CS 4649/7649 RIP
Robot Intelligence: Planning

Approaches to Classical Planning

Sungmoon Joo

School of Interactive Computing
College of Computing
Georgia Institute of Technology

Course Info.

- Course Website: joosm.github.io/RIP2014
  - Lecture slides
  - Project examples

- Email me(sungmoon.joo@cc.gatech.edu)
  - Introduce yourself, your github id
  - Project ideas
STRIPS: Blocks World Example

Constants:
A, B, C, Table
IsBlock(A), IsBlock(B)...

Ground Literals:
On(B, A) On(C, Table)
Clear(B) Clear(Table)

Actions: move(x, y, z)

PC: On(x, y) Clear(x) Clear(y)
D: On(x, y) A: On(x, z)

Domain Axioms:
On(y, z) ∧ On(z, x) ∧ (x ≠ Table) ⇒ (y = z)

STRIPS Planner*

*STRIPS (Stanford Research Institute Problem Solver)
Means – Ends Analysis

- Means-Ends Analysis
  - Search only relevant aspect of problem
  - What means (operators) are available to achieve the desired ends (goal)
  - Find difference between goal and current state
  - Find operator to reduce the difference
  - Perform means-ends analysis on new subgoals

Properties of a Planner

1) Sound: The planner produces valid plans
   - STRIPS is sound

2) Optimal: The planner produces optimal (shortest) plans
   - STRIPS is suboptimal

3) Complete: The planner finds a solution when there is one or returns that the solution is not possible.
   - STRIPS is incomplete!
Anomaly

*Called ‘anomaly’ because it seemed to make sense for a conjunctive goals to first achieve one goal and then achieve another goal, and then the complete goal would be achieved.

Airplane Logistics

Init: At(O1, A) At(O2, A) Plane(747)
      At(747, A) Have-Fuel(747)

Goal: At(O1, B) \land At(O2, B)
Airplane Logistics

<table>
<thead>
<tr>
<th>Stack</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>At(O1, B)</td>
<td>At(O1, A)</td>
</tr>
<tr>
<td>At(O2, B)</td>
<td>At(O2, A)</td>
</tr>
<tr>
<td>At(O1, B) ^ At(O2, B)</td>
<td>At(747, A) ^ HaveFuel(747)</td>
</tr>
<tr>
<td><strong>In(O1, 747)</strong></td>
<td><strong>At(O1, A)</strong></td>
</tr>
<tr>
<td><strong>At(747, B)</strong></td>
<td><strong>At(O2, A)</strong></td>
</tr>
<tr>
<td><strong>UnloadO1 747, B</strong></td>
<td><strong>At(O2, A)</strong></td>
</tr>
<tr>
<td><strong>At(O1, B)</strong></td>
<td><strong>At(O2, A)</strong></td>
</tr>
<tr>
<td><strong>At(O1, B) ^ At(O2, B)</strong></td>
<td><strong>HaveFuel(747)</strong></td>
</tr>
</tbody>
</table>

---

Airplane Logistics

<table>
<thead>
<tr>
<th>Stack</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>At(O1, B)</td>
<td>At(O1, A)</td>
</tr>
<tr>
<td>At(O2, B)</td>
<td>At(O2, A)</td>
</tr>
<tr>
<td>At(O1, B) ^ At(O2, B)</td>
<td>At(747, A) ^ HaveFuel(747)</td>
</tr>
<tr>
<td><strong>In(O1, 747)</strong></td>
<td><strong>At(O1, A)</strong></td>
</tr>
<tr>
<td><strong>At(747, B)</strong></td>
<td><strong>At(O2, A)</strong></td>
</tr>
<tr>
<td><strong>UnloadO1 747, B</strong></td>
<td><strong>At(O2, A)</strong></td>
</tr>
<tr>
<td><strong>At(O1, B)</strong></td>
<td><strong>At(O2, A)</strong></td>
</tr>
<tr>
<td><strong>At(O1, B) ^ At(O2, B)</strong></td>
<td><strong>HaveFuel(747)</strong></td>
</tr>
</tbody>
</table>

---

Airplane Logistics

<table>
<thead>
<tr>
<th>Stack</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>At(O1, B)</td>
<td>At(O1, A)</td>
</tr>
<tr>
<td>At(O2, B)</td>
<td>At(O2, A)</td>
</tr>
<tr>
<td>At(O1, B) ^ At(O2, B)</td>
<td>At(747, A) ^ HaveFuel(747)</td>
</tr>
<tr>
<td><strong>At(O1, A)</strong></td>
<td><strong>At(747, A)</strong></td>
</tr>
<tr>
<td><strong>At(747, A)</strong></td>
<td><strong>At(747, A)</strong></td>
</tr>
<tr>
<td><strong>UnloadO1 747, B</strong></td>
<td><strong>At(O2, A)</strong></td>
</tr>
<tr>
<td><strong>At(O1, B)</strong></td>
<td><strong>At(O2, A)</strong></td>
</tr>
<tr>
<td><strong>At(O1, B) ^ At(O2, B)</strong></td>
<td><strong>HaveFuel(747)</strong></td>
</tr>
<tr>
<td><strong>At(747, B)</strong></td>
<td><strong>In(O1, 747)</strong></td>
</tr>
<tr>
<td><strong>UnloadO1 747, B</strong></td>
<td><strong>At(O2, A)</strong></td>
</tr>
<tr>
<td><strong>At(O2, B)</strong></td>
<td><strong>At(O2, A)</strong></td>
</tr>
<tr>
<td><strong>At(O1, B) ^ At(O2, B)</strong></td>
<td><strong>HaveFuel(747)</strong></td>
</tr>
</tbody>
</table>
Airplane Logistics

