# Hybrid Symbolic-Probabilistic Plan Recognizer : Initial steps

**Dorit Avrahami-Zilberbrand** and **Gal A. Kaminka**
Computer Science Department Bar Ilan University, Israel
{avrahad1,galk,}@cs.biu.ac.il

### Abstract

It is important for agents to model other agents' unobserved plans and goals, based on their observable actions, a process known as plan recognition. Plan recognition often takes the form of matching observations of an agent's actions to a plan-library, a model of possible plans selected by the agent. In this paper, we present efficient algorithms that handle a number of key capabilities implied by plan recognition applications, in the context of hybrid symbolic-probabilistic recognizer. The central idea behind the hybrid approach is to combine the symbolic approach with probabilistic inference: the symbolic recognizer efficiently filters inconsistent hypotheses, passing only the consistent hypotheses to a probabilistic inference engine. There are few investigations that utilize an hybrid symbolic-probabilistic approach. The advantage of this kind of inference is potentially enormous. First, it can be highly efficient. Second, it can efficiently deal with richer class of plan recognition challenges, such as recognition based on duration of behaviors, recognition despite intermittently lost observations, and recognition of interleaved plans.

## Introduction

Plan recognition (Kautz & Allen 1986; Charniak & Goldman 1993; Carrbery 2001) focuses on mechanisms for recognizing the unobservable state of an agent, given observations of its interaction with its environment. Most approaches to plan recognition utilize a plan library, which encodes the behavioral repertoire of a typical observed agent. Observations are matched against this plan library in sequence, and resulting recognition hypotheses are often ranked according to their likelihood or via some other ranking method. In general, plan recognition libraries have a complex structure, and may explain a large number of possible resulting observation sequences.

The ability to perform plan recognition can be useful in a wide range of applications. Such applications include intrusion detection applications (Geib & Harp 2004), virtual training environments (Tambe & Rosenbloom 1995), visual monitoring (Bui 2003) and suspicious or anomalous behavior (Wu *et al.* 2003; Niu *et al.* 2004; Duong *et al.* 2005b). A number of key capabilities implied by these applications, are either handled in very expensive probabilistic approaches or are not addressed at all by these approaches (see Section for details): (i) handling lossy observations (where an observation or a component of an observation is intermittently lost); (ii) dealing with plan execution duration constraints; and (iii) interleaved plans (where an agent interrupts a plan for another, only to return to the first later).

In this paper we present our initial steps towards an efficient hybrid symbolic-probabilistic recognizer that handles these challenges. We build our hybrid recognizer on highly efficient symbolic algorithms presented in earlier published work (Avrahami-Zilberbrand & Kaminka 2005; Avrahami-Zilberbrand, Kaminka, & Zarosim 2005). To the best of our knowledge, these are the fastest *symbolic* plan-recognition algorithms today. The benefit of a symbolic recognizer is that it can act as a highly efficient filter, which only lets through valid hypotheses. A probabilistic reasoner will then rank the remaining hypotheses based on their likelihood.

Existing approaches often do not consider combining symbolic inference to reduce complexity and to allow richer class of plan recognition inference, as we propose to do. Pure probabilistic recognizers are much more computationally complex than their symbolic counterparts. In the hybrid recognizer, the probabilistic recognition components will only be ranking a small number of hypotheses, and thus we expect the hybrid recognizer to combine the best of both worlds.

## Background and Related Work

There has been considerable research exploring plan recognition. Here we only address those efforts that relate directly to the challenges addressed in this paper. There are few investigations that utilize an hybrid symbolic-probabilistic approach. (Geib & Harp 2004) developed PHATT, a hybrid recognizer, where a symbolic algorithm filters inconsistent hypotheses before they are considered probabilistically. PHATT assumes instantaneous, atomic actions, and takes a generate-and-test approach: With each observation, the symbolic algorithm generates a *pending set* of possible expected observations, which are matched against the next observation to maintain correct state history hypotheses. The size of the pending set may grow exponentially (Geib 2004). In contrast, our work incrementally main-

tains hypotheses implicitly, without predicting impending observations. Moreover, our system is taking durations into account, and addresses efficient matching of (lossy) multi-feature observations.

