Recognizing Plan/Goal Abandonment*

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

Robert P. Goldman,
SIFT Inc.
2119 Oliver Ave. South
Minneapolis, MN 55405
rpgoldman@siftech.com

Abstract

The ability to recognize when an agent abandons a plan is an open problem in the plan recognition literature and is a significant problem if these methods are to be applied in real systems. This paper presents an explicit, formal, and implemented solution to the problem of recognizing when an agent has abandoned one of its goals based on a theory of probabilistic model revision.

1 Introduction

There is a large body of research on the topic of intent recognition or task tracking, focused on identifying the goals of a user from observations of their actions. However, none of this work has directly addressed the problem of recognizing when a user has abandoned a goal.

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. A system attempting to find completions for these open plans will wind up considering unreasonable situations such as the first step of a time critical two step plan simple plan are taken but the plan is not elder a two step plan with a required seconds or minutes duration but not attempting the second step of the plan until days or weeks later. Unfortunately, existing plan recognition systems cannot draw such inferences.

2 Background

Recent work in execution based plan/intent recognition[Bui *et al.*, 2002; Geib and Goldman, 2001; Goldman *et al.*, 1999] has been based on a model of the execution of simple hierarchical task network (HTN) plans [Erol *et al.*, 1994].

The idea behind this approach is that initially the executing agent has a set of goals and chooses a set of plans to execute to achieve these goals. 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, and so on.

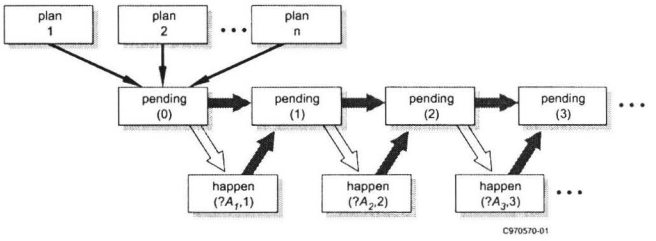

Figure 1: A simple model of plan execution.

This process is illustrated in Figure 1. In this light, the observed actions are nothing more than the observations of a hidden Markov model and the process of plan recognition is the inference of the underlying state of the model.Space prohibits a full exposition of this approach. We refer readers to [Geib and Goldman, 2001; Goldman *et al.*, 1999] for a more complete discussion.

3 Exact Solutions

Given our model of plan execution, a formal and general model of probabilistic abandonment of goals, will be forced to deal with an exponentially larger model and will be required to obtain the prior probabilities that each of the goals is abandoned. For real world applications this approach is simply untenable. A complete discussion of these issues is provided in the full paper.

4 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 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], and others have suggested that cases of an unexpectedly small $P(observations|Model)$ should be used as evidence of a model mismatch.

*This material is based upon work supported by DARPA/IPTO and the Air Force Research Laboratory under Contract No. F30602-02-C-0116

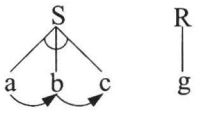

Figure 2: A very simple example plan library.

Instead of the general $P(observations|model)$ statistic we propose 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 $P(notContrib(G, s, t)|model, observations)$. 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.

4.1 Computing $notContrib$

Consider the plan library shown in Figure 2. 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. Next, 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. Let us assume that all of the elements of the pending set are equally likely to be selected for execution next. Note that this is an assumption that we will make for the rest of this paper. Nothing about the algorithm hinges on this uniformity assumption. It is made solely for ease of computation and discussion.

Given this assumption, the probability of seeing c at time 2 is given 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 (that is the probability that any other element of the pending set is chosen at time 2) is just:

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

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, $P(notContrib(q, s, t)|model, obs) =$

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

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 the rest of the plans continues as before.

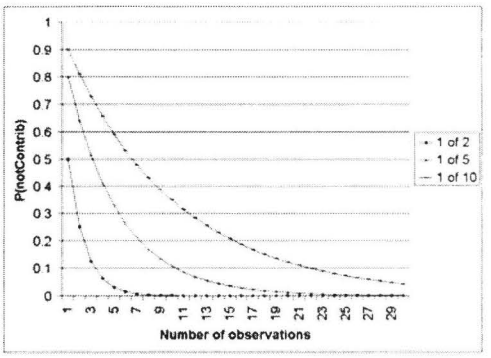

Figure 3: Required Evidence Theoretical Curves.

4.2 Evidential Requirements

This approach creates an interesting linkage between the size of the pending set, the number of elements that contribute to the goal of interest, and the number of actions that **don't** contribute to the goal that must be observed before the goal is considered abandoned.

Figure 3 shows three theoretical curves for the probability of $notContrib$ for different sets of values. The curves are labeled with the number of actions that contribute to a goal and the size of the pending set. Thus the curve labeled "1 of 2" shows the drop in the probability given each observation if there is one action that contributes to the desired goal out of a pending set of size two. Note that in these curves, we are again making the assumption that all of the actions in the pending set are equally likely to be chosen.

Notice that as the ratio of the number of contributing actions to the size of the pending set drops the number of actions required to drive $notContrib$ down to a particular threshold value increases significantly. We will see the effects of this in the empirical results.

