Regression Planning

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

- Nodes are propositions: a formula made up of assignments of values to features
- Arcs correspond to actions that can achieve one of the goals
- Neighbors of a node N associated with arc A specify what must be true immediately before A so that N is true immediately after.
- The start node is the goal to be achieved.
- $goal(N)$ is true if N is a proposition that is true of the initial state.

Defining nodes and arcs

- A node N can be represented as a set of assignments of values to variables:

  $$[X_1 = v_1, \ldots, X_n = v_n]$$

  This is a set of assignments you want to hold.
- The last action is one that achieves one of the $X_i = v_i$, and does not achieve $X_j = v_j'$ where v_j' is different to v_j.
- The neighbor of N along arc A must contain:
  - The prerequisites of action A
  - All of the elements of N that were not achieved by A

  N must be consistent.

Formalizing arcs using STRIPS notation

$$\langle G, A, N \rangle$$

where G is $[X_1 = v_1, \ldots, X_n = v_n]$ is an arc if

- $\exists i \; X_i = v_i$ is on the effects list of action A
- $\forall j \; X_j = v_j'$ is not on the effects list for A, where $v_j' \neq v_j$
- N is $\mathit{preconditions}(A) \cup \{X_k = v_k : X_k = v_k \notin \mathit{effects}(A)\}$ and N is consistent in that it does not assign different values to any variable.

Regression example

Actions

mc: move clockwise
mcc: move counterclockwise
puc: pick up coffee
dc: deliver coffee
pum: pick up mail
dm: deliver mail

Locations:

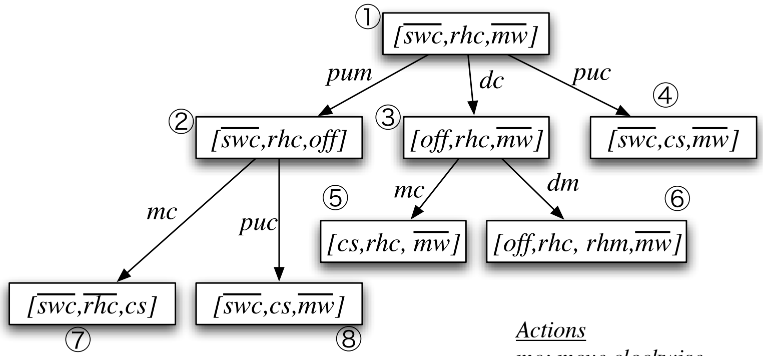
cs: coffee shop
off: office
lab: laboratory
mr: mail room

Feature values

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

$$\begin{tikzcd}
& & {[\neg swc]} \arrow[d, "dc"] & & & \\
& & {[off,rhc]} \arrow[dl, "mc"'] \arrow[drr, "mcc"] & & & \\
& {[cs,rhc]} \arrow[dl, "mc"'] \arrow[d, "puc"] \arrow[dr, "mcc"] & & & {[lab,rhc]} \arrow[d, "mc"] \arrow[dr, "mcc"] & \\
{[mr,rhc]} & {[cs]} & {[off,rhc]} & & {[off,rhc]} & {[mr,rhc]}
\end{tikzcd}$$

Find the errors



①
[$\overline{swc}$,rhc,$\overline{mw}$]
pum
dc
puc
②
[$\overline{swc}$,rhc,off]
③
[off,rhc,$\overline{mw}$]
④
[$\overline{swc}$,cs,$\overline{mw}$]
mc
puc
⑤
mc
dm
⑥
[cs,rhc, $\overline{mw}$]
[off,rhc, rhm,$\overline{mw}$]
[$\overline{swc}$,$\overline{rhc}$,cs]
[$\overline{swc}$,cs,$\overline{mw}$]
⑦
⑧

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

Actions
mc: move clockwise
mac: move anticlockwise
puc: pick up coffee
dc: deliver coffee
pum: pick up mail
dm: deliver mail

Loop detection and multiple-path pruning

- Goal G_1 is simpler than goal G_2 if G_1 is a subset of G_2.
  - It is easier to solve $[cs]$ than $[cs, rhc]$.
- If you have a path to node N have already found a path to a simpler goal, you can prune the path N.

Improving Efficiency

- You can define a heuristic function that estimates how difficult it is to solve the goal from the initial state.
- You can use domain-specific knowledge to remove impossible goals.
  - It is often not obvious from an action description to conclude that an agent can only hold one item at any time.

Comparing forward and regression planners

- Which is more efficient depends on:
  - The branching factor
  - How good the heuristics are
- Forward planning is unconstrained by the goal (except as a source of heuristics).
- Regression planning is unconstrained by the initial state (except as a source of heuristics)