Extended Abstract: Recognizing Plan/Goal Abandonment

Christopher W. Geib
Honeywell Laboratories
3660 Technology Drive
Minneapolis, MN 55418
christopher.geib@htc.honeywell.com

Robert P. Goldman
Sift Inc.
2119 Oliver Ave.
Minneapolis, MN 55410
rpgoldman@siftech.com

Introduction

This paper describes ongoing efforts to extend our work (Geib & Goldman 2001b; 2001a) (G&G) on the Probabilistic Hostile Agent Task Tracker (PHATT) to handle the problem of goal abandonment. As such, we will be discussing both probabilistic intent inference and the PHATT system and assume the reader is familiar with these areas. We refer the interested reader to the complete paper and to our earlier papers for more discussion of these issues.[1]

We are applying plan recognition methods, specifically PHATT, to two real-life applications: building assistant systems for elderly patients in assisted living environments and tracking hostile agents in computer network security. Both of these application areas require us to be able to recognize, and reason about, situations when the agents we observe have abandoned a plan or goal. Consider the case of an elder[2] who begins the process of taking her medication but gets distracted and does not complete the plan. If a system can recognize when that plan has been abandoned and will not be completed, the system can provide helpful reminders.

In general, the ability to infer abandoned goals is an operational requirement for any plan recognition system that is executing incrementally and continuously. Abandoning goals is something that any real observed agent will do. If a plan recognition system is unable to recognize this fact, the system will build up an ever increasing set of active or *open plans* that the agent has no intention of completing. Without some method of identifying abandoned plans a system attempting to find completions for these open plans will wind up considering unreasonable situations such as the elder begins making a sandwich which they didn't finish until several hours, days or even weeks later.

Existing plan recognition systems do not draw such inferences. Instead a number of different domain assumptions allow them to work around this problem:

- only consider the agent's current goal: The Lumière project, (Horvitz *et al.* 1998) discards all previous goals of an agent (although remembering a history of past help), therefore there is no need for an explicit treatment of abandoned goals.
- the agent has a single fixed goal: In tutorial applications (Gertner, Conati, & VanLehn 1998) there is often only one goal the system is concerned with: namely the agent's acquiring a skill or knowledge from the system. From the perspective of the system, effectively only those actions that are part of the educational process are relevant for recognition.
- the agent will always come back:(Kautz & Allen 1986) Some systems assume that while an agent may interrupt one goal in order to achieve another one, they will eventually return to the original goal.
- rely on a cooperative agent for disambiguation: (Lesh, Rich, & Sidner 2001) Other systems expect that agents can be counted upon to communicate that they have abandoned a goal, possibly in response to a direct query from the system.

However, for a number of reasons, none of these assumptions are acceptable in our domains of interest.

As we will discuss, explicitly computing the probability of a goal's abandonment can result in an explosion in the number of hypotheses that must be considered by the system making the algorithms computationally prohibitive for real world systems. Rather than explicitly computing the probability of goal abandonment, we believe the problem can best be viewed as a case of probabilistic model revision.

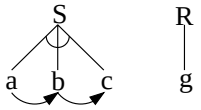

Figure 1: A very simple example plan library.

Background

The PHATT framework is based on a model of the execution of simple hierarchical task network (HTN) plans (Erol, Hendler, & Nau 1994) like the ones seen in Figure 1.

The central realization of the PHATT approach is that as plans are executed the agent is able to execute any one of

[1]This work was performed under the support of the U.S. Department of Commerce, National Institute of Standards and Technology, Advanced Technology Program Cooperative Agreement Number 70NANB0H3020

[2]Elderly patient.

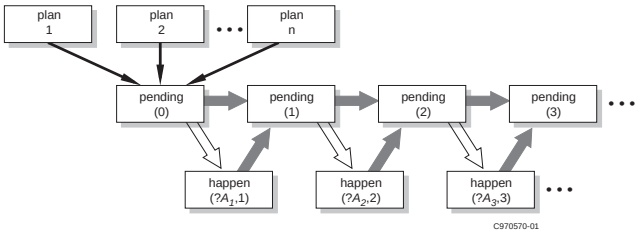

Figure 2: A simple model of plan execution.

