Corpus-based, Statistical Goal Recognition*

Nate Blaylock and **James Allen**
Department of Computer Science
University of Rochester
Rochester, New York, USA
{blaylock,james}@cs.rochester.edu

Abstract

Goal recognition for dialogue systems needs to be fast, make early predictions, and be portable. We present initial work which shows that using statistical, corpus-based methods to build goal recognizers may be a viable way to meet those needs. Our goal recognizer is trained on data from a *plan corpus* and is then used to determine the agent's most likely goal based on that data. The algorithm is linear in the number of goals, and performs very well in terms of accuracy and early prediction. In addition, it is more easily portable to new domains as it does not require a hand-crafted plan library.

1 Introduction

Much work has been done over the years in *plan recognition*, which is the task of inferring an agent's goal and plan based on observed actions. *Goal recognition* is a special case of plan recognition, in which only the goal is recognized.

Goal and plan recognition have been used in a variety of applications including intelligent user interfaces [Lesh *et al.*, 1999], dialogue systems [Carberry, 1990b; Allen *et al.*, 2000] and machine translation [Alexandersson, 1995].

We are especially interested in applying goal recognition to dialogue systems, in order to aid natural language understanding and intention recognition. We do not intend to use a goal recognizer to *directly* recognize communicative intentions (the goals behind a user's utterance); rather, we intend to use it to identify a user's domain goals to quickly help narrow the search space for more costly intention recognition routines (e.g., [Lochbaum, 1998; Chu-Carroll and Carberry, 2000]).

This application places several demands on our goal recognizer:

1. **Speed:** Dialogues happen in real-time, and the system is expected to understand a user's utterance and generate a response in a short amount of time.
2. **Early/partial prediction:** We need accurate goal predictions very early on in the exchange, as the system needs this information to better respond to a user's utterance. If full recognition is not immediately available, the system needs at least partial information to allow it to act on the user's utterance.
3. **Portability**: We want to be able to rapidly port our dialogue system to new domains.

We present initial work in which we use corpus-based methods to build a goal recognizer. Our recognizer is fast (linear in the number of possible goals), makes early predictions, and is easier to port to new domains than many systems, as it does not require a hand-crafted domain plan library.

We first discuss the corpus-based approach we take in goal recognition. We then report on initial experiments that we have performed and discuss their results. Finally, we comment on related work and then discuss future directions for our work.

2 The Corpus-based Approach

The use of statistical methods, based on corpora, has revolutionized the field of Natural Language Processing over the past 10+ years. The method is as follows: one uses a corpus of data to train a statistical model which is then used to make predictions on future data. Seemingly simple methods have yielded good results in many areas of NLP.[1]

We apply a similar approach to the task of goal recognition. We use a *plan corpus* (a list of goals and the plans an agent executed to achieve them) to train statistical models which can predict an agent's goal based on an observed sequence of actions. As we show below, initial work shows several possible advantages over previous work on goal and plan recognition: recognition is fast (linear in the number of goals), robust (can handle unknown actions and plans) and does not require a hand-crafted plan library.

2.1 Recognition using N-gram Models

We define the task of goal recognition as follows: given an observed sequence of n actions so far $A_1, A_2, \ldots, A_n$, (which, for compactness, we will represent as $A_{1,n}$), find the most likely goal G:

$$G* = \text{argmax}\ P(G|A_{1,n}) \quad (1)$$

*This is a correction of the paper which appears in the IJCAI '03 proceedings.

[1] For a good review, see [Manning and Schütze, 1999].

Goal #	Goal Description	Sessions
1	Find a file named 'core'	12
2	Find a file that contains the word 'motivating' and whose name ends in '.tex'	11
3	Find a machine that has low (<1.0) load; AND determine if Oren Etzioni is logged into the machine named chum	5
4	Compress all large files (>10,000 bytes) in the Testgrounds subdirectory tree	2
5	Compress all files in the directory named 'backups' [Don't use *]	3
6	Find a large file (>100,000 bytes) that hasn't been changed for over a month	8
7	Find a file that contains less than 20 words	8
8	Find a laser printer in Sieg Hall that has an active print job	6
9	Find a Sun on that has low (<1.0) load; AND determine if Dan Weld is active on the machine named chum	2
10	Find a file of length 4 in neal/Testgrounds subdirectory	1
11	See if Dan Weld is logged in to chum	1
	Total	59

