

# Efficient Approximate Inference for Online Probabilistic Plan Recognition

**Hung H. Bui**

School of Computing Science  
Curtin University of Technology  
PO Box U1987, Perth, WA 6001, Australia  
URL: <http://www.cs.curtin.edu.au/~buihh>  
Email: [buihh@cs.curtin.edu.au](mailto:buihh@cs.curtin.edu.au)

## Abstract

We present a new general framework for online probabilistic plan recognition called the Abstract Hidden Markov Memory Model (AHMEM). The new model is an extension of the existing Abstract Hidden Markov Model to allow the policy to have internal memory which can be updated in a Markov fashion. We show that the AHMEM can represent a richer class of probabilistic plans, and at the same time derive an efficient algorithm for plan recognition in the AHMEM based on the Rao-Blackwellised Particle Filter approximate inference method.

## Introduction

The ability to perform plan recognition can be very useful in a wide range of applications such as monitoring and surveillance, decision supports, and team work. However the plan recognizing agent's task is usually complicated by the uncertainty in the plan refinement process, in the outcomes of actions, and in the agent's observations of the plan. Dealing with these issues in plan recognition is a challenging task, especially when the recognition has to be done online so that the observer can react to the actor's plan in real-time.

The uncertainty problem has been addressed by the seminal work (Charniak & Goldman 1993) which phrases the plan recognition problem as the inference problem in a Bayesian network representing the process of executing the actor's plan. More recent work has considered dynamic models for performing plan recognition online (Pynadath & Wellman 1995; 2000; Goldmand, Geib, & Miller 1999; Huber, Durfee, & Wellman 1994; Albrecht, Zukerman, & Nicholson 1998). While this offers a coherent way of modelling and dealing with various sources of uncertainty in the plan execution model, the computational complexity and scalability of inference is the main issue, especially for dynamic models.

Inference in dynamic models such as the Dynamic Bayesian Networks (DBN) is more difficult than in a static model. Inference in a static network utilizes the sparse structure of the graphical model to make it tractable. In the dynamic case, the DBN belief state that we need to maintain usually does not preserve the conditional independence

properties of the single time-slice network, making exact inference intractable even when the DBN has a sparse structure. Thus, online plan recognition algorithms based on exact inference will run into problems when the belief state becomes too large, and will be unable to scale up to larger or more detailed plan hierarchies.

In order to achieve scalability, we need approximation methods that can utilize the special structure of the plan execution process for efficiency. In our previous work (Bui, Venkatesh, & West 2000a; 2000b), we have proposed a new framework for online plan recognition based on the Abstract Hidden Markov Models (AHMM). The AHMM is a stochastic model for representing the execution of a hierarchy of contingent plans (termed policies). Studying the structure of the AHMM reveals a set of special context-specific independence properties typical of a stochastic plan execution model which can be exploited for efficient inference. We have derived an approximate inference scheme for the AHMM based on the Rao-Blackwellised Particle Filter (RBPF), and shown that this algorithm scales well w.r.t. the number of levels in the plan hierarchy.

While our work on the AHMM is among the first to address the issue of complexity and scalability in online probabilistic plan recognition, the model that we have considered is somewhat limited in its representational power. Among these limitations of the AHMM is the inability to represent uninterrupted sequence of plans and actions. This is due to the fact that each policy in the AHMM is purely reactive on the current state and has no memory. This type of memoryless policies cannot represent a sequence of uninterrupted sub-plans since they have no way of remembering the sub-plan in the sequence that is currently being executed. Thus, the decision to choose the next sub-plan is only dependent on the current state, and not on the sub-plans that have been chosen in the past. Other models for plan recognition such that the Probabilistic State Dependent Grammar (PSDG) (Pynadath & Wellman 2000; Pynadath 1999) are more expressive and do not have this limitation. Unfortunately, the existing exact inference method for the PSDG in (Pynadath 1999) has been found by us to be flawed and inadequate.

The main motivation in this paper is to extend the existing AHMM framework to allow for policies with memories to be considered. We propose an extension of the

AHMM called the Abstract Hidden Markov mEmory Model (AHMEM). The expressiveness of the new model encompasses that of the PSDG (Pynadath & Wellman 2000), thus the new model removes the current restriction of the AHMM. More importantly, we show that the RBPF approximate inference method used for the AHMM can be extended to the more general AHMEM as well, ensuring that the new generalized model remains computationally attractive. To the best of our knowledge, we are the first to provide a scalable inference method for this general type of hierarchical probabilistic plan hierarchy.

The paper will be structured as follows. In section 2, we provide a more detailed discussion of the AHMM, PSDG and related models for online probabilistic plan recognition. We then introduce the generalized model AHMEM in section 3. The computational procedures for dealing with AHMEM is presented in section 4. Finally, we conclude and present directions for future work in section 5.

## Models for Online Probabilistic Plan Recognition

In the AHMM, an agent’s probabilistic plan is modeled by an abstract Markov policy (AMP). An AMP is an extension of a policy in Markov Decision Processes (MDP) defined within a subset of the environment state space so that it can select other more refined AMPs and so on to form a hierarchy of policies. The AMP is thus similar to a contingent plan that prescribes which sub-plan should be invoked at each applicable state of the world to achieve its intended goal, except that it can represent both the uncertainty in the plan refinement and in the outcomes of actions. The AMP model originates from the abstract probabilistic planning literature (Sutton, Precup, & Singh 1999; Parr & Russell 1997; Forestier & Varaiya 1978) and is equivalent to the concept of *options* introduced in (Sutton, Precup, & Singh 1999).

