Extended Abstract: Partial Observability in Collaborative Task Tracking

Christopher W. Geib and Robert P. Goldman

Honeywell Labs
3660 Technology Drive.
Minneapolis, Minnesota 55418
geib{goldman}@htc.honeywell.com

Introduction

Team-level intent recognition presents new and interesting challenges for the task tracking research and violates some of the implicit and explicit assumptions behind much existing research. While our work hasn't focused specifically on the problem of inferring team intent, we have looked at environments that have many of the same properties. In this extended abstract we briefly discuss one such environment and one of the central problems it presents. We then briefly sketch a plan recognition algorithm we have been working on that provides some leverage on this problem.

The Environment

Consider the case of a team of workers that are operating an oil refinery. Traditionally these teams are split into a hierarchy of at least three levels. First are *main operators* working within a centralized control room that has overall visibility and control across the whole of the plant. A second level, *subsystem operators*, move about the plant but are most often at one of the smaller control facilities/sheds. These satellite facilities provide visibility and control of particular subsystems. Third are *field operators* outside working on the actual hardware of the refinery that can affect individual control points for the refinery. For the purpose of this discussion, we will assume a team of three members made up of one of each of these kinds of operators.

There are some important points to note about this domain. First, the most reliable form of communication for this team is over a two-way radio. While the main and subsystem operators are often at a computer console where they may take control actions, most of their communication and coordination is not done through the computer but instead by radio. The field operator may and or may not have a connection into the computerized control system, but may always be reached by radio.

Second, while it is not obvious from this description, there are very few actions that could *not* be performed by any one of the three operators. While the different operators may make these changes through very different means, they are all able to take many of the same control actions. Finally, notice that while there is a hierarchy within the team of operators with the main operator being senior to the subsystem operator who is senior to the field operator, local conditions can force violation of this chain of command.

We will assume that for the purpose of this discussion that the goals of the team members are the same as the goals of the team. In the case of an oil refinery this is reasonable.

In the following we will assume a traditional Hierarchical Task Network (HTN) plan library (Erol, Hendler & Nau, 1994a,b) of the kind used in much work on plan recognition. This library is an And/Or tree representing the plans that are available to the team to achieve their objectives. We will assume the plan library also contains ordering constraints between the steps of each task network (plan or method).

Complete Observability

Consider the case of the main operator closing a valve remotely via his control panel. In this case, an intent recognition system watching the operator's actions would "see" the control action taken and would be able to use it to infer the plans and goals of the team. In contrast, consider the field operator closing the same valve by hand. While in a well-instrumented plant the main operator (and any intent recognizing computer system) will be able to observe the effects of this action, no control action will be seen by the system. Instead this would simply be registered as a change in the system's state. The field operator's actions, even under instruction from the main operator, are not mediated by the control system. To make matters worse, in less automated plants, the state of the valve and even the proximal effects of closing the valve may not be observable by the control system. In this case, only the effects that the closing has on the system may be observed. Those effects will be observed in parts of the plant that are causally "downstream." Thus the effect of the action may only be observed a considerable time after it is actually taken.

In sum, the actions of our team members are not all mediated by a computer. In fact, they are sometimes not even taking actions whose effects are directly observable by a computer.

We can no longer assume that a complete set of observations of the important actions will be available as input. In this domain, and other team based domains, it is critical to infer the execution of actions that are not directly observed but are implied by other observations. In our previous work (Geib and Goldman 2001a) we have identified two kinds of information that provide us grounds to infer the execution of unobserved actions: the observation of *unenabled actions*, and observations of state changes.

We define an *unenabled action* as an observed action whose preconditions (as given in the ordering constraints of the plan graph) have **not** been observed prior to the observation of the action. The plan graph has ordering constraints between the observed action and one or more actions (more generally, subgoals) that have not been observed. If we assume the plan library is a complete representation of the team's available plans, the execution of such an unenabled action gives us good reason to believe the enabling actions must have been performed without being observed.

Observations of state changes provide similar evidence for unobserved actions. If our representation of actions includes the effects of the action, we can use reports of state changes to infer the occurrence of unobserved actions. In particular, we can hypothesize the occurrence of actions that could cause the state changes. Observations of state changes have not played a part in previous plan recognition work. The loss of a complete observability makes it critical to include this information stream.