Table 1: Goals and their counts in the Unix corpus

Using Bayes' Rule, this becomes:

$$G* = \text{argmax } \frac{P(A_{1,n}|G)P(G)}{P(A_{1,n})} \quad (2)$$

Since $P(A_{1,n})$ is constant in the argmax, we can drop it:

$$G* = \text{argmax } P(A_{1,n}|G)P(G) \quad (3)$$

Using the Chain Rule, we can rewrite this as:

$$G* = \text{argmax } P(A_n|A_{1,n-1},G)P(A_{n-1}|A_{1,n-2},G) \ldots P(A_1|G)P(G) \quad (4)$$

These conditional distributions are very large and difficult to estimate, therefore, we make an n-gram assumption, i.e., we assume that an action A_i is only dependent on the goal G and the j actions preceding it ($A_{i-j,i-1}$). For a unigram model, we assume that A_i is independent of all other actions. In this case, our goal recognition equation becomes:

$$G* = \text{argmax } P(G)\prod_{i=1}^{n} P(A_i|G) \quad (5)$$

If we assume that A_i is independent of everything but G and A_{i-1}, we get a bigram model:

$$G* = \text{argmax } P(G)\prod_{i=2}^{n} P(A_i|A_{i-1},G) \quad (6)$$

We estimate $P(G)$, $P(A_i|G)$ and $P(A_i|A_{i-1},G)$ using a plan corpus. Then, for recognition, we first initialize our goal probabilities with $P(G)$ and make an initial prediction, based solely on these priors. Then, for each action we observe, we multiply each goal's score by the corresponding conditional n-gram probability, and make a new prediction.

This algorithm has the nice feature of compositionality — new observations produce conditional probabilities, which are simply multiplied with the previous predictions. This results in computational savings, since updates are linear in the number of goals. It also gives us a good model for early prediction (as desired for our dialogue system), since the model is based on actions observed so far and does not require all actions in the plan execution.

Goal Types	11
Goal Sessions	59
Action Types	22
Total Actions	412
Average Actions/Goal	7.0

Table 2: Statistics for the Unix corpus

3 Experiments

3.1 The Plan Corpus

We performed several experiments using Lesh's Unix plan corpus [Lesh, 1998]. The corpus was gathered from human Unix users (CS undergraduates) at the University of Washington. Users were given a task in Unix (a goal), and were instructed to solve it using a subset of Unix commands (no pipes, no `awk`, etc.) The students' commands and results were recorded, as well as whether or not they successfully accomplished the goal. There were 59 successful goal sessions which involved 11 different goals.[2] Table 1 shows the individual goals and the number of successful goal sessions for each.

We automatically removed unsuccessful commands, such as typos, from each execution. Remaining commands were stripped of arguments to a base command type form. This means our training set consisted only of action types (such as `ls`, `grep`, etc.), and did not consider flags or arguments. We hope to extend our model to incorporate this additional information in the future (see below).

Table 2 shows some statistics from the resulting plan corpus. There were 11 possible goals[3] and 22 different action types used in the goal sessions. On average, there were 7.0 actions per goal session.

[2] There was actually a twelfth goal, but it was essentially the same as goal 1, so we merged the two.

[3] These are actually goal schemas, since we do not recognize parameters.

We used cross-validation in testing each of the 59 test cases. Because of the small size of the Unix corpus, the training set was formed by removing only the test case from the training corpus for each test case.

3.2 Evaluation Metrics

As pointed out by Lesh [1998], there is a lack of agreed-upon metrics and benchmarks for reporting results for plan and goal recognizers. We use the following metrics to report our results as they measure the attributes we are seeking for a goal recognizer (as described above). A test case is the set of actions executed for a goal. For each test case, the recognizer makes an initial prediction, and then the actions are fed to it one by one, and it makes predictions after every action. Each metric is for a single test case. For reporting multiple test cases, the metric is averaged across all cases.

