Planning

Philipp Koehn

26 March 2019
Outline

- Search vs. planning
- STRIPS operators
- Partial-order planning
- The real world
- Conditional planning
- Monitoring and replanning
search vs. planning
Search vs. Planning

- Consider the task *get milk, bananas, and a cordless drill*

- Standard search algorithms seem to fail miserably:

  - After-the-fact heuristic/goal test inadequate

- After-the-fact heuristic/goal test inadequate
Search vs. Planning

- Planning systems do the following
  1. improve action and goal representation to allow selection
  2. divide-and-conquer by **subgoaling**
  3. relax requirement for sequential construction of solutions

- Differences

<table>
<thead>
<tr>
<th></th>
<th><strong>Search</strong></th>
<th><strong>Planning</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>States</strong></td>
<td>Data structures</td>
<td>Logical sentences</td>
</tr>
<tr>
<td><strong>Actions</strong></td>
<td>Program code</td>
<td>Preconditions/outcomes</td>
</tr>
<tr>
<td><strong>Goal</strong></td>
<td>Program code</td>
<td>Logical sentence (conjunction)</td>
</tr>
<tr>
<td><strong>Plan</strong></td>
<td>Sequence from $S_0$</td>
<td>Constraints on actions</td>
</tr>
</tbody>
</table>
strips operators
STRIPS Operators

- Tidily arranged actions descriptions, restricted language

\[
\begin{align*}
\text{At}(p) & \quad \text{Sells}(p, x) \\
\text{Buy}(x) & \\
\text{Have}(x)
\end{align*}
\]

- **ACTION:** \( \text{Buy}(x) \)
- **PRECONDITION:** \( \text{At}(p), \text{Sells}(p, x) \)
- **EFFECT:** \( \text{Have}(x) \)

- Note: this abstracts away many important details!

- Restricted language \( \implies \) efficient algorithm
  - Precondition: conjunction of positive literals
  - Effect: conjunction of literals
partial-order planning
Partially Ordered Plans

- Partially ordered collection of steps with
  - Start step has the initial state description as its effect
  - Finish step has the goal description as its precondition
  - causal links from outcome of one step to precondition of another
  - temporal ordering between pairs of steps

- Open condition = precondition of a step not yet causally linked

- A plan is complete iff every precondition is achieved

- A precondition is achieved iff it is the effect of an earlier step and no possibly intervening step undoes it
Example

Start

At(Home)  Sells(HWS,Drill)  Sells(SM,Milk)  Sells(SM,Ban.)

Have(Milk)  At(Home)  Have(Ban.)  Have(Drill)

Finish
Example

At(Home)  Sells(HW,Drill)  Sells(SM,Milk)  Sells(SM,Ban.)

At(SM)  Sells(SM,Milk)

Buy(Milk)

Have(Milk)  At(Home)  Have(Ban.)  Have(Drill)

Finish
Example

\begin{itemize}
  \item \texttt{At(Home)}
  \item \texttt{Sells(HW,Drill)}
  \item \texttt{Sells(SM,Milk)}
  \item \texttt{Sells(SM,Ban.)}
\end{itemize}

\begin{itemize}
  \item \texttt{At(x)}
  \item \texttt{Go(SM)}
  \item \texttt{At(SM)}
  \item \texttt{Sells(SM,Milk)}
  \item \texttt{Buy(Milk)}
  \item \texttt{Have(Milk)}
  \item \texttt{At(Home)}
  \item \texttt{Have(Ban.)}
  \item \texttt{Have(Drill)}
  \item \texttt{Finish}
\end{itemize}
Example

Start

At(Home)  Sells(HWS,Drill)  Sells(SM,Milk)  Sells(SM,Ban.)

At(HWS)  Sells(HWS,Drill)

Buy(Drill)

At(x)

Go(SM)

At(SM)  Sells(SM,Milk)

Buy(Milk)

Have(Milk)  At(Home)  Have(Ban.)  Have(Drill)

Finish
Example
Planning Process

- 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 to remove possible conflicts

- Gradually move from incomplete/vague plans to complete, correct plans

- Backtrack if an open condition is unachievable or if a conflict is unresolvable
function POP(\textit{initial}, \textit{goal}, \textit{operators}) \textbf{returns} \textit{plan}