The execution of an AMP leads to a special stochastic process called the *Abstract Markov Model* (AMM). The noisy observation about the environment state can then be modelled by making the state “hidden”, similar to the hidden state in the Hidden Markov Models (Rabiner 1989). The resulting stochastic process is termed the *Abstract Hidden Markov Model*. Intuitively, the AHMM models how an AMP causes the adoption of other policies and actions at different levels of abstraction, which in turn generate a sequence of states and observations. In the plan recognition task, an observer is given an AHMM corresponding to the actor’s plan hierarchy, and is asked to infer about the current policy being executed by the actor at all levels of the hierarchy, taking into account the sequence of observations currently available.

We have shown that the complexity of policy recognition scales reasonably well w.r.t. the number of abstraction levels in the policy hierarchy (Bui, Venkatesh, & West 2000a; 2000b). This is achieved by examining the Dynamic Bayesian Network representation of the AHMM to identify a set of context specific independence (CSI) properties (Boutilier *et al.* 1996) that are inherent in the execution model of an abstract policy. This leads to an im-

portant observation that given the right context, knowing the state of execution at a given level renders the details of the high level plans independent of the realization at the lower levels. Based on this general observation, we have proposed a hybrid sampling-based inference method, a variant of the Rao-Blackwellised particle filter (RBPF) (Doucet *et al.* 2000) to take advantage of these CSI properties in the AHMM, leading to an efficient algorithm for policy recognition. When applied to general DBN inference, Rao-Blackwellisation (Casella & Robert 1996) splits the network into two sets of variables: the set of variables that need to be sampled (termed the Rao-Blackwellising (RB) variables), and the set of remaining variables whose belief state conditioned on the RB variables need to be maintained (termed the Rao-Blackwellised (RB) belief state). The RB variables thus play a similar role to the cut-set variables in cut-set inference (Pearl 1988). The difference is that instead of summing over all the possible values of the cut-set variables which can be intractable, only a number of representative sampled values are used. In addition, we show that the AHMM representation and the hybrid policy recognition algorithm can also utilize a factored representation of the state space (Boutilier, Dearden, & Goldszmidt 2001).

The AHMM is closely related to a model for probabilistic plan recognition called the Probabilistic State-Dependent Grammar (PSDG), independently proposed in (Pynadath 1999; Pynadath & Wellman 2000). The PSDG can be described as the Probabilistic Context Free Grammar (PCFG) (Jelinek, Lafferty, & Mercer 1992), augmented with a state space, and a state transition probability table for each terminal symbol of the PCFG. In addition, the probability of each production rule is made state dependent. As a result, the terminal symbol now acts like primitive actions and the non-terminal symbol chooses its expansion depending on the current state. Our policy hierarchy is equivalent to a special class of PSDG where only production rules of the form  $X \rightarrow YX$  and  $X \rightarrow \emptyset$  are allowed. The first rule models the adoption of a lower level policy  $Y$  by a higher level policy  $X$ , while the second rule models the termination of the policy  $X$ . The PSDG model considered in (Pynadath 1999; Pynadath & Wellman 2000) allows for more general rules of the form  $X \rightarrow Y_1 \dots Y_m X$ , i.e., the recursion symbol must be located at the end of the expansion. Thus in a PSDG, a policy might be expanded into a sequence of policies at the lower level which will be executed one after another before control is returned to the higher level policy.

Although more expressive, the existing computational method for inference with the PSDG remains inadequate. (Pynadath 1999) considered the case where the states are fully observable and proposed an exact method for updating the belief state of the PSDG in a “compact” closed form. The proposed algorithm seemingly gets around the exponential blow up in the size of the belief state. Unfortunately, we have found a flaw in the derivation of the algorithm. Basically, it was assumed that the higher levels in the belief state are independent of the lower levels given the current level. In (Bui, Venkatesh, & West 2000a; 2000b), we have shown that this proposition is true only if we also know the terminating and starting times of all the

current policies. For more details about the flaw in the inference algorithm for PSDG, interested readers are referred to the longer version of this paper (Bui 2002).

Similar to our AHMM and the PSDG, the recent work by (Goldmand, Geib, & Miller 1999) also proposes a detailed model of the plan execution process. Using the rich language of probabilistic Horn abduction, they are able to model more sophisticated plan structures such as interleaved/concurrent plans, partially-ordered plans. However the work serves mainly as a representational framework, and provides no analysis on the complexity of plan recognition in this setting.

Other work in probabilistic plan recognition up to date has employed much coarser models for plan execution. Most have ignored the important influence of the state of the world to the agent’s planning decision (Goldmand, Geib, & Miller 1999) and assume that the agent’s actions can be observed directly and accurately. We note that this kind of simplifying assumptions is needed in previous work so that the computational complexity of performing probabilistic plan recognition remains manageable.

The AHMM and PSDG are also related to other hierarchical temporal models. In abstract probabilistic planning, Parr and Russell proposed a model for representing an abstract probabilistic plan called the Hierarchical Abstract Machines (HAM) (Parr & Russell 1997; Parr 1998). In this model, the policy is replaced by a stochastic finite automaton, which can call other automata at the lower level. The HAM framework allows for machines with arbitrary finite number of machine states and transition probabilities with the constraint that there is no recursion in the calling stack to keep the stack finite. Thus, like the PSDG, HAM can readily represent more complex plans involving sequences of policies, alternative policy paths, etc. The construction of the AHMEM framework in this paper is closely related to the HAM model, similar to the way AHMM is related to the *options* model (Sutton, Precup, & Singh 1999). Despite their representational similarity, our models are intended to use for plan recognition whereas the other models are used to speed up the process of finding the optimal policy for MDP.

