CSE 513T Theory of Artificial Intelligence & Machine Learning April 06, 2015

Lecture 23

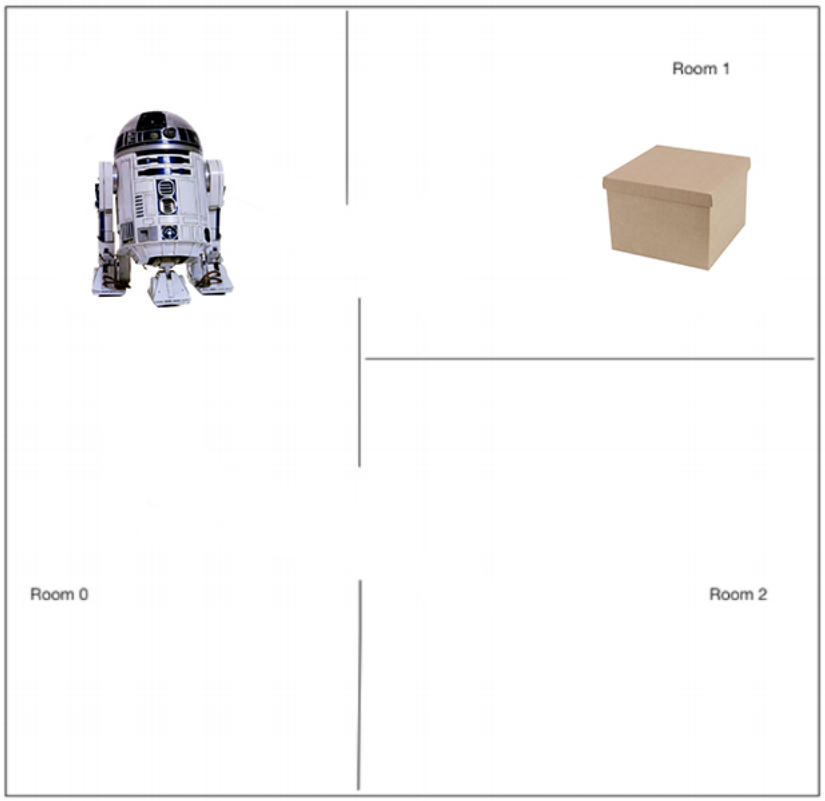
Lecturer: Brendan Juba *Scribe: Marli Komarek*

1 Today's Topics

1. Finish Logical Encoding of Actions and Change, Planning-As-Satisfiability
2. Markov Decision Processes, PAC Rules and Noisy STRIPS
3. Policies and Fault Tolerance
4. Partial Observability (if time)

2 From last time...

Figure 1: Robot and Box Problem



a goal ... $at_{b,2} \wedge \neg at_{r,2}$
$at_{b,2}$ is "box in room 2"
$at_{r,2}$ is "robot in room 2"

Move	A Plan	Corresponding Action Attributes
1	Move to Room 1	$move_{0,1}$
2	Pick up the Box	$pick_up_1$
3	Move to Room 0	$move_{1,0}$
4	Move to Room 2	$move_{0,2}$
5	Put down Box	put_down_2
6	Move to Room 0	$move_{2,0}$

Table 1: Example Moving Box from Room 1 to 2

Room	$at_{r,0}$	$at_{r,1}$	$at_{r,2}$	$at_{b,0}$	$at_{b,1}$	$at_{b,2}$	$holds_{r,b}$
0	1	0	0	0	1	0	0
1	0	1	0	0	1	0	0
2	0	1	0	0	1	0	0
3	1	0	0	1	0	0	1
4	0	0	1	0	0	1	1
5	0	0	1	0	0	1	1
6	1	0	0	0	0	1	0

Table 2: The Corresponding Environment

Action encoding - effect preconditions

$[q_{1,t} \wedge \ldots \wedge q_{s,t} \wedge a_{\alpha,t}] \Rightarrow e_{t+1}$

(e.g. $holds_{r,b,t}$ for the box-moving effect)

Action preconditions $[a_{alpha,t}] = p_t$

Frame axiom for action with preconditions

$[\neg\ell_t \wedge \neg q_{1,t} \wedge a_{x,t}] \Rightarrow e_{t+1}$

Axioms for actions: "there is a unique action at each step"

$\bigvee_{\alpha \in A} a_{\alpha,t}(t = 1, \ldots, T) \neg a_{\alpha,t} \vee \neg a_{\alpha',t}(t = 1, \ldots, T)$

