Hierarchical Instantiated Goal Recognition

Nate Blaylock
Cycorp, Inc.
3721 Executive Center Dr. Ste 100
Austin, Texas, USA
blaylock@cyc.com

James Allen
Department of Computer Science
University of Rochester
PO Box 270262
Rochester, New York, USA
james@cs.rochester.edu

Abstract

We present a goal recognizer that statistically recognizes hierarchical goal schemas and their corresponding parameter values. The recognizer is fast (quadratic in the number of possible goal schemas and observations so far), and also supports partial parameter and subgoal-level prediction as well as n-best prediction.

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, Rich, & Sidner 1999), traffic monitoring (Pynadath & Wellman 1995), and dialogue systems (Carberry 1990).

For most applications, there are several properties required in order for goal recognition to be useful:

1. **Speed:** Most applications use goal recognition "online", meaning they use recognition results before the observed agent has completed its activity. Ideally, goal recognition should take a fraction of the time it takes for the observed agent to execute its next action.
2. **Early/partial prediction:** In a similar vein, applications need accurate goal prediction as early as possible in the observed agent's task execution. Even if a recognizer is fast computationally, if it is unable to predict the goal until after it has seen the last action in the agent's task, it will not be suitable for applications which need recognition results *during* task execution. If full recognition is not immediately available, applications can often make use of partial predictions.

In our work, we model goals and subgoals as parameterized action schemas from the SHOP2 HTN planner (Nau *et al.* 2003). Thus, we can distinguish between recognition of a goal *schema* and its corresponding *parameter values*. We term *instantiated goal recognition* as the recognition of a goal schema together with its parameter values. Additionally, we consider the task of *hierarchical goal recognition* which is the recognition of the chain of the agent's currently active top-level goal and subgoals. As an illustration, consider the tree shown in Figure 1 which represents an execution trace of subgoals based on a hierarchical plan.

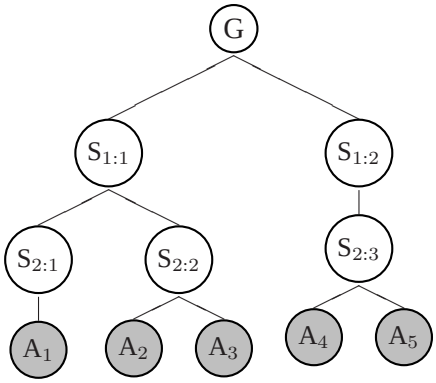

Figure 1: A Sample Hierarchical Plan Execution Tree

Here the root of the tree G is the agent's top-level goal. Leaves of the tree A_1–A_5 represent the atomic actions executed by the agent. Nodes in the middle of the tree represent the agent's various subgoals within the plan. For each executed atomic action, we can define a *goal chain* which is the subgoals which were active at the time it was executed. This is the path which leads from the atomic action to the top-level goal G. The goal chains associated with each atomic action in the tree in Figure 1 are shown in Figure 2. We cast hierarchical goal recognition as the recognition of the goal chain corresponding to the agent's last observed action.

Recognizing such goal chains can provide valuable information not available from a top-level goal recognizer. First, though not full plan recognition, hierarchical goal recognition provides information about which goal an agent is pursuing as well as a partial description of *how*.

Additionally, the prediction of subgoals can be seen as a type of partial prediction. As mentioned above, when a full prediction is not available, a recognizing agent can often make use of partial predictions. A hierarchical recognizer may be able to predict an agent's subgoals even when it is still not clear what the top-level goal is. This can allow a recognizer to potentially make predictions much earlier in

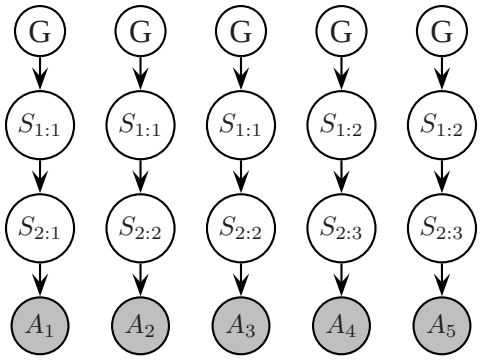

Figure 2: Goal Chains Corresponding to the Execution Tree in Figure 1

an execution stream.

