Planning

COMP3431 Robot Software Architectures

Planning

A planner finds sequences of actions that will cause transitions from an initial state through intermediates states to a goal state

Actions

- Transitions from one state to the next are achieved by *actions*.
- Must specify how actions work
- Must work out correct sequence of actions to reach goal

Action Models

- Action action(<parameters>)
  - PRECOND: <conditions that must be true to apply this actions>
  - EFFECTS: <conditions that become true or false after executing the action>

Action Example

Action Fly(p, from, to)

PRECOND: Plane(p) $\wedge$ At(p, from) $\wedge$ Airport(from) $\wedge$ Airport(to)

EFFECT: $\neg$At(p, from) $\wedge$ At(p, to))

- positive and negative literals in effects can be separated into an *add list* and and *delete list*

Example

Init: $\text{Airport(MEL)} \wedge \text{Airport(SYD)} \wedge \text{Plane(P1)} \wedge \text{Plane(P2)} \wedge \text{Cargo(C1)} \wedge \text{Cargo(C2)} \wedge \text{At(C1, SYD)} \wedge \text{At(C2, MEL)} \wedge \text{At(P1, SYD)} \wedge \text{At(P2, MEL)}$

Goal: $\text{At(C1, MEL)} \wedge \text{At(C2, SYD)}$

Action Load(c, p, a)

PRECOND: $\text{At(c, a)} \wedge \text{At(p, a)} \wedge \text{Cargo(c)} \wedge \text{Plane(p)} \wedge \text{Airport(a)}$

EFFECT: $\neg\, \text{At(c, a)} \wedge \text{In(c, p)}$

Action Unload(c, p, a)

PRECOND: $\text{In(c, p)} \wedge \text{At(p, a)} \wedge \text{Cargo(c)} \wedge \text{Plane(p)} \wedge \text{Airport(a)}$

EFFECT: $\text{At(c, a)} \wedge \neg\, \text{In(c, p)}$

Action Fly(p, from, to)

PRECOND: $\text{At(p, from)} \wedge \text{Plane(p)} \wedge \text{Airport(from)} \wedge \text{Airport(to)}$

EFFECT: $\neg\, \text{At(p, from)} \wedge \text{At(p, to)}$

Load(C_1, P_1, SYD)

Fly(P_1, SYD, MEL)

Unload(C_1, P_1, MEL)

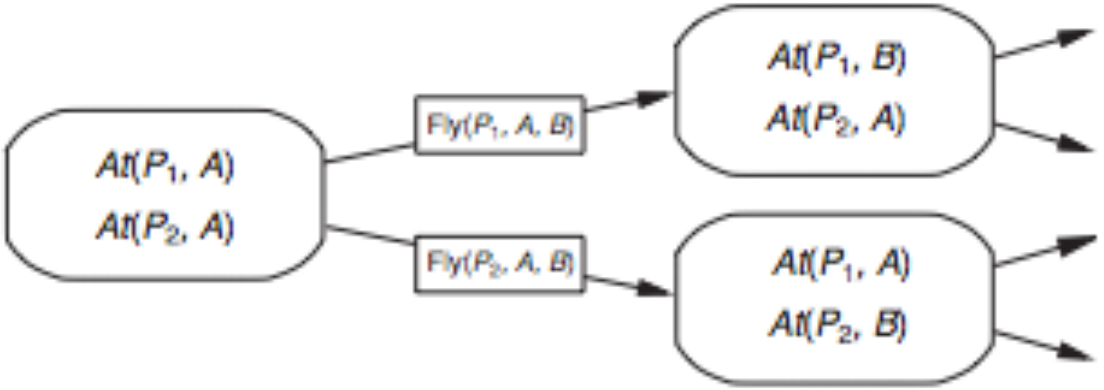
Load(C_2, P_2, MEL)

Fly(P_2, MEL, SYD)

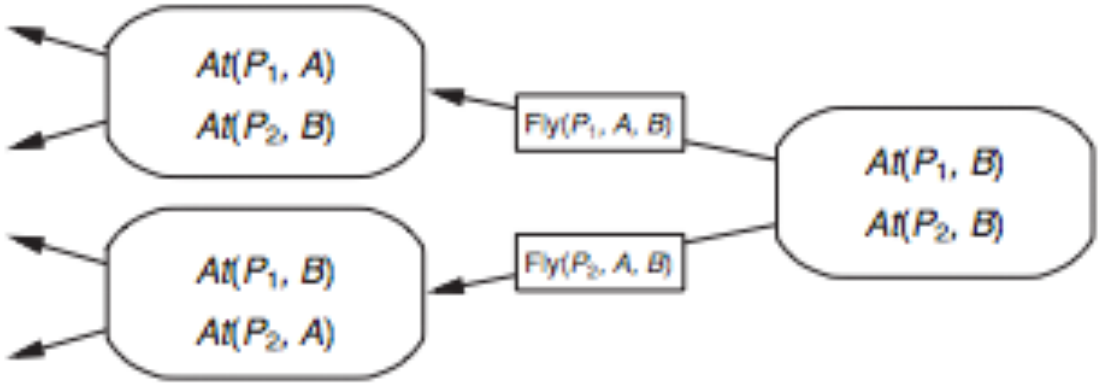
Unload(C_2, P_2, SYD)

