution (although possible) rarely occurs. Thus in toothbrushing, one can delay or do something between putting toothpaste on the toothbrush and brushing one’s teeth, it is not generally advisable.

At this mid-level, Cooper and Shallice [2000] presented a computational model for selection of of steps for routine tasks based on competitive activation within a hierarchically organized network of action schemas. Their activation model for sequential step selection was based on the Contention Scheduling theory of Norman and Shallice [1980]. This model was demonstrated in the routine task of preparing coffee. Under normal functioning, the model was able to generate a sequence of simple actions (pick up spoon, dip spoon in sugar bowl etc.) culminating in a drinkable cup of coffee.

According to Rasmussen *et al* [1983], human activity in such routine tasks is oriented towards a goal and controlled by a set of rules which have proven successful in the past. The sequence of steps is typically derived empirically during previous occasions, communicated from another person’s knowhow or a cookbook recipe. We are interested in capturing such procedures for household tasks.

In contrast, literature and work in AI planning [Weld, 1999] falls under goal controlled exploratory behavior category. Here attempts are made to reach the goal using knowl-

edge of different plans and a successful sequence is selected. Such planning is complementary to our work and is critical during execution of these tasks.

One source of common sense knowledge is the web. For instance, web-sites such as *eHow*<sup>1</sup> list the steps to perform activities<sup>2</sup>. Intel developed a system called *Probabilistic Activity Toolkit (PROACT)* to build activity models [Philipose *et al.*, 2003]. They automatically identified activities by observing the objects involved in the activity [Perkowitz *et al.*, 2004]. They found the relevance of various terms to a given activity from the web. For instance, the word *cup* is highly related to activity *making tea* because *cup* occurs frequently on the web-pages about *making tea*.

This idea of using the web as the information source is very attractive. However, while extracting knowledge from the web, one has to deal with high variance and noise in web information and large documents. We have better information on steps for household tasks from Open Mind Indoor Common Sense (OMICS) database. In OMICS, volunteers are prompted with household tasks and asked for steps to accomplish such a task. We still have to extract semantic information from the steps, as well as deal with issues of noise and consistency in the data.

### 3 Semantic Data Extraction

There are more than 150 tasks in OpenMind database like *making coffee*, *cleaning the floor*, *washing clothes* for which plans have been entered by the users. Each task in our database consists of a number of plans and each plan consists of a sequence of instructions. A sample plan from the database is shown below:

```

Task: wash clothes
Steps: collect clothes
       move to washing machine
       place clothes in washing machine
       add detergent to clothes
       close lid of washing machine
       start washing machine
  
```

Our objective is to make use of these plans for the given task to build a model. From this model, we can extract a plan, or derive a plan (which may not be present in the original set of plans), or generate a custom plan based on environmental constraints. In this section, we further describe how this data is pre-processed by extraction of action-object pairs.

#### 3.1 Extracting action-object pairs

We extract the action and object from a given step for better processing downstream. For extracting action-object pairs we first parse the instruction with Brill’s part-of-speech (POS) tagger [Brill, 1992]. We then identify the first verb as the action. If the verb is followed by a preposition, we combine the preposition with the action. Finally, we identify the first noun phrase as the object of the action. Since most of the steps in the tasks are instructional in nature, this simple procedure performs surprisingly well. The result of parsing the above plan is shown below:

<sup>1</sup><http://www.ehow.com>

<sup>2</sup>These activities correspond to our tasks.

```

Task: wash clothes
Action object pairs for steps:
  collect, clothes
  move to, washing machine
  place, clothes
  add, detergent
  close, lid
  start, washing machine

```

We are also interested in finding how the objects are related to various actions for every task. We, therefore, find the following conditional probability distribution for every task.

$$P(action|object) = \frac{f(action, object)}{f(object)} \quad (1)$$

where  $f(action, object)$  is the number of times  $action$  occurs with  $object$  and  $f(object)$  is the number of times  $object$  occurs in the given task across all the plans.

## 4 Discriminative Approach

A simple way of coming up with a plan is to select a random plan from the set of the plans in the database. This forms our baseline. In the discriminative approach, we select a plan from a subset representing the majority consensus. Majority consensus is reflected by the biggest cluster of plans in the task.