In this paper, we extend our recent work on hierarchical goal *schema* recognition (Blaylock & Allen 2006) to include parameter recognition and create a hierarchical *instantiated* goal recognizer that is computationally fast and supports partial prediction of subgoal levels as well as parameter values. We also report the results of our experiments with this recognizer on a corpus of plan recognition problems.

For the rest of the paper, we first briefly describe our hierarchical goal schema recognizer and then describe the addition of parameter recognition to produce an instantiated recognizer. We then report our experimental results, discuss related work, and then conclude.

Hierarchical Schema Recognition

This section briefly describes our recent work in creating a hierarchical goal schema recognizer (reported in (Blaylock & Allen 2006)), which we use here as a base in constructing an instantiated goal recognizer. The schema recognizer uses a novel type of graphical model which we term a *Cascading Hidden Markov Model* (CHMM), which is a stack of HMMs where the state node of one level is the output of the level above it. An example of a CHMM is shown in Figure 3.

Here, the dth HMM (i.e., the HMM which starts with the hidden state $X_{d:1}$) is a typical HMM with the output sequence $O_{1,n}$. As we go up the CHMM, the hidden level becomes the output level for the level above it, and so forth.

CHMMs are different than Hierarchical HMMs (HHMMs). In CHMMs, all levels transition in lock-step at every observation (each state outputs exactly one value), whereas states at higher levels in HHMMs can output any number of values before they transition. Effectively, the difference results in HHMMs modeling a tree-like structure (like that in Figure 1) and CHMMs modeling a list of chains (as in Figure 2). Note that converting a goal tree into a list of goal chains results in a loss of information. For example, in the chains in Figure 2, it is unclear if the three nodes labeled $S_{1:1}$ come from a single node within source tree, or possibly several. This presents a problem for parameter recognition which we deal with below.

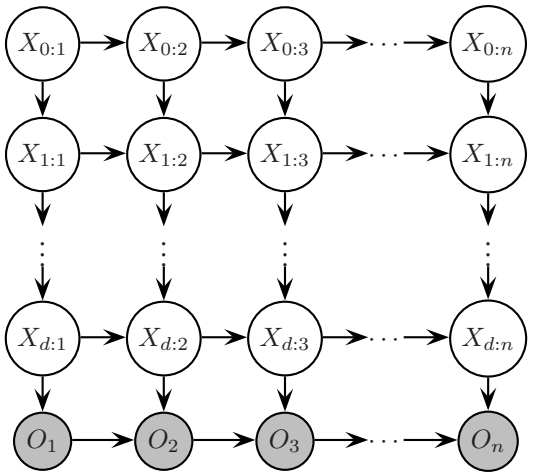

Figure 3: A Cascading Hidden Markov Model (CHMM)

However, this simplification allows us to do efficient reasoning with CHMMs. Our hierarchical schema recognizer works by modeling hierarchical plans as a list of goal chains, resulting in such duplication of subgoals which span multiple timesteps. We convert a corpus of plans into such lists of goal chains and use them to train the HMMs at each level of the CHMM. Online prediction is then made by computing the forward probability at each subgoal level and using that distribution to make an n-best prediction of each subgoal (separate 'no-predicts' are also possible if the probability of the n-best set is below a user-set threshold τ). The forward probability in CHMMs can be computed in time quadratic in the number of possible states (i.e., possible goal schemas).

Adding Parameter Recognition

In previous work, we introduced a recognizer for parameter values of an agent's top-level goal (Blaylock & Allen 2004) and integrated it into an instantiated goal recognizer for an agent's top-level goal (Blaylock & Allen 2005b). In this paper, we show how to create a hierarchical instantiated goal recognizer by adding parameter recognition based on that work with the work discussed in the previous section on hierarchical goal schema recognition.

We first briefly describe the top-level parameter recognition algorithm and then describe how it can be augmented to support parameter recognition within a hierarchical instantiated goal recognizer.

