Classical Planning

Deepak Kumar
November/December 2021

Blocksworld

• **Objects**    A, B, C, Table

• **Relations**  On\(^2\), Clear\(^1\)

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

• Example Knowledge Base

\[ \Delta = \{\text{On(C, Table)}, \text{On(A, C)}, \text{On(B, A)}, \text{Clear(B)}, \text{Clear(Table)}\} \]
STRIPS Representation

• STRIPS – Stanford Research Institute Problem Solver, 1971

• Actions are represented as operators (aka STRIPS Operator)

\textit{move}(x, y, z): \textit{move} x from y to z

\textbf{preconditions:} \textit{On}(x, y) \land \textit{Clear}(x) \land \textit{Clear}(z)

\textbf{delete:} \textit{Clear}(z), \textit{On}(x, y)

\textbf{add:} \textit{On}(x, z), \textit{Clear}(y), \textit{Clear}(Table)

• STRIPS Assumption

All literals not mentioned in \textit{delete} carry over to the next state.

Planning as search

• Initial State

• Goal

• Actions
Planning as Search

- **Initial State**
  \{On(B, A), Clear(Table), On(A, C), On(C, Table), Clear(B)\}

- **Goal**
  On(B, Table) ∧ On(A, B)

- **Actions**
  \textbf{move}(x, y, z): move x from y to z
  \textbf{preconditions}: On(x, y) ∧ Clear(x) ∧ Clear(z)
  \textbf{delete}: Clear(z), On(x, y)
  \textbf{add}: On(x, z), Clear(y), Clear(Table)

**Example**

- \textbf{move}(B, A, Table)
  \textbf{preconditions}: On(B, A) ∧ Clear(B) ∧ Clear(Table)
  \textbf{delete}: On(B, A), Clear(Table)
  \textbf{add}: On(B, Table), Clear(A), Clear(Table)

- KB = \{On(B, A), Clear(Table), On(A, C), On(C, Table), Clear(B)\}
Example

- move(B, A, Table)
  preconditions: On(B, A) \land \text{Clear}(B) \land \text{Clear}(\text{Table})
  delete: \text{On}(B, A), \text{Clear}(\text{Table})
  add: \text{On}(B, \text{Table}), \text{Clear}(A)

- KB = \{\text{On}(B, A), \text{Clear}(\text{Table}), \text{On}(A, C), \text{On}(C, \text{Table}), \text{Clear}(B)\}
Example

• move(B, A, Table)
  preconditions: On(B, A) ∧ Clear(B) ∧ Clear(Table)
  delete: On(B, A), Clear(B), Clear(Table)
  add: On(B, Table), Clear(A)

• KB = {Clear(Table), On(A, C), On(C, Table), On(B, Table), Clear(A)}

Planning Problem

• Given a goal, G (a wff)
• A current state Δ
• Find a sequence of actions to produce state, S such that S ⊨ G

Two ways to do this
• Forward Search / Progression Planning
• Backward Search / Regression Planning
Forward Search/ Progression Planning

Start with current state description
Apply STRIPS operators until a state description $S \models G$ is produced.

That is, very similar to our basic blind search problem...
Forward Search/ Progression Planning

Start with current state description
Apply STRIPS operators until a state description $S \models G$ is produced.

That is, very similar to our basic blind search problem...

Forward Search is impractical – # of rules & state descriptions are large

To improve, focus on goals – exploring islands
Forward Search/ Progression Planning- Islands

Identify islands in search space to focus search

e.g. Goal: On(A, Table) ∧ On(B, Table) ∧ On(C, B)
    On(C, Table) ∧ On(C, B) ∧ On(A, B)

Each conjunct is an island. Achieve one island, then another...

while Goal is not reached
    Select a conjunct of G
    Select an instance of an operator that adds it
    Create subgoals that are in its preconditions

Example

Goal: On(A, Table) ∧ On(B, Table) ∧ On(C, B)

move(x, y, z): move x from y to z
preconditions: On(x, y) ∧ Clear(x) ∧ Clear(z)
delete: Clear(z), On(x, y)
add: On(x, z), Clear(y), Clear(Table)
Example

Goal: On(A, Table) ∧ On(B, Table) ∧ On(C, B)

Select On(A, Table)

move(x, y, z): move x from y to z
preconditions: On(x, y) ∧ Clear(x) ∧ Clear(z)
delete: Clear(z), On(x, y)
add: On(x, z), Clear(y), Clear(Table)

move(A, y, Table) since On(A, Table) is in add
move(A, C, Table)
Example

Goal: On(A, Table) \land On(B, Table) \land On(C, B)

