CSE 513T Theory of Artificial Intelligence & Machine Learning April 5, 2018

Lecture 22

Lecturer: Brendan Juba *Scribe: Robert Flagg*

Contents

1. Last Lecture and Today's Goals
2. Markov Decision Processes and PAC rules
3. Policies and Fault Tolerance
4. Partial Observability

1 Last Lecture and Today's Goals

Last time we discussed STRIP planning and we made two big assumptions:

1. Environments are *fully* described by simple deterministic rules.
2. We have *complete* information about (the relevant part of) the environment's state.

Given these assumptions, we can formulate a simple sequence of actions to achieve goals. It is unnecessary to revise plans during execution.

Today we consider richer models of environments and relax both of the previous assumptions. We obtain corresponding richer solutions that can respond to unexpected environment states.

2 Markov Decision Process - MDP

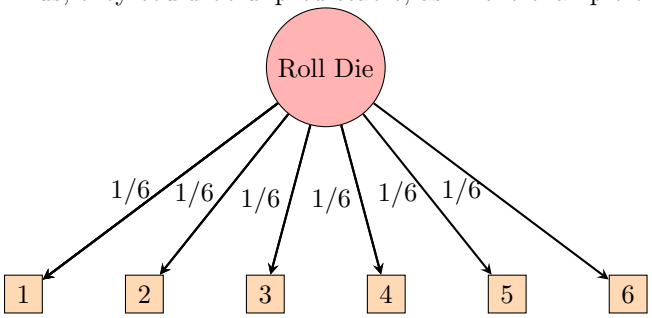
We start with a probabilistic environment model (by relaxing assumption no. 1).
Assume simply that the distribution over states at time t+1 only depends on the actual state at time t and the action we choose at time t.
This **"Markov Property"** can be expressed as follows:

$$Pr_D[\mathbf{x}_{t+1}|\mathbf{x}_1, \alpha_1, \mathbf{x}_2, \alpha_2, ..., \mathbf{x}_t, \alpha_t] = Pr[\mathbf{x}_{t+1}|\mathbf{x}_t, \alpha_t]$$

where $\mathbf{x}_{t+1}$ = state of env. at t+1, and $\mathbf{x}_1, \alpha_1, ..., \mathbf{x}_t, \alpha_t$ = history of states and actions at time 1,...,t

Given $\alpha_1, ..., \alpha_t$, the states $\mathbf{x}_1, ..., \mathbf{x}_t$ are a Markov chain, and this interactive random process is called a **Markov Decision Process** (or MDP).

We assume states $\mathbf{x}$ are given by n propositional attributes called a "factored" MDP.
Thus, they could be unpredictable, as in the example of the action of rolling a die:



The case we are interested in is when the rules *approximately* predict the environment, possibly with a nonzero error probability.

Example: Action pick_up (from last time): pick_$up_{i,t} \Rightarrow holds_{r,b,t+1}$ (if $at_{r,i}$ and $at_{b,i}$, etc...)

- Now we might only have that $pickup_{i,t} \Rightarrow holds_{r,b,t+1}$ is $(1-\epsilon)$-valid for $\epsilon > 0$ for all t, states where preconditions hold.
- Or the frame axiom might be $(1-\epsilon)$-valid, modeling the possibility of the environment disturbed by a process other than the agent.

In general, environments described by $(1-\epsilon)$-valid STRIPS formulation are called **"noisy STRIPS"**.

3 Policies and Fault Tolerance

We want plans to achieve goals in noisy STRIPS environments. It no longer suffices to execute a fixed sequence of actions since we can no longer predict how the environment will evolve.
More generally, we construct a *policy* that, in general, takes an entire history of (observed) states (and for convenience, previously chosen actions) to a choice of the next action.
This can be represented as follows:

$$\pi : X \times (A \times X)^* \Rightarrow A \tag{1}$$

where π = policy, X = current state, $(A \times X)^*$ = sequence of 0 or more states and actions, A = choice of next action.

