**Partial Order Planning**

Dr Pradip Biswas, PhD (Cantab)
Assistant Professor
Indian Institute of Science
https://cambum.net/index.htm

Any planning algorithm that can place into a plan without specifying which comes first is called a partial order plan.

**ADVANTAGES**

- **DIVIDE & CONQUER**
  Works on several subgoals independently, solves them with several subproblems and then combines the subplans.

- **FLEXIBILITY**
  The planner can work on important decisions first, rather than being forced to work on steps in chronological order.

- **LEAST COMMITMENT**
  We can delay our choice during search e.g. do not commit to an order of actions until it is required.

**Total-Order vs Partial-Order Plans**

**POP ALGORITHM**

- Formulate planning as a state space search problem
  - **States:** Unfinished plan
  - **Actions:** Steps added for completion
  - **Goal:** Finished plan

- Nodes are partial plans
- Arcs/Transitions are plan refinements
- Solution is a node (not a path).

**COMPONENTS OF POP**

- Actions
- Ordering Constraints
- Causal Links
- Open Preconditions

- A Solution is consistent plan with no open preconditions
CONSISTENT PLAN

- **Cycle Checking**: By ordering Constraint

- **Conflict Resolution**: Placing new action outside protection interval

A plan is **Complete** iff all preconditions are achieved

Partial Plan Representation

- **Plan** = (A, O, L), where
  - A: set of actions in the plan
  - O: temporal orderings between actions (a < b)
  - L: causal links linking actions via a literal

- Causal Link:
  Action Ac (consumer) has precondition Q that is established in the plan by Ap (producer).

<table>
<thead>
<tr>
<th>Clear b</th>
</tr>
</thead>
<tbody>
<tr>
<td>move-a-from-b-to-table</td>
</tr>
</tbody>
</table>

Threats to causal links

Step Aₙ threatens link (Ap, Q, Ac) if:
1. Aₙ has (not Q) as an effect, and
2. Aₙ could come between Ap and Ac, i.e.

O ∪ (Ap < Aₙ < Ac) is consistent

What’s an example of an action that threatens the link example from the last slide?

Initial Plan

For uniformity, represent initial state and goal with two special actions:

- **A₀**:
  - no preconditions,
  - initial state as effects
  - must be the first step in the plan.

- **A∞**:
  - no effects
  - goals as preconditions
  - must be the last step in the plan.

POP algorithm

POP((A, O, L), agenda, actions)
If agenda = () then return (A, O, L)
Pick (Q, aₙ) from agenda

aₙ = choose actions s.t. Q ∈ effects(aₙ)

If no such action aₙ exists, fail.

L' := L ∪ (aₙ, Q, aₙ)
O' := O ∪ (aₙ < aₙ)
agenda' := agenda - (Q, aₙ)

If aₙ is new, then A := A ∪ aₙ and ∀P ∈ preconditions(aₙ), add (P, aₙ) to agenda'

For every action aₙ that threatens any causal link (Ap, Q, Ac) in L'

- **Termination**
- **Goal Selection**
- **Action Selection**
- **Update goals**

- **Update causal links**: Demotion: aₙ < ap
- **Promotion**: ac < aₙ

POP

POP is sound and complete

- **POP Plan** is a solution if:
  - All preconditions are supported (by causal links), i.e., no open conditions.
  - No threats
  - Consistent temporal ordering
  - By construction, the POP algorithm reaches a solution plan
POP example: Sussman Anomaly

A0
(on C A) (on-table A) (on-table B) (clear C) (clear B)

A1: move B from Table to C
(on B C) -(on-table B) -(clear C)
(clear B) (clear C) (on-table B)

Ainf
(on A B) (on B C)

A2: move A from Table to B
(clear A) (clear B) (on-table A)
(on A B) -(on-table A) -(clear B)

A0
(on C A) (on-table A) (on-table B) (clear C) (clear B)

A1: move B from Table to C
(on B C) -(on-table B) -(clear C)
(clear B) (clear C) (on-table B)

Ainf
(on A B) (on B C)

Work on open precondition (on B C) and (clear B)

A0
(on C A) (on-table A) (on-table B) (clear C) (clear B)

A1: move B from Table to C
(clear B) (clear C) (on-table B)
(on A B) (on B C)

A0
(on C A) (on-table A) (on-table B) (clear C) (clear B)

A1: move B from Table to C
(clear B) (clear C) (on-table B)
(on A B) (on B C)

Ainf
(on A B) (on B C)

Work on open precondition (on A B)

A0
(on C A) (on-table A) (on-table B) (clear C) (clear B)

A1: move B from Table to C
(clear B) (clear C) (on-table B)
(on A B) (on B C)

Ainf
(on A B) (on B C)

A0
(on C A) (on-table A) (on-table B) (clear C) (clear B)

A1: move B from Table to C
(clear B) (clear C) (on-table B)
(on A B) (on B C)

Ainf
(on A B) (on B C)

Protect causal links

A0
(on C A) (on-table A) (on-table B) (clear C) (clear B)

A1: move B from Table to C
(clear B) (clear C) (on-table B)
(on A B) (on B C)

Ainf
(on A B) (on B C)

A0
(on C A) (on-table A) (on-table B) (clear C) (clear B)

A1: move B from Table to C
(clear B) (clear C) (on-table B)
(on A B) (on B C)

Ainf
(on A B) (on B C)

A0
(on C A) (on-table A) (on-table B) (clear C) (clear B)

A1: move B from Table to C
(clear B) (clear C) (on-table B)
(on A B) (on B C)

Ainf
(on A B) (on B C)

Work on open precondition (clear A)

A0
(on C A) (on-table A) (on-table B) (clear C) (clear B)

A1: move B from Table to C
(clear B) (clear C) (on-table B)
(on A B) (on B C)

Ainf
(on A B) (on B C)

A0
(on C A) (on-table A) (on-table B) (clear C) (clear B)

A1: move B from Table to C
(clear B) (clear C) (on-table B)
(on A B) (on B C)

Ainf
(on A B) (on B C)

Final plan

A0
(on C A) (on-table A) (on-table B) (clear C) (clear B)

A1: move B from Table to C
(clear B) (clear C) (on-table B)
(on A B) (on B C)

Ainf
(on A B) (on B C)

A0
(on C A) (on-table A) (on-table B) (clear C) (clear B)

A1: move B from Table to C
(clear B) (clear C) (on-table B)
(on A B) (on B C)

Ainf
(on A B) (on B C)
Partial-Order Planning vs State-Space Planning

Complexity: $O(b^n)$ worst-case

- Non-deterministic choices ($n$):
  - ProgWS, RegWS: $n = |\text{actions}|$
  - POP: $n = |\text{preconditions}| + |\text{link protection}|$
  - Generally an action has several preconditions

- Branching factor ($b$)
  - POP has smaller $b$:
    - No backtrack due to goal ordering
    - Least commitment: no premature step ordering
      - Does POP make the least possible amount of commitment?

Properties of POP Algorithm

- POP is
  - Sound
  - Complete
  - Systematic (no repetition)

- Extension for
  - Disjunction
  - Universals
  - Negation
  - Conditionals

- Currently not the most efficient method
  - Very sensitive to subgoal ordering

Heuristics for POP

- The number of distinct open preconditions
- Select precondition that can be satisfied in fewest number of ways
- Relax heuristics
  - Planning Graph