Introduction to AI

Lecture 10
Partial Order Planning

Dr. Tamal Ghosh
Department of CSE
Adamas University

Total Order Planning

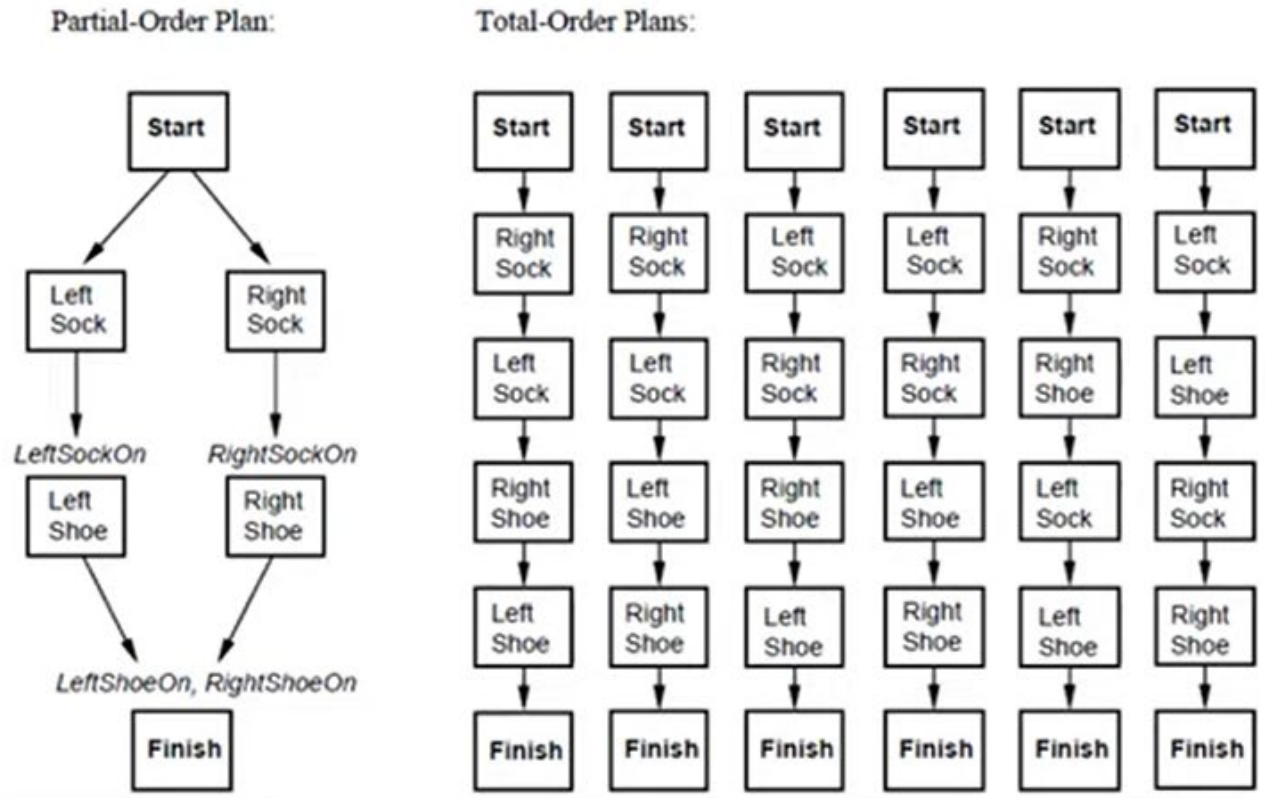
- Forward/backward state-space searches are forms of totally ordered plan search
  - explore only strictly linear sequences of actions, directly connected to the start or goal
  - cannot take advantages of problem decomposition

POP Example: Putting a pair of shoes

- Goal(RightShoeOn ^ LeftShoeOn)
- Init()
- Action: RightShoe
  - PRECOND: RightSockOn
  - EFFECT: RightShoeOn
- Action: RightSock
  - PRECOND: None
  - EFFECT: RightSockOn
- Action:LeftShoe
  - PRECOND: LeftSockOn
  - EFFECT: LeftShoeOn
- Action: LeftSock
  - PRECOND: None
  - EFFECT: LeftSockOn

POP: Shoes and Socks problem

- A partial-order plan for putting on shoes and socks, and the six corresponding linearizations into total-order plans