YOYO* (Kaminka, Pynadath, & Tambe 2002) is a hybrid-probabilistic plan recognition algorithm for multi-agent overhearing. The plan-library used by YOYO* includes information about the average duration of plan steps, which is used to calculate the likelihood of an agent terminating one step and selecting another without being observed to do so. In this, YOYO* addressed missing observations (though their likelihood of becoming lost is to be provided a-priori). However, in contrast to our work, YOYO* did not address matching multi-feature observations (where some features may be intermittently lost), not did it allow for interleaved plans.

Our work differs significantly from probabilistic approaches, though it complements them in principle (as demonstrated by Geib et al.). The first probabilistic plan recognition system was described in (Charniak & Goldman 1993) using Bayesian Networks (BN). Later work (Albrecht, Zukerman, & Nicholson 1997; Xiang & Gong 2003) has utilized Dynamic Bayesian Networks (DBN) for the same purpose. It is known that the exact inference of BN is intractable with respect to the network size ((Cooper 1990)), as it is in DBN (Kjærulff 1992; Boyen & Koller 1998).

There have been much work that utilize Hidden Markov Models and its extensions ((Rabiner 1989; Ghahramani & Jordan 1997; Han & Veloso 1999; Bui, Venkatesh, & West 2002; Brand, Oliver, & Pentland 1997; Murphy 2002; Fine, Singer, & Tishby 1998)). An *HMM* is the most simple case of *DBN*, where in each time slice there is only a single state variable and an observation node. HMMs explicitly represent uncertainty in observations. The complexity of the HMM is $O(TN^2)$ where $N$ is the number of the states and $T$ is the observation length. However, this model does not handle reactive recognition, nor different execution duration of plans, lossy observations, etc.

There has been recent work on using hidden semi-Markov models (HSMMs) in recognizing plans (Duong *et al.* 2005a). Hidden semi-Markov models allow for providing some probabilistic constraints over the duration of plans. However, the model does not allow for interleaved activities, nor does it address matching with multi-feature observations.

None of these probabilistic approaches consider combining symbolic inference to reduce complexity and to allow richer class of plan recognition inference, as we propose to do. Note that the hybrid recognizer we propose will complement these approaches, by allowing them to focus the computational efforts on (the small number of) hypotheses whose probability is greater than zero.

## Fast and Complete Symbolic Plan Recognition

We focus on a specific model of symbolic plan recognition, briefly described below. The reader is referred to (Avrahami-Zilberbrand & Kaminka 2005; Avrahami-Zilberbrand, Kaminka, & Zarosim 2005) for details.

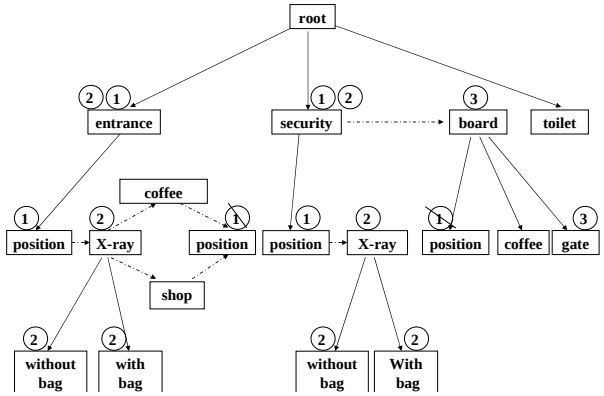


Figure 1: **Example plan library. Circled numbers denote timestamps.**

The plan library is a single-root directed acyclic connected graph, where vertices denote *plan steps*, and edges can be of two types: vertical edges decompose plan steps into sub-steps, and sequential edges specify the expected temporal order of execution. Each plan has an associated set of conditions on observable features of the agent and its actions. When these conditions hold, the observations are said to match the plan. At any given time, the observed agent is assumed to be executing a *plan decomposition path*, root-to-leaf through decomposition edges.

