Classical Planning

Deepak Kumar

December 2017

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

  \[ \Delta = \{ \text{On}(C, \text{Table}), \text{On}(A, C), \text{On}(B, A), \text{Clear}(B), \text{Clear}(\text{Table}) \} \]
Tell-A-sk-Do Systems

Can specify a goal as a wff:
Achieve On(A, B) ∧ On(B, C) ∧ On(C, Table)
Achieve On(A, B)
Achieve ∃x On(x, B)
Etc.

Planning Algorithms

• Situation Calculus – Logic-based Planning

• STRIPS

• Planning as Search
  • Forward Search/Progression Planning
  • Backward Search/regression Planning

• Non-linear Planning (Partial Order Planning)
Situation Calculus – Action Schema

• **Action Schema:**
  \( \text{move}(x, y, z) \) - move \( x \), from \( y \), to \( z \)

• **Function do():**
  if \( \alpha \) is an action and \( \sigma \) is a state then
  \( \text{do}(\alpha, \sigma) \rightarrow \sigma' \)
  \( \sigma' \) is the state resulting from doing \( \alpha \) in \( \sigma \)

• Facts are represented by fluents
  \{On(A, B, S_0), On(A, C, S_0), On(C, Table, S_0)\}

Representing States & Actions

• Modeling Actions with Axioms
  \( \forall x\forall y\forall s [\text{On}(x, y, s) \land \text{Clear}(x, s) \land \text{Clear}(z, s) \land x \neq z \Rightarrow \text{On}(x, z, \text{do}(\text{move}(x, y, z), s))] \)
  \( \forall x\forall y\forall s [\text{On}(x, y, s) \land \text{Clear}(x, s) \land \text{Clear}(z, s) \land x \neq z \Rightarrow \neg\text{On}(x, \text{do}(\text{move}(x, y, z), s))] \)
  \( \forall x\forall y\forall s [\text{On}(x, y, s) \land \text{Clear}(x, s) \land \text{Clear}(z, s) \land (x \neq z) \land (y \neq z) \Rightarrow \text{Clear}(y, \text{do}(\text{move}(x, y, z), s))] \)
  \( \forall x\forall y\forall s [\text{On}(x, y, s) \land \text{Clear}(x, s) \land \text{Clear}(z, s) \land (x \neq z) \land (z \neq \text{Table}) \Rightarrow \neg\text{Clear}(z, \text{do}(\text{move}(x, y, z), s))] \)

• Frame Axioms for Move-On
  \( \forall x\forall y\forall s [\text{On}(x, y, s) \land (x \neq u) \Rightarrow \text{On}(x, y, \text{do}(\text{move}(u, v, z), s))] \)
  \( \forall x\forall y\forall s [\neg\text{On}(x, y, s) \land [(x \neq u) \lor (y \neq z)] \Rightarrow \neg\text{On}(x, y, \text{do}(\text{move}(u, v, z), s))] \)
  \( \forall x\forall y\forall s [\text{Clear}(u, s) \land (u \neq z) \Rightarrow \text{Clear}(u, \text{do}(\text{move}(x, y, z), s))] \)
  \( \forall x\forall y\forall s [\neg\text{Clear}(u, s) \land (u \neq y) \Rightarrow \neg\text{Clear}(u, \text{do}(\text{move}(x, y, z), s))] \)
Planning using Situation Calculus

• To get a plan that achieves a goal: \( \Upsilon(s) \)

• We need to prove: \( \exists s \Upsilon(s) \)

• We’ll get the plan by using Green’s Trick: \( \exists s \Upsilon(s) \lor \text{Answer(s)} \)

• Goal: \( \exists s \text{On}(B, \text{Table}, s) \)

• Example 1: Prove: \( \neg [\exists s \text{On}(B, \text{Table}, s)] \lor \text{Answer(s)} \)
  i.e. Add the clause: \( \neg \text{On}(B, \text{Table}, s) \lor \text{Answer(s)} \) to KB (in clause form)
  Answer(do(move(B, A, Table), S_0))