If we ignore the state dependency, the DBN structure of the AHMM and PSDG is similar to the structure of the Hierarchical Hidden Markov Model (HHMM) (Fine, Singer, & Tishby 1998; Murphy & Pashkin 2001). However, while the HHMM is a type of Probabilistic Context Free Grammar (PCFG), the AHMM and PSDG are not due to the state dependency in the model.

## The Abstract Hidden Markov Memory Models

The aim of this section is to introduce the Abstract Hidden Markov Memory Models (AHMEM), an extension of the AHMM where the policy can have internal memory. In doing this, we achieve two purposes: (1) introduce a general model for plan recognition whose expressiveness encompasses that of the current AHMM and PSDG models, and (2) retain the computational attractiveness of the current AHMM framework. We first define the AHMEM model in subsection 3.1. The DBN structure of the new model is given in subsection 3.2.

## The Model

Consider an MDP-like model with  $S$  representing the states of the environment, and  $A$  representing the set of primitive actions available to the agent. Each action  $a \in A$  is defined by its transition probability from the current state  $s$  to the next state  $s'$ :  $\sigma_a(s, s')$ . The set of abstract policies will include every primitive actions. Furthermore, the AHMM framework (Bui, Venkatesh, & West 2000a) defines higher level abstract policies on top of a set of policies as follows:

**Definition 1 (Abstract Markov Policy (AMP)).** Let  $\Pi$  be a set of AMPs, an AMP  $\pi^*$  over  $\Pi$  is defined as a tuple  $\langle S_{\pi^*}, D_{\pi^*}, \beta_{\pi^*}, \sigma_{\pi^*} \rangle$  where:

- $S_{\pi^*} \subset \cup_{\pi \in \Pi} S_{\pi}$  is the set of applicable states.
- $D_{\pi^*} \subset \cup_{\pi \in \Pi} D_{\pi}$  is the set of destination states.
- $\beta_{\pi^*} : D_{\pi^*} \rightarrow (0, 1]$  is the set of stopping probabilities such that  $\beta_{\pi^*}(d) = 1, \forall d \in D_{\pi^*} \setminus S_{\pi^*}$ .
- $\sigma_{\pi^*} : S_{\pi^*} \times \Pi \rightarrow [0, 1]$  is the selection function where  $\sigma_{\pi^*}(s, \pi)$  is the probability that  $\pi^*$  selects the policy  $\pi$  at the state  $s$ .

An AMP as defined above is purely reactive, in the sense that it selects the next policy at the lower level based only on the current state  $s$ . This restricts the set of behaviours that an AMP can represent. For example, it will not be able to represent a plan consisting of a few sub-plans, one followed by another regardless of the state sequence. To represent this kind of plans, the agent needs to have some form of internal memory to remember the current stage of execution. Let  $M$  be a set of possible internal memory states. We first extend the definition of our policy to include a memory variable which takes on values in  $M$  and is updated after each stage of execution of the policy.<sup>1</sup>

**Definition 2 (Abstract Markov Policy with Memory (AMPE)).** Let  $\Pi$  be a set of AMPEs, an AMPE  $\pi^*$  over  $\Pi$  is defined as a tuple  $\langle S_{\pi^*}, D_{\pi^*}, \beta_{\pi^*}, \sigma_{\pi^*}, I_{\pi^*}^e, \sigma_{\pi^*}^e \rangle$  where:

- $S_{\pi^*} \subset \cup_{\pi \in \Pi} S_{\pi}$  is the set of applicable states.
- $D_{\pi^*} \subset \cup_{\pi \in \Pi} D_{\pi}$  is the set of destination states.
- $\beta_{\pi^*} : D_{\pi^*} \times M \rightarrow (0, 1]$  is the terminating probability.  $\beta_{\pi^*}(d, m)$  is the probability that the policy  $\pi^*$  would stop if the current state is  $d$  and the current memory value is  $m$ .
- $\sigma_{\pi^*} : S_{\pi^*} \times M \times \Pi \rightarrow [0, 1]$  is the policy selection probability.  $\sigma_{\pi^*}(s, m, \pi)$  is the probability that  $\pi^*$  selects the policy  $\pi$  at the state  $s$  and memory value  $m$ .
- $I_{\pi^*}^e : S_{\pi^*} \times M \rightarrow [0, 1]$  is the initial distribution of memory values.  $I_{\pi^*}^e(s, m)$  is the probability that the initial memory is  $m$  if  $\pi^*$  commences at state  $s$ .
- $\sigma_{\pi^*}^e : S_{\pi^*} \times M \times M \rightarrow [0, 1]$  is the memory transition probability.  $\sigma_{\pi^*}^e(s, m, m')$  is the probability that the next memory value is  $m'$  given that the current memory value is  $m$  and the current state is  $s$ .

<sup>1</sup>One can argue that we can always incorporate the memory variable in the environment state, and hence we gain no extra representational power just by introducing the memory variables. However, incorporating the memory variables in the state variable would blow up the size of the state space and thus defeat our purpose of keeping the model computationally feasible.

Subsequently, we will drop the subscript  $\pi^*$  if there is no confusion about the policy in the context. Note that all the states in  $D \setminus S$  are called terminal states and thus  $\beta(d, m) = 1, \forall m \in M, d \in D \setminus S$ . Also, the policy selection, memory initial and transition probability have to be proper probability distributions, i.e.  $\sum_{\pi \in \Pi} \sigma(s, m, \pi) = 1$ ,  $\sum_{m \in M} I^e(s, m) = 1$ , and  $\sum_{m' \in M} \sigma^e(s, m, m') = 1$ .

