Outline

- Partial-order planning
- STRIPS operators
- 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.

Search vs. Planning.
### Plan

<table>
<thead>
<tr>
<th>Preconditions/Outcomes</th>
<th>Logical sentence (conjunction)</th>
<th>Plan from 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actions</td>
<td>Lisp code</td>
<td>Lisp code</td>
</tr>
<tr>
<td>States</td>
<td>Lisp data structures</td>
<td>States</td>
</tr>
</tbody>
</table>

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

### Search

Planning systems do the following:

- Search
Principal difficulty: unconstrained branching, hard to apply heuristics

\[
[\cdots] (\text{Go(Supermarket)}, \text{Buy(Milk)}, \text{Buy(Bananas)}, \text{Go(HWS)}) = d
\]

Solution

\[\cdots \land \neg \text{Have(Milk)} \land \text{Have(Home, s)} \land \text{At(result, s)} \land \text{Plan(result)} = s\]

Query

\[\left[ (\cdots \land \neg \text{Have(Milk)} \land \text{Have(Home, s)} \land \text{At(result, s)} \land \text{Plan(result)} = s) \Rightarrow (\text{Buy(Milk)} \land \text{At(Supermarket, s)}) \right] \land \text{Query(s)} \land \text{Actions as Successor State Axioms}\]

Initial state

\[\text{At(Home, s)} \land \neg \text{Have(Milk, s)} \land \text{Plan(result)}\]

Planning in situation calculus
\( \text{Have}(x) \)

\( \boxed{\text{Buy}(x)} \)

\( \text{At}(d), \text{Sells}(d', x) \)

Effect: conjunction of literals

Precondition: conjunction of positive literals

Restricted language \( \leq \) efficient algorithm

Note: this abstracts away many important details

\( \text{EFFECT: } \text{Have}(x) \)

\( \text{PRECONDITION: } \text{At}(d), \text{Sells}(d', x) \)

\( \text{ACTION: Buy}(x) \)

Tidily arranged actions descriptions, restricted language

---

STRIPS operators
radially move from incomplete/vague plans to complete, correct plans

order one step wrt another

add a step to fulfill an open condition

add a link from an existing action to an open condition

Operators on partial plans:

Define: 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
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

Partially ordered plans
Return a plan with a precondition that has not been achieved
pick a plan step $s_{new}$ from $steps(plan)$

\begin{align*}
&\text{function Select-Subgoal}(plan) \rightarrow \text{return } s_{new}, c
\end{align*}

end

\begin{align*}
&\text{Resolve-THREATS}(plan)
\end{align*}

\begin{align*}
&\text{Choose-OPERATOR}(plan, \text{operators}, s_{new}, c)
\end{align*}

\begin{align*}
&s_{new}, c \rightarrow \text{Select-Subgoal}(plan)
\end{align*}

\begin{align*}
&\text{if Solution?}(plan) \text{ then return } plan
\end{align*}

\begin{align*}
&\text{loop do}
\end{align*}

\begin{align*}
&\text{NAME-MINIMAL-PLAN}(\text{initial, goal})
\end{align*}

\begin{align*}
&\text{function POP}(\text{initial, goal, operators}) \rightarrow \text{return } plan
\end{align*}

Pop algorithm sketch
Extensions for disjunction, universals, negation, conditionals

\( \text{POP is sound, complete, and systematic (no repetition)} \)

end

if not consistent(\( \text{plan} \)) then fail

promote: add \( S \) \( \rightarrow \) \( \text{segments}(\text{plan}) \)

Demotion: add \( \text{segments} \) \( \leftarrow \) \( S \) \( \text{to \ segments}(\text{plan}) \)

choose either

for each \( \text{segment} \) that threatens a link \( S \) \( \leftarrow \) \( S' \) in \( \text{links}(\text{plan}) \) do

procedure resolve-threats(\( \text{plan} \))

add \( \text{start} \) \( \rightarrow \) \( S \) \( \text{to \ segments}(\text{plan}) \)

add \( S \) \( \text{to \ steps}(\text{plan}) \)

if \( S \) \( \text{add} \) is a newly added step from operators then

if \( S \) \( \text{add} \) is the ordering constraint then

add \( \text{the \ ordering \ constraint} \) \( \text{segments} \) \( \rightarrow \) \( S \text{add} \text{to \ segments}(\text{plan}) \)

add \( \text{the \ causal \ link} \) \( S \text{add} \rightleftharpoons \) \( S \text{add} \text{to \ links}(\text{plan}) \)

if there is no such step then fail

choose a step \( S \) \( \text{add} \) from operators or \( \text{steps}(\text{plan}) \) that has \( \text{c} \) as an effect

procedure choose-operator(\( \text{plan}, \text{operators}, S \text{add}, c \))
Promotion: put after Buy(Home) (III)

Demotion: put before Go(HWS) (III)

Achieved by a causal link. E.g., Go(Home)lobber AT(HWS) implies that the condition A llobber is a potentially intervening step that destroys the condition

Promotion and Demotion
Example: Blocks World

Start State

| A | B | C |

Goal State

| A | B | C |

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

 várias inequações

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