• Example 2: Prove: \( \neg [\exists s \text{On}(A, B, s) \land \text{On}(B, \text{Table}, s)] \lor \text{Answer(s)} \)
  i.e. Add the clause: \( \neg \text{On}(A, B, s) \lor \neg \text{On}(B, \text{Table}, s) \lor \text{Answer(s)} \) to KB (in clause form)
  Answer(do(move(A, C, B), do(move(B, A, Table), S_0)))

Problems with Situation Calculus Planning

• Too many Frame Axioms

• Proof effort is too large for even simple problems

• There are more problems:
  
  • Frame Problem
  • Qualifications Problem
  • Ramification Problem
  • Etc.
STRIPS Planner

• STRIPS – STanford Research Institute Problem Solver, 1971

• Actions are represented as operators (aka STRIPS Operator)

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

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

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

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

Alternatively:

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

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

effects: \(\neg\text{Clear}(z), \neg\text{On}(x, y), \text{On}(x, z), \text{Clear}(y), \text{Clear}(\text{Table})\)
STRIPS Representation

• STRIPS – STanford Research Institute Problem Solver, 1971

• Actions are represented as operators (aka STRIPS Operator)

move(x, y, z): move x from y to z
  \textbf{preconditions:}\ On(x, y) \land \text{Clear}(x) \land \text{Clear}(z)
  \textbf{delete:}\ \text{Clear}(z), \text{On}(x, y)
  \textbf{add:}\ \text{On}(x, z), \text{Clear}(y), \text{Clear}(\text{Table})

• STRIPS Assumption

  All literals not mentioned in \textbf{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**

\begin{align*}
\text{On}(B, \text{Table}) \\
\land \\
\text{On}(B, \text{Table}) \land \text{On}(A, B)
\end{align*}

• **Actions**

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

\begin{itemize}
  \item \textbf{preconditions}: On(x, y) \land \text{Clear}(x) \land \text{Clear}(z)
  \item \textbf{delete}: \text{Clear}(z), \text{On}(x, y)
  \item \textbf{add}: \text{On}(x, z), \text{Clear}(y), \text{Clear}(\text{Table})
\end{itemize}

**Example**

• \textbf{move}(B, A, Table)

\begin{itemize}
  \item \textbf{preconditions}: On(B, A) \land \text{Clear}(B) \land \text{Clear}(\text{Table})
  \item \textbf{delete}: On(B, A), \text{Clear}(\text{Table})
  \item \textbf{add}: On(B, Table), \text{Clear}(A), \text{Clear}(\text{Table})
\end{itemize}

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

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

• KB = \{On(B, A), Clear(Table), On(A, C), On(C, Table), 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 $S \models G$

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
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, Floor) ∧ On(B, Floor) ∧ On(C, B)
On(C, Floor) ∧ 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, Floor) ∧ On(B, Floor) ∧ 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(y, y)
add: On(x, z), Clear(y), Clear(Table)
Example

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

Select On(A, Floor)

move(A, y, Floor)

since On(A, Floor) is in add

move(A, C, Floor)
Example

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

Select On(A, Floor)

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

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

Example

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

Select On(A, Floor)

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

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

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

Table

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

Select On(A, Floor)

move(A, y, Floor)

since On(A, Floor) is in add
move(A, C, Floor)

Clear(Floor) On(A, y) {y = C} On(A, C) Clear(A)

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

{x = B, z = Floor}
move(B, A, Floor)

Clear(B) Clear(Floor) On(B, A)

Example

Table

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

Select On(A, Floor)

move(A, y, Floor)

since On(A, Floor) is in add
move(A, C, Floor)

Clear(Floor) On(A, y) {y = C} On(A, C) Clear(A)

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

{x = B, z = Floor}
move(B, A, Floor)

Clear(B) Clear(Floor) On(B, A)
Example

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

Select On(A, Floor)

move(A, y, Floor)

since On(A, Floor) is in add

move(A, C, Floor)

Clear(Floor)

On(A, y)

\{y = C\}

On(A, C)

Clear(A)

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

This satisfies On(A, Floor)
Example

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

Select On(A, Floor)

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

Clear(Floor)

On(A, y)
{y = C}
On(A, C)

Clear(A)

Next, select On(B, Floor)

Already satisfied.