When an AMPE  $\pi^*$  is executed from a state  $s$ , it first initialises its memory value  $m$  according to the distribution  $I^e(s, m)$ . Then a policy at the lower level  $\pi$  will be selected according to the distribution  $\sigma(s, m, \pi)$ . This policy  $\pi$  will be executed until it terminates at some state  $s'$ . At the new state  $s'$ , the policy  $\pi^*$  itself will terminate with probability  $\beta(s', m)$ . If it does not terminate, the memory variable will be given a new value  $m'$  according to the transition probability  $\sigma^e(s', m, m')$ . Then a new policy at the lower level  $\pi'$  is selected with probability  $\sigma(s', m', \pi')$  and so on.

The above definition for the AMPE is similar to the HAM language (Parr & Russell 1997) used for abstract probabilistic planning. Each policy can be viewed as a stochastic finite-state machine in the HAM language, and the memory of the policy is equivalent to the internal state of the machine. Our model does not have a ‘‘choice’’ memory state however since all the choices that a policy makes can be specified in the policy selection function  $\sigma(\cdot)$ .

Using the AMPEs, we can construct a hierarchy of abstract policies in the same way as in the AHMM. We start with a set of primitive actions  $\Pi_0$ , and for  $k = 1, \dots, K$  build a set of new AMPEs  $\Pi_k$  on top of  $\Pi_{k-1}$ . Then, if a top-level policy  $\pi^K$  is executed, it invokes a sequence of level-(K-1) policies, each of which invokes a sequence of level-(K-2) policies and so on. A level-1 policy will invoke a sequence of primitive actions which leads to a sequence of states. The dynamic process of executing a top-level AMPE is termed the Abstract Markov Memory Model (AMEM). We can then introduce the hidden states and model the noisy observation of the state by an observation model  $\Pr(o_t | s_t) = \omega(s_t, o_t)$ . The resulting model is termed the Abstract Hidden Markov Memory Model (AHMEM).

Some special cases of the AHMEM are worth mentioning here. First, the AHMM itself is a special AHMEM. The memoryless policies of the AHMM are equivalent to AMPEs where the dependency on the memory variable is ignored (e.g. when  $M$  is a singleton set). The class of PSDG considered in (Pynadath 1999) can also be easily converted to an AHMEM. The terminal symbols in the PSDG are equivalent to the primitive actions. Each non-terminal symbol is equivalent to a memoryless policy. In addition, each sequence  $Y = Y_1 Y_2 \dots Y_n$  encountered on the RHS of a production  $X \rightarrow Y_1 Y_2 \dots Y_n$  or  $X \rightarrow Y_1 Y_2 \dots Y_n X$  is equivalent to a policy whose memory  $m$  taking on the values  $1, \dots, n$ .  $Y$  then simply selects the (memoryless) policy  $Y_i$  if  $m = i$ .

Note that in the AHMEM definition, we assume a balanced policy hierarchy for the ease of presentation (all the actions must appear at the same bottom level). However, we can also specify an unbalanced hierarchy by introducing some dummy policies which are equivalent to primitive actions at the higher levels in the hierarchy.

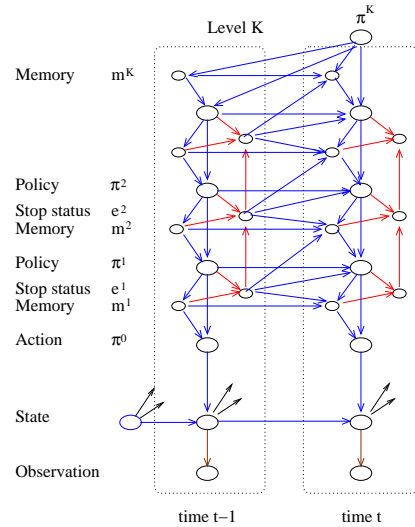


Figure 1: The 2-time-slice DBN for AHMEM

## DBN Representation of the AHMEM

The DBN structure for a AHMEM is very similar to that of an AHMM: at each time slice  $t$ , the variable  $s_t$  represents the current state,  $\pi_t^k$ ,  $k = 0, \dots, K$  represents the current policy at all levels,  $e_t^k$  represents whether  $\pi_t^k$  terminates at the current time, and  $l_t = \max\{k | e_t^k = T\}$  denotes the highest level of termination. In addition to these variables, we need to introduce the memory variable  $m_t^k$  for  $k \geq 1$  to represent the current memory value of the policy  $\pi_t^k$  (the primitive action  $\pi_t^0$  has no memory). The 2-time-slice DBN is given in Fig. 1. The structure of this network is described below. Note that if  $x$  is a variable, we use the notation  $x_{i:j}$  to denote the sequence  $(x_i, x_{i+1}, \dots, x_j)$ , and similarly  $x^{i:j}$  to denote  $(x^i, x^{i+1}, \dots, x^j)$ .

**Policy Termination and Selection** Consider the variable  $e_t^k$ . The parameter at this node can be specified in much the same way as in the network for the AHMM. The parent  $e_t^{k-1}$  plays the role of the context variable:  $e_t^{k-1} = F \Rightarrow e_t^k = F$ , and when  $e_t^{k-1} = T$ ,  $\Pr(e_t^k = T | \pi_t^k, s_t, m_t^k) = \beta_{\pi_t^k}(s_t, m_t^k)$ . Thus, the same CSI property for the AHMM still holds: when  $e_t^{k-1} = F$ ,  $e_t^k$  becomes independent of the remaining parents.