In our previous work (Goldman, Geib & Miller, 1999, Geib and Goldman, 2001a 2001b), we have put forward a theory and implementation of plan recognition that handles the issues of partial observability. We discuss this approach in the following section.

Our Approach

Our plan recognition framework PHATT (Probabilistic Hostile Agent Task Tracker) is based on the realization that plans are executed dynamically and that at any given moment the agent is able to choose to execute any of the actions that have been enabled by its previous actions. To formalize this slightly, 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 the set of pending primitive actions. As the episode proceeds, the agent will repeatedly execute one of the pending actions, and generate a new set of pending actions from which further actions will be chosen.

The new 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 1. To provide some intuition for the probabilistically-inclined, the sequence of pending sets can be seen as a Markov chain, and the addition of the action executions with unobserved actions makes it a hidden Markov model.

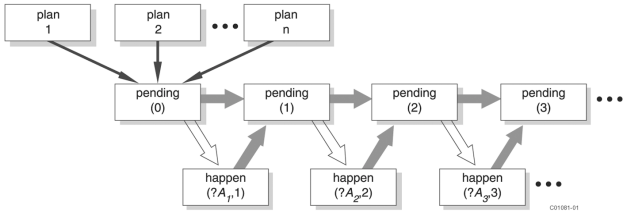

Figure 1: A simple model of pending set generation

This view of plan execution provides a simple conceptual model for the generation of execution traces. To use this model to perform probabilistic plan recognition, we use the observations of the agent's actions as an execution trace. By stepping forward through the trace, and hypothesizing goals the agent may have, we can generate the agent's resulting pending sets. Once we have reached the end of the execution trace we will have the complete set of pending sets that are consistent with the observed actions and the sets of hypothesized goals that go with each of these sets. Once we have this set, we establish a probability distribution over it. We can then determine which of the possible goals the agent is most likely pursuing.

Notice that observations of the agent's actions are used to construct the execution traces. As we have pointed out, in cases of team intent recognition, the observations will not, in general, be a complete record of the execution trace. Instead it will be necessary to consider execution traces containing unobserved actions. For each set of observations we must construct a *set* of possible execution traces, inserting hypothesized unobserved actions to complete them.

Since any of the actions in the agent's pending set could be added to the hypothesized execution trace, this process results in a significant expansion of the search space. This space can be pruned using ordering constraints provided by the observations. We need only consider execution traces that are consistent with the observation stream. Therefore, we reject sequences that do not contain all the observed actions, and those in which actions appear in different order. As we will see, the execution traces are also filtered to be consistent with the unobserved actions that are implied by unenabled actions and observed state changes.

Inferring Unobserved actions

Beyond the completely observable cases we have discussed so far, PHATT is designed to infer the goals of an agent given that his/her behavior is only partially observable. The central idea behind this work is the production of a probability distribution over the set of all pending sets. This is generated using the observations as an execution trace of the agent's actions. Since each pending set represents the results of at least one execution trace, we generated the pending sets by stepping through

observations. In the case of complete and correct observations, this is sufficient. However, in the case of only partially observed action sequences, the observation stream no longer represents the complete execution trace. Instead, for each set of observations we must construct a **set** of possible execution traces, inserting hypothesized unobserved actions to complete them.

Since any of the actions in the agent's pending set could be added to the hypothesized execution trace, this process results in a significant expansion of the search space. This space can be pruned using ordering constraints provided by the observations. We are only interested in execution traces consistent with the observations. Therefore, if a sequence does not contain all the observed actions or doesn't obey the ordering constraints imposed by the sequence or plan library, it cannot generate one of the pending sets we are interested in and can be filtered from consideration. The execution traces are also filtered to be consistent with the unobserved actions that are implied by unenabled actions and observed state changes.

To summarize, we handle unobserved actions by extending the observed sequence of actions with hypothesized unobserved actions consistent with the observed actions, observed state changes, and the plan graph to create a set of possible execution traces. Then we follow the plan recognition algorithm as before. We use the set of execution traces to construct the pending sets and then the probability distribution over the sets of hypotheses of goals and plans implicated by each of the traces and pending sets.