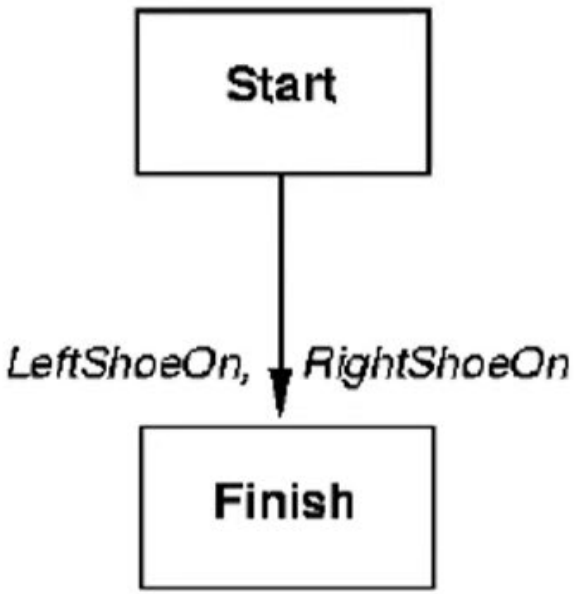


How to define Partial Order Plan

- A set of actions, that make up the steps of the plan
- A set of ordering constrain $A \prec B$
  - A before B
- A set of causal links: $A \xrightarrow{p} B$
  - A achieves P for B $RightSock \xrightarrow{RightSockOn} RightShoe$
  - May be conflicts if C has the effect of ¬P and if C comes after A and before B
- A set of open preconditions:
  - A precondition is open, if it is not achieved by some action in the plan

The Initial Plan

- Initial plan contains:
- Start:
  - PRECOND: none
  - EFFECT: Add all propositions that are initially true
- Finish:
  - PRECOND: Goal state
  - EFFECT: none
- Ordering constraints: $Start \prec Finish$
- Causal links: {}
- Open preconditions:
  - {preconditions of Finish}



Next

- Successor function
  - Arbitrarily picks one open precondition p on an action B and generates a successor plan, for every possible consistent way of choosing an action A, that achieves p
- Consistency:
  - Causal link $A \xrightarrow{p} B$ and the ordering constraint are added ( ) $(A \prec B \quad Start \prec A \quad A \prec Finish)$
  - Resolve conflict: add $B \prec C$ or $C \prec A$
- Goal test:
  - There are no open preconditions

Example Final Plan

- The final plan has the following components:
- Actions: {RightSock, RightShoe, LeftSock, LeftShoe, Start, Finish}
- Orderings: {RightSock ‹ RightShoe, LeftSock ‹ LeftShoe}
- Open preconditions: {}
- Links:

$$RightSock \xrightarrow{RightSockOn} RightShoe$$

$$LeftSock \xrightarrow{LeftSockOn} LeftShoe$$

$$RightShoe \xrightarrow{RightShoeOn} Finish$$

$$LeftShoe \xrightarrow{LeftShoeOn} Finish$$

Example Algorithm for POP

- POP: A sound, complete partial order planner using STRIPS representation

```
function POP(initial, goal, operators) returns plan
  plan ← MAKE-MINIMAL-PLAN(initial, goal)
  loop do
    if SOLUTION?(plan) then return plan
    S_need, c ← SELECT-SUBGOAL(plan)
    CHOOSE-OPERATOR(plan, operators, S_need, c)
    RESOLVE-THREATS(plan)
  end
```

where:
- c is a precondition of a step S_{need}
- RESOLVE-THREATS: orders steps as needed to ensure intermediate steps don't undo preconditions needed by other steps

POP Example / Changing a flat tire

- Consider the problem of changing a flat tire.
- The goal is to have a good spare tire, properly mounted onto the car's axle,
- The initial state has a flat tire on the axle and a good spare tire in the trunk.
- There are just four actions:
  - removing the spare from the trunk,
  - removing the flat tire from the axle,
  - putting the spare on the axle, and
  - leaving the car unattended overnight.
- We assume that the car is in, particularly bad neighborhood, so that the effect of leaving it overnight is that the tires disappear.