Now consider the variable  $\pi_t^k$ . The parent  $e_{t-1}^k$  is the context variable: if  $e_{t-1}^k = F$ ,  $\pi_t^k = \pi_{t-1}^k$  and  $\pi_t^k$  is independent of the remaining parents; otherwise,  $\Pr(\pi_t^k | \pi_{t-1}^{k+1}, s_{t-1}, m_{t-1}^{k+1}) = \sigma_{\pi_t^k}(s_{t-1}, m_{t-1}^{k+1}, \pi_{t-1}^k)$  and  $\pi_t^k$  is independent of  $\pi_{t-1}^k$ .

**Memory Update** Consider the variable  $m_t^k$ . There are two parents of this node that act like context variables:  $e_{t-1}^{k-1}$  and  $e_{t-1}^k$ . If  $e_{t-1}^{k-1} = F$ , the policy at the lower level has not terminated and memory value is not being updated at this time. Thus,  $m_t^k = m_{t-1}^k$ , and  $m_t^k$  becomes independent of all the remaining parents. If  $e_{t-1}^{k-1} = T$  then there are two cases. If

$e_{t-1}^k = F$ , the policy at level  $k$  continues from the previous time, and thus the memory value will be updated according to the memory transition model:  $\Pr(m_t^k | \pi_t^k, s_{t-1}, m_{t-1}^k) = \sigma_{\pi_t^k}^e(s_{t-1}, m_{t-1}^k, m_t^k)$ . If  $e_{t-1}^k = T$ , the policy at level  $k$  has just started at state  $s_{t-1}$ , and thus the memory value will be initialised:  $\Pr(m_t^k | \pi_t^k, s_{t-1}) = I_{\pi_t^k}^e(s_{t-1}, m_t^k)$ . In this case,  $m_t^k$  is independent of the previous memory value  $m_{t-1}^k$ .

If we ignore the dependency on the state variable, and group the pair of variables  $(\pi_t^k, m_t^{k+1})$  together, we obtain a Hierarchical HMM (Murphy & Pashkin 2001) where the pair  $(\pi_t^k, m_t^{k+1})$  form a level  $k$  state variable in the equivalent HHMM. However, this conversion can only be done if there is no state dependency in our AHMEM.

**Conditional Independence Properties** Even though AMPEs are more expressive than memoryless policies, they remain “autonomous”, in the sense that the higher layers have no influence over the state of an AMPE during its execution. The only way the higher layers can influence the current state of an AMPE is through the conditions at the start: either through the starting state or the starting time. Thus, the conditional independence theorem for policies in the AHMM still holds in this more general setting. We state the theorem for the AHMEM below. The proof for the AHMM case directly extends to this general case.

**Theorem 1.** *Let  $\tau_t^k$  and  $b_t^k$  be two random variables representing the starting time and the starting state, respectively, of the current level- $k$  policy  $\pi_t^k$ :  $\tau_t^k = \max\{t' < t | e_{t'}^k = T\}$  and  $b_t^k = s_{\tau_t^k}$ . Then given the policy  $\pi_t^k$  and its starting state and time, the set of current policies and memories at the higher levels are independent of the set of current policies, memories and state at the lower level:*

$$\pi_t^{k+1:K}, m_t^{k+1:K} \perp \pi_t^{0:k-1}, m_t^{1:k}, s_t | \pi_t^k, b_t^k, \tau_t^k \quad (1)$$

### Approximate Inference for AHMEM

In this section, we look at the online inference problem in the AHMEM. Assume that at time  $t$ , we have a sequence of observations about the environment state  $o_{1:t-1} = (o_1, \dots, o_{t-1})$ . We need to compute the belief state of the DBN which is the joint distribution of all the current variables given this observation sequence:  $\Pr(\pi_t^{0:K}, m_t^{1:K}, s_t, l_t, o_t | o_{1:t-1})$ . From this, we can answer various queries about the current status of the plan execution. For example, the marginal probability of  $\pi_t^k$  tells us about the current policy the actor is executing at some level  $k$ ; the probability  $\Pr(m_t^k | \pi_t^k)$  tells us about the current stage of execution of a policy  $\pi_t^k$ ; the probability  $\Pr(e_t^k | \pi_t^k)$  tells us if  $\pi_t^k$  will end after the current time, etc.

Since there is no compact closed form representation for the above belief state, doing exact inference with the structure of the AHMEM will be intractable when  $K$  is large. However, theorem 1 suggests that we can apply the Rao-Blackwellised Particle Filter (RBPF) to this problem in a similar way as in the AHMM (Bui, Venkatesh, & West 2000b). The idea is to use the sequence of state and terminating status variables as the Rao-Blackwellising (RB) vari-

ables:  $r_t = (s_t, l_t)$ . Two main steps in the RBPF procedure are: (1) updating the Rao-Blackwellised belief state  $\mathcal{B}_t = \Pr(\pi_t^{0:K}, m_t^{1:K}, l_t, s_t, o_t | r_{1:t-1})$  using exact inference, and (2) sampling the RB variable  $r_t$  from the current RB belief state.