Figure 1 shows an example portion of a plan library, inspired by the plan hierarchies of RoboCup soccer teams (e.g. (Kaminka & Tambe 2000)). An observed agent is assumed to change its internal state in two ways. First, it may follow a sequential edge to the next plan step. Second, it may reactively interrupt plan execution at any time, and select a new (first) plan (we later address the case of interleaving, where the agent may resume an interrupted plan sequence).

The recognizer operates as follow: First, it matches observations to specific plan steps in the library. Then, after matching plan steps are found, they are tagged by the timestamp of the observation. These tags are then propagated up the plan library, so that complete plan-paths (root to leaf) are tagged to indicate they constitute hypotheses as to the internal state of the observed agent when the observations were made. The propagation process tags paths along decomposition edges. However, the propagation process is not a simple matter of following from child to parent. A plan may match the current observation, yet be *temporally inconsistent*, when a history of observations is considered.

A plan is temporally consistent in time stamp $t$ if one of three cases holds: (a) the node in question was tagged at time $t-1$ (i.e., it is continuing in a self-cycle); or (b) the node follows a sequential edge from a plan that was successfully tagged at time $t-1$; or (c) the node is a first child (there is no sequential edge leading into it). A first child may be selected at any time (e.g., if another plan was interrupted). If neither of these cases is applicable, then the node is not part of a temporally-consistent hypothesis, and its tag should be deleted, along with all tags that it has generated in climbing

up the graph. The tags made on the plan-library are used to save information from one run to the next.

Figure 1 shows the process in action (the circled numbers in the figure denote the time-stamps). Assume that the matching algorithm matches at time $t = 1$ the multiple instances of the $position$ plan. At time $t = 1$, Propagate begins with the four $position$ instances. It immediately fails to tag the instance that follows $coffee$ and $shop$, since these were not tagged at $t = 0$. The $position$ instance under $board$ is initially tagged, but in propagating the tag up, the parent $board$ fails, because it follows $security$, and $security$ is not tagged $t = 0$. Therefore, all tags $t = 1$ will be removed from $board$ and its child $position$. The two remaining instances successfully tag up and down, and result in possible hypotheses $root \rightarrow entrance \rightarrow position$ and $root \rightarrow security \rightarrow position$.

The basic model described above may be used to recognize plan(s) whose execution follows in some order, yet can be interrupted. It also allows for plans to have self-cycles, and thus be non-instantaneous. However, it cannot recognize more complex temporal behavior, such as maintaining the selection of a specific plan-step within some bounded interval, or interrupting a sequence of plan steps under one node, to execute another, only to return to it to the first sequence later (plan interleaving). We briefly described these extensions below. The reader is referred to (Avrahami-Zilberbrand, Kaminka, & Zarosim 2005) for details.

**Managing Durations** Instances of the same plan step can vary in the duration of their execution. For example, depending on the line to the security check, a passenger may take a long time or short time to execute the $position$ plan in Figure 1. As a result, we may have multiple observation time-stamps $(t, t+1, \ldots t+k)$ that are all consistent with a single plan, and only reflect the duration that its execution requires between one and $k + 1$ time-stamps. However, often some bounds are known on execution duration. For instance, in an airport terminal, there exist a difference in the plans of a passenger who stands at the check-in area for a few minutes, and a passenger who is held there for half an hour. We thus want to take into account constraints on the duration of plan-steps. To handle different execution duration the constraints in the propagation algorithm are modified as described in (Avrahami-Zilberbrand, Kaminka, & Zarosim 2005).

**Interleaved plans** Many plan recognition algorithms cannot cope with modeling an agent that is pursuing multiple plans (i.e., for multiple goals), by interleaving plan steps. Here, the agent may begin with one plan, and interrupt its execution to execute another, only to return to the remaining plan steps in the first plan. This challenge is handled by the symbolic algorithm by adding a $memoryFlag$ in each of the first children. this flag will hold the latest time-stamp tag, in the sequential link chain from this child. This flag is used to disqualify plans that are in the middle of the chain and are not the ones that we paused at.

