PARTIAL ORDER PLANNING

PRADIPTA BISWAS
M. TECH.
SCHOOL OF INFORMATION TECHNOLOGY

PARTIAL ORDER PLAN

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

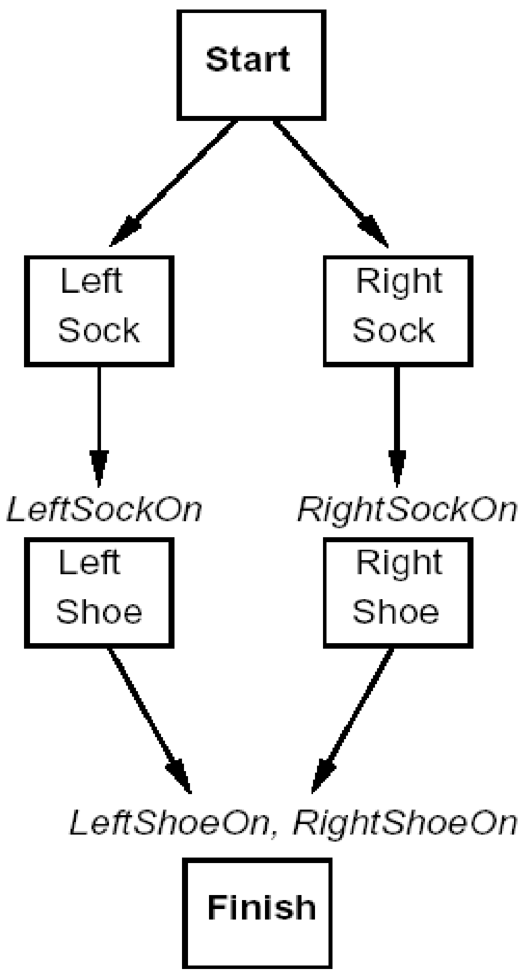
 The planner can work on important decisions first, rather than being forced to work on steps in chronological order

- LEAST COMMITMENT

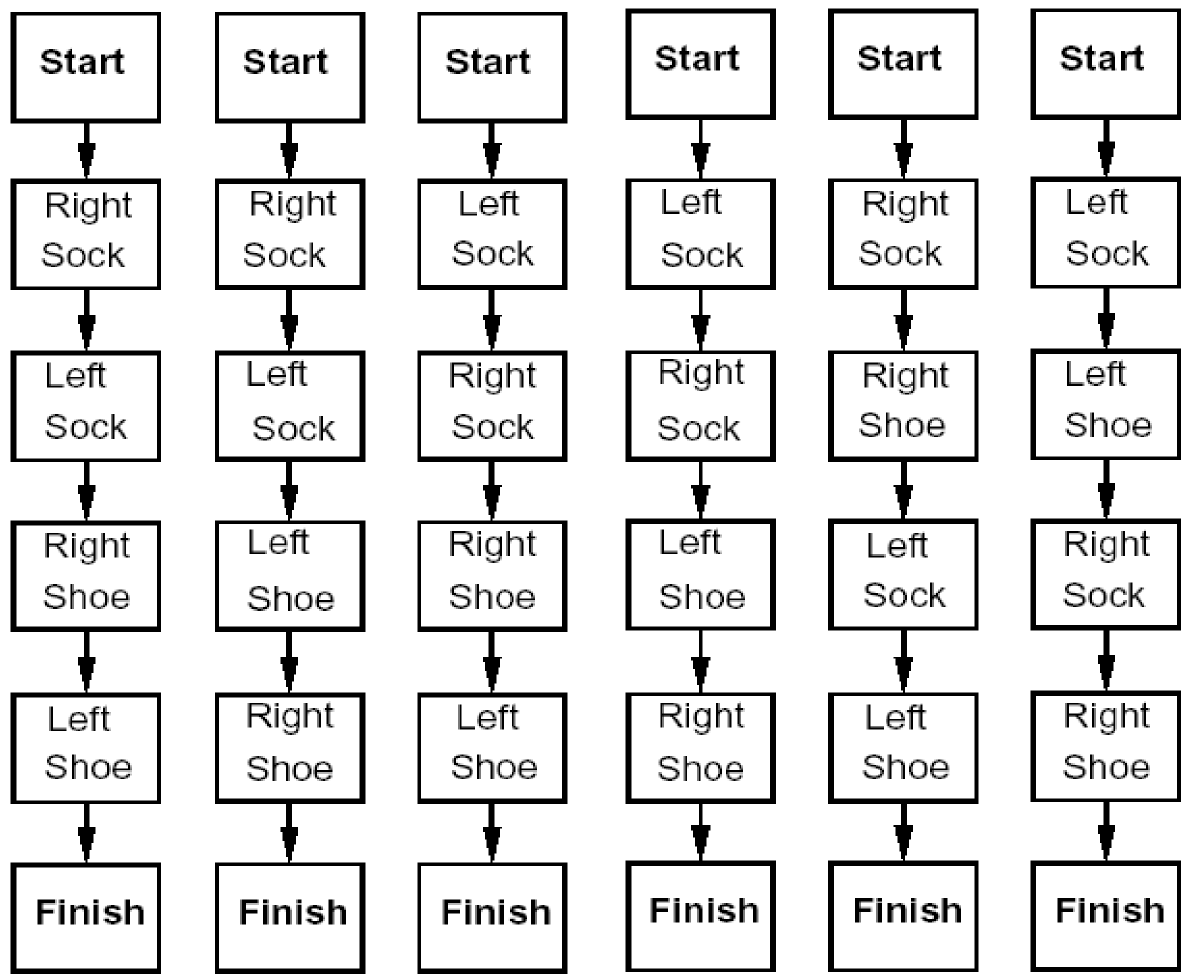
 We can delay our choice during search
 e.g. do not commit to an order of actions until it is required

