SATPLAN

• One approach: Extract SAT problem from planning graph
• Another approach: Make a sentence for depth n, that has a satisfying assignment iff a plan exists at depth n
  • Variables:
    - Every proposition at every even depth index: clean\(_i\), garb\(_i\)
    - Every action at every odd depth index: cook\(_i\)

Constructing SATPLAN sentence

• Initial sentence (clauses): garb\(_0\), clean\(_0\), quiet\(_0\), ¬present\(_0\), ¬dinner\(_0\)
• Goal (at depth 4): ¬garb\(_4\), present\(_4\), dinner\(_4\)
• Action\(_t\) → (Pre\(_t\) ∧ Eff\(_t\)) [in clause form]
• Cook\(_t\) → (clean\(_t\) ∧ dinner\(_{t+1}\))
• Explanatory Frame Axioms: For every state change, say what could have caused it
  • garb\(_t\) ∧ ¬garb\(_{t+2}\) → (dolly\(_{t+2}\) v carry\(_{t+2}\)) [in clause form]
  • Conflict exclusion: For all conflicting actions a and b at depth t, add ¬a \lor ¬b
• One's precondition is inconsistent with the other's effect

Planning Assumptions

• Assumed complete and correct model of world dynamics
• Assumed know initial state
• Assumed world is deterministic
• These assumptions hold in domains such as scheduling machines in factories but not in many other domains.
Conditional Planning

- POP with these new ways of fixing threats and satisfying preconditions increases the branching factor in the planning search and makes POP completely impractical.
- People are working on conditional planning versions of GraphPlan and SatPlan.
- Instead of constructing conditional plans ahead of time, just plan as necessary when you have the information.

Replanning

- One place where replanning can help is to fill in the steps in a very high-level plan.
Replanning

- One place where replanning can help is to fill in the steps in a very high-level plan.
- Another is to overcome execution errors.

Replanning Cycle

World

Planner

Init state

Plan

Execute

Goto Airport

Goto Gate

Replanning

- One place where replanning can help is to fill in the steps in a very high-level plan.
- Another is to overcome execution errors.

Universal Plan

Assume
- Offline computation is cheap
- Space is plentiful
- Online computation is expensive

- Plan for every possible initial state
- Store: initial state → first step
- World is completely observable

Triangle Tables

Fikes & Nilsson

<table>
<thead>
<tr>
<th>Init</th>
<th>Pre(A1)</th>
<th>Eff(A1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>At HW</td>
<td>Sells(HW, Drill)</td>
<td>At SMSells(SM, Bananas)</td>
</tr>
<tr>
<td>At HW</td>
<td>Buy Drill Eff(A2)</td>
<td>At HW</td>
</tr>
<tr>
<td>At SM</td>
<td>Go SM Eff(A3)</td>
<td>At HW</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Goal Conds</th>
<th>Pre(A2)</th>
<th>Eff(A2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Have Drill</td>
<td>Sells(SM, Bananas)</td>
<td>Have Bananas</td>
</tr>
</tbody>
</table>

Triangle Tables

Fikes & Nilsson

<table>
<thead>
<tr>
<th>Init</th>
<th>Pre(A1)</th>
<th>Eff(A1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>At HW</td>
<td>Sells(HW, Drill)</td>
<td>At SMSells(SM, Bananas)</td>
</tr>
<tr>
<td>At HW</td>
<td>Buy Drill Eff(A2)</td>
<td>At HW</td>
</tr>
<tr>
<td>At SM</td>
<td>Go SM Eff(A3)</td>
<td>At HW</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Goal Conds</th>
<th>Pre(A2)</th>
<th>Eff(A2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Have Drill</td>
<td>Sells(SM, Bananas)</td>
<td>Have Bananas</td>
</tr>
</tbody>
</table>

Execute Highest True Kernel (rectangle including lower left corner and some upper right corner)
Triangle Tables
Fikes & Nilsson

<table>
<thead>
<tr>
<th>Goal</th>
<th>Conds</th>
<th>Have Bananas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Init</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre(A1)</td>
<td>Sells(HW, Drill)</td>
<td>Go HW ERP(A1)</td>
</tr>
<tr>
<td>Pre(A2)</td>
<td>At HW</td>
<td>Buy Drill ERP(A2)</td>
</tr>
<tr>
<td>Pre(A3)</td>
<td>At HW</td>
<td>Go SM ERP(A3)</td>
</tr>
<tr>
<td>Pre(A4)</td>
<td>Sells(SM, Bananas)</td>
<td>At SM Buy Ban ERP(A4)</td>
</tr>
</tbody>
</table>

Execute Highest True Kernel (rectangle including lower left corner and some upper right corner)

Hybrid Architectures

- Reactive lower level
- Deliberative higher level

robot

images

motor commands

Primitive Behaviors

sensory

predicates

planner/replanner

predictions

images

motor commands

Primitive Behaviors

Robot
Hybrid Architectures

- Reactive lower level
- Deliberative higher level

Diagram: Flowchart showing interactions between Planner/Replanner, Sensory Predicates, Primitive Behaviors, and Robot with labels for perceptions, predictions, actions, motor commands, images, sensors, and predicates.