**Lossy Features** We take each observation to consist of a tuple of observed features, including states of the world that pertain to the agent (e.g.,a person's role – passenger or cop), actions taken (e.g., $shop$), and execution conditions maintained (e.g., $velocity = 200$). In most applications an implicit assumption was made that all relevant features were in fact observable. However, in realistic settings, some features may be intermittently unobservable. For instance, due to a sensor failure, a plan recognition system might only know the position of another agent, but not its velocity or heading. (Avrahami-Zilberbrand, Kaminka, & Zarosim 2005) has shown how to efficiently determine which plans match a set of observations and deals with lossy observations, using structure called LFDT. The LFDT is a decision tree that automatically constructed prior to run-time and allows efficient mapping from observations to plans that may match them.

**Missing Observations** An underlying assumption in LFDT is that every change in internal state (in our terms, change in plan path) is somehow reflected in observations. However, in realistic settings, this assumption is sometimes violated (e.g., in overhearing applications (Kaminka, Pynadath, & Tambe 2002)). Some internal decision-making may be permanently or intermittently unobservable, for all of the plans along a specific plan decomposition path. In this case, an entire observation is essentially missing (all features are unobservable). For example, a passenger that is not in the camera zone for certain amount of time. (Avrahami-Zilberbrand, Kaminka, & Zarosim 2005) proposed some small changes in the propagation algorithms, to allow them to address this difficulty.

## Hybrid Approach: Disambiguating Hypotheses using Probabilities

After getting all *current state hypotheses* from the symbolic recognizer, the next step is to rank these hypotheses, e.g., to determine the maximum-posterior probability hypothesis. Each such hypothesis called *probable current state hypothesis*.

We follow in the footsteps of *Hierarchical Hidden Markov Model* (HHMM) (Fine, Singer, & Tishby 1998) in representing probabilistic information for the plan library. Each plan-step in the plan library denoted by $q_i^d$, where $i$ is the plan-step index and $d$ is the depth in the hierarchy (the depth index of the root is 1 and the index of the last level is D). For each plan step $q_i^d$ in the plan library we define the following probabilities:

The first probability is the probability to follow a sequential edge from the $ith$ plan-step to the $jth$ plan-step, meaning how likely it is that the agent will complete execution of the current plan step and go on to its next plan-step in the sequential chain. For each internal state $q_i^d$, there is a state transition probability matrix denoted by $A^{q^d} = (a_{i,j}^{q^d})$, where $a_{i,j}^{q^d} = P(q_j^{d+1}|q_i^{d+1})$ is the probability of making a horizontal transition from the $ith$ plan-step to the $jth$ plan-step. Note that self cycle edges are also represented by sequential edges probabilistic matrix.

The second probability signifies how likely it is that the agent will interrupt its execution of the current plan step and give back the control to the plan step's parent. This probability is denoted by $a_{i,end}^{q^d}$, the probability that the plan-step i going to the end state and returns the control to its parent $q_i^d$.

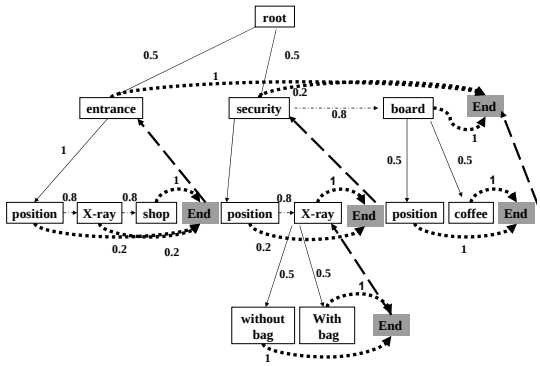


Figure 2: An example of plan library represented as HHMM.

The third probability is the vertical transition, meaning how likely the agent will execute each of the plan's first children. This probability is denoted by $\Pi^{q^d} = \pi^{q^d}(q^{d+1}) = P(q_i^{d+1}|q^d)$, the probability that plan-step $q^d$ will initially activate the plan-step $q_i^{d+1}$. It is also possible to represent the probable duration of a step (e.g., as in (Kaminka, Pynadath, & Tambe 2002; Duong *et al.* 2005b)).