Progression and Regression

- Forward Search



- Backward Search



Backward Regression

$$g' = (g - Add(a)) \cup Precond(a)$$

- g' is the regression from goal g over action a
- I.e. going backwards from g, we look for an action, a, that has preconditions and effects that satisfy g'

Planning and TR Programs

Action :-

goal → do_nothing

precond → action

….

start → action

- TR Programs list actions from a plan, keeping preconditions
- Each rule below should be the regression of the rule above

Sussman's Anomaly

- Goal: On(A, B) ∧ On(B, C)
- Try achieving On(A, B) first

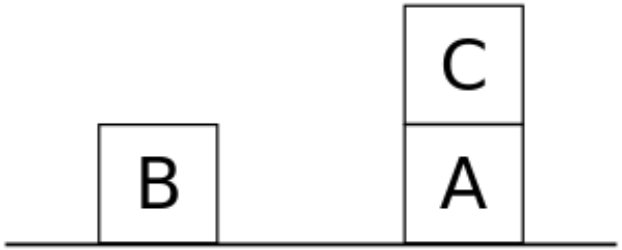
  [move(c,a,floor), move(a,floor,b),
  move(a,b,floor), move(b,floor,c)]

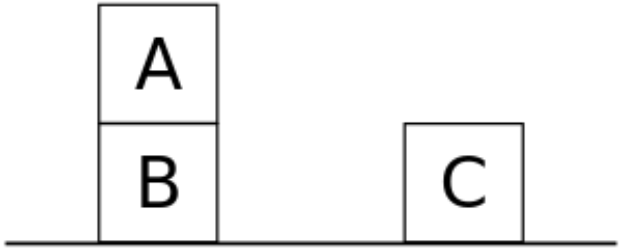
- Trying On(B, C) first

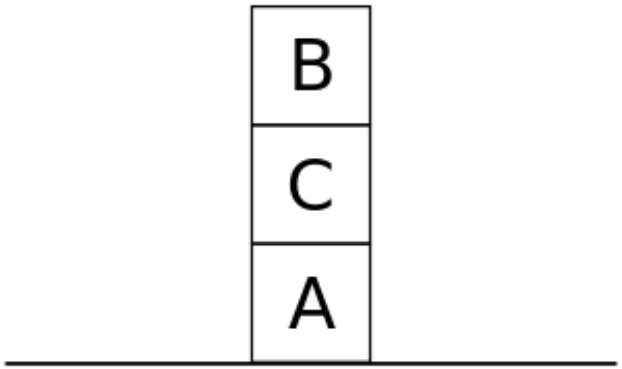
  [move(b,floor,c), **move(b,c,floor)**,
  move(c,a,floor), move(a,floor,b)]

- Should be:

  [move(c,a,floor), move(b,floor,c), move(a,floor,b)]







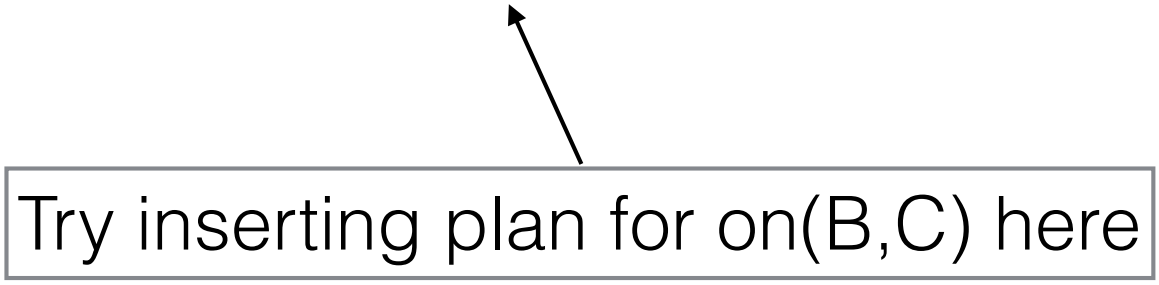
WARPLAN

Warren, D. H. D. (1974). *Warplan: A system for generating plans.*
Memo No. 76, Department of Computational Logic, University of Edinburgh.