4.3 Estimating $P(abandoned(g)|Obs)$

If we compute $P(notContrib(g, s, t)|model, obs)$ for each g and threshold our explanations as described in the previous section, we can now produce explanations of the observations in which goals have been abandoned. By considering the complete and covering set of such explanations for the observations we can estimate the probability of a specific goal's abandonment. It is given by:

$$P(abandoned(g)|Obs) \approx \frac{\sum_e^{Exp_{A(g)}} P(e|Obs)}{\sum_e^{Exp} P(e|Obs)}$$

where Exp represents the set of all explanations for the observations, and $Exp_{A(g)}$ represents the set of explanations in which goal g is marked as abandoned.

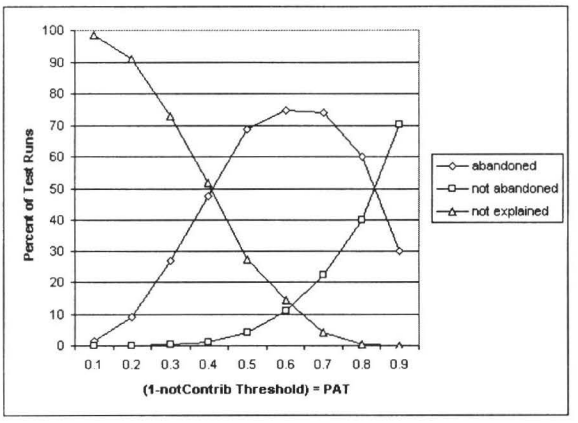

Figure 4: Empirical Accuracy

5 Accuracy

To test this theory we have extended Geib and Goldman's Probabilistic Hostile Agent Task Tracker (PHATT) to estimate goal abandonment. We will not cover the details of the PHATT algorithm here; instead we refer the interested reader to [Geib and Goldman, 2001]. We used a very simple plan library with three root goals each having eight unordered steps. To generate test cases, we chose an ordering for the actions for each of the three goals. These plans were then randomly interleaved preserving the intra-plan ordering. To simulate goal abandonment, at each time step one of the goals could be chosen for abandonment. If chosen, all remaining steps of the plan were removed from the observation stream. When given to PHATT, each of these test sequences produced one of three possible results:

abandoned where PHATT believed with probability greater than 0.5 that the correct goal had been abandoned.

not abandoned where PHATT did not believe believe with probability greater than 0.5 that the correct goal had been abandoned

not explained where PHATT was unable to explain the set of observations. In all cases, this was a result of the system believing that a goal was abandoned before it actually had been. The system was therefore unable to account for the remaining actions in that test data point.

The results of one thousand such randomly generated data points at each of nine $notContrib$ threshold values between 0.1 and 0.9 can be seen in Figure 4. To aid in understanding, the X-axis plots (1 - the notContrib threshold) which we will call the *probability of abandonment threshold* (PAT). The y-axis plots the percentage of test points for each of the possible results at that threshold value. The results confirm our intuitions.

The number of test sequences that are **not explained** drops to zero as the PAT is raised. The PAT is specifying how much evidence the algorithm needs to have before it can consider a goal abandoned. Since the system's failure to explain a test sequence is a result of prematurely believing a goal has been abandoned, as the PAT rises and more evidence is required this number should drop to zero.

The number of test sequences that are **not abandoned** rises as the PAT is raised. As the threshold rises the system requires more and more evidence to be convinced that the goal has been abandoned. This allows more data points to reach the end of the observation sequence without being convinced of the goal's abandonment.

Finally, **abandoned** rises and peaks giving the algorithm a maximum accuracy of about seventy five percent at a PAT of between 0.6 and 0.7. The algorithm's subsequent dip is again a result of the increasing confidence in abandonment required by the rising PAT. As with **not abandoned** at this point the system's accuracy is falling prey to the limited length of the test sequences. Since each test sequence is of limited length the test runs are ending before enough evidence can be observed for the higher PAT values. Figure 3 will show us why.

Consider the curve in Figure 3 labeled "1 of 5." This is approximately the ratio of contributing actions to the size of the pending set in this example. At a PAT of 0.9 the system will require approximately ten actions in a row that do not contribute to the goal in order to convince itself of the goals abandonment. Since the longest any of the tests can be is twenty three actions if the plan is abandoned more than half way through the observation stream it will have a hard time producing enough evidence to convince the system.

6 Conclusions

This paper presents a solution to the problem of recognizing when an agent has abandoned a goals based on probabilistic model revision. A number of issues are covered in more detail in the full version of this paper available from the author.

References

[Bui *et al.*, 2002] Hung H. Bui, Svetha Venkatesh, and Geoff West. Policy recognition in the abstract hidden markov model. In *Technical Report 4/2000 School of Computer Science, Curtin University of Technology*, 2002.

[Erol *et al.*, 1994] Kutluhan Erol, James Hendler, and Dana S. Nau. 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)*, pages 249–254, 1994.

[Geib and Goldman, 2001] Christopher W. Geib and Robert P. Goldman. Plan recognition in intrusion detection systems. In *Proceedings of DISCEX II, 2001*, 2001.

[Goldman *et al.*, 1999] Robert P. Goldman, Christopher W. Geib, and Christopher A. Miller. A new model of plan recognition. In *Proceedings of the 1999 Conference on Uncertainty in Artificial Intelligence*, 1999.

[Jensen *et al.*, 1990] Finn Verner Jensen, Bo Chamberlain, Torsten Nordahl, and Frank Jensen. Analysis in hugin of data conflict. In *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence*, 1990.

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