Top-Level Parameter Recognition

Our "flat" parameter recognizer (Blaylock & Allen 2004) worked for the recognition of a single top-level goal, and used a tractable form of Dempster-Shafer theory (DST) to combine evidence (using Dempster's rule of combination) from observed actions. A corpus was used to learn $P((G^j = A_i^k)|G^S, A_i^S)$ or the probability that the value of the kth parameter of observed action A_i is equal to the jth parameter of the goal G, given both the goal and action schemas as well as the two parameter positions. Note that in this dis-

tribution, the *value* of the parameter is not considered, only its *position*. This approach has the advantage that it can predict parameter values which were never seen in the training data (because it only focuses on *relations*), but the disadvantage that it can only predict parameter values which have appeared as parameter values of actions in the current observation stream. This issue is discussed in more detail in (Blaylock 2005).

When the parameter recognizer observes a new action, at the most atomic level, a DST basic probability assignment (bpa) called a *local bpa* is created for the pair of each goal parameter position j and action parameter position k, which reflects the probability discussed above. The local bpas of a single observed action (for a single goal parameter) can be combined by Dempster's rule of combination into an *action bpa* which represents the evidence from the single observed action. Upon each new action observation, the action bpa is then combined with a running *prediction bpa* which keeps track of the combined evidence for a single goal parameter position.

Parameter values are predicted by choosing the focal element of the prediction bpa with the highest weight. It is then comparing to the product of the ignorance measure of the bpa ($m(\Omega)$) and a user-set ignorance weight value (ψ), and predicted if it is higher than that threshold.

Application of this parameter recognition algorithm to the hierarchical case is not straightforward. For the rest of this section, we first describe our basic approach to hierarchical instantiated goal recognition and then discussed several problems (and their solutions) to incorporating the parameter recognizer.

Basic Algorithm

The basic algorithm of the hierarchical instantiated goal recognizer is as follows: Upon observing a new action, we first run the hierarchical schema recognizer and use it to make preliminary predictions at each subgoal level. At each subgoal level, we associate a set of parameter recognizers, one for each possible goal schema, which are then updated in parallel. This is necessary because each parameter recognizer is predicated on a particular goal schema. Each parameter recognizer is then updated by the process described below to give a new prediction bpa for each goal parameter position. If the schema recognizer makes a prediction at that level (i.e., its prediction was above the threshold τ), we then use the parameter recognizers for the predicted goal schemas to predict parameter values in the same way described above. This is similar to the algorithm used by our instantiated top-level goal recognizer (Blaylock & Allen 2005b), and combines the partial prediction capabilities of the schema and parameter recognizers to yield an instantiated goal recognizer.

Dealing with Uncertain Output

One problem when moving the parameter recognizer from a "flat" to a hierarchical setting is that, at higher subgoal levels, the hidden state (subgoal predictions) of the level below becomes our output action. This has two consequences: first, instead of the output being a single goal schema, it is now a probability distribution over possible goal schemas (computed by the schema recognizer at the lower level). Also, instead of having a single parameter value for each subgoal parameter position, we now have the prediction bpa from the parameter recognizer at the level below.

We solve both of these problems by weighting evidence by its associated probability. In general, a bpa in DST can be weighted using Wu's weighting formula (Wu 2003):

$$m'(A) = \begin{cases} wm(A) + 1 - w & : \quad A = \Omega \\ wm(A) & : \quad \text{otherwise} \end{cases} \tag{1}$$

where m is the bpa to be weighted and w is the weight. This equation essentially weights each of the focal points of the bpa and redistributes lost probability to Ω, or the ignorance measure of the bpa.

To handle uncertain parameter values in the output level, we change the computation of local bpas. Here we have an advantage because the uncertainty of parameter values at the lower level is actually itself represented as a bpa (the prediction bpa of the parameter recognizer at the lower level). We compute the local bpa by weighting that prediction bpa with the probability of positional equality computed from the corpus ($P((G^j = A_i^k)|G^S, A_i^S)$).

Using these local bpas, we can then calculate the action bpa (or evidence from a single action or subgoal) in the same way as the original parameter recognizer. However, we still have the lingering problem of having uncertain subgoal schemas at that level. Modifying the update algorithm for this case follows the same principle we used in handling uncertain parameters. To handle uncertain goal schemas, we compute an action bpa for each possible goal schema. We then introduce a new intermediate result called an *observation bpa* which represents the evidence for a parameter position given an entire observation (i.e., a set of uncertain goal schemas each associated with uncertain parameter values). To compute the observation bpa, first each action bpa in the observation is weighted according to the probability of its goal schema (using Equation 1). The observation bpa is then computed as the combination of all of the action bpas. This effectively weights the contributed evidence of each uncertain goal schema according to its probability (as computed by the schema recognizer).