\textit{plan} ← MAKE-MINIMAL-PLAN(\textit{initial}, \textit{goal})

loop do
    if SOLUTION?(\textit{plan}) then return \textit{plan}
    \textit{S}_{\text{need}}, \textit{c} ← SELECT-SUBGOAL(\textit{plan})
    CHOOSE-OPERATOR(\textit{plan}, \textit{operators}, \textit{S}_{\text{need}}, \textit{c})
    RESOLVE-THREATS(\textit{plan})
end

function SELECT-SUBGOAL(\textit{plan}) \textbf{returns} \textit{S}_{\text{need}}, \textit{c}

pick a plan step \textit{S}_{\text{need}} from STEPS(\textit{plan})
    with a precondition \textit{c} that has not been achieved
return \textit{S}_{\text{need}}, \textit{c}
**Partially Ordered Plans Algorithm**

**procedure CHOOSE-OPERATOR**(plan, operators, S<sub>need</sub>, c)

choose a step S<sub>add</sub> from operators or STEPS(plan) that has c as an effect
if there is no such step then fail
add the causal link S<sub>add</sub> → S<sub>need</sub> to LINKS(plan)
add the ordering constraint S<sub>add</sub> < S<sub>need</sub> to ORDERINGS(plan)
if S<sub>add</sub> is a newly added step from operators then
    add S<sub>add</sub> to STEPS(plan)
    add Start < S<sub>add</sub> < Finish to ORDERINGS(plan)

**procedure RESOLVE-THREATS**(plan)

for each S<sub>threat</sub> that threatens a link S<sub>i</sub> → S<sub>j</sub> in LINKS(plan) do
    choose either
    Demotion: Add S<sub>threat</sub> < S<sub>i</sub> to ORDERINGS(plan)
    Promotion: Add S<sub>j</sub> < S<sub>threat</sub> to ORDERINGS(plan)
    if not CONSISTENT(plan) then fail
end
Clobbering and Promotion/Demotion

- A **clobberer** is a potentially intervening step that destroys the condition achieved by a causal link. E.g., $Go(Home)$ clobbers $At(Supermarket)$:

**Demotion**: put before $Go(Supermarket)$

**Promotion**: put after $Buy(Milk)$
Properties of Partially Ordered Plans

- Nondeterministic algorithm: backtracks at choice points on failure
  - choice of $S_{add}$ to achieve $S_{need}$
  - choice of demotion or promotion for clobberer
  - selection of $S_{need}$ is irrevocable

- Partially Ordered Plans is sound, complete, and systematic (no repetition)

- Extensions for disjunction, universals, negation, conditionals

- Can be made efficient with good heuristics derived from problem description

- Particularly good for problems with many loosely related subgoals
Example: Blocks World

"Sussman anomaly" problem

Start State

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

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

Goal State

Clear(x) On(x,z)
PutOnTable(x)

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

+ several inequality constraints
Example

On(C,A) On(A,Table) Cl(B) On(B,Table) Cl(C)

On(A,B) On(B,C)

FINISH
Example
Example
Example
the real world
The Real World

![Diagram of a car with a person falling out]

- \( \neg \text{Flat(Spare)} \)
- \( \text{Intact(Spare)} \)
- \( \text{Off(Spare)} \)
- \( \text{On(Tire1)} \)
- \( \text{Flat(Tire1)} \)

- \( \text{On(x)} \)
- \( \neg \text{Flat(x)} \)

- \( \text{Remove(x)} \)
- \( \text{Off(x)} \)
- \( \text{ClearHub} \)

- \( \text{Puton(x)} \)
- \( \text{On(x)} \)
- \( \neg \text{ClearHub} \)

- \( \text{Intact(x)} \)
- \( \text{Flat(x)} \)

- \( \text{Inflate(x)} \)
- \( \neg \text{Flat(x)} \)
Things Go Wrong

- **Incomplete information**
  - Unknown preconditions, e.g., $\text{Intact}(\text{Spare})$?
  - Disjunctive effects, e.g., $\text{Inflate}(x)$ causes $\text{Inflated}(x) \lor \text{SlowHiss}(x) \lor \text{Burst}(x) \lor \text{BrokenPump} \lor \ldots$.

- **Incorrect information**
  - Current state incorrect, e.g., spare NOT intact
  - Missing/incorrect postconditions in operators.

- **Qualification problem** can never finish listing all
  - required preconditions of actions
  - possible conditional outcomes of actions