- *Accuracy:* The number of correct predictions divided by the total number of observed actions.
- *Converged:* Whether or not the final prediction was correct (i.e., whether the recognizer *finished* with the correct answer).
- *Convergence point:* If the recognizer converged, at which point in the input it started giving only the correct answer. This is reported as a quotient of the action number (i.e., after observing x actions) over the total number of actions for that case.[4]

3.3 Unigram model

For our first experiment, we trained unigram models on the data (based on Equation 5). To provide for unseen data, where $P(A_i|G)$ was 0, $P(A_i|G)$ was set to be a very small constant. This smoothing technique allows goal G to still remain possible, in case other evidence makes it likely, despite the fact that action A_i was not seen in relation to it in the training data.

With the unigram model, our recognizer achieved an accuracy of 55.4%, with 78.0% of the cases converging. For cases which converged, the average point of convergence was after 3.1 actions (out of an average of 7.6 actions).

While these were encouraging results, we wanted to see if we could do better.

3.4 Bigram model

In our next experiment, we used a bigram model (based on Equation 6), which encodes at least some ordering into the actions by considering both the current action and the preceding action. Our intuition was that this would give us better performance than the unigram model, because of this temporal information.

We prepend a special `start` action to the front of each execution, which handles the special case of Equation 6 in which $n = 1$, i.e., when we've only seen one action so far. This also encodes information about which actions tend to begin executions, which may or may not be correlated to the goal.

For the case where $P(A_i|A_{i-1}, G)$ equals 0, i.e., the bigram was never seen in conjunction with the goal, we use a unigram back-off model which uses the estimation $P(A_i|A_{i-1}, G) \approx P(A_i|G)$. As with the unigram model above, if $P(A_i|G)$ equals 0, we smooth this with a small constant.

Contrary to our expectations, the bigram model performed almost exactly the same as the unigram model, with an accuracy of 55.6% and with 78.0% of cases converging. For the tests cases which converged, they converged on average after 3.1/7.6 actions.

The fact that the bigram model performed about the same does not seem to be due to a lack of data. 89.8% of bigrams in the test set were seen at least once in the training data. The lack of improvement seems be tied more to the domain itself. The unigram model seems to do well, despite lack of temporal knowledge, due to the fact that some goals (such as goal 8, for example) are highly correlated with a single command (such as `lpq`), which immediately boosts the probability of the correct goal. Other goals (goal 1, for example), were not uniquely correlated to any one command, and predictions for these goals were wrong more often. Pairs of commands did not do any better at distinguishing these cases of confusion. For example, the use of `cd` followed by `ls` does not seem to uniquely pick out any goal any better than the individual correlations for `cd` and `ls`.

It is actually illustrative to look at a goal-by-goal breakdown of results. Table 3 shows convergence results for each goal type. For the 13 cases which didn't converge, 2 (for goals 10 and 11) were because the goal did not appear in the training data; and 7 are the recognizer confusing goal 1 for goal 6 and vice-versa. This seems mostly to be due to the fact that these two goals are almost indistinguishable by only bare commands. Without the flags, many sessions with these goals simply become strings of `ls` and `cd`, and ordering information does not discriminate any better either. We expect that bigram and other higher-order n-gram models would perform better on more typical plan recognition domains.

Goal#	Convergence	Competitors
1	6/12 (50.0%)	6:5,2:1
2	11/11 (100.0%)	none
3	4/5 (80.0%)	9:1
4	2/2 (100.0%)	none
5	2/3 (66.7%)	4:1
6	6/8 (75.0%)	1:2
7	8/8 (100.0%)	none
8	6/6 (100.0%)	none
9	1/2 (50.0%)	3:1
10	0/1 (0.0%)	6:1
11	0/1 (0.0%)	3:1

Table 3: Convergence by goal in the unigram experiment

[4] It is necessary to report the total number of actions as well. Because this statistic is only for the test cases which converged, it is possible that the average actions per session is different from that of the entire corpus.

Abstract Goal	Goals which Instantiate
Find a file with attributes	1, 2, 6, 7, 10
Find a machine with attributes	3, 9, 11
Compress files with attributes	4, 5
Find a printer with attributes	8

Table 4: Abstract goal types

	Unigram	Bigram	Abstract Goals
Accuracy	55.4%	55.6%	83.9%
Converged	78.0%	78.0%	100.0%
Conv. Point	3.1/7.6	3.1/7.6	1.4/7.0

Table 5: Results of the experiments on the Unix corpus

3.5 Goal Abstractions

In our final experiment, we defined an abstraction hierarchy for goal types. The abstract types and goals they encompass are shown in Table 4.

