Partial-order planning

STRIPS operators

Search vs. planning
Standard search algorithms seem to fail miserably:

Consider the task get milk, bananas, and a cordless drill.

Search V S. Planning
Planning systems do the following:

1) Open up action and goal representation to allow selection
2) Divide-and-conquer by subgoal solving
3) Relax requirement for sequential construction of solutions

<table>
<thead>
<tr>
<th>Constraints on actions</th>
<th>Sequence from $S_0$</th>
<th>Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical sentence (conjunction)</td>
<td>Lisp code</td>
<td>Goal</td>
</tr>
<tr>
<td>Preconditions/outcomes</td>
<td>Lisp code</td>
<td>Actions</td>
</tr>
<tr>
<td>Logical sentences</td>
<td>Lisp data structures</td>
<td>States</td>
</tr>
</tbody>
</table>

Plan

Search
Principal difficulty: unconstrained branching, hard to apply heuristics

\[ [\cdots \land (S \land H) \land (\neg H) \land (S_0 \land H) \land \neg (H \land S_0) \land (\neg H \land S_0)] = p \]

Solution

\[ \cdots \lor (\text{PlanResult}(p, s) \land \text{Have}(s)) \]
Tidily arranged actions descriptions, restricted language

Note: this abstracts away many important details!

\[
\begin{align*}
\text{Effect: conjunction of literals} \\
\text{Precondition: conjunction of positive literals} \\
\text{Precondition: efficient algorithm} \\
\text{Restricted language} \Rightarrow \text{efficient algorithm}
\end{align*}
\]

\[
\begin{align*}
\text{Effect: } & \text{Have}(x) \\
\text{Precondition: } & \text{At}(d) \wedge \text{Sells}(d', x) \\
\text{Action: } & \text{Buy}(x)
\end{align*}
\]
Gradually move from incomplete/vague plans to complete, correct plans

**Operators on partial plans:**

- **add a link** from an existing action to an open condition
- **add a step** to fulfill an open condition
- **order** one step wrt another

**Defn:** open condition is a precondition of a step not yet fulfilled

**Planning search:** node = partial plan

**Standard search:** node = concrete world state

---

**State space vs. Plan space**
A plan is complete iff every precondition is achieved and no possibly intervening step undoes it.
Return solution, c
with a precondition c that has not been achieved
pick a plan step, sneed from steps(plan)

function select-subgoal(plan) returns sneed, c

end

(resolve-threavts(plan)
choose-operator(plan, operators, sneed, c)
select-subgoal(plan) sneed, c

if solution? (plan) then return plan

loop do
plan -> make-minimal-plan(initial, goal)

function pop(initial, goal, operators) returns plan

Pop algorithm sketch
end
if not CONSISTENT(plan) then Fail

Promotion: Add $S^p \succ S^q$ to ORDERINGs(plan)

Demotion: Add $S^p \prec S^q$ to ORDERINGs(plan)

choose either

for each $S^q_{thread}$ that threatens a link $S^p \prec S^q$ in LINKS(plan) do

procedure RESOLVE-THREATS(plan)

add $S^p \succ S^q$ to Finish ORDERINGs(plan)
add $S^q \succ S^p$ to STEPS(plan)

if $S^q \succ S^p$ is a newly added step from operators then

add the ordering constraint $S^q \succ S^p \succ S^q$ to ORDERINGs(plan)
add the causal link $S^q \prec S^p$ to LINKS(plan)

if there is no such step then Fail

choose a step $S^q \succ S^p$ from operators or STEPS(plan) that has $c$ as an effect

procedure CHOOSE-OPERATOR(plan, operators, $S^q\prec S^p$, c)

POF algorithm cont'd.
Extensions for disjunction, universal, negation, conditionals

POP is sound, complete, and systematic (no repetition)

POP algorithm contd.
A clobberer is a potentially intervening step that destroys the condition achieved by a causal link. E.g., \( \text{Go} \left( \text{Home} \right) \) clobbers \( \text{At} \left( \text{HWS} \right) \).
Example: Blocks World

+ several inequality constraints

\[
\begin{align*}
\text{PutOnTable}(x) & \quad \text{Clear}(x) \land \text{On}(x, z) \\
\text{Clear}(x) & \quad \neg \text{On}(x, z) \\
\text{Clear}(z) & \quad \text{On}(x, y) \\
\text{Clear}(y) & \quad \neg \text{On}(x, y)
\end{align*}
\]

"Sussman anomaly" problem
Example cont.
Example contd.
Based on AMAI slides © S. Russell and P. Norvig, 1998

Example cont'd.
Example cont.