Uncertain Transitions at the Prediction Level

In the original parameter recognizer, we assumed that the agent only had a single goal that needed to be recognized, thus we could assume that the evidence we saw was always for the same set of parameter values. However, in hierarchical recognition this is not the case, as subgoals may change at arbitrary times throughout the observations (see Figure 2).

This is a problem because we need to separate evidence. In the example of Figure 2, for example, subgoals $S_{2:1}$ and $S_{2:2}$ give us evidence for the parameter values of $S_{1:1}$, but presumably not (directly) $S_{1:2}$. Ideally, if we knew the start and end times of each subgoal, we could simply reset the prediction bpas in our parameter recognizers at that level after the third observation when the subgoal switched at that level.

In hierarchical recognition, we do not know a priori when goal schemas begin or end. We can, however, provide a rough estimation by using the transition probabilities estimated for each HMM in the schema recognizer. We use this probability (i.e., the probability that a new schema does not begin at this timestep) to weight the prediction bpa at each new timestep.

Basically, this provides a type of decay function for evidence gathered from previous timesteps. Assuming we could perfectly predict schema start times, if a new schema started, we would have a 0 probability, and thus weighting would result in a reset prediction bpa. On the other hand, if a new subgoal did not start, then we would have a weight of 1 and thus use the evidence as it stands.

Uncertain Transitions at the Output Level

A more subtle problem arises from the fact that subgoals at the output level may correspond to more than one timestep. As an example, consider again the goal chain sequence shown in Figure 2. At level 2, the subgoal $S_{2:2}$ is output for two timesteps by $S_{1:1}$.

This becomes a problem because Dempster's rule of combination makes the assumption that combined evidence bpas represent independent observations. For the case in Figure 2, when predicting parameters for $S_{1:1}$, we would combine output evidence from $S_{2:2}$ two separate times (as two separate observation bpas), as if two separate events had occurred.

The predictions for $S_{2:2}$ will of course likely be different at each of the timesteps, reflecting the progression of the recognizer at that level. However, instead of being two independent events, they actually reflect two estimates of the same event, with the last estimate presumably being the most accurate (because it itself has considered more evidence at the output level).

To reflect this, we change the update algorithm to additionally keep track of the prediction bpa formed with evidence from the last timestep of the most recently *ended* subgoal at the level below, which we will call the *last subgoal prediction (lsp) bpa*. At a new timestep, the prediction bpa is formed by combining the observation bpa with this lsp bpa. If we knew the start and end times of subgoals, we could simply discard this prediction bpa if a subgoal had not yet ended, or make it the new lsp if it had. Not knowing schema start and end times gives us a similar problem at the output level. As we discussed, we need a way of distinguishing which observed output represents a new event versus which represents an updated view of the same event.

We handle this case in a similar way to that above. We calculate the probability that the new observation starts a new timestep by the weighted sum of all same transition probabilities at the level below. This estimate is then used to weight the prediction bpa from the last timestep and then combine it with the lsp bpa to form a new lsp bpa. In cases that there is high probability that a new subgoal was begun, the prediction bpa will have a large contribution to the lsp bpa, whereas it will not if the probability is low.

Total Sessions	5000
Goal Schemas	10
Action Schemas	30
Ave Actions/Session	9.5
Subgoal Schemas	28
Ave Subgoal Depth	3.8
Max Subgoal Depth	9

Table 1: The Monroe Corpus

Complexity

Space precludes a detailed discussion of algorithm complexity here. Instead, we provide a sketch of the analysis and refer the reader to (Blaylock 2005) for details.

First, we must analyze the complexity of the modified parameter recognizer (which deals with output with uncertain goal schemas). In this analysis, $|S|$ is the number of possible goal schemas; q is the maximum parameters for a goal schema; i is the current timestep; and D is the depth of the hierarchical plan.

The modified algorithm computes the observation bpa by combining (worst case) $|S|$ action bpas — each with a maximum size of iq (limited by the number of unique parameter values seen, as described in (Blaylock & Allen 2004)). Thus, the total complexity for the update of a single parameter position is $O(|S|i^2q^3)$ and for the parameter recognizer of a single goal schema (with q parameter positions), this becomes $O(|S|i^2q^4)$. As q is constant and likely small, we drop it, which gives us $O(|S|i^2)$.