Select On(A, Table)

move(A, y, Table)
since On(A, Table) is in add
move(A, C, Table)

Clear(Table)
On(A, y)
\{y = C\}
On(A, C)

Example

Goal: On(A, Table) \land On(B, Table) \land On(C, B)

Select On(A, Table)

move(A, y, Table)
since On(A, Table) is in add
move(A, C, Table)

Clear(Table)
On(A, y)
\{y = C\}
On(A, C)

Example

move(x, y, z): move x from y to z
preconditions: On(x, y) \land Clear(x) \land Clear(z)
delete: Clear(z), On(x, y)
add: On(x, z), Clear(y), Clear(Table)
Example

Goal: \( \text{On}(A, \text{Table}) \land \text{On}(B, \text{Table}) \land \text{On}(C, B) \)

Select \( \text{On}(A, \text{Table}) \)

- \( \text{move}(A, y, \text{Table}) \) since \( \text{On}(A, \text{Table}) \) is in add
  - \( \text{move}(A, C, \text{Table}) \)

Clear(\text{Table}) false

- On(A, y) \{y = C\}
  - On(A, C) true

Move(x, A, z) since Clear(A) is in add

- \{x=B, z = \text{Table}\}
  - \( \text{move}(B, A, \text{Table}) \)

Clear(B) true

Clear(\text{Table}) true

On(B, A) true

\( \text{move}(x, y, z): \) move \( x \) from \( y \) to \( z \)

preconditions: \( \text{On}(x, y) \land \text{Clear}(x) \land \text{Clear}(z) \)

delete: \( \text{Clear}(z), \text{On}(x, y) \)

add: \( \text{On}(x, z), \text{Clear}(y), \text{Clear}(\text{Table}) \)
Example

Select On(A, Table)

move(A, y, Table)
since On(A, Table) is in add

move(A, C, Table)

Clear(A)

Clear(Table)

On(A, y)

{y = C}

On(A, C)

Plan: [move(B, A, Table), move(A, C, Table)]

This satisfies On(A, Table)

Example

Select On(A, Table)

move(A, y, Table)
since On(A, Table) is in add

move(A, C, Table)

Clear(A)

Clear(Table)

On(A, y)

{y = C}

On(A, C)

Goal: On(A, Table) \land On(B, Table) \land On(C, B)

Goal: On(A, Table) \land On(B, Table) \land On(C, B)

Move(x, A, z)
since Clear(A) is in add

{x = B, z = Table}

move(B, A, Table)

Clear(B)

Clear(Table)

On(B, A)

move(B, A, Table)

Clear(B)

Clear(Table)

On(B, A)

preconditions: On(x, y) \land Clear(x) \land Clear(z)

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

add: On(x, z), Clear(y), Clear(Table)
Example

Goal: \( \text{On}(A, \text{Table}) \land \text{On}(B, \text{Table}) \land \text{On}(C, B) \)

Select \( \text{On}(A, \text{Table}) \)

move(\(A, y, \text{Table}\))

since \( \text{On}(A, \text{Table}) \) is in add

\( \text{move}(A, C, \text{Table}) \)

Clear(\(A\))

\( \{y = C\} \)

\( \text{On}(A, C) \)

Clear(\(\text{Table}\))

Move(\(x, A, z\))

since Clear(\(A\)) is in add

\( \{x = B, z = \text{Table}\} \)

\( \text{move}(B, A, \text{Table}) \)

Clear(\(B\))

Clear(\(\text{Table}\))

\( \text{On}(B, A) \)

Plan: [\( \text{move}(B, A, \text{Table}), \text{move}(A, C, \text{Table}) \)]

This satisfies \( \text{On}(A, \text{Table}) \)

Next, select \( \text{On}(B, \text{Table}) \)

Already satisfied.

Goal:

\( \text{On}(A, \text{Table}) \land \text{On}(B, \text{Table}) \land \text{On}(C, B) \)
Example

Goal: \( \text{On}(A, \text{Table}) \land \text{On}(B, \text{Table}) \land \text{On}(C, B) \)

Select \( \text{On}(A, \text{Table}) \)

\( \text{move}(A, y, \text{Table}) \) since \( \text{On}(A, \text{Table}) \) is in add

\( \text{move}(A, C, \text{Table}) \)

Clear(\text{Table}) \( \checkmark \)

\( \text{On}(A, y) \)

\( \{y = C\} \)

\( \text{On}(A, C) \)

Clear(A) \( \times \)