As we discussed above, a dialogue system needs to be able to reason quickly, accurately and early about a user's goals and intentions. If the user's specific goal cannot be determined, a dialogue system can often use partial information in formulating a response to the user. These abstract goal classes represent that partial information. If we are unable to determine a specific goal, the system can perhaps determine what the user's abstract goal is, and this may be enough to formulate a response at that point in the dialogue.

The probability of an abstract goal is calculated by simply summing the probabilities of each of its children. For this experiment, we use the same unigram model as before and perform an additional summing at each step to calculate the most likely abstract goal.

The recognizer predicted abstract goals very well, with 83.9% accuracy and 100.0% of the cases converging. The average convergence point was after 1.4/7.0 actions. Table 5 summarizes the results of all three experiments.

It is important to note that abstract goal recognition does not occur in competition to, but rather in conjunction with the specific goal recognition. Of course, it is best to recognize the specific goal, which our unigram model seemed to do fairly well, but we can also recognize the abstract goal earlier and more accurately. There may be many cases where it will be sufficient for the dialogue system to have just the abstract goal in order to help the user.

4 Discussion

These initial results are encouraging, but there still remain significant challenges for scaling up the system to sufficiently complex domains. In this section, we discuss our goal recognizer in light of the desiderata for dialogue systems we mentioned above. We then discuss several challenges that remain.

4.1 Desiderata

Speed Because it involves a simple probability lookup, our recognizer is linear in the number of goals. Speed is a big advantage to this approach. By comparison, logic-based reasoning recognizers like [Kautz, 1991] are exponential in the size of the plan library.[5] Several systems [Vilain, 1990; Lesh, 1998] improve on this time complexity, but at the expense of expressiveness.

Early/partial prediction Our recognizer was able to make correct predictions 3.1 actions through the input with 55.4% overall accuracy, and give correct abstract results 1.4 actions through the input with 83.9% overall accuracy. This early prediction is crucial to our domain of dialogue systems.

Most work does not report how early the recognizer makes correct predictions. Lesh [1998] simulates a task-completion agent, which, upon recognizing the user's goal, steps in to complete the task. He reports[6] a convergence point of 8.7 for an average plan length of 26.8 actions for the task of searching for a printer with attributes (4 predicates) and a convergence point of 4.9 actions for an average plan length of 17.9 actions for a restricted cooking domain, both with 100.0% accuracy since Lesh uses a 'strict consistency' approach. For most domains, however, (like the full Unix corpus), it is unclear if Lesh's system could make such early predictions. Because it is based on 'strict consistency', a goal hypothesis must become logically impossible before it can be ruled out. In fact, even in for the domains he reports statistics in, the recognizer rarely recognizes the user's *actual* goal exclusively, rather, based on the set of possible goals, it tries to recognize a *sufficient* goal that covers all possible goals. As Lesh points out, many domains do not lend themselves to the prediction of sufficient goals.

Portability The portability of our goal recognizer depends on the existence of a plan corpus. If a plan corpus exists or can be created for the new domain (see section below), all we have to do is use it to train models for the new domain.[7] On the other hand, most goal recognizers (e.g., [Vilain, 1990; Carberry, 1990a; Kautz, 1991; Charniak and Goldman, 1993; Paek and Horvitz, 2000]), require a complete, hand-crafted plan library in order to perform recognition, which can require a significant amount of knowledge engineering for each domain.

Hong's recognizer [Hong, 2001] only requires knowledge of plan operators, not the library, but it is unable to make early predictions, as it usually does not make end-goal predictions

[5] Granted, these systems are performing plan recognition and not just goal recognition, which makes the comparison unfair. However, very little work has been done on goal recognition in its own right, so plan recognizers are all we have to compare against.

[6] Lesh reports the average length of plan with and without the task-completion agent, which we used to extrapolate these convergence points.

[7] Models could also be trained for different individuals within a single domain by using a user- (or group-) specific corpus. With other approaches, user-specific models have greatly improved recognition (e.g., [Lesh, 1998]).

until after it has seen the entire executed plan. Lesh [1998] requires only knowledge of plan operators and domain goal predicates.

