RECOGNIZING AGENT PLAN/GOAL ABANDONMENT

A significant extension to work on abductive probabilistic plan recognition

Honeywell

PROBLEM

The ability to recognize when an agent abandons a plan or goal is important to successful implementation of plan recognition in real systems.

HISTORICAL APPROACHES

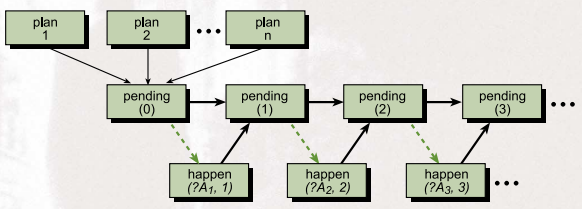
Existing plan recognition systems have adopted a number of methods to work around the problem.

- Only worry about the agent's current goal [Horvitz *et al*., 1998].
- Assume the agent only has one goal [Conati *et al*.,1997; Gertner *et al*., 1998]
- Assume the agent will always come back [Katz and Allen, 1986]
- Rely on a cooperative agent for disambiguation

THE EXACT SOLUTION

Background

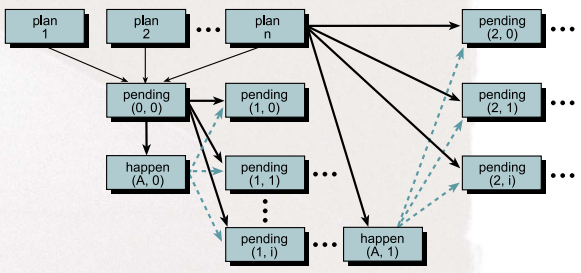
[Goldman *et al*., 2002; Geib and Goldman, 2001b; 2001b] provide an algorithm for plan recognition based on a model of the execution of hierarchical task network (HTN) plans. As the agent executes one of the set of pending actions, a new set of pending actions is generated from which the next action will be chosen, and so forth.



For each valid explanatory hypothesis (set of goals, chosen plans, observed actions), there is at least one corresponding series of pending sets which are generated from previous sets by removing the action just executed and adding actions enabled by the executed action.

Introducing plan/goal abandonment in the model critically changes the way in which pending sets are generated.

The execution of an action no longer generates a unique new pending set because the agent also chooses a set (possibly the empty set) of goals to be abandon, greatly multiplying the possible alternatives.



Thus, computing the exact probability of an explanation considering goal abandonment presents two problems:

- The existing search space is multiplied by 2^n (the number of possible sets of abandoned goals)
- The need for a probability distribution over the set of goals the agent could abandon

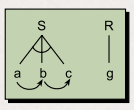
These issues makes explicitly computing the exact probability of goal abandonment unrealistic.

APPROXIMATION BY MODEL REVISION

Laskey [1991], Jensen [Jensen et al., 1990], Habbema [1976] have suggested that unexpectedly small values for the statistic of *P(observations\model)* indicate a model mismatch. In our case this mismatch can be interpreted as the abandonment of a goal.

In this light, consider computing 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*), (we denote this as *P(notContrib(g,s,t)|model, observations*)). If this probability gets unexpectedly small, it is evidence of a mismatch between the model which predicts that the agent is still working on the goal and the reality that the agent has abandoned it.

Computing "notContrib"



Given this very simple plan library and assuming the following sequence of observations: *happen (a,0), happen (b,1), happen (g,2), happen (g,3)*, 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* is then: $1 - (m / |PS_2|)$

To handle more than one time step and to handle partially ordered plans, this formula must be generalized slightly.

If $m_{G,i}$ represents the number of elements in the pending set at time *i* that contribute to goal *G*, *then (s-1)* is the last time we saw an action contribute to *G* and *t* is the current time, *P(notContrib(G,s,t)|model,obs)* = $\prod_{i=s}^{t} 1 - (m_{G,i} / |PS_i|)$

By allowing the user to set a threshold on this statistic and marking any explanation that drops below this threshold as abandoned we can now generate explanations that involve the abandonment of goals.

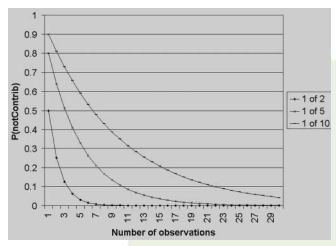
Estimating abandonment

Estimating the probability that a given goal has actually been abandoned requires a further step. If one computes *P(notContrib(G,s,t)|model, obs)* for each *G* and threshold, one can produce explanations of the observations in which goals have been abandoned and estimate the probability of a specific goal's abandonment:

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

Evidential Theory

Note 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.

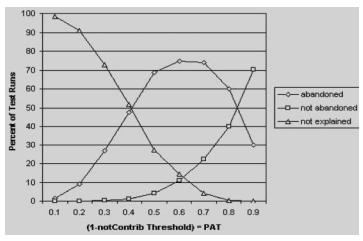


Theoretical Evidential Curves
Three theoretical curves for the probability of "*notContrib*" for different sizes of the pending set with only a single action contributing to the desired goal.

IMPLEMENTATION

Accuracy

Investigations into the algorithm's accuracy employed a very simple plan library with three root goals, each having eight unordered steps. These plans were randomly interleaved preserving the intra-plan ordering. At each time step, one of the goals could be chosen for abandonment. A thousand randomly generated data points were run, at each of nine *notContrib* threshold values between 0.1 and 0.9.



Empirical Accuracy

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

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

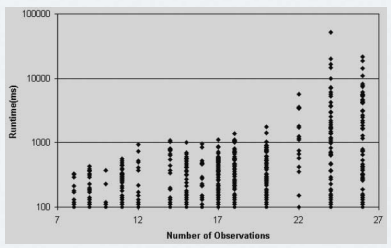
not explained—where PHATT was unable to explain the set of observations. In all cases, this was a result of the system believing that a goal was abandoned before it actually had been. The system was therefore unable to account for the remaining actions in that test data point.

Note that the number of test sequences "not explained" drops to zero as the PAT is raised. Likewise, the number "not abandoned" rises as the PAT (Probability of Abandonment Threshold) rises exactly as expected.

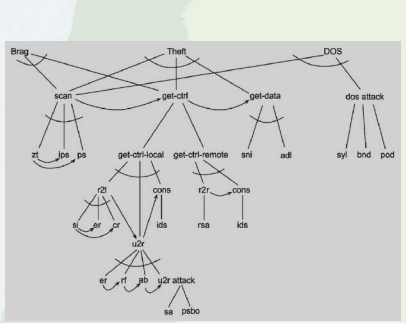
Runtime

Runtime tests were done using the test library:

The chart below shows seventeen hundred randomly generated two-goal observation streams with the PAT set at 0.75. The system believed 1138 of the runs had at least one abandoned goal, but still completed the majority of runs in less than one second.



Runtimes of 1700 Runs with 2 Goals Each



Plan/Goal Library Used to Generate Runtimes

C03111