- WARPLAN tries to interleave actions by protecting goals.
  - Achieve on(A,B): [move(c,a,floor), move(a,floor,b)]
  - Protect on(A,B)
  - Now try on(B,C) by appending actions to end of plan
    - If it tries to undo a protected goal, move backwards through plan and try to slot new plan in.

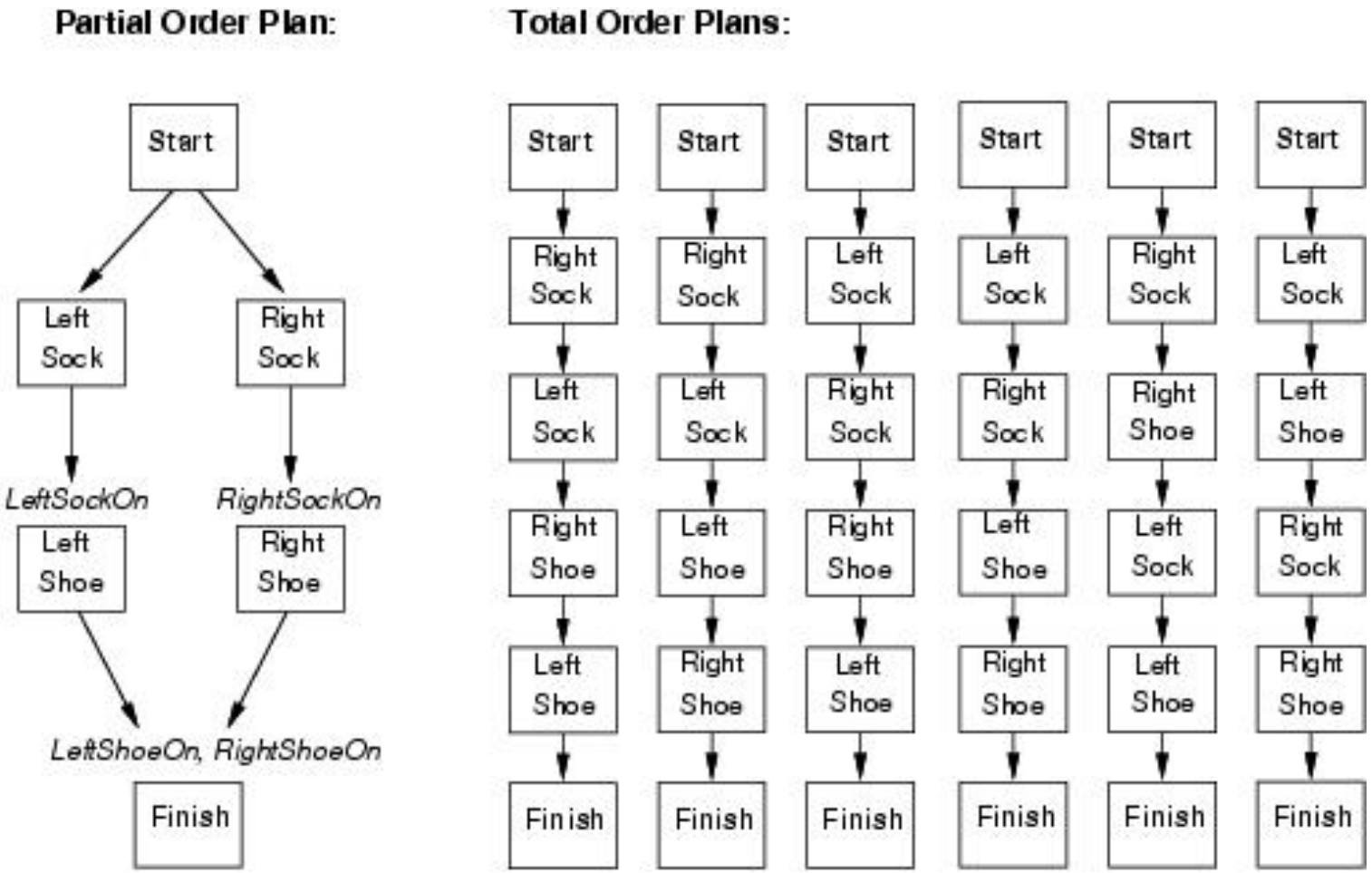
Warplan

- [move(c,a,floor), move(a,floor,b), **move(a,b,floor)**, ..]
- [move(c,a,floor), .., move(a,floor,b)]



