**CSE 513T Theory of Artificial Intelligence & Machine Learning** November 15, 2016

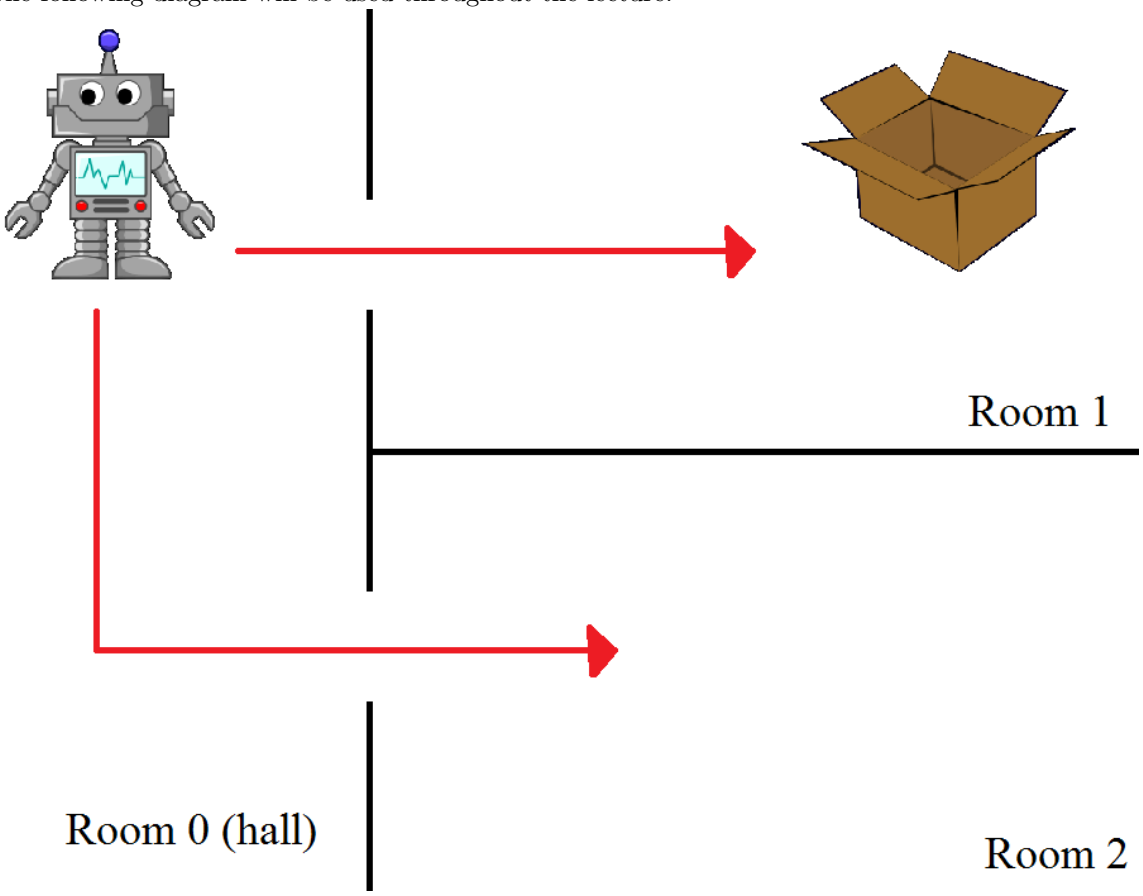
Lecture 22

*Lecturer: Brendan Juba* *Scribe: Anna Gautier*

# 0 Today's Topics:

1. Classical STRIPS Planning
2. Logical Encoding of Actions and Change
3. Planning as Satisfiability

The following diagram will be used throughout the lecture:




# 1 Classical Strips Planning

## 1.1 Planning

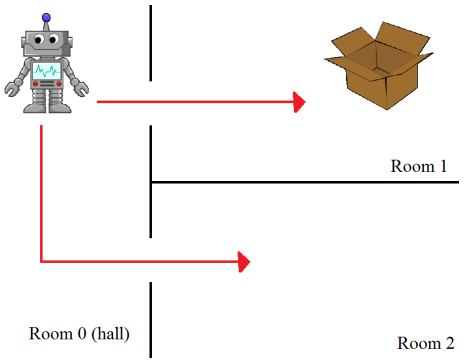
Planning is how to produce intelligent behavior and choose which actions to perform. An agent autonomously selects series of actions in pursuit of some goal. To understand the object of actions and their relationships, we need to model (or find a representation of) the environment in which the agent operates.