Figure 2 shows portion of the plan library after converting it to HHMM. First, we add an *end state* for each level, then we add edge from each plan-step to this *end state*. This edge represent the probability to interrupt. Each plan-step should also have a self-cycle edge; We omitted this edge, for presentation clarify.

We will use these probabilities to rank the hypotheses that we got from the symbolic recognizer. To calculate the probability for each hypothesis, we traverse the plan library from the leaf of each selected hypothesis in time-stamp $t-1$, to the leaf of each of the current state hypotheses in time stamp $t$ that we got from the symbolic recognizer. While traversing the plan library we multiply the appropriate probabilities (continue or interrupt or move-to-next) for each plan-step from the leaf of the previous path to the leaf of the current path. If there is more then one way to get from the leaf of the previous hypothesis to the leaf of the current hypothesis, then it should be calculated too, and the most probable way should be taken.

$$\hat{X}_i = \arg\max_{X_i} P(X_i|observations) = \arg\max_{X_i} \sum_{W_k \in W} P(W_k) \times P(X_i|W_k) \quad (1)$$

Formally, the symbolic algorithm filtered all previous hypotheses (paths) $W = W_1, W_2, ..., W_r$, and current possible hypotheses (paths) $X = X_1, X_2, ..., X_l$, based on the observations. To calculate the best hypothesis, we need to calculate for each current hypothesis the probability $P(X_i, t+1|observations)$, meaning the probability to get path $X_i$ in time-stamp $t+1$ given sequence of observations, and find the best hypothesis. By assuming a Markov property, we get Equation 1. To calculate it we use the calculated probability in previous time-stamp $P(W_k)$ and multiply it by $P(X_i|W_k)$, for each possible current path $X_i$, where $X_i = x_i^1, ..., x_i^m$ and previous path $W_k = w_k^1, ..., w_k^n$ that the symbolic algorithm had returned (note that the upper index denotes depth in the plan library). This probability is described in Equation 1. Note that we omitted the plan-step index, and left only the depth index, for presentation clarify.

The calculation of this probability is calculated in two cases, for a given $w \in W$ and $x \in X$. In the first case, $x$ and $w$ have a common parent, and $x$ is a path made by first child edges from this common parent. Here, we calculate the probability of climbing up vertices in $w$ (by taking *interrupt* edges) until we hit a common parent to $x$ and $w$, and then climb down (by taking first child decomposition edges) to $x$. In the second case, $x$ is reached by following a sequential edge from a vertex in $w$ to a vertex in $x$.

In both cases, the probability of climbing up from a leaf $g^n$ at depth $n$, to a parent $g^j$ (where $j < n$) is given by

$$\gamma_g^j = \prod_{d=n}^{j} a_{g,\text{end}}^d \quad (2)$$

and the probability of climbing down from a parent $g^j$ to a leaf $g^m$ is given by

$$\beta_g^j = \prod_{i=j}^{m} \pi^{g^i}(g^{i+1}) \quad (3)$$

Note that we omitted the plan-step index, and left only the depth index, for presentation clarity.

Using $\gamma_w^j$ and $\beta_x^j$, and summing over all possible $j$'s, we can calculate the probability for the two cases in which a move from $w_n$ to $x_m$ is possible (Equation 4). The first case is covered in the term $\gamma_w^j \times \text{Eq}(x^j, w^j) \times \beta_x^j$, where the function $\text{Eq}(x^j, w^j)$ returns 1 if $x^j = w^j$, and 0 otherwise. The overall probability calculation covers all ways of interrupting a plan, climb up to a common parent, and then following first-child decompositions to a new plan. The second case is covered by the term $\gamma_w^j \times a_{w,x}^j \times \beta_x^j$, where $a_{w,x}^j$ returns the probability of taking a sequential edge from $w^j$ to $x^j$, given that such an edge exists, and that the observed agent is done in $w_j$.

$$\begin{aligned} P(X_i|W_k) &= \sum_{j=n-1}^{1} [\gamma_w^j \times \text{Eq}(x^j, w^j) \times \beta_x^j \\ &\quad + \gamma_w^j \times a_{w,x}^j \times \beta_x^j] \\ &= \sum_{j=n-1}^{1} \gamma_w^j \beta_x^j [\text{Eq}(x^j, w^j) + a_{w,x}^j] \end{aligned} \quad (4)$$