Frame problem specifying what stays the same between steps - simple version $[\ell_c \wedge not(\neg\ell_{t+1})] = \ell_{t+1}$

Explanatory Frame Axioms

$(\forall \ell)\ [\neg\ell_t \wedge \ell_{t+1}] \bigvee_{\alpha:\ell effect} a_{\alpha,t}$

more compact, one clause per literal

$\neg[\neg\ell_t \wedge \ell_{t+1}] \bigvee_{\alpha:\ell effect} a_{\alpha,t}$

Proposition

For any time horizon T, the formula

$\bar{\bigwedge}_{t=1}$([unique α_t] $\wedge$ [preconditions]$_{\alpha \in A}$ $\wedge$ [effects α_t]$_{\alpha \in A}$ $\wedge$ [frame axioms])

$\wedge(\ell_{1,0} \wedge \ell_{2,0} \wedge \ldots \wedge \ell_{n,0})$

such that the initial state is the unique satisfying assignment has a unique satisfying assignment for each T-step plan $\alpha_1, \alpha_2, \ldots, \alpha_T$ the assignment to $x_{1,t}, x_{2,t}, \ldots, x_{n,t}$ is the state of the environment after $\alpha_1, \ldots, \alpha_t$

This tells us that we can test for the existence of a plan achieving $G(\mathbf{x})$) by adding "$\wedge G(\bar{x}_T)$" ... satisfying assignment if and only if there is a plan of length T such that the goal G is satisfied on $\bar{x}_T$

We can read the plan directly from the unique action literal set to 1 in any satisfying assignment

"planning-as-satisfiability" using a SAT-solver to solve planning problems.

Sketch of proof, by induction on t

1H for $\alpha_1, \ldots, \alpha_t$ the unique satisfying assignment to $\mathbf{x}_0, \ldots, \mathbf{x}_t$ is the state of the environment in $t = 0, \ldots, t$
Base: immediate from statement
IS: for plan extended by α_{t+1}
we can only set $a_{\alpha_{t+1}} = 1$ if α_{t+1} is legal in the environment state x_t
we can't set any other α''s $a_{\alpha' t+1}$ to 1
(we must set same $a_{\alpha,t+1}$ to 1)

Literals that are not an effect of α_{t+1} are uniquely fixed in $\mathbf{x}_{t+1}$ by the frame axioms
The effects of α_{t+1} are uniquely determined in $\mathbf{x}_{t+1}$ by the actions axioms

SAT may be NP-complete, but so is STRIPS planning
e.g. stupid SAT domain - 1st n actions set an assignment actions with preconditions allow you to mark clauses as satisfied

Goal: mark all clauses

	Strong assumptions in STRIPS	Generalized Domain
1	Effects of actions deterministically determine next state of environment	Markov Decision Process (MDP)
2	Complete information about environment's state	Partially Observable Markov Decision Process (POMDP)

3 Markov Decision Process (MDP)

We (only) assume that the environment's next state is drawn from a fixed probability distribution that depends (only) on the current state and the action we take.

$P_{r_D}[\mathbf{x}_{t+1}|(\alpha_t, \mathbf{x}_t), (\alpha_{t-1}, \mathbf{x}_{t-1}), \ldots, (\alpha_1, \mathbf{x}_1), \mathbf{x}_0] = P_r[\mathbf{x}_{t+1}|\alpha_t, \mathbf{x}_t]$

(Fixing the actions, the resulting process is a Markov chain)
States given by vectors of attributes – "*factored* MDP"

Example:
States describing the contents and order of a deck of cards, "shuffle" action produces a uniform random permutation of the cards
Example:
We had a rule
$[pick_up_{i,t} \wedge at_{b,i,t}] \rightarrow holds_{r,b,t+1}$
this rule might (only) hold with probability $(1 - \varepsilon)$ (for all possible states) and with probability
$\varepsilon, holds_{r,b,t+1} = 0$
Likewise, some frame axioms might be violated with probability $\leq \varepsilon$, etc.