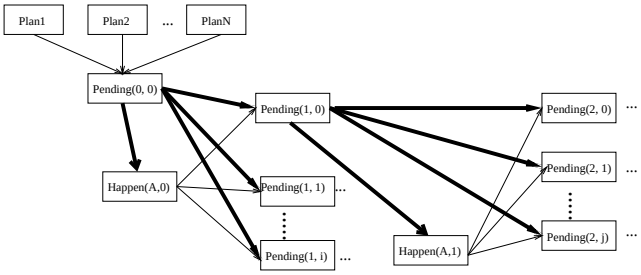

Figure 3: A model of plan execution and pending set generation with abandoned goals.

the actions in its plans that have been enabled by its previous actions. To formalize this, initially the executing agent has a set of goals and chooses a set of plans to execute to achieve these goals. The initial actions of the set of plans chosen determines a *pending set* of primitive actions. The agent executes one of the pending actions, generating a new pending set from which the next action will be chosen.

Thus, for each possible explanatory hypothesis, there is a unique corresponding sequence of pending sets. Each pending set is generated from the previous set by removing the action just executed and adding newly enabled actions. Actions become enabled when their required predecessors are completed. This process is illustrated in Figure 2.

Given this, plan recognition in PHATT can be viewed as using probabilistic simulation of the generation of pending sets to produce the complete, exclusive and exhaustive covering set of explanations for a given set of observations. Conditional probabilities of root goals or plan features can then be computed from the set of explanations. We refer the interested reader to (Geib & Goldman 2001b; 2001a) for a more extensive explanation of this process.

Exact Solutions

Adding plan abandonment to the model of plan execution critically changes the way in which pending sets are progressed. We will assume that immediately after each action in the observation sequence, the agent is able to abandon one or more of the goals in its pending set. This means that the previous pending set and the action observed no longer uniquely determine the new pending set. Our model of plan execution moves from the one seen in Figure 2 to that shown in Figure 3.

Now we no longer have a deterministic model of hypothesis progression. Previously, a pending set and an action execution yielded a unique pending set for the next time instant. However, now there are multiple possible alternatives. In the figure these are denoted: Pending(t,0), Pending(t,1),... Pending(t,i), where t represents the time step. As before, the execution of an action generates a unique new pending set by enabling new actions. However, now the agent also chooses a set (possibly the empty set) of goals to be abandoned.

Since the agent may abandon more than a single goal at a time, the number of pending sets equals the number of possible subsets of the goals in the pending set that would have resulted without considering abandonment. I.e. the new hypotheses correspond to the power-set of the goals in the original pending set. Thus, where the previous algorithm had to consider a single pending set with n goals, our new algorithm will have to consider 2^n pending sets for the next observation.

This kind of explosion of the explanation space makes explicitly computing the probability of goal abandonment computationally very expensive.

Model Revision

Rather than explicitly considering all of the possible plans that could be abandoned, the problem can be looked at as a question of model fitting or model revision. If we are using a model of plan execution that does not consider plan abandonment to recognize observation streams in which the agent is abandoning plans, we expect that the computed probabilities for the observation streams will be quite low. Laskey (1991), Jensen (Jensen *et al.* 1990), Habbema (1976) and others (Rubin 1984) have suggested that cases of an unexpectedly small $P(observations|Model)$ should be used as evidence of a model mismatch.

In the case where we are interested in recognizing a specific kind of model failure (namely that the agent is no longer executing a plan that it has begun) the statistic of $P(observations|model)$ is not sufficient. While this statistic will drop rather rapidly as we fail to see evidence of the agent carrying out the steps of their plan, it does not provide us with sufficient information to determine which of the agent's goals has been abandoned, the critical information that we need in order to repair the model.

Instead of the general $P(observations|model)$ statistic we propose the use of a more specific statistic. Our intuitions about how we recognize plans will help us to find this statistic. Suppose I am observing someone that I initially believe has two high level goals, call them α and β. At the outset they mostly alternate the steps of the plans, but at about half way through the agent stops working on β and instead only executes actions that are part of α. As the string of actions that contribute only to α gets longer, and we see no more actions that contribute to β we should start to suspect that the agent has abandoned β.