We perform hierarchical agglomerative clustering where we group similar plans, and merge similar groups into larger groups [Salton, 1989]. For each task, we find the similarity between two plans  $p_i$  and  $p_j$  as the following.

$$Sim(p_i, p_j) = \frac{len(largest\ matching\ sequence)}{len(p_i)len(p_j)} \quad (2)$$

where  $len(p_i)$  is the individual number of steps in the plan  $p_i$  and  $len(largest\ matching\ sequence)$  is the number of common steps in plans  $p_i$  and  $p_j$ . When individual plans are compared we use the similarity criteria as given in the equation above, but when clusters are compared, we use group average to compare clusters on the basis of average similarity. In our system, we can specify the desired number of clusters. We empirically found that plans for our tasks fall in five or fewer categories. Therefore, we choose to have at the most five clusters of plans for each task.

We have found these clusters to correspond to distinct techniques for accomplishing the task. For example for *making coffee* task, the different clusters correspond to using coffee maker, instant coffee, and espresso machine. Since a majority of people entered plans to make coffee using the coffee-maker, that cluster was the largest. After clustering, we randomly select a plan from the largest cluster.

## 5 Generative Approach

So far we have selected a plan from the original set of plans. In this section, we describe how we can instead *generate* a plan. We propose to construct a model for each task, called *Task Model*, using the first order Markov chains [Rabiner, 1989]. The primary motivation for the Markov chain is the inherent sequential nature of steps in a given plan. We model each plan as a first order Markov chain, where each step depends on the previous one with no hidden states.

### 5.1 Constructing the Task Model

All the plans are encapsulated between the start and end steps. To build the Task Model, we link the first step to the start state with probability 1.0. We keep linking all other steps in order with probability 1.0, ending in the end state. We repeat this for all plans in the database with appropriate probability computations. For example, we initially have a transition from A to B with probability 1.0. When we get a new transition from A to C, we create an additional link from A and recompute the probabilities of links from A as 0.5 each. It is possible for the same instruction to occur more than once in the model. For instance, in case of *wash the clothes* task, *open the lid* can occur before putting the clothes in the washing machine and after the washing is done to take out the clothes.

After executing the above procedure for *washing clothes* task we combine all these links to generate a graph. All these states are represented in the graph as nodes and joined according to their transition probabilities. A sample construction is shown in figure 2.

There are various advantages of building such Task Model. Firstly, we do not have to store all individual plans for a task. We store a model for each task which makes the storage space linear in the number of steps, rather than linear in the number of total plans in the database. Secondly, models evolve with technological advancements and encompass new information as newer plans for tasks come up. When we have a new version of the database of plans entered by non experts, we can either generate a new model from the data using the same procedure described here or update the existing model with new probabilities and new states. Finally, having a model allows us to generate consensus plans and more complex plans that do not exist in the database or use available objects in the environment.

The following subsections discuss details of how the generated Task Model can be used to derive a plan with the different generative techniques. We first describe technique three, followed by techniques four and five using the most probable sequence.

### 5.2 Plans using the first order Markov Chains

We have already captured step sequence information in our Task Model. To derive a locally optimal plan, we go through the most probable sequence of steps. At every state starting from start, we choose the next state as the one with the highest probability. The state at time  $t$  is found using the following equation:

$$NextState(t) = \arg \max_{s_i} p(s_i|s_j) \quad (3)$$

where  $t$  is the current time step,  $s_j$  is the state at time  $t - 1$ , and  $s_i$  are all the successor states of  $s_j$ . To avoid cycles, we remove all the incoming links to the step that we visit<sup>3</sup>. A

<sup>3</sup>Note that this will not prevent us from using the same instruction again in the plan generation. This is because the same instruction may be represented by more than one states. For instance, "close the lid" instruction may be represented by two states - one that occurs before "put the clothes" state and the other that occurs after "take out the clothes" state.