To see the process in action in the previous example, assume that the symbolic algorithm matches at time $t = 6$ the multiple instances of the *coffee* plan, and at time $t = 5$ the multiple instances of *x-ray*. To calculate the probability of *P(board→coffee — security→x-ray)* we traverse the tree and multiply the probabilities on the arrows. We will get *P(board→coffee — security→x-ray)* = $1 \times 0.8 \times 0.5 = 0.4$

**Algorithm 1** calcProb(SBR $t-1$ results $M$, Plan Library $g$, Time-stamp $t$)

1: **for all** $v \in MnotVisited$ **do**
2: $PropagateProbUpAndDown(v, g, t, root(g))$
3: $Normalize(M, g, t)$
4: $FindMax(M, g, t)$

(interrupting x-ray under security, then following sequential link to board and then choosing coffee). The probability: *P(entrance →coffee — entrance→x-ray) = 0.8* (following sequential link from x-ray to coffee in the entrance). Therefore, the probable option is that the agent drinks coffee in the entrance area, and not in the boarding area. (Pay attention that these results should be multiplied by *P(security→x-ray)* and *P(entrance→x-ray)*. We assumed here that these probabilities are equal, and therefore we omitted them).

We presented above a naive algorithm for calculating the probabilities of the hypotheses in time-stamp $t$. However, this calculation can be expensive, if we go over all leaves of the paths in $t-1$ and for each of these leaves traverse the plan library until getting to all leaves of paths we got in time-stamp $t$. The complexity in the worst case is $O(N^2T)$, where $N$ is the plan library size, and $T$ is the number of observations.

We developed set of algorithms that calculates the probability in Equation 1 in time $O(NDT)$, where $D$ is the depth of the plan library. The central idea behind these algorithms is summing the probabilities for paths in $t-1$ while propagating up along the hierarchy, i.e,. going over the plan library from leaves of paths in $t-1$, and summing up the calculated probabilities. While propagating up and summing the $t-1$ paths, we check if there is a child or a sequential edge to path that is tagged with $t$. If the answer is yes, then we propagate down the calculated probability.

This process is described in Algorithms 1-3. The $calcProb$ algorithm (Algorithm 1) calculates the probability of all valid hypotheses that the symbolic algorithm had returned, normalizes these probabilities and find the hypothesis with the maximum probability. To calculate these probabilities, it goes over all matching paths $M$ in time stamp $t-1$, we got from SBR (which we did not visited yet) and calls algorithm 2. The $PropagateProbUpAndDown$ algorithm (Algorithm 2) calculate $Z$, which holds the probability from paths in $t-1$. It goes up from leaf of $v$, which is one of the paths from $t-1$, and add the probabilities while going up the hierarchy. If there is a child or sequential edge from this path to plan that tagged with time-stamp $t$, it calls to algorithm 3. Algorithm 3 goes down along the hierarchy and multiply probabilities until reaching the leaf of path in $t$. When reaching to leaf node it add it to the probability of this node and save this probability there. Therefore, the probabilities of the leaves are saved in the tree in its leaves, and can be collected, normalized and analyzed later by algorithm 1.

There is a significant difference between the HHMM and our hybrid approach. First, the hybrid approach deals with interleaved execution and duration constraints, which the HHMM cannot handle. An initial extension to include con-