We can formalize this idea as the probability that none of the observed actions in a subsequence (from say s to t) contribute to one of the goals (call it G), and we denote it $P(noneContrib(G, s, t)|model, observations)$. If this probability gets unexpectedly small we consider this as evidence of a mismatch between the model and the real world.

Namely the model predicts that the agent is still working on the goal, while the agent may have abandoned it. The following section will detail how to compute this probability.

For a Single Explanation

Consider the plan library shown in Figure 1. The first plan is a very simple plan for achieving S by executing a, b, and c and the second plan for R has only the single step of g. Given this plan library, assume the following sequence of observations:

$$happen(a, 0), happen(b, 1), happen(g, 2), happen(g, 3).$$

In this case we know that at time 0 and 1 that the agent has as a goal achieving S. In PHATT, we compute the probability of seeing c at time 2 by: $(m/|PS_2|)$ where m is the number of elements in the pending set that have c as the next action. The probability that we don't see c (the probability that any other element of the pending set is chosen at time 2) is:

$$1 - (m/|PS_2|)$$

or more generally the probability that we have seen b at time $(s-1)$ and not seen c by time t:

$$\prod_{i=s}^{t} 1 - (m/|PS_i|)$$

To handle partially ordered plans, this formula must be generalized slightly. With partially ordered plans it is possible for more than a single next action to contribute to the specified root goal. Thus, if $m_{q,i}$ represents the number of elements (with any next action) in the pending set at time *i* that contribute to goal *q*, *(s-1)* is the last time we saw an action contribute to *q* and *t* is the current time, $P(noneContrib(q, s, t)|model, obs) =$

$$\prod_{i=s}^{t} 1 - (m_{q,i}/|PS_i|)$$

Thus, under the assumptions that we have made we can compute the probability of the subsequence of actions not contributing to a given plan or goal. By computing this value and setting a threshold, we can consider any drop in this probability below the threshold as sufficient evidence of a model mismatch and revise the model to reflect the goals abandonment. This requires removing all the elements from the current pending set that contribute to the abandoned goal. Modeling of the execution of the rest of the goals can continue as before.

Conclusions

The full paper not only presents this argument in fuller detail but also provides implementation details as well as some preliminary results about the system's runtime, effectiveness, accuracy and limitations of this approach.

The ability to recognize when an agent abandons a plan is an open problem in the plan recognition literature. This paper presents a solution to the problem of recognizing when an agent has abandoned a goals based on probabilistic model revision. As such, it represents a significant extension to recent work on abductive probabilistic plan recognition.

References

Erol, K.; Hendler, J.; and Nau, D. S. 1994. UMCP: A sound and complete procedure for hierarchical task network planning. In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems (AIPS 94)*, 249–254.

Geib, C. W., and Goldman, R. P. 2001a. Plan recognition in intrusion detection systems. In *Proceedings of DISCEX II, 2001*.

Geib, C. W., and Goldman, R. P. 2001b. Probabilistic plan recognition for hostile agents. In *Proceedings of the FLAIRS 2001 Conference*.

Gertner, A.; Conati, C.; and VanLehn, K. 1998. Procedural help in andes: Generating hints using a bayesian network student model. In *Proceedings of the 15th National Conference on Artificial Intelligence*.

Habbema, J. 1976. Models for diagnosis and detections of combinations of diseases. In *Decision Making and Medical Care*. North Holland.

Horvitz, E.; Breese, J.; Heckerman, D.; Hovel, D.; and Rommelse, K. 1998. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*.

Jensen, F. V.; Chamberlain, B.; Nordahl, T.; and Jensen, F. 1990. Analysis in hugin of data conflict. In *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence*.

Kautz, H., and Allen, J. F. 1986. Generalized plan recognition. In *AAAI-86*, 32–38.

Laskey, K. B. 1991. Conflict and surprise: Heuristics for model revision. In *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*.

Lesh, N.; Rich, C.; and Sidner, C. 2001. Collaborating with focused and unfocused users. In *Proceedings of the 8th International Conference on User Modeling*.

Rubin, D. 1984. Bayesianly justifiable and relevent frequency calculations for the applied statistician. In *The Annals of Statistics*, volume 12(4), 1151–1172.