## 1.2 STRIPS

STRIPS domains are a particularly simple class of actions and environments used by 'Shakey' the robot at SRI in the 1960s. The planner used in those robots was called 'STRIPS'

A STRIPS domain consists of an an environment with a list of attribute. For example, we have attributes encoding the robot's location, the box's location and whether or not the robot is holding the box.
$at_{r,i}$ ≡robot(r) is in room i
$at_{r,i}$ ≡box(b) is in room i
$holds_{r,b}$ ≡robot(r) is holding box(b)

In our example the attributes are set as follows:




$at_{r,0} = 1$, $at_{r,1} = 0$, $at_{r,2} = 0$
$at_{b,0} = 0$, $at_{b,1} = 1$, $at_{b,2} = 0$
$holds_{r,b} = 0$

# 2 Actions and Changes

## 2.1 Basic Concepts

Some examples of actions include: 'move from room i to room j' ($Move_{i,j}$), 'pick up box' ($PickUp_b$), 'put down box' ($PutDown_b$)

### 2.1.1 Effects of actions

Specific actions have certain effects on attributes.

For example, $Move_{0,1}$ has the effect of setting $at_{r,0} = 0$ and $at_{r,1} = 1$. We can write these as literals that will must be satisfied for $Move_{0,1}$: $(\neg at_{r,0}, at_{r,1})$.

### 2.1.2 Preconditions of actions

Some actions need preconditions to determine when they are satisfied.

For example to $PickUp_b$, the robot and the box need to be in the same room. We can express this in literals by

$$(at_{r,0} \wedge at_{b,0}) \vee (at_{r,1} \wedge at_{b,1}) \vee (at_{r,2} \wedge at_{b,2})$$

### 2.1.3 Conditioned effects

Some actions have effects on attributes conditional on the values of other attributes.

For example if the robot is holding the box, the location of the box should change when the robot moves. More formally, if $holds_{r,b} = 1$ is satisfied and the robot takes action $Move_{i,j}$, then we set $at_{b,i} = 0$ and $at_{b,j} = 1$

In general: Conditional effects of an action are given by an effect's literals together with preconditions

Note: Even if the conditional effect's preconditions are not satisfied, the agent may still execute the action

#### 2.1.4 Goal

A STRIPS goal is given by a conjunctions (AND of literals, over the state attributes)

#### 2.1.5 Plan

A plan in a STRIPS domain is a sequence of actions such that the preconditions of each action is satisfied

The plan achieves the goal is the goal formula is satisfied on the sate of the environment after the final action of the plan

#### 2.1.6 A STRIPS Problem

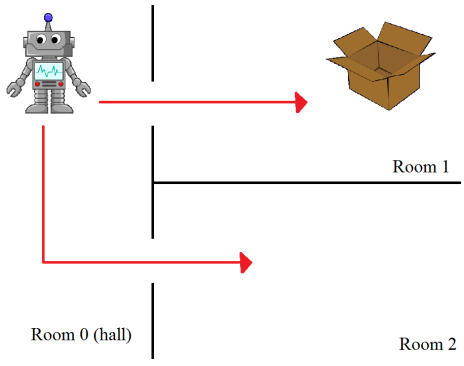
A STRIPS Problem consists of

1. A STRIPS domain: A complete description of attributes, actions, and effects
2. An initial state: A setting of the state attributes
3. A STRIPS goal

#### 2.1.7 Solution

A solution is a plan that transforms the initial state to one in which the goal condition is satisfied in the STRIPS domain

### 2.2 Our Example




In our robot state we start from the initial conditioned state described above. Suppose the goal we want to achieve is $(at_{b,2} \wedge \neg at_{r,2})$. Then the following is a solution:

1. Move to Room 1 ($Move_{0,1}$)

2. Pick up the box ($PickUp_b$)
3. Move to Room 0 ($Move_{1,0}$)
4. Move to Room 2 ($Move_{0,2}$)
5. Put down the box ($PutDown_b$)
6. Move to Room 0 ($Move_{2,0}$)