Move(x, A, z) since Clear(A) is in add

\( \{x = B, z = \text{Table}\} \)

\( \text{move}(B, A, \text{Table}) \)

Clear(B) \( \checkmark \)

Clear(\text{Table}) \( \checkmark \)

\( \text{On}(B, A) \)

Next, select \( \text{On}(B, \text{Table}) \)

Already satisfied. \( \checkmark \)

Next, select \( \text{On}(C, B) \)

... \( \text{move}(C, \text{Table}, B) \)

Plan: \[\text{move}(B, A, \text{Table}), \text{move}(A, C, \text{Table})\]

This satisfies \( \text{On}(A, \text{Table}) \)

Plan: \[\text{move}(B, A, \text{Table}), \text{move}(A, C, \text{Table}), \text{move}(C, \text{Table}, B)\]

Next, select \( \text{On}(B, \text{Table}) \)

Already satisfied. \( \checkmark \)

Next, select \( \text{On}(C, B) \)

... \( \text{move}(C, \text{Table}, B) \)
Sussman Anomaly

- Goal: $\text{On}(A, B) \land \text{On}(B, C)$

- Select: $\text{On}(A, B)$ to get plan: $[\text{move}(C, A, \text{Table}), \text{move}(A, \text{Table}, B)]$

- Select: $\text{On}(B, C)$
  
it will undo $\text{On}(A, B)$

- Selecting $\text{On}(B, C)$ first it will simply $\text{move}(B, \text{Table}, C)$ and then will have to undo it to achieve $\text{On}(A, B)$.

Forward Search/ Progression Planning - Islands

Identify islands in search space to focus search

e.g. Goal: $\text{On}(A, \text{Table}) \land \text{On}(B, \text{Table}) \land \text{On}(C, B) \land \text{On}(C, \text{Table}) \land \text{On}(C, B) \land \text{On}(A, B)$

Each conjunct is an island. Achieve one island, then another...

while Goal is not reached
Select a conjunct of $G$
Select an instance of an operator that adds it
Create subgoals that are in its preconditions

Can work, but tends to undo earlier achieved islands...as we saw in Sussman Anomaly.
Backward Search/Regression Planning

- Regress goals through STRIPS operators.

\[
\begin{align*}
&\text{Backward Search/Regression Planning} \\
&\quad \text{• Regress goals through STRIPS operators.}
\end{align*}
\]
Backward Search/Regression Planning

Goal

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

move(x, y, z):
move x from y to z
preconditions: On(x, y) ∧ Clear(x) ∧ Clear(z)
delete: Clear(z), On(x, y)
add: On(x, z), Clear(y), Clear(Table)

New subgoal
Backward Search/Regression Planning

move(x, y, z): move x from y to z
preconditions: On(x, y) ∧ Clear(x) ∧ Clear(z)
deltes: Clear(z), On(x, y)
add: On(x, z), Clear(y), Clear(Table)
Backward Search/Regression Planning

Goal

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

move(A, Table, B)
(x/A, y/Table, z/B)

move(B, Table, C)
(x/B, y/Table, z/C)

move(C, Table, A)
(x/C, y/Table, z/A)

move(x, y, z): move x from y to z
preconditions: On(x, y) ∧ Clear(x) ∧ Clear(z)
delete: Clear(z), On(x, y)
add: On(x, z), Clear(y), Clear(Table)

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

move(A, B, C)
(x/A, y/B, z/C)

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

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

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

move(A, Table, C)
(x/A, y/Table, z/C)

move(B, Table, C)
(x/B, y/Table, z/C)

move(C, Table, A)
(x/C, y/Table, z/A)

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

move(A, B, C)
(x/A, y/B, z/C)

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

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

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

move(A, Table, C)
(x/A, y/Table, z/C)

move(B, Table, C)
(x/B, y/Table, z/C)

move(C, Table, A)
(x/C, y/Table, z/A)
Backward Search/Regression Planning

move(x, y, z):
move x from y to z
preconditions: On(x, y) ∧ Clear(x) ∧ Clear(z)
delete: Clear(z), On(x, y)
add: On(x, z), Clear(y), Clear(Table)
Backward Search/Regression Planning

- Regress goals through STRIPS operators.
- Keeps branching factor lower than forward search for most problem domains.
- Can use least commitment planning
  i.e. do not commit to variable instantiations if possible

Avoids Sussman Anomaly.
Backward Search/Regression Planning

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

Goal

- On(A, Table)
- Clear(A)
- Clear(B)
- On(B, C)
- On(C, Table)

move(x, y, z):
- move x from y to z
  - preconditions: On(x, y) ∧ Clear(x) ∧ Clear(z)
  - delete: Clear(z)
  - add: On(x, z), Clear(y), Clear(Table)

Non-linear Planning
(aka Partial Order Planning)

- So far all plans produced are linear plans
  - [move(B, A, Table), move(A, C, Table), move(B, Table, C), ...]

