Conditional Planning

Section 11.3
Outline

- Fully observable environments
- Partially observable environments
- Conditional POP
Uncertainty

- The agent might not know what the initial state is
- The agent might not know the outcome of its actions
- The plans will have branches rather than being straight line plans, includes \textit{conditional steps}

\[
\text{if} \ < \text{test} > \ \text{then plan}_A \ \text{else plan}_B
\]

- \textit{Full observability}: The agent knows what state it currently is, does not have to execute an \textit{observation action}
  Simply get plans ready for all possible contingencies
Modeling uncertainty

- Actions sometimes fail $\rightarrow$ disjunctive effects

- Example: moving left sometimes fails

  $\text{Action}(\text{Left}, \text{Precond: } \text{AtR}, \text{Effect: } \text{AtL } \lor \text{AtR})$

- Conditional effects: effects are conditioned on secondary preconditions

  $\text{Action}(\text{Suck}, \text{Precond: } ;, \text{Effect: } (\text{when AtL: CleanL}) \land (\text{when AtR: CleanR}))$

- Actions may have both disjunctive and conditional effects:

  Moving sometimes dumps dirt on the destination square only when that square is clean

  $\text{Action}(\text{Left}, \text{Precond: } \text{AtR};, \text{Effect: } \text{AtL } \lor (\text{AtL } \land \text{when CleanL: } \neg \text{CleanL}))$
The vacuum world example

- Double Murphy world
  - the vacuum cleaner sometimes deposits dirt when it moves to a clean destination square
  - sometimes deposits dirt if Suck is applied to a clean square
- The agent is playing a game against nature
Perform and-or search

- Left Suck
- Right Suck
- Loop
- Goal
In the “double-Murphy” vacuum world, the plan is:

[  
  \textit{Left},  
  \textit{if} \hspace{1mm} AtL \land CleanL \land CleanR  
  \text{ then } [ ]  
  \text{ else } Suck  
  \]
function \textsc{And-Or-Graph-Search} (\textit{problem})
returns a conditional plan, or failure

\textsc{Or-Search}(\text{Initial-State}[\textit{problem}], \textit{problem}, [])

function \textsc{Or-Search} (\textit{state}, \textit{problem}, \textit{path})
returns a conditional plan, or failure

\begin{verbatim}
if \textsc{Goal-Test}[\textit{problem}](\textit{state}) then return the empty plan
if \textit{state} is on \textit{path} then return failure
for each action, state-set in \textsc{Successors} [\textit{problem}](\textit{state}) do
    \textit{plan} ← \textsc{And-Search} (\textit{state}, \textit{problem}, [\textit{state} | \textit{path}])
    if \textit{plan} ≠ failure then return [\textit{action} | \textit{plan}]
return failure
\end{verbatim}
function **AND-SEARCH** (*state-set, problem, path*)
returns a conditional plan, or failure

for each \( s_i \) in state-set do
    \( \text{plan}_i \leftarrow \text{OR-SEARCH}(S_i, \text{problem}, \text{path}) \)
    if \( \text{plan} = \text{failure} \) then return \( \text{failure} \)

return

[if \( s_1 \)
    then \( \text{plan}_1 \)
  else if \( s_2 \)
    then \( \text{plan}_2 \)
  else ... if \( s_{n-1} \)
    then \( \text{plan}_{n-1} \)
  else \( \text{plan}_n \) ]
The vacuum cleaner sometimes deposits dirt when it moves to a clean destination square.

- It sometimes deposits dirt if *suck* is applied to a clean square.
- + move sometimes fails.
First level of the search

GOAL

Left

Suck
No acyclic solutions

A cyclic solution is to try going left until it works. Use a *label*.

\[L_1: \text{Left, if atR then } L_1 \text{ else if CleanL then } [] \text{ else Suck}\]
Partially observable environments

- The agent knows only a certain amount of the actual state (e.g., local sensing only, does not know about the other squares)
  - *Automatic sensing*: at every time step the agent gets all the available percepts
  - *Active sensing*: percepts are obtained only by executing specific sensory actions

- *Belief state*: The set of possible states that the agent can be in

- “Alternate double Murphy world”: dirt can sometimes be left behind when the agent leaves a clean square
Part of the search

CleanL ~CleanL
Suck

Suck

CleanR ~CleanR

Left

Right
Conditional POP (CNLP algorithm)

GOAL

A

INIT

LEFT

atL

cleanL

cleanR

Dangling Edge
Duplicate the goal and label it

A

atL

left

cleanL

cleanR

INIT

B

atL

cleanL

cleanR

GOAL

GOAL
Conditional POP (CNLP algorithm)

INIT

LEFT

GOAL

SUCK

A

atL

~cleanL

cleanL
cleanR

atL

B

GOAL

atL

cleanL
cleanR

Sec. 11.3 – p.17/18
Classical planning is NP
Conditional planning is harder than NP
Had to go back to state space search
Many problems are intractable