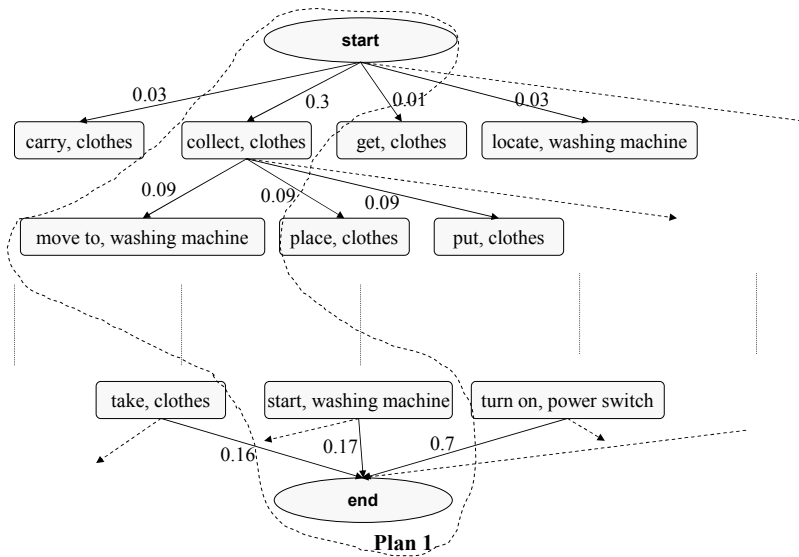


Figure 2: Portion of a Task Model. The dashed outline shows one of the plans.

sample output for the task *wash the clothes* is given below. It is in the format *step*  $\rightarrow$  *transition probability*  $\rightarrow$  *step*.

```

start -> 0.36 -> get, clothes
get, clothes -> 0.3333 -> locate, machine
locate, machine -> 0.4285 -> move to, machine
move to, machine -> 0.2727 -> fetch, clothes
fetch, clothes -> 0.5 -> open, machine
open, machine -> 0.8 -> put, clothes
put, clothes -> 0.5294 -> add, detergent
add, detergent -> 0.5 -> close, machine
close, machine -> 0.2857 -> start, machine
start, machine -> 0.6666 -> end

```

Putting these steps together, we get a plan. The evaluation of its *goodness* is discussed in the next section.

### 5.3 Generating globally optimal plans

So far we have considered dependency of a state on the previous state to reduce the computation. For global optimality, we consider the relation of present state to all the previous states. Therefore, our formulation of selecting the next state as given in equation (3) changes to the following:

$$NextState(t) = \arg \max_i p(s_i | s_1, s_2, \dots, s_{i-1}) \quad (4)$$

where  $t$  is the time step on which we are trying to find the best state,  $s_1, s_2, \dots, s_{i-1}$  are the states that occurred till  $t-1$  and are linked through a path in the model, and  $i$  iterates over all the possible states at step  $t$ .

To implement this idea, we computed the probability along different possible sequences from start to end in our Task Model. The sequence that gives the maximum probability is chosen for the plan. In technique 4, we select a random sequence if there are multiple choices with the largest probability. In technique 5, we select the shortest of these sequences if there is a tie. The results of all these techniques along with the analysis are given in the Results section.

### 5.4 Plan with environmental constraints

In a real-time system, we want to consider the environmental constraints while generating a plan. Such constraints may be provided by the user or obtained by the system using sensors. Sensors may provide information about what objects are available in the environment. However, our data is represented in the units of action-object pairs. To convert the restrictions on objects to action-object pairs, we assume that the most likely action is one that occurs most frequently with the object. We make use of the  $P(action|object)$  probabilities that we found earlier in equation (1). For handling a constraint to use a given object in the task, we find the most probable action to be associated with that object:

$$\arg \max_{action} P(action|object) \quad (5)$$

In plan generation, we choose children of the current step with restrictions. If there is more than one choice, we choose the one with highest probability. If there are no constraints for a step, we choose the one with the maximum probability on its link. In order to avoid cycles, we remove all the incoming links at each visited step.