Plan:
- Load(O₁, 747, A)
- Fly(747, A, B)
- Land(O₁, 747, A)
- Unload(O₂, 747, B)
- At(O₂, B)
- At(747, B)
- HaveFuel(747)

No action that adds this!
Was there a way to solve this problem?

<table>
<thead>
<tr>
<th>Load(o,p,loc)</th>
<th>Unload(o,p,loc)</th>
<th>Fly(p,from,to)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC: At(o,loc)</td>
<td>PC: In(o,p)</td>
<td>PC: At(p,from)</td>
</tr>
<tr>
<td>At(p,loc)</td>
<td>At(o,loc)</td>
<td>Have-Fuel(p)</td>
</tr>
<tr>
<td>Add: In(o,p)</td>
<td>Delete: At(o,loc)</td>
<td>Add: At(p,to)</td>
</tr>
<tr>
<td>Delete: At(p,loc)</td>
<td>Delete: In(o,p)</td>
<td>Delete: At(p,from)</td>
</tr>
</tbody>
</table>

Init: At(O_1, A) \land At(O_2, A) \land Plane(747) \land At(747, A) \land Have-Fuel(747)

Goal: At(O_1, B) \land At(O_2, B)

Properties of a Planner

1) Sound: The planner produces valid plans
   - STRIPS is sound

2) Optimal: The planner produces optimal (shortest) plans
   - STRIPS is suboptimal

3) Complete: The planner finds a solution when there is one or returns that the solution is not possible.
   - STRIPS is incomplete!
STRIPS: Shakey SRI 1966-1972

- “SHAKEY: Experimenation in Robot Learning and Planning”
- Cameras, Tactile Sensors
- 1.35 MB of code

http://www.ai.sri.com/shakey/

Shakey Unblocks a Door

**Shakey: Pushing**

```
PUSH(OB,X,Y)

Preconditions:
(IXX) [INROOM(ROBOT,BX) ∧ INROOM(OB,RX) ∧ LOCINROOM(X,Y,RX)] ∧ PUSHABLE(OB)

Delete List:
AT(ROBOT,$1,$2)
NEXTTO(ROBOT,$1)
AT(OB,$1,$2)
NEXTTO(OB,$1)
NEXTTO($1,OB)

Add List:
*AT(OB,X,Y)
NEXTTO(ROBOT,OB)
```


---

**Shakey: Unblocking, why not Pushing?**

```
UNBLOCK(DX,RX,BX)

Preconditions:
BLOCKED(DX,RX,BX) ∧ INROOM(ROBOT,RX) ∧ PUSHABLE(BX)

Delete List:
AT(ROBOT,$1,$2)
BLOCKED(DX,RX,BX)
AT(BX,$1,$2)
NEXTTO(ROBOT,$1)
NEXTTO(BX,$1)
NEXTTO($1,BX)

Add List:
*UNBLOCKED(DX,RX)
NEXTTO(ROBOT,BX)
```

Linear Planning Analysis

- Stack search was greedy to achieve sub-goals
- "Depth First" with incomplete backtracking
- Not effective when sub-goals interact
- But we like Means-Ends because it is efficient
- Is there a better way?

Observations

- Forward search was undesirable due to options in initial state
- Goal state has fewer conjuncts than ground literals in initial state

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

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

- Let's search backwards!
Regression Planning

Regression Planning (Variables Show Up)
Regression Planning

- Just as with forward (progression) planning this algorithm would be complete. However:
  - Still large branching factor (potentially larger)
  - When do we instantiate the variables?

Regression Planning

- Still large branching factor (potentially larger)
- When do we instantiate the variables?
- Introduces new concept:

  "Least Commitment Planning"

  Make choices only when they are relevant to solving
  The current part of the problem
Partial Order Planning

- Based on the concept of “Least Commitment”
- Nodes are partial plans
- Arcs/Transitions are **plan refinements**
- Solution is a node, not a path (search in plan space!!)