4.2 Challenges

Domain size With only 11 goals and 22 action types, the Unix domain is quite small, and it is unclear whether our recognizer would scale to larger domains. One immediate fault of our recognizer is that it does not handle parameterized goals. Each of the 11 goals above is treated as an atomic unit or goal schema without instantiated parameters. One straightforward way to handle parameters would be to treat a goal schema as an abstract goal, with each possible set of parameter instantiations as a separate, more specific goal. However, this would explode the number of goals and, in the case of 2 or more parameters, lead to a multiple-inheritance abstraction hierarchy, which is not supported by our current abstract goal score calculation model.

Hierarchical plans Another shortcoming of the current recognizer is that, although it handles goal abstraction, it does not handle hierarchical plans. Complex plans covering longer time-scales are less likely to be identifiable from a few observations alone (which tend to reflect more immediate subgoals). Ideally, we would want to recognize subgoals for partial results, even if we still cannot recognize the high-level goal.

Data collection In some domains (like operating systems), it may be possible to collect enough data from users to train a recognizer. In most domains, however, it will be infeasible to collect enough data on users solving goals in order to build effective statistical models. Furthermore, even it this data could be collected, the inner structure of the user's hierarchical plan would not be explicit from the data (i.e., we can only observe the primitive actions, not the subgoal structure that motivates the actions).

As the next step in our research, we plan to explore the use of AI planners to generate artificial plan corpora to be used for training. The approach we plan to take combines planning and Monte-Carlo simulation to generate plan corpora. The idea is to generate plans stochastically (allowing distributions over different aspects of the planning process, such as the goals, situations and action decompositions). By combining "context-free" Monte-Carlo simulation techniques with richly context-dependent planning algorithms, we hope to obtain a corpus that captures likely user behavior. In addition, this generated corpus has the big advantage that the subgoal hierarchy that generates the observed actions is also known.

5 Related Work

As mentioned above, [Vilain, 1990; Lesh, 1998] improve speed over [Kautz, 1991], but do so at the expense of expressiveness. Also, these goal recognizers are typically not able to make early predictions, as they are unable to distinguish between consistent goals, even if one is more likely than the other.

There are several lines of research which incorporate probabilistic reasoning into plan and goal recognition. [Carberry, 1990a] and [Bauer, 1994] use Dempster-Shafer theory and [Charniak and Goldman, 1993], [Pynadath and Wellman, 1995], and [Paek and Horvitz, 2000] use Belief Networks to represent the likelihood of possible plans and goals to be attributed to the user. All of these methods, however, require a complete plan library as well as the assignment of probability distributions over the library.

[Appelt and Pollack, 1991] and [Goldman *et al.*, 1999] cast plan recognition as weighted abduction. However, this also requires a plan library and the acquisition of weights for abductive rules. Abduction is also computationally hard, and it is unclear whether such routines would be fast enough for complex domains.

Probably the closest work to ours is [Albrecht *et al.*, 1998], which uses a dynamic belief network to do goal (quest) recognition in a multi-user dungeon (MUD) domain. The belief network takes into account actions, locations, and previous quest in recognizing the player's current quest. Similar to our own work, their model uses bigram independence assumptions and uses a corpus to estimate conditional probability distributions.

Although our approaches are similar, there are a few significant differences. Albrecht *et al.* encode state into their model (in the form of location and previous quest), whereas our system only considers actions. We use abstract goals for partial recognition, whereas their system only makes full predictions. Our system also uses an (n-1)gram backoff strategy for smoothing.

6 Conclusions and Future Work

We have presented our initial work on using statistical, corpus-based techniques for goal recognition. Our recognizer is fast (linear time), does early and partial prediction, and can be ported to new domains more easily than many recognizers. We showed several initial experiments using the goal recognizer, in which we achieved high recognition rates with fairly high-accuracy early prediction.

There are several areas of future work that we are interested in. Several were alluded to above: scaling to larger domains, incorporating hierarchical plans, and using AI planners to generate artificial plan corpora.

In addition, we plan to collect a larger human-generated corpus in the Unix domain. With more training data, we would like to explore using trigram and 4-gram models as well as more advanced data mining techniques to train the statistical model.

Finally, as mentioned above, we would like to see how performance can be improved by training on a user-specific corpus. Similarly, we would like to see how different planners (say a reactive versus a deliberative planner) predict user behavior. Perhaps different planners could be used to model different domains (say domains with time pressure), or even different personality types.