Total-Order vs Partial-Order Plans

Partial Order Plan:



Total Order Plans:



POP ALGORITHM

- Formulate planning as a state space search problem

- **States:** Unfinished plan
- **Action:** 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: $Ap \xrightarrow{Q} Ac$

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

move-a-from-b-to-table $\xrightarrow{\text{(clear b)}}$ move-c-from-d-to-b

Threats to causal links

Step A_t <u>threatens</u> link (A_p, Q, A_c) if:

1. A_t has (not Q) as an effect, and
2. A_t could come between A_p and A_c, i.e. $O \cup (A_p < A_t < A_c)$ 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:

- A_0:
  - 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, a_{need}) from agenda

a_{add} = **choose**(actions) s.t. Q $\in$effects(a_{add})

If no such action a_{add} exists, **fail**.

L' := L $\cup$ (a_{add}, Q, a_{need}) ; O' := O $\cup$ ($a_{add} < a_{need}$)

agenda' := agenda - (Q, a_{need})

If a_{add} is new, then A := A $\cup$ a_{add} and

$\forall$P $\in$preconditions(a_{add}), add (P, a_{add}) to agenda'

For every action a_t that threatens any causal link (a_p, Q, a_c) in L'

choose to add $a_t < a_p$ or $a_c < a_t$ to O.

If neither choice is consistent, **fail**.

POP((A', O', L'), agenda, actions)

Termination

Goal Selection

Action Selection

Update goals

Protect causal links

- Demotion: $a_t < a_p$
- Promotion: $a_c < a_t$

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)

(on A B) (on B C)