- Search for plans was done by searching through a space of states

- Instead, search a space of plans!
State Space vs Plan Space

Plan component

Plan transformation operator

Incomplete plan
Non-linear Planning
(aka Partial Order Planning)

• So far all plans produced are linear plans

  [move(B, A, Table), move(A, C, Table), move(B, Table, C), ...]

• Search for plans was done by searching through a space of states

• Instead, search a space of plans!

Planning in Plan Space

• A set of plan components – plan state

  STRIPS rules

• Plan operators
  • Adding steps to the plan
  • Reorder steps already in plan
  • Changing a partially ordered plan into a fully ordered plan
  • Changing a plan schema (replacing un-instantiated variables)
SRIPS Rule Representation

Add conditions
- Clear(y)
- On(x, z)

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

Operator
move(x, y, z)

Partial Order Planning

Goal State
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Initial State
<table>
<thead>
<tr>
<th>C</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table
Partial Order Planning

Diagram showing the Initial State and Goal State with actions and constraints for Partial Order Planning.

Extend by adding an action to achieve one of the conjuncts of the goal.
Partial Order Planning

Goal State

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

Table

Initial State

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

Table

Pick a goal

<table>
<thead>
<tr>
<th>Nil</th>
</tr>
</thead>
</table>

finish

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

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

start

True

Partial Order Planning

Goal State

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

Table

Initial State

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

Table

Pick a goal

<table>
<thead>
<tr>
<th>Nil</th>
</tr>
</thead>
</table>

finish

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

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

start

True
Partial Order Planning

Goal State

Initial State

Goal State

Initial State
Goal State

Initial State

Threat Analysis

b < a what about c?
Threat Analysis

Goal State

Initial State

start

Clear(Table)  On(A, Table)  Clear(A)  On(C, Table)  move(C, A, Table)  Clear(Table)  Clear(C)  On(B, Table)  move(B, Table, C)

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

Clear(B)  On(B, Table)  Clear(C)  On(B, Table)

On(C, A)  Clear(Table)  On(C, Table)  move(C, A, Table)  Clear(Table)  Clear(C)  On(B, Table)  move(B, Table, C)

Clear(A)  On(A, Table)  Clear(A)  On(A, Table)  move(A, Table, B)  Clear(Table)  Clear(C)  On(B, Table)  move(B, Table, C)

On(A, B)  Clear(B)  On(B, Table)

Clear(C)  On(C, Table)  move(A, Table, B)  Clear(Table)  Clear(C)  On(B, Table)  move(B, Table, C)

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

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

Clear(A)  On(A, Table)  Clear(A)  On(A, Table)  move(A, Table, B)  Clear(Table)  Clear(C)  On(B, Table)  move(B, Table, C)

Clear(B)  On(B, Table)  Clear(C)  On(B, Table)

On(C, A)  Clear(Table)  On(C, Table)  move(C, A, Table)  Clear(Table)  Clear(C)  On(B, Table)  move(B, Table, C)

Clear(A)  On(A, Table)  Clear(A)  On(A, Table)  move(A, Table, B)  Clear(Table)  Clear(C)  On(B, Table)  move(B, Table, C)

Clear(B)  On(B, Table)  Clear(C)  On(B, Table)

On(C, A)  Clear(Table)  On(C, Table)  move(C, A, Table)  Clear(Table)  Clear(C)  On(B, Table)  move(B, Table, C)

Clear(A)  On(A, Table)  Clear(A)  On(A, Table)  move(A, Table, B)  Clear(Table)  Clear(C)  On(B, Table)  move(B, Table, C)

Clear(B)  On(B, Table)  Clear(C)  On(B, Table)

On(C, A)  Clear(Table)  On(C, Table)  move(C, A, Table)  Clear(Table)  Clear(C)  On(B, Table)  move(B, Table, C)

Clear(A)  On(A, Table)  Clear(A)  On(A, Table)  move(A, Table, B)  Clear(Table)  Clear(C)  On(B, Table)  move(B, Table, C)

Clear(B)  On(B, Table)  Clear(C)  On(B, Table)
what about c?
1. c < a
2. b < c
That is
b < c < a
is a Total Ordering