An MDP that, on each state, is described by $(1 - \varepsilon)$ - valid STRIPS rules is a "noisy STRIPS" domain

4 Policies and Fault Tolerance

We still want to be able to achieve goals in noisy STRIPS, but it no longer suffices to blindly follow a sequence of actions. More generally we want a *policy*

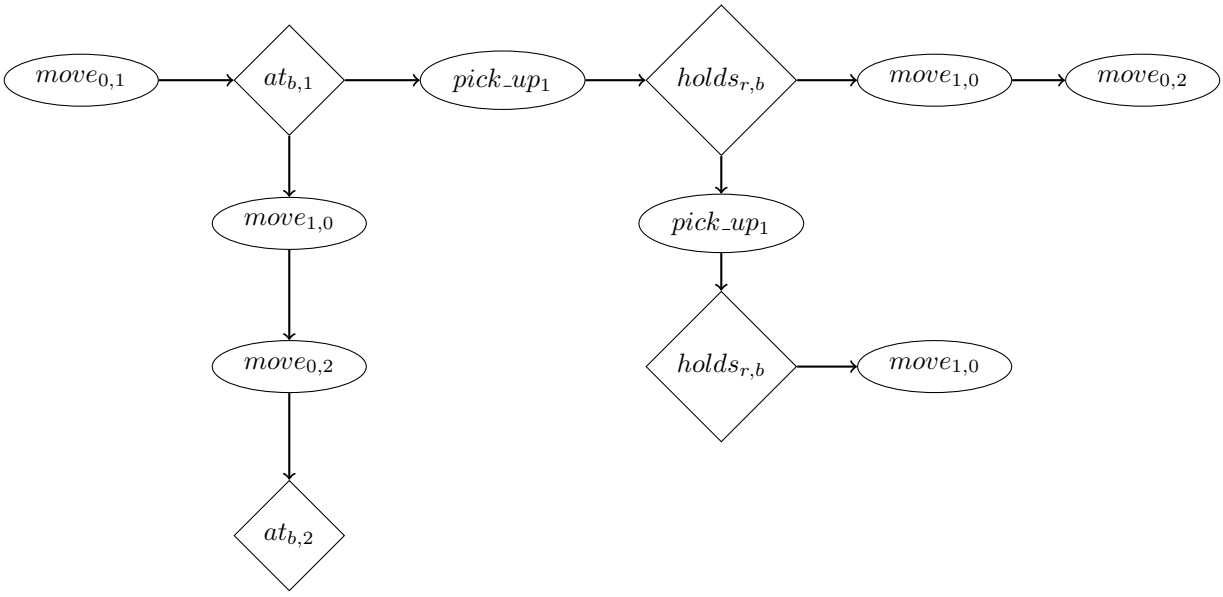
$\Pi : \chi \times (A \times \chi)^* \to A$

χ is the current state

A is the choice of next action

A *very* natural kind of policy decision tress We take the convention that the entire T-horizon policy is given by a single (common) tree

Example:



In noisy STRIPS... we call a transition to a state that violates one of the STRIPS rules is a *fault.* (If faults have probability $\leq \varepsilon$, independent, then k faults happen at k points $\omega p < \varepsilon^k \leftarrow$ for short plans, get small quickly)

A classical plan is a kind of policy that (in general) handles 0 fails (a "*conformant*" policy)

Generally a k-fault tolerant policy is one that checks the results of its actions on each step, if one is violated, it executes a $k-1$ fault tolerant recovery policy and otherwise continues a k-fault tolerant policy (tolerates k faults)

Notice

k-fault tolerant policies built over conform ant policies (classical plans) are rank-k decision tree policies (rank-k decision tree policies are actually more general)

We could consider more sophisticated kinds of policies but ultimately we want to be able to find them (so we'll focus on rank-k decision tree policies - extend definition to include trees with internal nodes with only one child, with the convention that rank doesn't increase for those nodes)

5 Partial Observability

For example: when the robot can only see the room that it is currently in while trying to find the box.
We extend the MDP model by including a random process that produces "observations"
(depending only on the current state & action)

Room	$at_{r,0}$	$at_{r,1}$	$at_{r,2}$	$at_{b,0}$	$at_{b,1}$	$at_{b,2}$	$holds_{r,b}$
0	1	$*$	$*$	0	$*$	$*$	0
1	$*$	1	$*$	$*$	1	$*$	0
2	$*$	1	$*$	$*$	1	$*$	0
3	1	$*$	$*$	1	$*$	$*$	1
4	$*$	$*$	1	$*$	$*$	1	1
5	$*$	$*$	1	$*$	$*$	1	1
6	1	$*$	$*$	0	$*$	$*$	0

Table 3: The Robot and Box Example with Partial Observability

Main example: masking process applied to environment's state
(recall hides some the attribute's values)