The complexity of an update for the instantiated recognizer can be calculated from the runtime of the schema recognizer plus the runtime of each of the $D|S|$ parameter recognizers (one per each goal schema per level). Thus the total runtime complexity is $O(D|S|^2 + D|S|^2i^2) = O(D|S|^2i^2)$, or quadratic in the number of possible goal schemas and the number of actions observed so far.

Experiments

We now report on our experiments, first on the parameter recognizer itself, and then on the hierarchical instantiated goal recognizer. For the experiments, we used 4500 plan sessions from the artificial Monroe corpus (Blaylock & Allen 2005a) for training and the remaining 500 for testing. This is the same data used in the experiments on our instantiated top-goal recognizer (Blaylock & Allen 2005b) and allows us to make comparisons below.

Before we describe the experiments and their results, however, we briefly describe the metrics we use to report results, which are also the same used in (Blaylock & Allen 2005b).

Metrics

We report results for individual subgoal depths, as well as totals.[1] For each depth, we use the same metrics used in

[1] Predictions on "filler" subgoals inserted to make the plan trees of constant depth were not counted here. See (Blaylock 2005) for

(Blaylock & Allen 2005b) to measure results. These were introduced to try to measure the general requirements of goal recognizers described above. *Precision* and *recall* report the number of correct predictions divided by total predictions and total prediction opportunities, respectively. *Convergence* and *convergence point* stem from the fact that, oftentimes, the recognizer will be unsure very early on in a session, but may at some point 'converge' on the correct answer, predicting it from that point on until the end of the plan session. *Convergence* measures the percentage of plan sessions where the correct answer was converged upon.[2] For those plan sessions which converge, *convergence point* reports the average action observation after which it converged divided by the average number of actions for the converged sessions. This is an attempt to measure how *early* in the plan session the recognizer was able to zero in on the correct answer.

In reporting results for parameter recognition, we additionally use *recall/feasible* and *convergence/feasible*, which measure recall and convergence for those cases which it was possible for the parameter recognizer to get the right answer. As described above, our algorithm for parameter recognition can never predict a parameter which is has not yet seen as the parameter of an observed action.

In reporting results for instantiated recognition, *parameter Percentage (p%)* reports, for all correct predictions, the percentage of the goal parameters that were predicted. *Convergence Parameter Percentage (c/p%)* reports the same for all sessions which converged. These are an attempt to measure the specificity of correct predictions which are made. For hierarchical recognition, we make a change to how convergence and convergence point are calculated. Some subgoals only span one timestep (e.g., they only result in one executed atomic action), in which case, it does not make sense to report convergence or a convergence point. For all levels, we only report convergence and convergence point for subgoals which correspond to at least two timesteps.

Parameter Recognition Results

To test the parameter recognizer in its own right, we assumed perfect schema recognition — i.e., for each new observation, we gave the parameter recognizer information about the chain of goal schemas at that time point, including information about which schemas began at that timestep. This perfect information about schema transitions meant that the parameter recognizer did not need to deal with (a) uncertain transitions at the prediction level and (b) uncertain schemas at the output level. Note that there was still uncertainty at the output level at higher levels because *parameter values* were still potentially uncertain, even though the schema was known.

Top-level Results To help interpret the results, we compare performance at the top level to that of the flat recognizer (which only made predictions at the top level). For convenience, the results of the flat parameter recognizer on the same data set are shown in Table 3.

The hierarchical parameter recognizer performed slightly better in both the 1-best and 2-best cases. In 1-best, precision moved from 94.3 percent to 98.6 percent, although there was a drop in recall from 27.8 percent to 25.8 percent. In the 2-best recognizer, results were slightly better all around.

The reason for the improvement in performance is likely attributable to the fact that (perfect) subgoal schema information was present in the hierarchical recognizer. This allowed parameter values to be considered given the immediate child subgoal, giving better context for predictions.

Other Levels The hierarchical parameter recognizer performed well at other levels as well, with precision staying (for the 1-best case) in the high 90's and even up to 100 percent for levels 7 and 8. This performance inched up for the 2-best case (with 100 percent precision for levels 4–8).