Bounding the number of unobserved actions

The algorithm that we have sketched does not provide a mechanism for determining when to stop hypothesizing unobserved actions. As it stands this algorithm will generate and infinite stream of explanations of ever increasing length. To know when to stop adding actions to the explanations we compute the likelihood that various actions have been executed and were not observed.

Not all actions are equally likely to be executed without detection. Some actions are harder to hide than others. For example, the probability that a person could make a small change in the amount of air entering one of the refinery's subsystems undetected is much higher than the probability that they could successfully shut down the entire refinery unnoticed.

By capturing the probability that an action can be executed without observation, it is possible for our algorithm to generate the probability of a sequence of actions being executed unobserved. However, computing the probability of the unobserved actions in the execution trace is only half of the solution. We must also bound the probability of the unobserved actions within an explanation that we are willing to accept.

If no threshold were placed on the likelihood of the unobserved actions, the process of inferring actions would proceed to successively more and more unlikely explanations by adding more and more unobserved actions. To prevent the generation of this infinite sequence of ever less likely explanations, we require that the user provide a threshold probability value for the unobserved actions. We then build the explanation traces and compute the likelihood of the unobserved actions being executed unnoticed. If the addition of an unobserved action would drop the probability that the set of unobserved actions were executed unnoticed below the user's threshold, the explanation is not considered.

Assumptions

In our implementation of this algorithm we have assumed the given observations are true and correctly ordered. Thus if we have a sequence of three observations: **a, b, c**, we know **a** happened before **b**, which happened before **c**.

We also assume we need not question the validity of observations. However, there are environments where this assumption must be questioned. Consider a military example, if we receive a report of troops massing at a particular location, we must first determine the validity of the report before considering the effect this would have on our assessment of the enemy's goals. It is straightforward to complicate the model by including a traditional model of noisy observations.

Summary

We handle unobserved actions by extending the observed sequence of actions with hypothesized unobserved actions consistent with the observed actions, observed state changes, and the plan graph to create a set of possible execution traces. Then we follow the plan recognition algorithm as before. We use the set of execution traces to construct the pending sets and then the probability distribution over the sets of hypotheses of goals and plans implicated by each of the traces and pending sets.

To prevent the generation of an infinite sequence of ever less likely explanations, we also require two things. First, we compute the probability that the unobserved actions in the sequence could actually be performed without being observed. Second, we bound the probability of the explanations that we are willing to accept for our execution traces. In effect, we extend our algorithm to allow the user to specify how unlikely an explanation they are willing to accept and then use this bound to limit the unobserved actions that are added to the execution trace.

This approach has the added benefit that unobserved actions are added relative to the likelihood of their being executed unobserved. So, for example, small short term fluctuations in temperature, that don't disturb the process, might be inserted in an execution trace while the loss of a primary feed pump, or the loss of input oxygen, that is easy to observe, cannot be added to the trace. We refer the interested reader to (Geib & Goldman, 2001a,b) for a more complete discussion of these issues.

References

Kutluhan Erol, K., Hendler, J. and Nau, D.S., "UMCP: A Sound and Complete Procedure for Hierarchical Task Network Planning," *Artificial Intelligence Planning Systems: Proceedings of the Second International Conference (AIPS-94)*, Chicago, 1994.

Kutluhan Erol, K., Hendler, J. and Nau, D.S., "HTN Planning: Complexity and Expressivity," *Artificial Intelligence Planning Systems: Proceedings of the Second International Conference (AIPS-94)*, Chicago, 1994.

Geib, C.W., and Goldman, R.P., "Probabilistic Plan Recognition for Hostile Agents," *Proceedings of the FLAIRS01 Conference*, Key West, 2001a.

Geib, C.W. and Goldman, R.P., "Plan Recognition in Intrusion Detection Systems," *Proceedings of the DISCEX-II Conference*, 2001b.

Goldman, R.P., Geib, C.W., and Miller, C.A., "A New Model of Plan Recognition," In *Proceedings of the 1999 Conference on Uncertainty in Artificial Intelligence*, Stockholm, 1999.