POP Example / Flat Tire

$Init(At(Flat, Axle) \wedge At(Spare, Trunk))$

$Goal(At(Spare, Axle))$

$Action(Remove(Spare, Trunk),$

PRECOND: $At(Spare, Trunk)$

EFFECT: $\neg At(Spare, Trunk) \wedge At(Spare, Ground))$

$Action(Remove(Flat, Axle),$

PRECOND: $At(Flat, Axle)$

EFFECT: $\neg At(Flat, Axle) \wedge At(Flat, Ground))$

$Action(PutOn(Spare, Axle),$

PRECOND: $At(Spare, Ground) \wedge \neg At(Flat, Axle)$

EFFECT: $\neg At(Spare, Ground) \wedge At(Spare, Axle))$

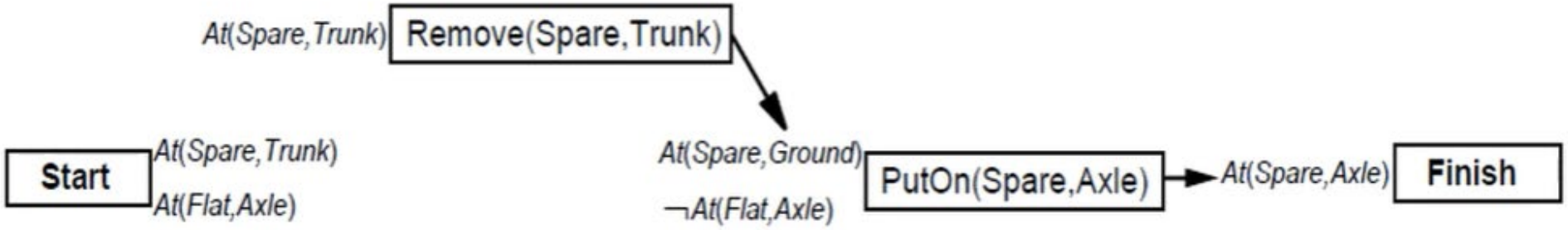
$Action(LeaveOvernight,$

PRECOND:

EFFECT: $\neg At(Spare, Ground) \wedge \neg At(Spare, Axle) \wedge \neg At(Spare, Trunk)$
$\wedge \neg At(Flat, Ground) \wedge \neg At(Flat, Axle))$

POP Algorithm / Sequence of Events

- Start : At (Spare,Trunk) ^ At (Flat, Axle) (init)
- Finish : with precondition At (Spare, Axle). (that is goal)
- Sequence of functions :
- 1. Pick the only open precondition, At (Spare, Axle) of Finish. Choose the only applicable action, PutOn(Spare, Axle).
- 2. Pick the At (Spare, Ground) precondition of PutOn(Spare, Axle). The only applicable action, to achieve it is Remove(Spare,Trunk)