Ainf

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

A0

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

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

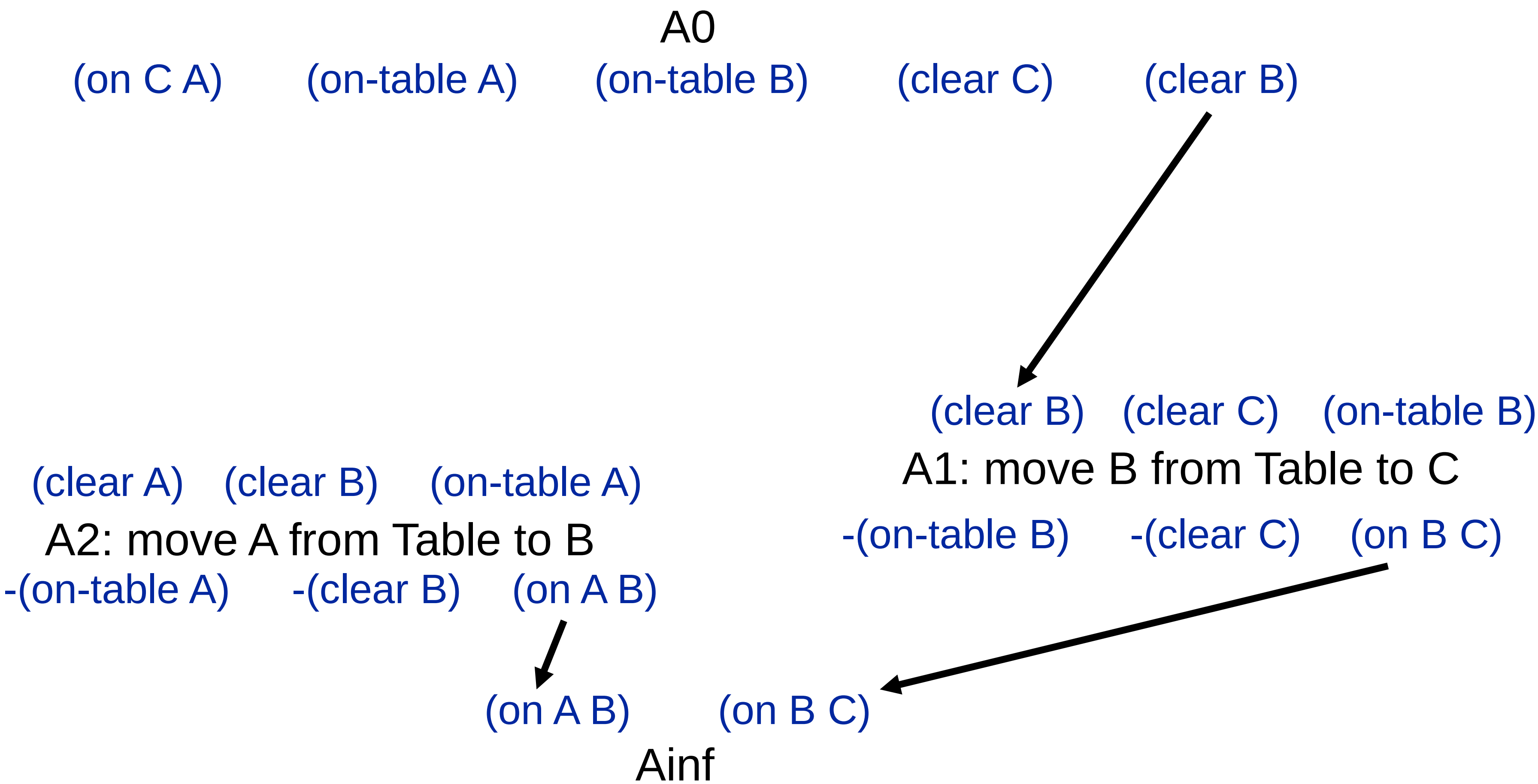
A1: move B from Table to C

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

(on A B) (on B C)

Ainf

Work on open precondition (on A B)



Protect causal links

A0

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

(A0, (clear B), A1) theatened by A2
--> Promotion

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

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

(on A B) (on B C)
Ainf

Work on open precondition(clear A) and protect links

A0

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

(on C A) (clear C)