### The Rao-Blackwellised Belief State

We first look at the core component of the RB belief state, the conditional joint distribution  $\mathcal{C}_t = \Pr(\pi_t^{0:K}, m_t^{1:K}, s_t | r_{1:t-1})$ . The full RB belief state can be obtained directly from  $\mathcal{C}_t$  by adding in the network representing the conditional distribution of  $o_t$  and  $e_t^{1:K}$ :  $\mathcal{B}_t = \Pr(o_t | s_t) \Pr(e_t^{1:K} | \pi_t^{0:K}, m_t^{1:K}, s_t) \mathcal{C}_t$ . When only memoryless policies are considered, this conditional distribution has a simple chain structure (Bui, Venkatesh, & West 2000a). With the addition of the memory variables, it follows from theorem 1 that  $\mathcal{C}_t$  has the following Bayesian network factorized form:

$$\mathcal{C}_t = \left( \prod_{k=1}^K \Pr(\pi_t^k, m_t^k | \pi_t^{k-1}, r_{1:t-1}) \right) \Pr(\pi_t^0, s_t | r_{1:t-1}) \quad (2)$$

The joint distribution  $\mathcal{C}_t$  thus also has the following potential representation. We first form the set of cliques:  $c_0 = \{\pi_t^0, s_t\}$ ,  $c_k = \{\pi_t^k, m_t^k, \pi_t^{k-1}\}$ ,  $k = 1 \dots, K$ . Note that the set of cliques  $c_{0:K}$  form a chain of cliques in this order, therefore we term  $\mathcal{C}_t$  the *policy-clique chain*.  $\mathcal{C}_t$  can be factored into the product of potentials on these cliques:  $\mathcal{C}_t \propto \prod_{k=0}^K \psi_k(c_k)$ . When  $\psi_k = \Pr(\pi_t^k, m_t^k | \pi_t^{k-1}, r_{1:t-1})$  and  $\psi_0 = \Pr(\pi_t^0, s_t | r_{1:t-1})$ , i.e. when the potential factorization is the same as the directed network factorization in (2), the potentials are said to be in *canonical* form.

Any potential representation of the clique chain can be canonicalized by first perform message passing (exact inference) to compute the marginal at each clique. The canonical form can then be computed directly from these marginals. Once we have the canonical form of the chain  $\mathcal{C}_t$ , it becomes a Bayesian network and can be sampled upward from the bottom level using forward sampling.

### Updating the RB Belief State

The procedure for updating the Rao-Blackwellised belief state as usual has two stages: (1) absorbing the new evidence, i.e. from  $\mathcal{C}_t$ , we need to compute  $\mathcal{B}_{t+} = \Pr(\pi_t^{0:K}, m_t^{1:K} | r_{1:t})$ ; and (2) projecting to the new time slice, i.e. from  $\mathcal{B}_{t+}$ , we need to compute  $\mathcal{C}_{t+1}$ . In principle, we are using exact inference to update the belief state of the DBN when the RB variables are known. Since the RB belief state has a tractable structure, the complexity of maintaining it using exact inference would be minimal. Note that in the AHMM, belief update is done by a series of arc-reversal operations on the directed Bayesian network representation. Here, we provide a straight-forward generalization to networks with memory nodes using a simplified version of exact inference in junction tree (Jensen 1996).

Let us look at the 2-time-slice DBN network again under the context that we know the RB variables  $l_t = l$  and

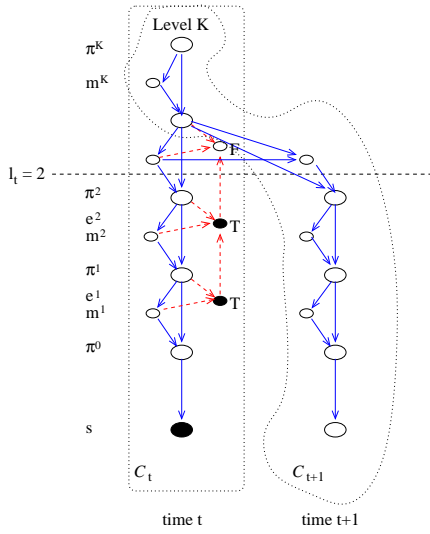


Figure 2: The 2-time-slice DBN conditionalized on the RB variables

$s_t = s$ . Fig. 2 shows an example of this 2-time-slice network when  $l_t = 2$ . Using the context specific independence properties of the AHMEM, all the policies from above level  $l$  remain unchanged, while all the links across time slices from level  $l$  and below can be removed. This greatly simplifies the network structure, allowing the updating operations to be performed efficiently.

The absorbing operation can be done as follows. We obtain the new potentials for  $\mathcal{B}_{t+}$  by absorbing the evidence  $s_t$  and  $e_t^{1:K}$  into the potentials for  $\mathcal{C}_t$ :

$$\begin{aligned} \psi_1 &\leftarrow \psi_1 \times \psi_0(\pi_t^0, s) \\ \psi_k &\leftarrow \psi_k \times \Pr(e_t^k | e_t^{k-1}, s, m_t^k, \pi_t^k) \\ &= \psi_k \times \beta_{\pi_t^k}(s, m_t^k), k = 2, \dots, l \\ \psi_k &\leftarrow \psi_k \times \Pr(e_t^k | e_t^{k-1}, s, m_t^k, \pi_t^k) \\ &= \psi_k \times (1 - \beta_{\pi_t^k}(s, m_t^k)), k = l + 1 \\ \psi_k &\leftarrow \psi_k \times \Pr(e_t^k | e_t^{k-1}, s, m_t^k, \pi_t^k) \\ &= \psi_k, k = l + 2, \dots, K \end{aligned}$$