**Algorithm 2** PropagateProbUpAndDown(SBR $t-1$ path $v$, Plan Library $g$, Time-stamp $t$, End Plan $r$)

1: $B \leftarrow leaf(v)$
2: $Z \leftarrow Prob(B, t-1)$
3: $S \leftarrow sequentialEdgeTagged(B, t-1, g, t)$
4: **for all** $s \in S$ **do**
5: $calcDownProb(Z, A_{B,s}, B)$
6: **if** $\exists selfCycleEdgeTagged(B, t-1, B, t)$ **then**
7: $calcDownProb(Z, A_{B,B}, B)$
8: **while** $B \neq r$ **do**
9: $Z \leftarrow Z * A_{B,end}$
10: $B \leftarrow Parent(B)$
11: $C \leftarrow childrenTagged(B, t-1) notVisited$
12: **for all** $c \in C$ **do**
13: $Z \leftarrow Z + PropagateProbUpAndDown(leaf(c), g, t, B)$
14: $C \leftarrow childrenTagged(B, t)$
15: **for all** $c \in C$ **do**
16: **if** $\neg Visited(B)$ **then**
17: $calcDownProb(Z, 1, B)$
18: **else**
19: $MultProbabilityOfPlan(leaf(c), A_{c,end} * \Pi_{B,c})$
20: $S \leftarrow sequentialEdges(B, t)$
21: **for all** $s \in S$ **do**
22: $calcDownProb(Z, A_{B,s}, B)$
23: $return(Z)$

**Algorithm 3** calcDown(probability $Z$, probability $p$, Plan $B$, Plan Library $g$, Time-stamp $t$)

1: $Z \leftarrow Z \times p$
2: **if** $isLeaf(B)$ **then**
3: $AddProbabilityToPlan(B, Z)$
4: **else**
5: $C \leftarrow childrenTagged(B, t)$
6: **for all** $c \in C$ **do**
7: $calcDownProb(Z, \Pi_{B,c} * A_{B,B}, B)$

sideration of duration, the *Switching Hidden Semi-Markov Model (S-HSMM)*, appears in (Duong *et al.* 2005b). However, these methods do not allow for interleaving plans, nor do they allow for lossy observations, either at the level of features or complete set of observations. Second, since the symbolic algorithm had already given us the possible paths, we do not need to consider all possible paths. Hopefully, many paths had been disqualified by the symbolic algorithm, due to ordering constraints or duration constraints. Therefore, we expect that the performance in practice will decrease significantly. Our initial analysis indicates that it may be possible to exploit the hybrid nature of the model to provide exact probabilistic inference at lower complexity than the S-HSMM model.

**Complexity Analysis:** the run-time complexity of the algorithm is $O(TDN)$. We first propagate the $t-1$ probability bottom up the hierarchy, not visiting plans that already been visited $O(N)$ (Algorithm 2, Lines 9–13). Then, calculating down for different depths (Algorithm 2, Lines 14–19), adding to path tagged with $t$ the probability to interrupt and reselect the plan (Algorithm 2 Line 19) is $O(ND)$.

The complexity analysis of HHMM is $O(T^3)$, where T is the length of the observation sequence. (Murphy & Paskin 2001) showed how to reduce the run-time complexity to $O(TQ^{2D})$ by representing HHMM as dynamic baysian network (DBN), where Q is the maximum states in each level. And reduced it even to $O(TDQ^D$ by using approximate DBN inference techniques. In contrast, our work has the run-time complexity of $O(TDN) \cong O(TDQ^D)$, in exact inference.

## Summary and Future Work

Agents must often rely on their observations of others, to infer their unobservable internal state, such as goals, plans, or selected plans. This paper addresses efficiency in the context of Hybrid Symbolic-Probabilistic plan recognition, relaying on hierarchical plan-based representation. In the hybrid recognizer, the probabilistic recognition components will only be ranking a small number of hypotheses, and thus we expect the hybrid recognizer to combine the best of symbolic and probabilistic worlds. The algorithms we propose are efficient, and can handle intermittent failures in observations, plans with duration, and lossy observations, both of complete observations and of only a subset of observable features.

**Acknowledgments.** This research was supported by Israel's Ministry of Trade and Commerce and by the AVNET 37 Consortium. Special thanks to Nadav Zilberbrand and K.Ushi.

## References

Albrecht, D.; Zukerman, I.; and Nicholson, A. 1997. Bayesian models for keyhole plan recognition in adventure game. *User Modeling and User-Adapted Interaction* 8(1–2):5–47.

Avrahami-Zilberbrand, D., and Kaminka, G. A. 2005. Fast and complete symbolic plan recognition. In *IJCAI-05*.