It is interesting to note that recall begins quite low (25.8 percent for level 0) and then climbs as we go down levels, reaching 100 percent for levels 7 and 8. As mentioned above, high absolute recall is not to be expected in plan recognition, as ambiguity is almost always present. The closer we move to the actual observed action, however, the higher precision gets. This can be attributed to two factors. First, subgoals at lower levels are closer to the observed input, and thus deal with less uncertainty about what the parameter values are.

Second, and probably most important, is that lower-level subgoals span fewer timesteps than those at higher levels, meaning that, if parameter values are available, they will be seen after a shorter number of actions. In the case of levels 7 and 8, all subgoals only spanned one timestep, and thus only had one chance to get the right parameter values. It turns out that parameter values at these levels always directly corresponded to the action parameters, which is why precision and recall reach 100 percent here.

Overall, the performance of the parameter recognizer was very encouraging, especially the performance at lower levels which had high recall. This is an important factor in our ability to do specific and accurate partial prediction in the instantiated goal recognizer, which we move to now.

Instantiated Recognition Results

Here we example the results of experiments on the instantiated hierarchical goal recognizer, which combined both the schema and parameter recognizers as described above. We first look at the results at the top level (i.e., level 0) and then the other levels.

Top-level Results To help interpret the results, we compare performance at the top level to that of our "flat" goal recognizer (which only made predictions at the top level). For convenience, the results of the flat instantiated recognizer on the same data set are shown in Table 5.

Hierarchical instantiated results at the top level closely mirror results of the hierarchical schema recognizer (Blaylock & Allen 2005b). This also happened for the flat recognizer and is to be expected, as schema recognition per-

details.

[2]This essentially measures how many *last* predictions were correct, i.e., whether we *ended* predicting the right answer.

	1-best ($\psi = 2.0$)					
level	**prec.**	**recall**	**recall/feas.**	**conv.**	**conv./feas.**	**conv. pt**
0	98.6%	25.8%	52.0%	44.7%	56.3%	5.0/9.9
1	99.7%	26.4%	52.0%	39.9%	55.2%	4.1/6.3
2	96.7%	53.0%	76.4%	51.6%	57.7%	2.5/4.8
3	98.7%	73.8%	89.4%	73.8%	74.1%	3.1/4.1
4	99.3%	80.0%	94.6%	80.9%	80.9%	3.3/3.8
5	97.5%	82.6%	91.1%	53.1%	53.1%	2.2/3.9
6	99.9%	98.3%	99.3%	50.0%	50.0%	2.0/4.0
7	100%	100%	100%	N/A	N/A	N/A
8	100%	100%	100%	N/A	N/A	N/A
total	98.5%	51.7%	76.5%	51.6%	61.2%	3.5/5.7

	2-best ($\psi = 2.0$)					
level	**prec.**	**recall**	**recall/feas.**	**conv.**	**conv./feas.**	**conv. pt**
0	97.7%	40.1%	80.8%	76.0%	95.8%	4.7/9.0
1	99.9%	41.3%	81.2%	63.6%	88.0%	3.5/5.7
2	99.6%	65.9%	95.1%	82.9%	92.8%	2.8/4.7
3	99.8%	81.0%	98.2%	97.6%	97.9%	3.4/4.5
4	100%	83.3%	98.5%	97.6%	97.6%	3.3/3.9
5	100%	89.7%	99.0%	93.0%	93.0%	2.5/3.9
6	100%	99.1%	100%	100%	100%	2.5/4.0
7	100%	100%	100%	N/A	N/A	N/A
8	100%	100%	100%	N/A	N/A	N/A
total	99.5%	62.4%	92.4%	78.6%	93.2%	3.5/5.6

Table 2: Results of Parameter Recognition

	1-best ($\psi = 2.0$)					
level	**prec.**	**recall**	**recall/feas.**	**conv.**	**conv./feas.**	**conv. pt**
top	94.3%	27.8%	55.9%	46.9%	59.1%	5.1/10.0

	2-best ($\psi = 2.0$)					
level	**prec.**	**recall**	**recall/feas.**	**conv.**	**conv./feas.**	**conv. pt**
top	97.6%	39.2%	78.9%	76.2%	96.1%	4.8/9.0