The projecting operation can be done by adding in time-slice  $t + 1$  to  $\mathcal{B}_{t+}$  (see Fig. 2) and marginalizing the now redundant variables  $\pi_t^{0:l}, m_t^{1:l+1}$  in the old time slice. To add in the new time-slice  $t + 1$ , we form a set of new cliques:  $c_{l+1}^e = \{m_{t+1}^{l+1}, m_{t+1}^{l+1}, \pi_{t+1}^{l+1}\}$ ,  $c_{l+1}^i = \{\pi_{t+1}^{l+1}, m_{t+1}^{l+1}, \pi_{t+1}^{l+1}\}$ ,  $c_k^e = \{\pi_{t+1}^k, m_{t+1}^k, \pi_{t+1}^{k-1}\}$  for  $k = 1, \dots, l$ , and  $c_0^i = \{\pi_{t+1}^0, s_{t+1}\}$ . The potentials for these cliques represent the conditional probability of the variables in the new time-slice given the current time slice and can be obtained directly from the parameters of the policies:

$$\begin{aligned} \psi_{l+1}^e &= \sigma_{\pi_{t+1}^e}^e(s, m_{t+1}^{l+1}, m_{t+1}^{l+1}) \\ \psi_{l+1}^i &= \sigma_{\pi_{t+1}^i}^i(s, m_{t+1}^{l+1}, \pi_{t+1}^l), \end{aligned}$$

```

Input: potentials for  $\mathcal{C}_t$ , evidence  $s$  and  $l$ 
Output: new potentials for  $\mathcal{C}_{t+1}$ 
Begin
/* absorbing */
 $\psi_1 \leftarrow \psi_1 \times \psi_0(\pi_t^0, s)$ 
 $\psi_{l+1} \leftarrow \psi_{l+1} \times (1 - \beta_{\pi_{t+1}^l}(s, m_t^{l+1}))$ 
For  $k = 2, \dots, l$ 
     $\psi_k \leftarrow \psi_k \times \beta_{\pi_t^k}(s, m_t^k)$ 
/* projecting */
Construct  $c_{l+1}^e$  and  $c_{l+1}^i$ 
Perform message passing along the path
     $c_1 \rightarrow c_2 \rightarrow \dots \rightarrow c_{l+1} \rightarrow c_{l+1}^e \rightarrow c_{l+1}^i$ 
 $\psi_{l+1} = \psi_{l+1}^i$ 
 $\psi_0 = \sigma_{\pi_{t+1}^0}^0(s, s_{t+1})$ 
For  $k = 1 \dots l$ 
     $\psi_k = \sigma_{\pi_{t+1}^k}^k(s, m_{t+1}^k, \pi_{t+1}^{k-1}) I_{\pi_{t+1}^k}^e(s, m_{t+1}^k)$ 
End

```

Figure 3: Updating  $\mathcal{C}_t$

$$\begin{aligned} \psi_k^i &= \sigma_{\pi_{t+1}^k}^k(s, m_{t+1}^k, \pi_{t+1}^{k-1}) I_{\pi_{t+1}^k}^e(s, m_{t+1}^k), k = 1, \dots, l \\ \psi_0^i &= \sigma_{\pi_{t+1}^0}^0(s, s_{t+1}) \end{aligned}$$

The set of cliques of the 2-time-slice network now forms a junction tree with  $c_{l+1}^i$  at the junction. To marginalize the variables of the old time-slice, we can perform message passing between the cliques with the schedule  $c_1 \rightarrow c_2 \rightarrow \dots \rightarrow c_{l+1} \rightarrow c_{l+1}^e \rightarrow c_{l+1}^i$ . After this, we obtain the new potentials for  $\mathcal{C}_{t+1}$ : for  $k = 0, \dots, l + 1$ ,  $\psi_k = \psi_k^i$ , and for  $k = l + 2, \dots, K$ ,  $\psi_k$  is unchanged.

The overall algorithm for updating the RB belief state is given in Fig. 3. Since the potentials from level  $l + 2$  to level  $K$  stay unchanged, the complexity of the algorithm is  $O(l)$  where  $l$  is the highest level of termination. Furthermore, if one of these potentials is in canonical form, it remains in canonical form after the updating procedure.

### RBPF for AHMEM

We now provide the full Rao-Blackwellised Particle Filter algorithm for policy recognition in the AHMEM (Fig 4). The general structure of the algorithm is the same as the RBPF procedure for AHMM (Bui, Venkatesh, & West 2000b). At each time step  $t$ , the algorithm maintains a set of  $N$  samples, each consists of a value for  $s_{t-1}$  and  $l_{t-1}$ , and a parametric representation of  $\mathcal{C}_t$ . The only differences are in the details of how to obtain new samples and update the RB belief state.

In order to obtain each new sample  $(s_t, l_t)$ , we first need to canonicalize the potentials of  $\mathcal{C}_t$ . This would involve performing the junction tree inference algorithm for  $\mathcal{C}_t$ . However, assuming that  $\mathcal{C}_{t-1}$  is in canonical form, we only have to canonicalize the potentials of  $\mathcal{C}_t$  between level 0 and level  $l + 1$ . Hence, only these cliques need to participate in the message passing schedule with complexity  $O(l)$ . Thus the

```

Begin
For  $t = 1, 2, \dots$ 
  /* sampling step */
  For each sample  $i = 1, 2, \dots, N$ 
    Absorb  $o_t$  into  $\mathcal{C}_t^{(i)}$ .
    Canonicalize  $\mathcal{C}_t^{(i)}$ , obtain normalization factor  $w_t$ .
    Sample  $s_t^{(i)}, l_t^{(i)}$  from  $\mathcal{C}_t^{(i)}$  and  $\Pr(e_t^{1:K} | s_t, \pi_t^{0:K}, m_t^{1:K})$ 
    Update weight  $w^{(i)} = w_t$ 
  /* re-sampling step */
  Normalize the weight  $\tilde{w}^{(i)} = \frac{w^{(i)}}{\sum_{i=1}^N w^{(i)}}$ 
  Re-sample the sample set according to  $\tilde{w}^{(i)}$ 
  /* exact step */
  For each sample  $i = 1, 2, \dots, N$ 
    Compute  $\mathcal{C}_{t+1}^{(i)}$  from  $\mathcal{C}_t^{(i)}$  and  $s_t^{(i)}, l_t^{(i)}$ 
    Compute  $h^{(i)} = \mathcal{C}_{t+1}^{(i)}(\pi_{t+1}^k)$ 
  /* Estimation step */
  Compute the estimator  $\Pr(\pi_{t+1}^k | o_{1:t}) \approx \hat{f} = \frac{1}{N} \sum_{i=1}^N h^{(i)}$ 
End

```

Figure 4: RBPF for AHMEM

complexity of the sampling step is  $O(l)$ . Since the complexity of the updating step is also  $O(l)$ , the overall complexity of the algorithm for dealing with each sample at each time step is  $O(l)$ . Note that the distribution for  $l$  usually decays exponentially, thus the average complexity is effectively constant-bounded.