Acknowledgments

We would like to thank Neal Lesh and Jun Hong for sharing their data with us. We would also like to thank the anonymous reviewers for their helpful comments.

This material is based upon work supported by Department of Education grant no. P200A000306; ONR research grant no. N00014-01-1-1015; and National Science Foundation grant no. E1A-0080124. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the above-mentioned organizations.

References

[Albrecht *et al.*, 1998] David W. Albrecht, Ingrid Zukerman, and Ann E. Nicholson. Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User-Adapted Interaction*, 8:5–47, 1998.

[Alexandersson, 1995] Jan Alexandersson. Plan recognition in VERBMOBIL. In M. Bauer, editor, *IJCAI 95 Workshop on The Next Generation of Plan Recognition Systems: Challenges for and Insight from Related Areas of AI (Working Notes)*, pages 2–7, Montreal, Canada, 1995.

[Allen *et al.*, 2000] J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent. An architecture for a generic dialogue shell. *Journal of Natural Language Engineering special issue on Best Practices in Spoken Language Dialogue Systems Engineering*, 6(3):1–16, December 2000.

[Appelt and Pollack, 1991] Douglas E. Appelt and Martha E. Pollack. Weighted abduction for plan ascription. *User Modeling and User-Adapted Interaction*, 2:1–25, 1991.

[Bauer, 1994] M. Bauer. Integrating probabilistic reasoning into plan recognition. In A. Cohn, editor, *Proceedings of the 11th European Conference on Artificial Intelligence*, pages 620–624, Amsterdam, Netherlands, August 1994. John Wiley & Sons.

[Carberry, 1990a] Sandra Carberry. Incorporating default inferences into plan recognition. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 471–478, 1990.

[Carberry, 1990b] Sandra Carberry. *Plan Recognition in Natural Language Dialogue*. ACL-MIT Press Series on Natural Language Processing. MIT Press, 1990.

[Charniak and Goldman, 1993] Eugene Charniak and Robert P. Goldman. A Bayesian model of plan recognition. *Artificial Intelligence*, 64(1):53–79, 1993.

[Chu-Carroll and Carberry, 2000] Jennifer Chu-Carroll and Sandra Carberry. Conflict resolution in collaborative planning dialogues. *International Journal of Human-Computer Studies*, 53(6):969–1015, 2000.

[Goldman *et al.*, 1999] Robert P. Goldman, Christopher W. Geib, and Christopher A. Miller. A new model of plan recognition. In *Uncertainty in Artificial Intelligence: Proceedings of the Fifteenth Conference (UAI-1999)*, pages 245–254, San Francisco, CA, 1999. Morgan Kaufmann Publishers.

[Hong, 2001] Jun Hong. Goal recognition through goal graph analysis. *Journal of Artificial Intelligence Research*, 15:1–30, 2001.

[Kautz, 1991] Henry Kautz. A formal theory of plan recognition and its implementation. In J. Allen, H. Kautz, R. Pelavin, and J. Tenenberg, editors, *Reasoning about Plans*, pages 69–125. Morgan Kaufman, San Mateo, CA, 1991.

[Lesh *et al.*, 1999] Neal Lesh, Charles Rich, and Candace L. Sidner. Using plan recognition in human-computer collaboration. In *Proceedings of the Seventh International Conference on User Modeling*, Banff, Canada, June 1999. Springer-Verlag. Also available as MERL Tech Report TR-98-23.

[Lesh, 1998] Neal Lesh. *Scalable and Adaptive Goal Recognition*. PhD thesis, University of Washington, 1998.

[Lochbaum, 1998] Karen E. Lochbaum. A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4):525–572, 1998.

[Manning and Schütze, 1999] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts, 1999.

[Paek and Horvitz, 2000] Tim Paek and Eric Horvitz. Conversation as action under uncertainty. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, Stanford, CA, June 2000.

[Pynadath and Wellman, 1995] David. V. Pynadath and Michael. P. Wellman. Accounting for context in plan recognition, with application to traffic monitoring. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 472–481, Montreal, Canada, 1995. Morgan Kaufmann.

[Vilain, 1990] Marc Vilain. Getting serious about parsing plans: a grammatical analysis of plan recognition. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 190–197, 1990.