3. Pick the: At(Flat, Axle) precondition of PutOn(Spare, Axle)

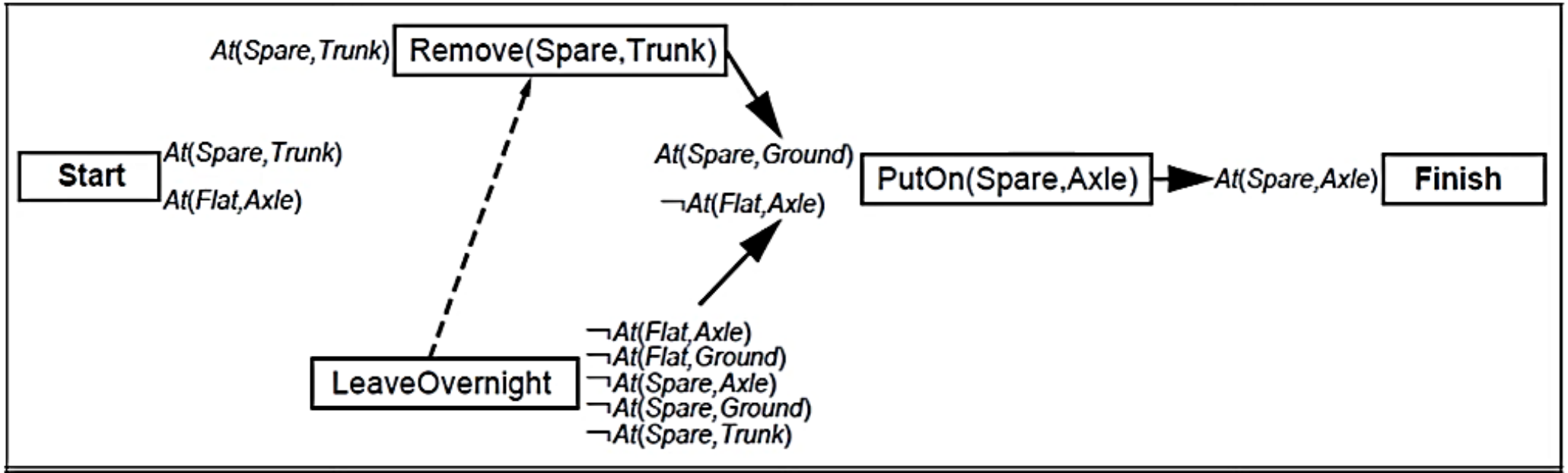


Figure 11.9 The plan after choosing *LeaveOvernight* as the action for achieving $\neg At(Flat, Axle)$. To avoid a conflict with the causal link from $Remove(Spare, Trunk)$ that protects $At(Spare, Ground)$, *LeaveOvernight* is constrained to occur before $Remove(Spare, Trunk)$, as shown by the dashed arrow.

- 4. The only remaining open precondition at this point is the At (Spare,Trunk), precondition of the action Remove(Spare,Trunk)
- 5. Consider again the : At (Flat, Axle) precondition of PutOn(Spare, Axle). This time, we choose Remove(Flat, Axle).
- 6. Once again, pick the At (Spare, Tire) precondition of Remove(Spare,Trunk) and choose Start to achieve it. This time there are no conflicts.
- 7. Pick the At (Flat, Axle) precondition of Remove(Flat, Axle), and choose Start to achieve it.

Final Solution

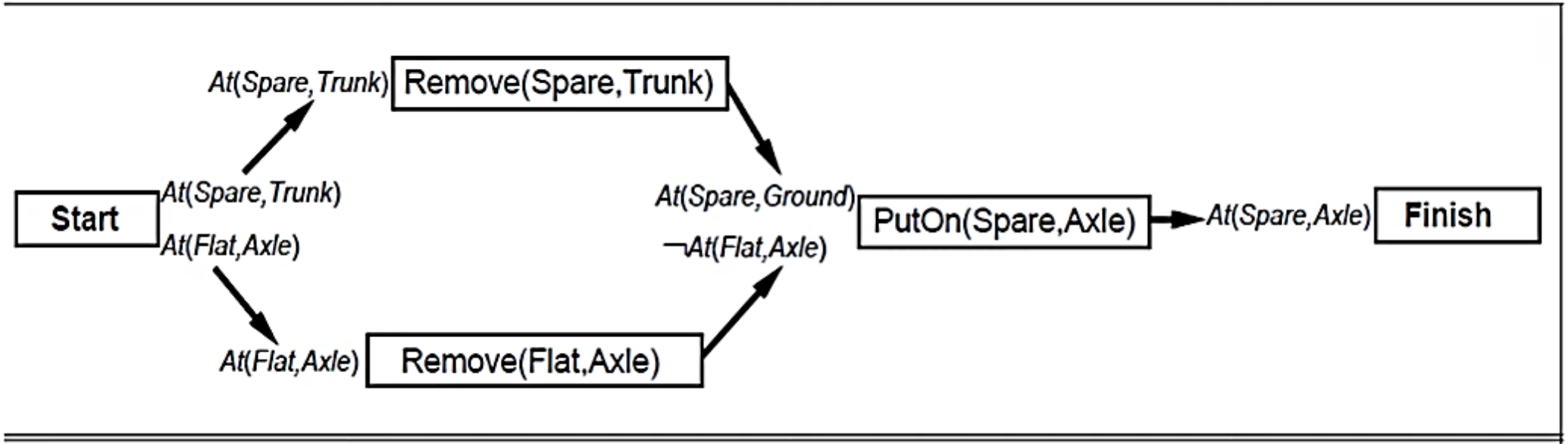


Figure 11.10 The final solution to the tire problem. Note that $Remove(Spare, Trunk)$ and $Remove(Flat, Axle)$ can be done in either order, as long as they are completed before the $PutOn(Spare, Axle)$ action.