A3: move C from A to Table

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

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

A1: move B from Table to C

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

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

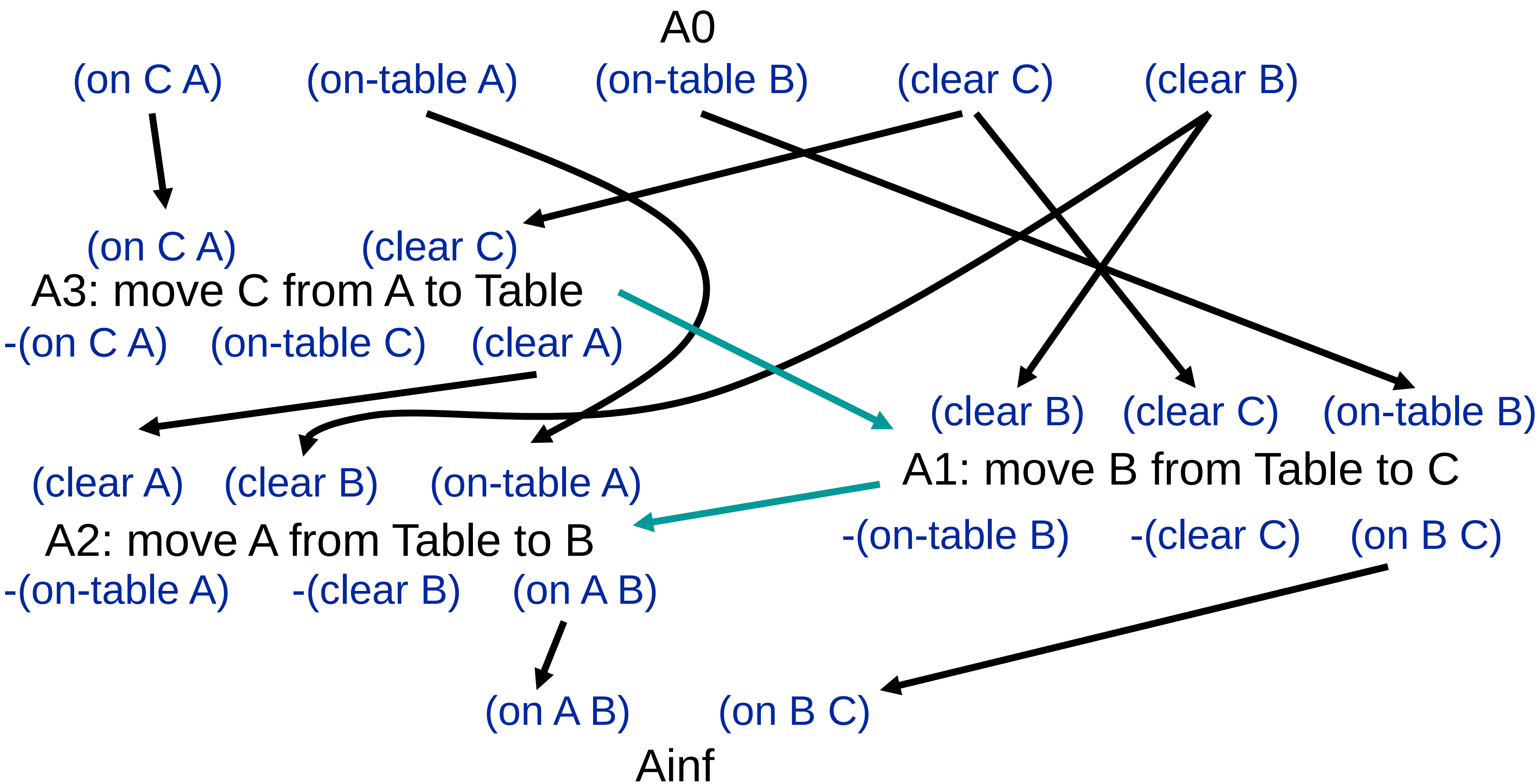
A2: move A from Table to B

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

(on A B) (on B C)

Ainf

Final plan


A0
(on C A)
(on-table A)
(on-table B)
(clear C)
(clear B)
(on C A)
(clear C)
A3: move C from A to Table
-(on C A)
(on-table C)
(clear A)
(clear B)
(clear C)
(on-table B)
A1: move B from Table to C
-(on-table B)
-(clear C)
(on B C)
(clear A)
(clear B)
(on-table A)
A2: move A from Table to B
-(on-table A)
-(clear B)
(on A B)
(on A B)
(on B C)
Ainf

Plartial-Order Planning vs State-Space Planning

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

- Non-deterministic choices (n):
  - ProgWS, RegWS: n = |actions|
  - POP: n = |preconditions| + |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

THANK YOU