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

Example

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

Select On(A, Floor)

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

Clear(Floor)

On(A, y)
{y = C}
On(A, C)

Clear(A)

Next, select On(B, Floor)

Already satisfied.
Example

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

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

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

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

Next, select \(\text{On}(B, \text{Floor})\)
- Already satisfied.

Next, select \(\text{On}(C, B)\)
- ... \(\text{move}(C, \text{Floor}, B)\)

Plan: \[\text{move}(B, A, \text{Floor}), \text{move}(A, C, \text{Floor}), \text{move}(C, \text{Floor}, B)\]
Sussman Anomaly

- **Goal**: On(A, B) \(\land\) On(B, C)

- **Select**: On(A, B) to get plan: [move(C, A, Floor), move(A, Floor, B)]

- **Select**: On(B, C)

  it will undo On(A, B)

- **Selecting On(B, C) first** it will simply move(B, Floor, C) and then will have to undo it to achieve On(A, B).

**Forward Search/ Progression Planning- Islands**

Identify islands in search space to focus search

e.g. **Goal**: On(A, Floor) \(\land\) On(B, Floor) \(\land\) On(C, B) 
\(\land\) On(C, Floor) \(\land\) On(C, B) \(\land\) 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.

\[ \text{Clear(Table)} \]
\[ \text{On(A, C)} \]
\[ \text{On(C, Table)} \]
\[ \text{On(B, A)} \]
\[ \text{Clear(A)} \]

\[ \text{On(A, B)} \]
\[ \text{On(B, C)} \]
\[ \text{On(C, Table)} \]

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)
Backward Search/Regression Planning

Backward Search/Regression Planning

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)
Backward Search/Regression Planning

**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), On(x, y)
- **add**: On(x, z), Clear(y), Clear(Table)

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

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

**On(A, B)**
- On(B, C)
- On(C, Table)
Backward Search/Regression Planning

On(B, Table) Clear(C) On(A, B) On(C, Table)
move(B, Table, C) x/B, y/Table, z/C

On(A, B) On(B, C) On(C, Table)
goal
move(A, Table, B) x/A, y/Table, z/B

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

Backward Search/Regression Planning

On(B, Table) Clear(C) On(A, B) On(C, Table)
move(B, Table, C) x/B, y/Table, z/C

On(A, B) On(B, C) On(C, Table)
goal
move(A, Table, B) x/A, y/Table, z/B

On(A, Table)
Clear(A)
On(B, C)
On(B, Table)
Clear(B)
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)
Backward Search/Regression Planning

1. On(A, Table)
2. Clear(A)
3. Clear(B)
4. On(B, C)
5. 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)

1. On(A, C)
2. Clear(A)
3. Clear(B)
4. On(B, Table)
5. Clear(B)
6. On(C, Table)

1. On(A, C)
2. Clear(A)
3. Clear(B)
4. On(B, Table)
5. Clear(B)
6. On(C, 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

Non-linear Planning
(aka Partial Order Planning)

• So far all plans produced are linear plans
  
  \([\text{move}(B, A, \text{Table}), \text{move}(A, C, \text{Table}), \text{move}(B, \text{Table}, C), ...]\)

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

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

State Space vs Plan Space

Plan component
Plan transformation operator
Incomplete plan

State Space vs Plan Space

State Space vs Plan Space

State Space vs Plan Space

State Space vs Plan Space
Non-linear Planning
(aka Partial Order Planning)

• So far all plans produced are linear plans

  \[\text{move}(B, A, \text{Table}), \text{move}(A, C, \text{Table}), \text{move}(B, \text{Table}, C), \ldots\]\n
• 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

Partial Order Planning

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

<table>
<thead>
<tr>
<th>C</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td>Initial State</td>
<td></td>
</tr>
</tbody>
</table>
Partial Order Planning

Goal State

Initial State

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

Goal State

Initial State
Partial Order Planning

Goal State

Initial State

Goal State

Initial State
Threat Analysis
Threat Analysis

b < a
what about c?
1. c < a
2. b < c

That is
b < c < a
is a Total Ordering