It is important to note here that even though we are associating the observed objects to their most likely actions, these actions are not forced in the plan. It is possible that the plan generation process neglects such steps to be consistent with the rest of the steps. A sample output for the *washing clothes* task with restriction to make use of *water*, *clothes*, and *washing machine* is given below:

```

Found restriction: "feed, water" with
probability 0.2
Found restriction: "put, clothes" with
probability 0.32
Found restriction: "start, washing machine"
with probability 0.15
The plan:
start -> 0.36 -> get, clothes

```

```

get, clothes -> 1 -> put, clothes
put, clothes -> 1 -> feed, water
feed, water -> 1 -> feed, detergent
feed, detergent -> 1 -> set, timings
set, timings -> 1 -> start, washing machine
start, washing machine -> 0.6666 -> end

```

This plan is different from the one that we derived earlier without any constraints. Because of this method’s bias towards choosing the step that fulfills one of the restrictions, the corresponding transition probabilities are set to 1.0.

## 6 Results and Analysis

We selected 105 tasks, each with at least 25 plans in the OMICS database and compared the following techniques:

1. Baseline: a plan selected randomly.
2. Discriminative approach: a plan selected from the largest cluster.
3. Generative approach: a plan generated from the corresponding Task Model.
4. A plan generated from the Task Model by considering the probability of the whole sequence.
5. A plan generated from the Task Model by considering the probability of the whole sequence, and choosing a plan with minimal length if there is a tie.

All these techniques made use of the same data and preprocessing. In order to compare these techniques<sup>4</sup>, we based our evaluation on the following criteria:

1. *Completeness.* Plan for a task should be complete. A plan for cleaning the floor, which spilled the water and the soap on the floor but did not mop it to dry, would be an incomplete plan.
2. *Correct sequence.* The sequence of steps should be consistent. A plan, which poured coffee from carafe into a mug before adding water to coffee-maker, would be rated low using this criteria.
3. *It should make sense.* For example if a coffee making plan used both a coffee-maker and instant coffee, it would be rated low.
4. *Not too many details of interaction with objects.* Given two plans with same number of objects, a plan with higher level description is preferable to low level detailed instructions. For making coffee for the step of adding filter to coffee-maker,  
 Preferable: add filter  
 Less preferable: find filter, take one filter, add filter

We asked 10 users to rank five plans for each of 105 household tasks. This evaluation took users about 2 hours to fill. For every task we averaged over the 10 evaluations for each of the 5 plans. The maximum score for a technique for a given

<sup>4</sup>The implementation that made use of the environmental constraints was difficult to judge and evaluate; therefore, we did not include it in our evaluation. However, it is a special case of technique 3 and inherits advantages of that technique.

Technique	Name	Avg. score	Overall ranking
1	Baseline	267.63	5
2	Discriminative	257.45	3
3	First order Markov chain	245.90	1
4	First order Markov chain with full sequence	250.09	2
5	First order Markov chain with full shortest sequence	266.00	4

Table 1: Results of the user judgments

task would be 5 (the worst) and the minimum score would be 1 (the best). We then add these scores for all the 105 tasks for each technique. Table 1 summarises the results, from which we make the following observations:

- Selecting a plan randomly from a given set of plans (baseline) gives the worst performance. This indicates that if we rely on the *knowledge* of just one person or one source, then we have high likelihood of a bad plan.
- Technique 2, which is a discriminative approach, does better than the baseline. A random plan from the consensus cluster is better than a random plan.
- Technique 3 using first order Markov Chains does the best. This is a generative model that *learns* the *knowledge* from the given data and generates a plan. This approach is attractive for a number of reasons. First, it considers a step as a unit instead of a plan, thus not confining itself to a particular plan like the first two techniques. Second, it is able to remove some noise and spurious data through the process of learning. Third, it captures the consensus at the level of steps and their sequence.
- Techniques 4 and 5 do not do as well as 3. We believe the main reason is the lack of sufficient number of plans required to perform the inferencing on long sequences of steps.

The scores reported in table 1 gives an idea of the performance of various techniques. To compare these techniques, we performed a paired two-tailed *t*-test. This test determines if the outcome of two different techniques come from the same distribution. If the original distributions are significantly different, then the one that provides better results is said to be performing significantly better over the other. This statement about significance is associated with a level of confidence. The *p*-values for different techniques are given in table 2. We can see that with the confidence interval of 95%, techniques 3 (locally optimal plan) and 4 (globally optimal plan) perform statistically significant over the baseline. Both of these techniques are based on the first order Markov chain generative models.