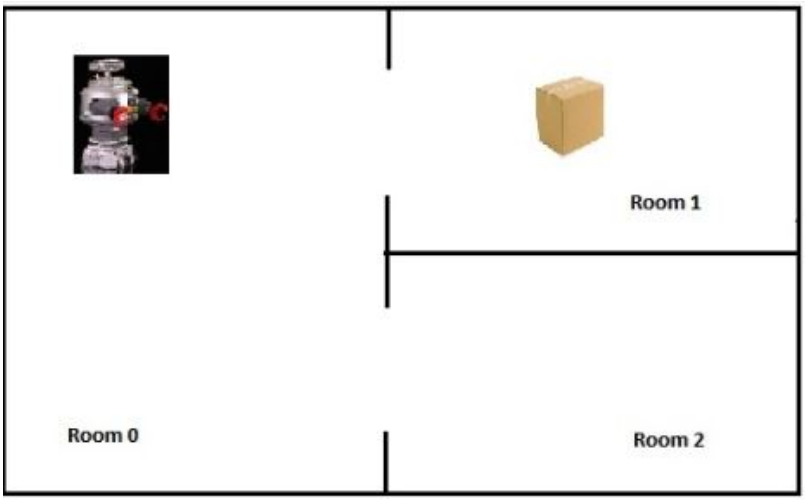
A concrete example of a policy is one computed by decision trees, (extended by sequences of "action nodes" of out-degree 1 along the "edges" between branching nodes).
We take the convention that an entire length-T policy is given by a single decision tree.

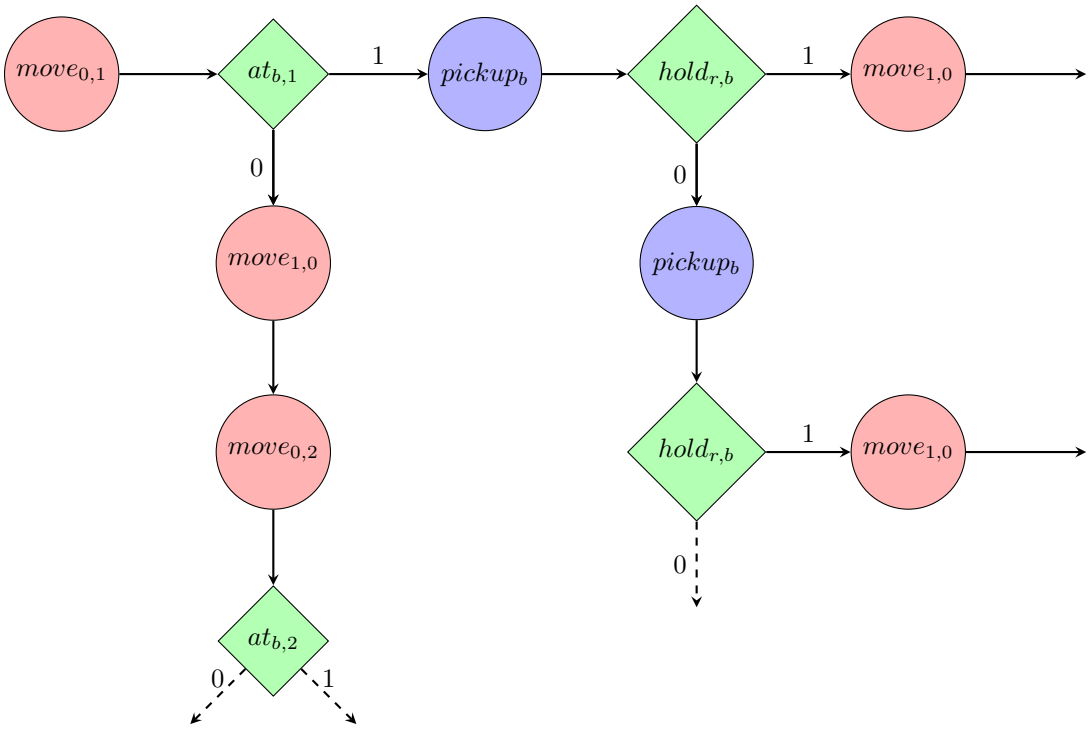
A STRIPS plan is one that handles 0 faults. It is just a sequence of actions. It *does not examine the environment* and is called a **"conformant policy"**.

A policy that handles k faults generally has the form:
- we check the effects of each of its actions
- if some outcome fails to hold, we execute a recovery policy that handles k-1 faults.
- otherwise, if the actions continued as intended, we must continue with a policy that continues to handle k faults.

Such a decision tree is of **rank** $\leq$ **k**. Although we could define richer kinds of policies, rank-k decision trees are nice because we'll be able to find them in *polynomial time*.