Table 3: Results of Flat Parameter Recognition on the Monroe Corpus (cf. (Blaylock & Allen 2005b))

	1-best ($\tau = 0.7, \psi = 2.0$)						**2-best** ($\tau = 0.95, \psi = 2.0$)					
level	**prec.**	**recall**	**p%**	**conv.**	**c/p%**	**conv. pt**	**prec.**	**recall**	**p%**	**conv.**	**c/p%**	**conv. pt**
0	82.5%	56.4%	24.0%	90.8%	49.8%	5.6/10.3	88.2%	60.2%	23.2%	91.0%	49.9%	5.2/10.3
1	81.3%	52.8%	23.5%	67.6%	26.5%	3.1/6.1	93.8%	75.4%	16.6%	94.8%	18.9%	2.4/5.6
2	85.4%	44.3%	22.5%	45.8%	38.5%	3.4/4.7	89.7%	62.0%	42.1%	84.4%	45.2%	3.6/4.8
3	72.9%	41.7%	82.4%	41.2%	90.6%	3.0/3.5	90.6%	74.4%	81.8%	99.0%	71.0%	3.9/4.5
4	73.6%	50.0%	99.9%	61.8%	100%	3.7/3.7	90.8%	68.6%	96.5%	100%	80.9%	3.8/3.8
5	58.8%	45.7%	100%	6.2%	100%	4.2/4.2	98.2%	76.4%	81.4%	100%	53.1%	2.0/3.9
6	69.3%	69.3%	100%	0.0%	N/A	N/A	98.3%	98.3%	99.2%	100%	50.0%	4.0/4.0
7	95.2%	95.2%	100%	N/A	N/A	N/A	100%	100%	100%	N/A	N/A	N/A
8	100%	100%	100%	N/A	N/A	N/A	100%	100%	100%	N/A	N/A	N/A
total	79.0%	50.4%	44.1%	61.7%	46.4%	3.9/6.8	91.3%	68.7%	47.2%	92.5%	43.7%	3.6/6.1

Table 4: Results of Instantiated Recognition

	1-best ($\tau = 0.7, \psi = 2.0$)						**2-best** ($\tau = 0.95, \psi = 2.0$)					
level	**prec.**	**recall**	**p%**	**conv.**	**c/p%**	**conv. pt**	**prec.**	**recall**	**p%**	**conv.**	**c/p%**	**conv. pt**
top	93.1%	53.7%	20.6%	94.2%	40.6%	5.4/10.0	95.8%	56.6%	21.8%	97.4%	41.1%	5.5/10.1

Table 5: Results of Flat Instantiated Recognition on the Monroe Corpus (cf. (Blaylock & Allen 2005b))

formance limits performance of the instantiated recognizers. The addition of parameter predictions serves to degrade the precision and recall of schema recognition results.

In the 2-best case, precision decreased from 95.8% in the flat recognizer to 88.2% in the hierarchical recognizer. However, recall increased from 56.6% to 60.2% and parameter percentage from 21.8% to 23.2%. This difference was more stark in convergence parameter percentage, which rose from 41.1% to 49.9%.

Although moving to a hierarchical recognizer seemed to degrade performance at the top level, at least in precision, this may be somewhat misleading because of the relative short length 9.5 actions of plans in the Monroe domain. We believe that longer, more complex plans will make it much more difficult to predict the top-level goal early on — requiring partial recognition of subgoals. Of course, this remains to be tested.

Other Levels

Precision and recall at other levels also closely mirror the performance of the schema recognizer. Precision dips in the middle levels but this levels out for 2-best prediction, which achieves precision ranging from the high 80's to 100 percent (with recall ranging in the 60's and 70's for high levels and high 90's and 100 percent for the lower levels).

Parameter prediction for levels 1 and 2 remains in the 20's, with a sudden jump to 82.4 percent at level 3, 99.9 percent at level 4, and 100 percent for the lower levels, for the 1-best level. Note that the drop in parameter prediction at several levels in the 2-best case is due to the fact that the recognizer gets more cases right (i.e., increases recall), but that many of the new correct predictions have less instantiated parameter values. Thus the decrease in number reflects that the recognizer is getting more correct predictions, but it does not reflect a decrease in performance for the cases it got correct in 1-best prediction.

