

Planning: Forward and CSP Planning

CPSC 322 – Planning 2

Textbook §11.2

Lecture Overview

- 1 Recap
- 2 Forward Planning
- 3 CSP Planning

Planning

- With CSPs, we looked for solutions to essentially **atemporal** problems.
 - find a single variable assignment (state) that satisfies all of our constraints.

Now consider a problem where we are **given**:

- A description of an **initial state**
- A description of the effects and preconditions of **actions**
- A **goal** to achieve

...and want to **find a sequence of actions** that is possible and will result in a state satisfying the goal.

- note: here we want not a **single state** that satisfies our constraints, but rather a **sequence of states** that gets us to a goal

Feature-Based Representation

- **Features** helped us to represent CSPs **more compactly** than states could.
 - The main idea: factor states into joint variable assignments
 - This allowed us to represent constraints compactly by discussing only relevant variables rather than full states.
- Now what we want to find is a **sequence of variable assignments** that goes from an initial state to a goal
- Instead of having one variable for every feature, we must instead have one variable for every feature at each time step, indicating the value taken by that feature at that time step!
- What do we need to build this representation?
 - the state space is easy: joint assignment to variables
 - initial state and goal state are also easy
 - the key is **modeling actions**

STRIPS Actions

- In STRIPS, an action has **two parts**:
 - 1 **Precondition**: a logical test about the features that must be true in order for the action to be legal
 - 2 **Effects**: a set of assignments to variables that are caused by the action
- If the feature V has value v after the action a has been performed, what can we conclude about a and/or the state of the world?
 - either $V = v$ was true in the state of the world immediately preceding execution of action a , or a sets $V = v$, or both.

Example

STRIPS representation of the action **pick up coffee**, PUC :

- **preconditions** $Loc = cs$ and $RHC = \overline{rhc}$
- **effects** $RHC = rhc$

STRIPS representation of the action **deliver coffee**, $DelC$:

- **preconditions** $Loc = off$ and $RHC = rhc$
- **effects** $RHC = \overline{rhc}$ and $SWC = \overline{swc}$

Note that Sam doesn't have to want coffee for Rob to deliver it; one way or another, Sam doesn't want coffee after delivery.

Lecture Overview

- 1 Recap
- 2 Forward Planning
- 3 CSP Planning

Forward Planning

Idea: search in the state-space graph.

- The nodes represent the states
- The arcs correspond to the actions: The arcs from a state s represent all of the actions that are legal in state s .
- A plan is a path from the state representing the initial state to a state that satisfies the goal.

Example state-space graph

Actions

mc: move clockwise

mac: move anticlockwise

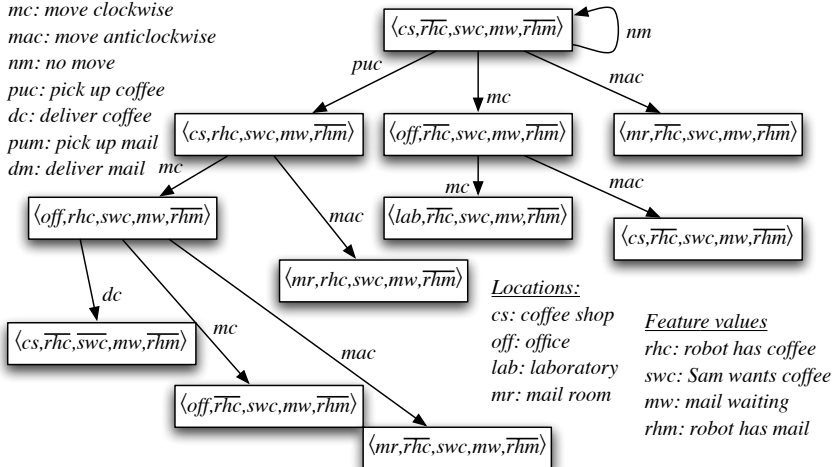
nm: no move

puc: pick up coffee

dc: deliver coffee

pum: pick up mail

dm: deliver mail



What are the errors (none involve room locations)?