If an estimation is required, we need to perform message passing for the entire chain  $\mathcal{C}_t$ . Thus the complexity is  $O(NK)$ . The algorithm given in Fig. 4 assumes that we want to query about the current policy that the actor is executing at level  $k$ :  $\pi_t^k$ . Other types of queries are also possible, as long as the probability required can be computed from the RB belief state. For example, we can ask the question: if the actor is currently executing  $\pi_t^k$ , what is the current stage of execution of this policy? To answer this query, we need to compute the conditional probability  $\Pr(m_t^k | \pi_t^k, o_{1:t-1})$ . This can easily be achieved by replacing the  $h$  function in the algorithm with  $h = \mathcal{C}_t(m_t^k | \pi_t^k)$ .

## Conclusion

In conclusion, we have presented the Abstract Hidden Markov Memory Models (AHMEM), a general framework for representing and recognizing probabilistic plans online. The new framework extends the AHMM by allowing the policies to have internal memories which are updated in a Markov fashion. This allows the AHMEM to represent a richer set of hierarchical probabilistic plans, including those belonging to the class of PSDG previously considered. Furthermore, we have shown that the Rao-Blackwellised Particle Filter (RBPF) approximate inference method used for the AHMM can be extended to the AHMEM, resulting in efficient and scalable procedures for plan recognition in this general setting.

Our work demonstrates the advantage of phrasing probabilistic plan recognition as inference in DBN so that suitable approximate methods can be employed to cope with the complexity issue. The DBN structure of a plan execution process is a complex network, but at the same time has its special characteristics which can be utilized for efficient approximate inference. A similar approach can be applied to more general models to consider more complex plan constructs such as multi-agent plans, interleaving plans etc.

## References

- Albrecht, D. W.; Zukerman, I.; and Nicholson, A. E. 1998. Bayesian models for keyhole plan recognition in an adventure game. *User Modelling and User-adapted Interaction* 8(1-2):5-47.
- Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context-specific independence in Bayesian networks. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence*.
- Boutilier, C.; Dearden, R.; and Goldszmidt, M. 2001. Stochastic dynamic programming with factored representations. *Artificial Intelligence*. to appear.
- Bui, H. H.; Venkatesh, S.; and West, G. 2000a. On the recognition of abstract Markov policies. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-2000)*.
- Bui, H. H.; Venkatesh, S.; and West, G. 2000b. Policy recognition in the Abstract Hidden Markov Model. Technical Report 4/2000, Department of Computer Science, Curtin University of Technology, Perth, WA, Australia.
- Bui, H. H. 2002. Efficient approximate inference for online probabilistic plan recognition. Technical Report 1/2002, School of Computing Science, Curtin University of Technology, Perth, WA, Australia.
- Casella, G., and Robert, C. P. 1996. Rao-Blackwellisation of sampling schemes. *Biometrika* 81-94.
- Charniak, E., and Goldman, R. P. 1993. A Bayesian model of plan recognition. *Artificial Intelligence* 64:53-79.
- Doucet, A.; de Freitas, N.; Murphy, K.; and Russell, S. 2000. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence*.
- Fine, S.; Singer, Y.; and Tishby, N. 1998. The hierarchical Hidden Markov Model: Analysis and applications. *Machine Learning* 32.
- Forestier, J.-P., and Varaiya, P. 1978. Multilayer control of large Markov chains. *IEEE Transactions on Automatic Control* 23(2):298-305.
- Goldman, R.; Geib, C.; and Miller, C. 1999. A new model of plan recognition. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence*.
- Huber, M. J.; Durfee, E. H.; and Wellman, M. P. 1994. The automated mapping of plans for plan recognition. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence*.

- Jelinek, F.; Lafferty, J. D.; and Mercer, R. L. 1992. Basic methods of probabilistic context free grammar. In Laface, P., and Mori, R. D., eds., *Recent Advances in Speech Recognition and Understanding*, 345–360. Springer-Verlag.
- Jensen, F. 1996. *An Introduction to Bayesian Networks*. Springer.
- Murphy, K., and Pashkin, M. 2001. Linear time inference in hierarchical HMMs. In *NIPS-01*.
- Parr, R., and Russell, S. 1997. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems (NIPS-97)*.
- Parr, R. 1998. *Hierarchical control and learning for Markov Decision Processes*. Ph.D. Dissertation, University of California, Berkeley.
- Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann.
- Pynadath, D. V., and Wellman, M. P. 1995. Accounting for context in plan recognition, with application to traffic monitoring. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*.
- Pynadath, D. V., and Wellman, M. P. 2000. Probabilistic state-dependent grammars for plan recognition. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence*.
- Pynadath, D. V. 1999. *Probabilistic grammars for plan recognition*. Ph.D. Dissertation, Computer Science and Engineering, University of Michigan.
- Rabiner, L. R. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2):257–286.
- Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDP and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112:181–211.