Conditional Planning

By

Rong Liang

What’s Conditional Planning?

- It’s a planning method for handling bounded indeterminacy.
  - Bounded Indeterminacy – actions can have unpredictable effects, but the possible effects can be determined.
    Ex: flip a coin (outcome will be head or tail)

- It constructs a conditional plan with different branches for the different contingencies that could arise.

- It’s a way to deal with uncertainty by checking what is actually happening in the environment at predetermined points in the plan. (Conditional Steps)

Example:
- Check whether SFO airport is operational. If so, fly there; otherwise, fly to Oakland.
Three kind of Environments

- **Fully Observable**
  - The agent always knows the current state

- **Partially Observable**
  - The agent knows only a certain amount about the actual state. (much more common in real world)

- **Unknown**
  - The agent knows nothing about the current state

Conditional Planning in Fully Observable Environments

- Agent used conditional steps to check the state of the environment to decide what to do next.

- Plan information stores in a library
  
  Ex: Action(Left) → Clean v Right

- Syntax:
  
  **If** <test> **then** plan_A **else** plan_B
AND-OR-Graph-Search

- Modify Minimax Algorithm
  - MAX node → OR node
    - It returns a single move/plan for an action
  - MIN node → AND node
    - It returns a series of plans for each action

Example of AND-OR-Graph

<table>
<thead>
<tr>
<th>Plan:</th>
<th>Take her out</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Action:</td>
<td>Acquire information</td>
<td>Ask her</td>
<td></td>
</tr>
<tr>
<td>AcquireInfo Action:</td>
<td>Ask her friend</td>
<td>Borrow her diary</td>
<td>Borrow her PDA</td>
</tr>
<tr>
<td>AskHerFriend Action:</td>
<td>Go to Friend</td>
<td>Send Message</td>
<td></td>
</tr>
</tbody>
</table>
Function \textsc{AND-OR-GRAPH-SEARCH} (\textit{problem}) returns a conditional plan, or failure

\textsc{OR-SEARCH} (INITIAL-STATE[\textit{problem}], \textit{problem},[])}

Function \textsc{OR-SEARCH}(\textit{state,problem}, \textit{path}) returns a conditional plan, or failure

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 \textit{action}, \textit{state-set} in \textsc{SUCCESSORS}[\textit{problem}](\textit{state}) do
  \textit{plan} \leftarrow \textsc{AND-SEARCH( state-set, problem, [state | path])}
  if \textit{plan} \neq failure then return [\textit{action} | \textit{plan}]
return failure

Function \textsc{AND-SEARCH}( \textit{state-set,problem}, \textit{path}) returns a conditional plan, or failure

for each \textit{S(i)} in \textit{state-set} do
  \textit{plan}(i) \leftarrow \textsc{OR-SEARCH( S(i), problem, path)}
  if \textit{plan} = failure then return failure
return [if \textit{s(1)} then \textit{plan}(1) else if \textit{s(2)} then \textit{plan}(2) else... if \textit{S(n-1)} then \textit{plan}(n-1) else \textit{plan}(n)]

\textbf{Good Thing About \textsc{AND-OR} Graph Search}

- The way it deal with cycle
  - If the current state is identical to a state on the path from the root, then it returns with failure; it means that if there is a noncyclic solution, it must be reachable from the earlier incarnation of the current state, so the new incarnation can be discarded
  - Algorithm can terminate in every finite state space

- But it doesn’t check whether the current state is a repetition of a state on some other path from the root
Failure of AND-OR-Graph-Search

- “Triple Murphy” Vacuum World – there are no longer any acyclic solution, and this algorithm would return with failure

Is there Any Solution?
Possible Solution For AND-OR-Graph-Search Failure

- Solution:
  - Cyclic solution – keep trying Left or Right until it is clean, but it doesn’t guaranteed succeed.

```
[L1 : Left, if AtR then L1 elseif CleanL then [] else Suck]
```

Partially Observable Environments

- It used the same AND-OR-Graph-Search algorithm, but the belief states will defy differently.

- Three choices for belief states:
  1. Sets of full state descriptions
     Ex: \{(AtR and CleanR and CleanL), (AtR and CleanR and not CleanL)\}
     (not good, the size will become $O(2^n)$)
  2. Logical sentences that capture exactly the set of possible worlds (Open-world Assumption)
     Ex: AtR and CleanR
     (not that good, it can’t represent all domains)
  3. Knowledge Propositions – describe the agent’s knowledge (Closed-world Assumption)
     Ex: $K(P) \rightarrow$ means the agent knows that P is true, if it doesn’t appear, it’s assumed false.
Partially Observable Environments

- Any scheme capable of representing every possible belief state will require $O(\log_2(2^{2n})) = O(2^n)$ bit to represent each one in the worst case.

- Two kind of Sensing
  1. Automatic – auto. Check the state
  2. Active – agents must use sensory action to check the state of environment. ex: CheckDirt

Advantages?
Disadvantage?

Disadvantages

- Agents are not capable of making tradeoffs between the probability of success and the cost of plan construction.

- Conditional Planning is harder than NP
  - NP means that a candidate solution can be checked to see whether it really is a solution in polynomial time.

- Use a lot of space $O(2^n)$
Questions
or
Comments?