A plan consists of:
- A: Set of actions
- O: Set of orderings for actions (a < b)
- Q: Set of causal links

A causal link \( q_t \in Q \) is defined as follows:
- Action b has a precondition that is established by Action w

\[
\begin{align*}
w &= Work \\
q_1 &= Have($) \\
b &= Buy(Mercedes)
\end{align*}
\]
Partial Order Planning: Threats

- A threat to \((a,q,b)\) is defined as an action \(t\) that:
  - Has \(\neg q\) as an effect
  \[ \neg q \in t_{\text{add}} \]
  - Could occur between \(a\) and \(b\)
  \[ O \cup a < t < b \quad \text{is inconsistent} \]

- What action \(t\) would be a threat to causal link \(q_i\)?

\[
\begin{align*}
w &= \text{Work} & q_1 &= \text{Have(\$)} & b &= \text{Buy(Mercedes)} \\
\end{align*}
\]

Partial Order Planning: Initialization

Since we’re only talking about actions, let’s turn states into actions:

- \(A_0\)
  - No preconditions
  - Initial state as effects
  - Must be the first step in the plan (all actions \(\geq A_0\) in \(O\))

- \(A_N\)
  - No effects
  - Goals as preconditions
  - Must be the last step in the plan (all actions \(\leq A_N\) in \(O\))
**POP Algorithm: Simplified UCPOP’ (Weld)**

**UCPOP (Universal, Conditional Partial-Order Planner)**

S. Joo (sungmoon.joo@cc.gatech.edu) 8/28/2014 29

---

**POP Algorithm**

```plaintext
POP ((A, O, Q), agenda, actions)
If agenda is empty then return (A, O, Q)
(q, a_{max}) = Choose (pair) from agenda
a_{max} = Choose (actions) s.t. q \in Add(a_{max})
If no such action a_{max} exists, Fail
Q' = Q \cup \{a_{max}\}
O' = O \cup \{a_{max}\}
agenda' = agenda - \{q, a_{max}\}
If a_{max} is new, then A = A \cup a_{max} and
\forall p \in PC(a_{max}), add (p, a_{max}) to agenda'
For every action a_i that threatens any causal link (a_i, Q, a_j) in Q'
Choose to add a_i \prec a_j or a_j \prec a_i to O
If neither choice is consistent, Fail
POP((A', O', Q'), agenda, actions)
```

The magic "Choose" enables backtracking.

---

**Example**

Agenda = \{(Have($), buy(Merc.))\}

O = \{(buy(Merc.), Have(Merc.), Finish)\}

\(\alpha_N = \text{Finish}\)

PC: Have ($)  
D: Have ($)  
A: Have(x)  
\(\text{buy(Mercedes)}\)  
PC: Have(Mercedes)
POP Algorithm

\[
\text{POP}((A, O, Q), \text{agenda, actions})
\]

1. If \(\text{agenda} = \emptyset\) then return \((A, O, Q)\)
2. \((q, a_{new}) = \text{Choose}\) (pair) from agenda
3. \(a_{add} = \text{Choose}\) (action) s.t. \(q \in \text{Add}(a_{add})\)
4. If no such action \(a_{add}\) exists, \text{Fail}
5. \(\Gamma' = \Gamma \cup \{(q, a_{new})\}\)
6. \(\Gamma' = \Gamma \cup \{(q, a_{new})\}\)
7. agenda' = agenda - \((q, a_{new})\)
8. If \(a_{add}\) is new, then \(A = A \cup a_{add}\) and \(\forall p \in PC(a_{add})\), add \((p, a_{add})\) to agenda'
9. For every action \(a\) that threatens any causal link \((a, q, a)\) in \(\Gamma'\)
10. Choose to add \(a < a\) or \(a < a\) to \(O\)
11. If neither choice is consistent, \text{Fail}
12. \text{POP}((A', O', \Gamma'), \text{agenda, actions})

\[
\text{Agenda} = \{(\text{work, Have}(\$), \text{buy(Mercedes)})\}
\]

\[
\text{O} = \{(\text{work}, \text{buy(Mercedes)})\}
\]

\[
\text{work} : \text{Have}(\$) \quad a_{new} = \text{Finish}
\]

\[
\text{buy(Mercedes)} \quad \text{PC} : \text{Have(Mercedes)}
\]

DONE!

S. Joo (sungmoon.joo@cc.gatech.edu) 8/28/2014 31

S. Joo (sungmoon.joo@cc.gatech.edu) 8/28/2014 32
POP Algorithm Details

POP Algorithm Details

Protect Causal Links

use O:

Demotion:
\[ a_t < a_i \]

Promotion:
\[ a_t > a_j \]

POP terminates

For every action \( a_i \) that threatens any causal link \( (a_i, Q, a_j) \) in \( Q' \):
Choose to add \( a_i < a_j \) or \( a_j < a_i \) to O
if neither choice is consistent, Fail

POP((A, O, Q), agenda, actions)

If agenda=∅ then return(A, O, Q)
\((q, a_{next}) = \text{Choose}(pair) \text{ from agenda}\)
\(a_{add} = \text{Choose}(actions) \text{ s.t. } q \in \text{Add}(a_{add})\)
If no such action \( a_{add} \) exists, Fail
\(Q' = Q \cup (a_{add}, q, a_{next})\)
\(O' = O \cup (q, a_{add} < a_{next})\)
agenda' = agenda - (q, a_{next})
If \( a_{add} \) is new, then \( A = A \cup a_{add} \) and
\( \forall p \in PC(a_{add}), \text{ add } (p, a_{add}) \text{ to agenda}'\)

S. Joo (sungmoon.joo@cc.gatech.edu) 8/28/2014