## 7 Conclusion and Future Work

In this paper we discussed the problem of selecting or extracting a plan to perform a task from the given set of plans that have been collected by distributed knowledge capture techniques. We proposed a discriminative as well as generative

Technique	$p$ -value
2	0.0838
3	0.0004
4	0.0036
5	0.7866

Table 2:  $p$ -values from paired two-tailed  $t$ -test. The tests were done by making pairs of ranks given by baseline and the technique specified in the first column.

approaches to derive a plan. We used hierarchical agglomerative clustering for the former approach and first order Markov Chains for the latter one.

Although the discriminative approach gave better results than the baseline, the generative approach did even better. For the generative approach we first construct a Task Model from the available plans. We use First Order Markov Chains to find the most likely sequence of steps. We can also incorporate information about available objects in the plan generation process. Our experiments showed reasonable plans as outputs in discriminative as well as generative models. Since the goodness of these results cannot be measured objectively, we used human subjects to evaluate our results. We also showed that the differences among techniques were statistically significant.

In our plan representation and generation processes, we perform shallow Natural Language Processing (NLP) for finding action-object pairs. Further NLP can improve the results. For instance, we can find the synonyms of the actions as well as objects and merge them to simplify the Task Model before generating the plans. In future work, we also plan to automate the specification of environmental constraints for objects in the environment using RFID or vision sensors.

## 8 Acknowledgments

This work was done while Chirag Shah was a summer intern at Honda Research Institute USA, Inc. Thanks are also due to anonymous reviewers and the users of the Open Mind Indoor Common Sense web-site for their data and feedback.

## References

[Brill, 1992] Eric Brill. A simple rule-based part-of-speech tagger. In *Proceedings of ANLP-92, 3rd Conference on Applied Natural Language Processing*, pages 152–155, Trento, IT, 1992.

[Cooper and Shallice, 2000] Richard Cooper and Tim Shallice. Contention scheduling and the control of routine activities. *Cognitive NeuroPsychology*, 17(4):297–338, 2000.

[Gupta and Kochenderfer, 2004] Rakesh Gupta and Mykel Kochenderfer. Common sense data acquisition for indoor mobile robots. In *Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, July 25-29 2004.

[Norman and Shallice, 1980] D. Norman and T. Shallice. *Attention to Action: Willed and automatic control of behavior*, pages 1–18. Plenum Press, New York, 1980.

[Perkowitz *et al.*, 2004] Mike Perkowitz, Matthai Philipose, Kenneth Fishkin, and Donald J. Patterson. Mining models of human activities from the web. In *Proceedings of the 13th Conference on World Wide Web*, pages 573–582. ACM Press, 2004.

[Philipose *et al.*, 2003] Matthai Philipose, Kenneth P. Fishkin, Mike Perkowitz, Donald Patterson, and Dirk Haehnel. The probabilistic activity toolkit: Towards enabling activity-aware computer interfaces. Technical Report IRS-TR-03-013, Intel Research Laboratories, November 2003.

[Rabiner, 1989] Lawrence R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

[Rasmussen, 1983] Jens Rasmussen. Skills, rules and knowledge: Signals, signs, and symbols, and other distinctions in human performance models. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13(3):257–266, May/June 1983.

[Salton, 1989] Gelad Salton, editor. *Automatic Text Processing: The transformation, analysis, and retrieval of information by computer*. Addison Wesley, 1989.

[Schank and Abelson, 1977] R. C. Schank and R. Abelson. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum Associates Ltd., Hove, UK, 1977.

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

[Schmidt, 1975] R. A. Schmidt. A schema theory of discrete motor skill learning. *Psychological Review*, 82(4):225–260, 1975.

[Stork, 1999] David G. Stork. The Open Mind Initiative. *IEEE Expert Systems and their Applications*, 14(3):19–20, May/June 1999.

[Waterman, 1986] D. A. Waterman. *A Guide to Expert Systems*. Addison Wesley, 1986.

[Weld, 1999] Daniel S. Weld. Recent advance in ai planning. *AI Magazine*, 20(2):93–123, Summer 1999.