Planning: Forward and Regression Planning

CPSC 322 Lecture 16

February 12, 2007
Textbook §11.2

Lecture Overview

1. Recap
2. Forward Planning
3. Regression Planning

Feature-Based Representation

We need two things to replace the tabular representation:

1. Modeling when actions are possible:
   - Precondition of an action: a function (proposition) of the state variables that is true when the action can be carried out
2. Modeling state transitions in a “factored” way:
   - causal rules: explain how the value of a variable describing a feature at time step t depends on the action taken at time $t-1$
   - frame rules: explain how the value of a variable describing a feature at time step t depends on the value of the variable that describes the same feature at time step $t-1$

STRIPS

- The previous representation was feature-centric:
  - for every feature, where does its value come from?
- STRIPS is an action-centric representation:
  - for every action, what does it do?
- The STRIPS assumption:
  - all variables not explicitly changed by an action stay unchanged
- 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

Lecture Overview

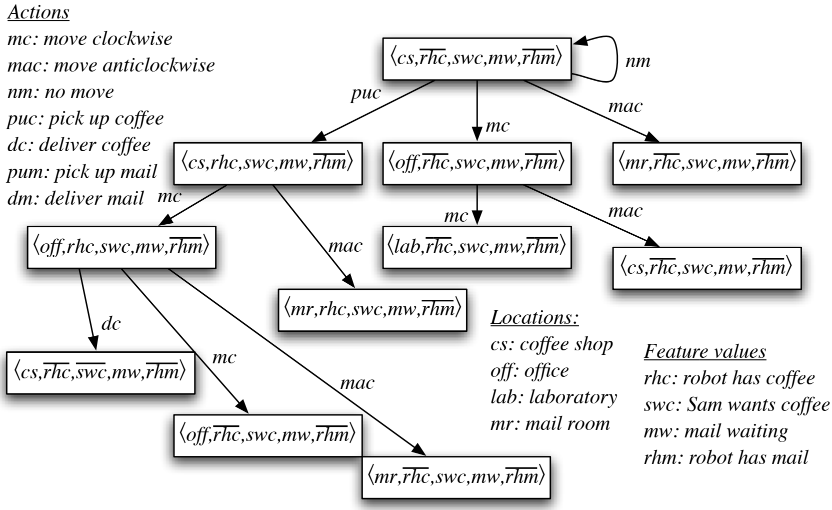
1. Recap
2. Forward Planning
3. Regression Planning

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

$\langle cs,\overline{rhc},swc,mw,\overline{rhm}\rangle$

nm

puc

mc

mac

$\langle cs,rhc,swc,mw,\overline{rhm}\rangle$

$\langle off,\overline{rhc},swc,mw,\overline{rhm}\rangle$

$\langle mr,\overline{rhc},swc,mw,\overline{rhm}\rangle$

mc

mac

mc

mac

$\langle off,rhc,swc,mw,\overline{rhm}\rangle$

$\langle lab,\overline{rhc},swc,mw,\overline{rhm}\rangle$

$\langle cs,\overline{rhc},swc,mw,\overline{rhm}\rangle$

$\langle mr,rhc,swc,mw,\overline{rhm}\rangle$

dc

mc

mac

$\langle cs,\overline{rhc},\overline{swc},mw,\overline{rhm}\rangle$

$\langle off,\overline{rhc},swc,mw,\overline{rhm}\rangle$

$\langle mr,\overline{rhc},swc,mw,\overline{rhm}\rangle$

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 (none involve room locations)?

Actions

mc: move clockwise

mac: move anticlockwise

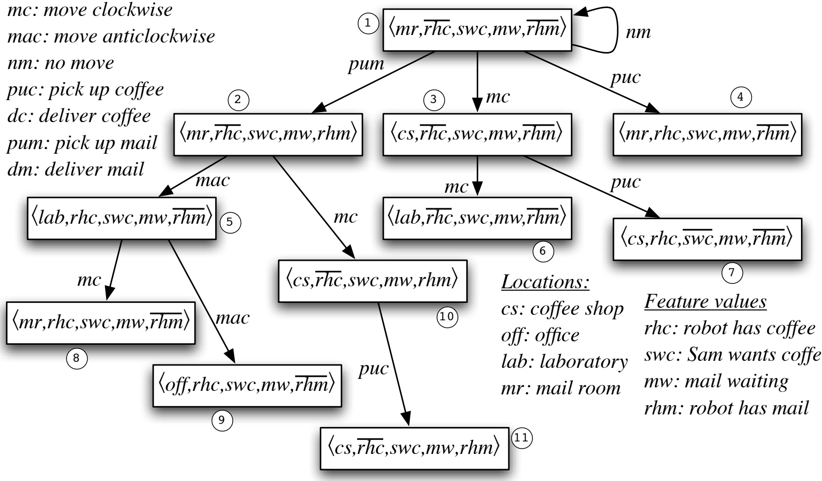
nm: no move

puc: pick up coffee

dc: deliver coffee

pum: pick up mail

dm: deliver mail


1 $\langle mr,\overline{rhc},swc,mw,\overline{rhm}\rangle$
nm
pum
mc
puc
2 $\langle mr,\overline{rhc},swc,mw,rhm\rangle$
3 $\langle cs,\overline{rhc},swc,mw,\overline{rhm}\rangle$
4 $\langle mr,rhc,swc,mw,\overline{rhm}\rangle$
mac
mc
mc
puc
5 $\langle lab,rhc,swc,mw,\overline{rhm}\rangle$
6 $\langle lab,\overline{rhc},swc,mw,\overline{rhm}\rangle$
7 $\langle cs,rhc,\overline{swc},mw,\overline{rhm}\rangle$
10 $\langle cs,\overline{rhc},swc,mw,rhm\rangle$
mc
mac
puc
8 $\langle mr,rhc,swc,mw,\overline{rhm}\rangle$
9 $\langle off,rhc,swc,mw,\overline{rhm}\rangle$
11 $\langle cs,\overline{rhc},swc,mw,rhm\rangle$

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

Forward planning representation

- The search graph can be constructed on demand: thus, we 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.

Lecture Overview

1. Recap
2. Forward Planning
3. Regression Planning

Defining nodes and arcs

Idea: search backwards from the goal description: nodes correspond to subgoals, and arcs to actions.

- A node N is a partial assignment of values to variables: $(X_1 = v_1, \ldots, X_n = v_n)$
- Start node: the goal condition
- The goal test: a proposition that is true of the initial state
- An action to node N achieves one of the $X_i = v_i$, and does not achieve any $X_j = v_j$ where v'_j is different from v_j.
- Any node that neighbours N via arc A must contain:
  - The prerequisites of action A
  - All of the elements of N that were not achieved by A