Actions

mc: move clockwise

mac: move anticlockwise

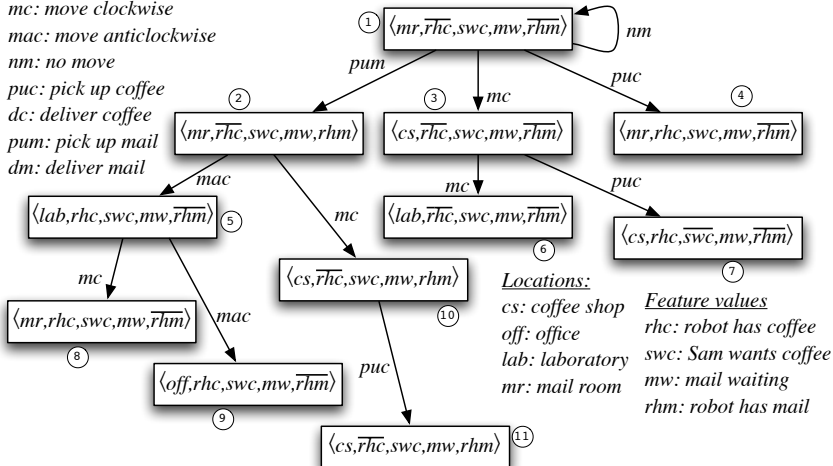
nm: no move

puc: pick up coffee

dc: deliver coffee

pum: pick up mail

dm: deliver mail



Improving Search Efficiency

Forward search can use **domain-specific knowledge** specified as:

- a **heuristic function** that estimates the number of steps to the goal
- **domain-specific pruning** of neighbors:
 - don't go to the coffee shop unless "Sam wants coffee" is part of the goal and Rob doesn't have coffee
 - don't pick-up coffee unless Sam wants coffee
 - unless the goal involves time constraints, don't do the "no move" action.

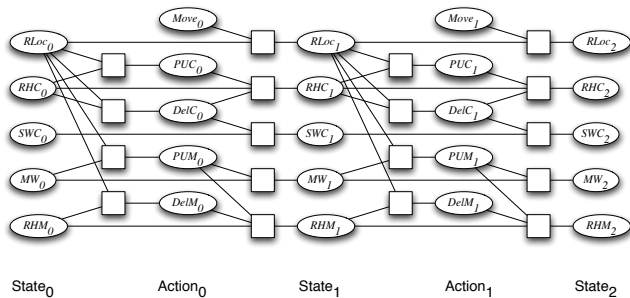
Lecture Overview

- 1 Recap
- 2 Forward Planning
- 3 CSP Planning**

Planning as a CSP

- We don't have to worry about searching forwards if we set up a planning problem as a CSP
- To do this, we need to “unroll” the plan for a fixed number of steps
 - this is called the **horizon**
- To do this with a horizon of k :
 - construct a **variable for each feature at each time step** from 0 to k
 - construct a boolean **variable for each action at each time step** from 0 to $k - 1$.

CSP Planning: Robot Example



Do you see why CSP planning doesn't have to search forwards?

CSP Planning: Constraints

As usual, we have to express the preconditions and effects of actions:

- **precondition constraints**
 - hold between state variables at time t and action variables at time t
 - specify when actions may be taken
- **effect constraints**
 - between state variables at time t , action variables at time t and state variables at time $t + 1$
 - explain how state variables at time $t + 1$ are affected by the action taken at time t
 - this includes both causal and frame axioms
 - basically, it goes back to the feature-centric representation the book discusses before STRIPS
 - of course, solving the problem this way doesn't mean we can't *encode* the problem using STRIPS

CSP Planning: Constraints

Other constraints we must/may have:

- **initial state constraints** constrain the state variables at time 0
- **goal constraints** constrain the state variables at time k
- **action constraints**
 - specify which actions cannot occur simultaneously
 - note that without these constraints, there's nothing to stop the planner from deciding to take several actions simultaneously
 - when the order between several actions doesn't matter, this is a good thing
 - these are sometimes called mutual exclusion (mutex) constraints
- **state constraints**
 - hold between variables at the same time step
 - they can capture physical constraints of the system
 - they can encode maintenance goals