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<sub>n</sub>, garb<sub>n</sub>
    - Every action at every odd depth index: cook<sub>n</sub>

---

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 Example

<table>
<thead>
<tr>
<th>Action</th>
<th>Preconditions</th>
<th>Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadGate</td>
<td>InLobby</td>
<td>KnowWhether(Gate1)</td>
</tr>
<tr>
<td>BoardPlane1</td>
<td>Gate1, AtGate1</td>
<td>AtPlane, ~AtGate1</td>
</tr>
<tr>
<td>BoardPlane2</td>
<td>~Gate1, AtGate2</td>
<td>OffPlane, ~AtGate2</td>
</tr>
<tr>
<td>GotoLobby</td>
<td>AtHome</td>
<td>AtLobby, ~AtHome</td>
</tr>
<tr>
<td>GotoGate1</td>
<td>AtLobby</td>
<td>AtGate1, ~AtLobby</td>
</tr>
<tr>
<td>GotoGate2</td>
<td>AtLobby</td>
<td>AtGate2, ~AtLobby</td>
</tr>
</tbody>
</table>
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
- 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
Hybrid Architectures

- Reactive lower level
- Deliberative higher level