The following table describes the attributes throughout this plan:

| Step | $at_{r,0}$ | $at_{r,1}$ | $at_{r,2}$ | $at_{b,0}$ | $at_{b,1}$ | $at_{b,2}$ | $hold_{r,b}$ |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 3 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 4 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 5 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |

# 3 Logical Encoding of Actions and Changes

Our objective is to represent the entire planning problem as a CNF, and then use a SAT-Solver to find a plan. in particular, we'll also encode the choose of actions as propositional attributes so that given a satisfying assignment, we can "read off" the plan directly

## 3.1 Small Problems

1. The setting of the state and action attributes needs to change during the execution of the plan. We call attributes that change their value 'fluent'
2. The fix is to include a copy of each fluent attribute for each step including a copy of X, $X_t$ for t=0,1,2,.....T where T is the time horizon
3. Next we choose how to fix the time horizon. First we note that if there are n fluents all valued at either 0 or 1, there are only $2^n$ possible states. So it a plan has more that $2^n$ steps there is an unnecessary loop or inefficiency. Setting $T = 2^n$ ensures that if there is a solution, we will find it. Typically though, we would not be interested in such a long plan so we can just consider the bounded version of STRIPS planning which is given a planning problem and a time horizon T, and returns a plan with at most T steps!

The following table describes the attributes throughout this plan:

| Time Step | Fluents | | | |
|---|---|---|---|---|
| t=0 | $x_{1,0}$ | $x_{2,0}$ | $\cdots$ | $x_{n,0}$ |
| t=1 | $x_{1,1}$ | $x_{2,1}$ | $\cdots$ | $x_{n,1}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| t=T | $x_{1,T}$ | $x_{2,T}$ | $\cdots$ | $x_{n,T}$ |

To solve any problem in time horizon T, we need to assume the existence of adding a WAIT action that preserves the state of the environment. A shorter plan can be extended to one that is precisely T steps by adding a WAIT actions. This ensure that if we have a solution that takes less that T steps, we also have a solution that takes T steps. That way when we SAT solve we only need to check the goal is

satisfied at the end step, not at any step along the way.

We can optimize the length of a plan by trying to solve a sequence of instances of bounded horizon planning by using a Binary search on T

## 3.2 Encoding Actions as Multiple Clauses

### 3.2.1 Effects of actions

For every effect of an action we have a clause. If there are r effects of some action $\alpha$ we have

$$\begin{aligned}[a_{\alpha,t}] &\Rightarrow e_{1,t+1}\\ [a_{\alpha,t}] &\Rightarrow e_{2,t+1}\\ &\vdots\\ [a_{\alpha,t}] &\Rightarrow e_{r,t+1}\end{aligned}$$

### 3.2.2 Preconditions of Actions

For every precondition of an action we have a clause. If there are r preconditions of some action $\alpha$ we have

$$\begin{aligned}[a_{\alpha,t}] &\Rightarrow p_{1,t}\\ [a_{\alpha,t}] &\Rightarrow p_{2,t}\\ &\vdots\\ [a_{\alpha,t}] &\Rightarrow p_{r,t}\end{aligned}$$

### 3.2.3 Conditional Effects of Actions

If there are r effects of some action $\alpha$ conditional on $q_1$, $q_2$, ..., $q_s$ being true, we have clauses

$$\begin{aligned}[q_{1,t} \wedge q_{2,t} \wedge \cdots \wedge q_{s,t} \wedge a_{\alpha,t}] &\Rightarrow e_{1,t+1}\\ [q_{1,t} \wedge q_{2,t} \wedge \cdots \wedge q_{s,t} \wedge a_{\alpha,t}] &\Rightarrow e_{2,t+1}\\ &\vdots\\ [q_{1,t} \wedge q_{2,t} \wedge \cdots \wedge q_{s,t} \wedge a_{\alpha,t}] &\Rightarrow e_{r,t+1}\end{aligned}$$

## 3.3 What we still need

1. Rules describing what doesn't change as a result of an action in a step
2. Rules specifying that a unique action is taken at each step. This is straightforward and can we accomplished by adding the following clause, where A is the set of all possible actions (including WAIT)

$$\left(\bigvee_{\alpha \in A} a_{\alpha,t}\right) \wedge \left(\bigwedge_{\alpha \neq \alpha' \in A} \neg(a_{\alpha,t} \wedge a_{\alpha',t})\right)$$