Related Work

Although parameter prediction has been included in logical-based (e.g., (Kautz 1991)) and case-based (e.g., (Cox & Kerkez To appear)) plan recognizers, relatively little attention has been given it in work on probabilistic plan recognizers. For example, (Pynadath & Wellman 2000) and (Bui 2003) build hierarchical goal recognizers but treat goals as atomic (basically like our goal schemas).

Probabilistic systems which do include parameter recognition typically use probabilities for goal schema prediction, but logical-based methods for filling in parameters (e.g., (Bauer 1995)). The recognizer in (Charniak & Goldman 1993) dynamically constructs a Belief Network (BN) with nodes for action and goal schemas, objects (possible parameter values), and relations that use the latter to fill slots in the former. For a given parameter slot, however, they consider all objects of the correct type to be equally likely, whereas we distinguish these using probabilities learned from a corpus. As the network grows at least with the number of observed actions (and likely the number of goal schemas), it is unclear if this approach would be scalable in general.

Conclusions

We have presented a hierarchical goal recognizer which statistically recognizes both goal schemas and their parameter values. The recognizer we have described here has two nice features (which correspond to the two desired traits of goal recognizers described above). First, recognition is fast and scalable, running in time quadratic to the number of possible goal schemas and number of observations. Second, the recognizer supports two forms of partial goal recognition: partial parameter prediction and threshold subgoal level prediction. This allows the recognizer to make predictions earlier in the agent's task execution than would be possible for all-or-nothing prediction.

An obvious need for future work is to test the recognizer on more corpora, especially real data. This is hindered by the lack of appropriate corpora for plan recognition in the field, especially since our approach expects corpora labeled with hierarchical plan structure.

Acknowledgments

We would like to thank the reviewers for their helpful comments. This material is based upon work supported by a grant from DARPA under grant number F30602-98-2-0133; two grants from the National Science Foundation under grant number IIS-0328811 and grant number E1A-0080124; ONR contract N00014-06-C-0032; and the EU-funded TALK project (IST-507802). 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

Bauer, M. 1995. A Dempster-Shafer approach to modeling agent preferences for plan recognition. *User Modeling and User-Adapted Interaction* 5(3–4):317–348.

Blaylock, N., and Allen, J. 2004. Statistical goal parameter recognition. In *Proceedings of the Fourteenth 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., *User Modeling 2005*, number 3538 in Lecture Notes in Artificial Intelligence. Edinburgh: Springer.

Blaylock, N., and Allen, J. 2005b. Recognizing instantiated goals using statistical methods. In *IJCAI Workshop on Modeling Others from Observations (MOO-2005)*.

Blaylock, N., and Allen, J. 2006. Fast hierarchical goal schema recognition. In *Proceedings of AAAI-06*. To appear.

Blaylock, N. J. 2005. *Towards Tractable Agent-based Dialogue*. Ph.D. Dissertation, University of Rochester, Dept. of Computer Science.

Bui, H. H. 2003. A general model for online probabilistic plan recognition. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*.

Carberry, S. 1990. *Plan Recognition in Natural Language Dialogue*. MIT Press.

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

Cox, M. T., and Kerkez, B. To appear. Case-based plan recognition with novel input. *International Journal of Control and Intelligent Systems*.

Kautz, H. 1991. A formal theory of plan recognition and its implementation. In Allen, J.; Kautz, H.; Pelavin, R.; and Tenenberg, J., eds., *Reasoning about Plans*. San Mateo, CA: Morgan Kaufman.

Lesh, N.; Rich, C.; and Sidner, C. L. 1999. Using plan recognition in human-computer collaboration. In *Proceedings of the Seventh International Conference on User Modeling*. Banff, Canada: Springer-Verlag.

Nau, D.; Au, T.-C.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research* 20:379–404.

Pynadath, D. V., and Wellman, M. P. 1995. Accounting for context in plan recognition, with application to traffic monitoring. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. Montreal, Canada: Morgan Kaufmann.

Pynadath, D. V., and Wellman, M. P. 2000. Probabilistic state-dependent grammars for plan recognition. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-2000)*.

Wu, H. 2003. *Sensor Data Fusion for Context-Aware Computing Using Dempster-Shafer Theory*. Ph.D. Dissertation, Carnegie Mellon University, Robotics Institute.