

Given:

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

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

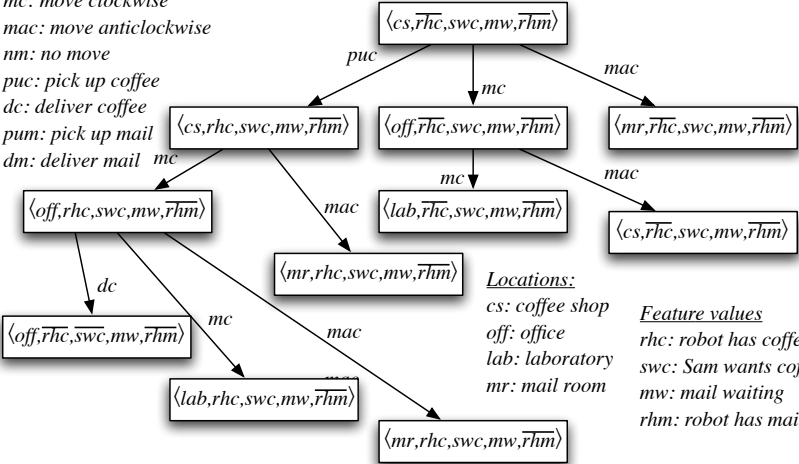
Idea: search in the state-space graph.

- 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



Locations:

- cs*: coffee shop
- off*: office
- lab*: laboratory
- mr*: mail room

Feature values

- rhc*: robot has coffee
- swc*: Sam wants coffee
- mw*: mail waiting
- rhm*: robot has mail

What are the errors?

Actions

mc: move clockwise

mac: move anticlockwise

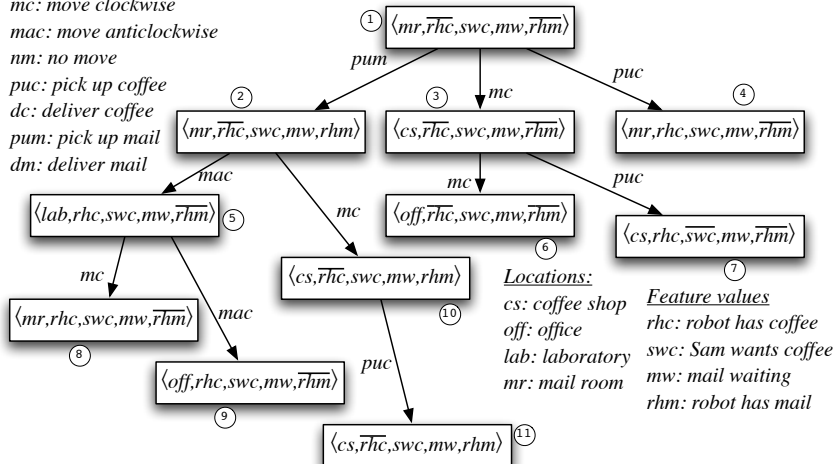
nm: no move

puc: pick up coffee

dc: deliver coffee

pum: pick up mail

dm: deliver mail



Forward planning representation

- The search graph can be constructed on demand: you only construct reachable states.
- If you want a cycle check or multiple path-pruning, you need to be able to find repeated states.
- There are a number of ways to represent states:
 - ▶ As a specification of the value of every feature
 - ▶ As a path from the start state

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.