Example: Clumsy Robot Policy
We return to the example of last lecture, where our robot attempts to find a box, pick it up and move it. But now the robot is "clumsy", in that it can fail to pick up a box or drop a box.





By taking a recovery action, we can increase the probability that the robot will (eventually) succeed. In general, we call a violation of our noisy STRIPS rules a "fault". If the probability of a fault $p < \frac{1}{T}$ and the faults at each step are independent then the probability that the number of faults (N) is greater than k is given by the binomial:

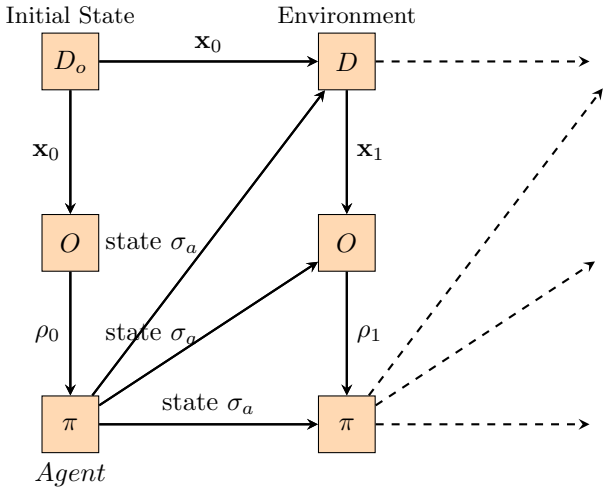
$$Pr[N > k] \leq \binom{T}{k+1} P_{k\geq 2}^{k+1} < (T_p)^{k+1} \tag{2}$$

Thus, to succeed w.p. $\geq 1-\epsilon$ it suffices to handle k faults s.t. $(T_p)^{k+1} < \epsilon$ where $k \sim \frac{log\frac{1}{\epsilon}}{log\frac{1}{T_p}}$

4 Partial Observability

Now we look at relaxing our second STRIP planning assumption, where we assumed complete information about (the relevant part of) the environment's state. Typically, we might expect to have sensors that provide information on the immediate environment and not much else. To model this, we extend the MDP model using an auxiliary random process that produces *observations*, given the last action and the current state of the environment.
For our factored MDP's, a natural kind of observation process is our masking processes that simply hide subsets of the attributes.
In summary, the resulting **Partially Observable Markov Decision Process** (**POMDP**) is an interactive process given by sets of state X, actions A, and observations O, and a pair of families of distributions, distributions D(x,a) over (next) states X given a current state x ∈ X and action at A and $D(x,a)$ over (next) observations $o \in O$ given a current state $x \in X$ and previous action $a \in A$ (We'll also have a distribution D_o over initial states).

Given a previous state $\mathbf{x}_t$ and an agent's choice of action a_t, the POMDP produces a next state $\mathbf{x}_{t+1}$ from $D(\mathbf{x}_t, a_t)$ and an observation $\rho_{t+1} \in \{0,1,\star\}^{n+|A|}$ from $P(\mathbf{x}_{t+1}, a_t)$. This process repeates for t=1,2,...,T.



As before we will consider STRIPS-like goals of the form. reach x ∈ X satisying a goal property $G : X \Rightarrow \{0,1\}$.
Unlike in STRIPS or MDP's, it may be impossible to determine $G(\ x_t)$ based on our observation $\mathbf{p}_t$

Example: In our robot and box example, we can't observe $at_{b,2}$ whenever $\neg at_{b,2} = 1$

In order to achieve high confidence that this goal is satisfied, we need to use our frame axiom to predict the following:
i) Since:
- $at_{b,2,t-1} = 1$ and
- we didn't take an action that would set $at_{b,2,t} = 0$

then frame axiom for $at_{b,2}$ is then $at_{b,2,t} = 1$ with probability $1-\epsilon$

ii) We must also use the robot's knowledge that if it is in room 0 ($\neq 2$) , then it is in particular not also in room 2, so we have $\neg at_{r,2,t}$.
- Thus verifying that the goal is achieved involved reasoning/inference.
We'll also need to extend our notion of policy to account for $>$ two outcomes.
In general, we test a *literal*, take one branch if observed true, and take the other branch if not true.

Goal: $at_{b,2} \wedge \neg at_{r,2}$