Avrahami-Zilberbrand, D.; Kaminka, G. A.; and Zarosim, H. 2005. Fast and complete plan recognition: Allowing for duration, interleaved execution, and lossy observations. In *IJCAI workshop on Modeling Other agents from Observations (MOO-04)*.

Boyen, X., and Koller, D. 1998. Tractable inference for complex stochastic processes. In *Proc. Fourteenth Annual Conference on Uncertainty in AI (UAI)*, 33–42.

Brand, M.; Oliver, N.; and Pentland, A. 1997. Coupled hidden markov models for complex action recognition. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, 994. Washington, DC, USA: IEEE Computer Society.

Bui, H.; Venkatesh, S.; and West, G. 2002. Policy recognition in the abstract hidden markov models. *Journal of Artificial Intelligence Research* 17:451–499.

Bui, H. 2003. A general model for online probabilistic plan recognition. In *IJCAI-03*.

Carrbery, S. 2001. Techniques for plan recognition. *User Modeling and User-Adapted Interaction* 11:31–48.

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

Cooper, G. F. 1990. The computational complexity of probabilistic inference using bayesian belief networks (research note). *Artif. Intell.* 42(2-3):393–405.

Duong, T.; Bui, H. H.; Phung, D.; and Venkatesh, S. 2005a. Activity recognition and abnormality detection with the switching hidden semi-markov models. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR-2005)*.

Duong, T. V.; Bui, H. H.; Phung, D. Q.; and Venkatesh, S. 2005b. Activity recognition and abnormality detection with the switching hidden semi-markov model. In *CVPR (1)*, 838–845.

Fine, S.; Singer, Y.; and Tishby, N. 1998. The hierarchical hidden markov model: Analysis and applications. *Machine Learning* 32(1):41–62.

Geib, C. W., and Harp, S. A. 2004. Empirical analysis of a probalistic task tracking algorithm. In *AAMAS workshop on Modeling Other agents from Observations (MOO-04)*.

Geib, C. W. 2004. Assessing the complexity of plan recognition. In *AAAI-04*.

Ghahramani, Z., and Jordan, M. I. 1997. Factorial hidden Markov models. *Machine Learning* 29:245–275.

Han, K., and Veloso, M. 1999. Automated robot behavior recognition applied to robotic soccer. In *Proceedings of the IJCAI-99 Workshop on Team Behavior and Plan-Recognition*. Also appears in Proceedings of the 9th International Symposium of Robotics Research (ISSR-99).

Kaminka, G. A., and Tambe, M. 2000. Robust multi-agent teams via socially-attentive monitoring. *JAIR* 12:105–147.

Kaminka, G. A.; Pynadath, D. V.; and Tambe, M. 2002. Monitoring teams by overhearing: A multi-agent plan recognition approach. *Journal of Artificial Intelligence Research* 17.

Kautz, H. A., and Allen, J. F. 1986. Generalized plan recognition. In *AAAI-86*, 32–37. AAAI press.

Kjærulff, U. 1992. A computational scheme for reasoning in dynamic probabilistic networks. In *UAI-1992*, 121–129. San Mateo, CA: Morgan Kaufmann.

Murphy, K. P., and Paskin, M. A. 2001. Linear-time inference in hierarchical hmms. In *NIPS*, 833–840.

Murphy, K. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. Dissertation, UC Berkeley.

Niu, W.; Long, J.; Han, D.; and Wang, Y.-F. 2004. Human activity detection and recognition for video surveillance. In *Proceedings of the IEEE Multimedia and Expo Conference*, 719–722.

Rabiner, L. R. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2):257–286.

Tambe, M., and Rosenbloom, P. S. 1995. RESC: An approach to agent tracking in a real-time, dynamic environment. In *IJCAI-95*.

Wu, G.; Wu, Y.; Jiao, L.; Wang, Y.-F.; and Chang, E. Y. 2003. Multi-camera spatio-temporal fusion and biased sequence-data learning for security surveillance. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, 528–538. New York, NY, USA: ACM Press.

Xiang, T., and Gong, S. 2003. On the structure of dynamic bayesian networks for complex scene modelling. In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*.