- check that goals before and after are preserved

Partially Ordered Plans



Partial-Order Planning

Init: $\text{Tire(Flat)} \wedge \text{Tire(Spare)} \wedge \text{At(Flat, Axle)} \wedge \text{At(Spare, Boot)}$

Goal: At (Spare, Axle)

Action Remove(obj, loc)

PRECOND: At(obj, loc)

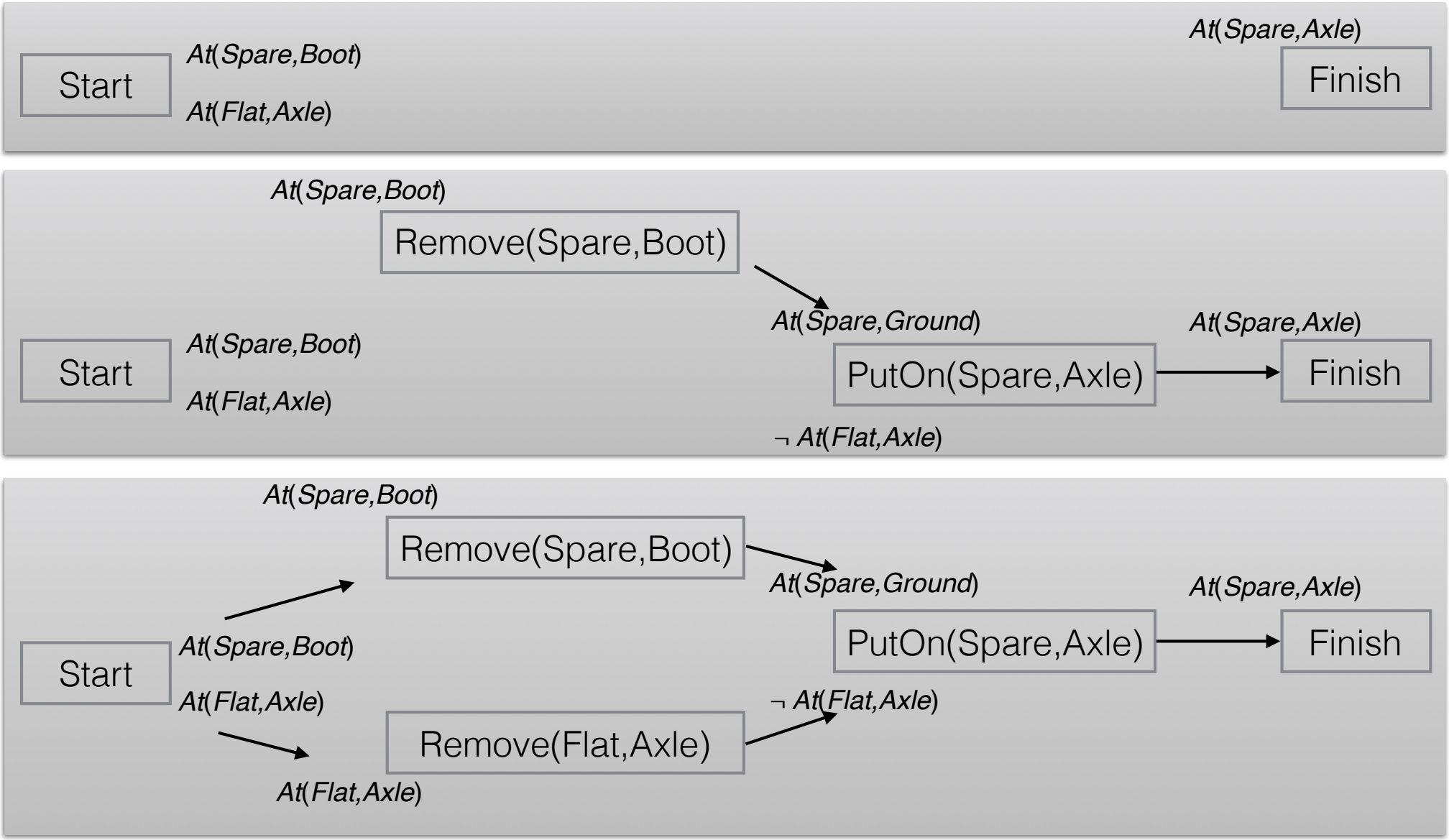
EFFECT: $\neg$ At(obj, loc) $\wedge$ At(obj, Ground)

Action PutOn(t, Axle)

PRECOND: $\text{Tire(t)} \wedge \text{At(t, Ground)} \wedge \neg \text{At(Flat, Axle)}$

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

Partial-Order Planning



Forward Planning

- Forward planners are now among the best.
- Use heuristics to estimate costs
- Possible to use heuristic search, like A*, to reduce branching factor.