Outline

Partial-order planning ◊
STRIPS operators ◊
Search vs. planning ◊
Search vs. Planning

After-the-fact heuristic/goal test inadequate

Standard search algorithms seem to fail miserably:
Consider the task get milk, bananas, and a cordless drill
| Planning                        | Plan from 0
|--------------------------------|-----------------|
| Goal                           | Explicit/implicit
| Actions                         | Applicability conditions/transition
| States                          | Data structures
| Search                          | 1) Open up action and goal representation to allow selection
|                                 | 2) Divide-and-conquer by subgoals
|                                 | 3) relax requirement for sequential construction of solutions
|                                 | Constraints on actions
|                                 | Logical sentence (conjunction)
|                                 | Preconditions/outcomes
|                                 | Logical sentences

Search vs. Planning contd.
A complete set of STRIPS operators can be translated into a set of successor-state axioms. Effect: conjunction of literals
Precondition: conjunction of positive literals

\[ \text{Note: this abstracts away many important details!} \]

\[ \text{ACTION:} \ Buy(p, x) \]
\[ \text{PRECONDITION:} \ At(p) \wedge Sells(p, x) \]
\[ \text{EFFECT:} \ \text{Have}(x) \]

Tidily arranged actions descriptions, restricted language
and no possibly intervening step undoes it

A precondition is achieved iff it is the effect of an earlier step

A plan is complete iff every precondition is achieved

Open condition = precondition of a step not yet causally linked
temporal ordering between pairs of steps
causal links from outcome of one step to precondition of another

Finish step has the goal description as its precondition
Start step has the initial state description as its effect

Partially ordered plans
Example
Example
Example
If a conflict is unsolvable
Backtrack if an open condition is unachievable or
Gradually move from incomplete/vague plans to complete, correct plans

Order one step wrt another to resolve possible conflicts
Add a step to fulfill an open condition
Add a link from an existing action to an open condition

Operators on partial plans:
Function SELECT-SUBGOAL(plan) returns Succeeded, c

end

RESOLVE-THREATS(plan)

CHOOSE-OPERATOR(plan, operators, Succeeded, c)

Succeeded, c → SELECT-SUBGOAL(plan)

IF SOLUTION(plan) then return plan

loop do

plan → MAKE-MINIMAL-PLAN(initial, goal)

end

Function POP(initial, goal, operators) returns plan

Chapter 11
end

if not CONSISTENT(plan) then fail

PROMOTION: Add $s_i \rightarrow S_i$ to ORDERINGS(plan)

DEMOPTION: Add $S_i \leftarrow s_i$ to ORDERINGS(plan)

choose either

for each $S_i \in$ ORDERINGS(plan) do

procedure RESOLVE-THREATS(plan)

if $S_{new}$ is a newly added step from operators then

add the ordering constraint $S_{new} \rightarrow S_{old}$ to ORDERINGS(plan)

add the causal link $S_{new} \leftarrow c$ to LINKS(plan)

end

if there is no such step then fail

choose a step $S_{old}$ from operators or STEPS(plan) that has $c$ as an effect

procedure CHOOSE-OPERATOR(plan, operators, $S_{need}$, $c$)

POP algorithm cont.
Promotion: put after Buy(\text{Milk})

Demotion: put before Go(\text{Supermarket})

\begin{itemize}
  \item \text{Promotion} achieved by a causal link. E.g., Go(\text{Home}) clodbers At(\text{Supermarket}).
  \item A clodberer is a potentially intervening step that destroys the condition
\end{itemize}
Particularly good for problems with many loosely related subgoals
Can be made efficient with good heuristics derived from problem description
Extensions for disjunction, universals, negation, conditionals
POP is sound, complete, and systematic (no repetition)

- Selection of $S^{need}$ is irrevocable
- Choice of demotion or promotion for clanger
- Choice of $S^{add}$ to achieve $S^{need}$

Non-deterministic algorithm: backtracks at choice points on failure.

Properties of POP
Example: Blocks World

Start State

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
</table>

Goal State

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
</table>

C(F) E; >™ š;<œ«<?

PutOnTable(x)

Clear(x) On(x,z) Clear(y)

~On(x,z) Clear(z) On(x, Table)

~On(x,z) Clear(y)

Clear(x) On(x,y)

~On(x,z) Clear(y)

Clear(z) On(x, Table)

Several inequality constraints

"Sussman anomaly" problem
Example contd.
Example cont'd.
Example cont'd.
Example cont'd.