CS 1571 Introduction to AI
Lecture 19

Planning: STRIPS

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Competition results

Simulated annealing competition:
1. Nathaniel Wetzel
2. David Bradley
3. Francesco DeSensi

Extra credit ☺

Planning

Planning problem:
- find a sequence of actions that achieves some goal
- an instance of a search problem
- the state description is very complex

Methods for modeling and solving a planning problem:
- State space search
- Situation calculus based on FOL
  - Typically Resolution refutation

Planning problems

Properties of (real-world) planning problems:

- The description of the **state of the world is very complex**
- **Many possible actions** to apply in any step
- **Actions are typically local**
  - \- they affect only a small portion of a state description
- **Goals** are defined as conditions and **refer only to a small portion of state**
- Plans consists of a **long sequence of actions**

- The state space search and situation calculus frameworks may be too cumbersome and inefficient to represent and solve planning problems

Situation calculus: problems

Frame problem refers to:
- The need to represent a large number of frame axioms

Solution: combine positive and negative effects in one rule

$$On(u,v,DO(MOVE(x,y,z),s)) \Leftrightarrow (\neg((u=x)\wedge(v=y))\wedge On(u,v,s))\vee$$
$$\vee(((u=x)\wedge(v=z))\wedge On(x,y,s)\wedge Clear(x,s)\wedge Clear(z,s))$$

Inferential frame problem:
- – We still need to derive properties that remain unchanged

Other problems:
- **Qualification problem** – enumeration of all possibilities under which an action holds
- **Ramification problem** – enumeration of all inferences that follow from some facts

Solutions

- **Complex state description and local action effects:**
  - avoid the enumeration and inference of every state component, focus on changes only
- **Many possible actions:**
  - Apply only actions that make progress towards the goal
  - Understand what the effect of actions is and reason with the consequences
- **Sequences of actions in the plan can be too long:**
  - Many goals consists of independent or nearly independent sub-goals
  - Allow goal decomposition & divide and conquer strategies

STRIPS planner

Defines a **restricted FOL representation language** as compared to the situation calculus

Advantage: leads to more efficient planning algorithms.

- State-space search with structured representations of states, actions and goals
- Action representation avoids the frame problem

STRIPS planning problem:

- much like a standard search problem

STRIPS planner

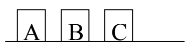
- **States:**
  - conjunction of literals, e.g. *On(A,B), On(B,Table), Clear(A)*
  - represent facts that are true at a specific point in time
- **Actions (operators):**
  - **Action:** *Move (x,y,z)*
  - **Preconditions:** conjunctions of literals with variables

    On(x,y), Clear(x), Clear(z)
  - **Effects.** Two lists:
    - **Add list:** *On(x,z), Clear(y)*
    - **Delete list:** *On(x,y), Clear(z)*
    - Everything else remains untouched (is preserved)

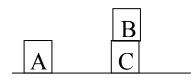
STRIPS planning

Operator: *Move (x,y,z)*

- **Preconditions:** *On(x,y), Clear(x), Clear(z)*
- **Add list:** *On(x,z), Clear(y)*
- **Delete list:** *On(x,y), Clear(z)*



$Move(B, Table, C)$



$On(B, Table)$ $Clear(C)$	delete / add	$On(B, C)$
$On(A, Table)$ $On(C, Table)$ $Clear(A)$ $Clear(B)$ $Clear(Table)$	unchanged	$On(A, Table)$ $On(C, Table)$ $Clear(A)$ $Clear(B)$ $Clear(Table)$

STRIPS planning

Initial state:

- Conjunction of literals that are true

Goals in STRIPS:

- A goal is a partially specified state
- Is defined by a conjunction of ground literals
  - No variables allowed in the description of the goal

Example:

$On(A,B) \wedge On(B,C)$

Search in STRIPS

Objective:

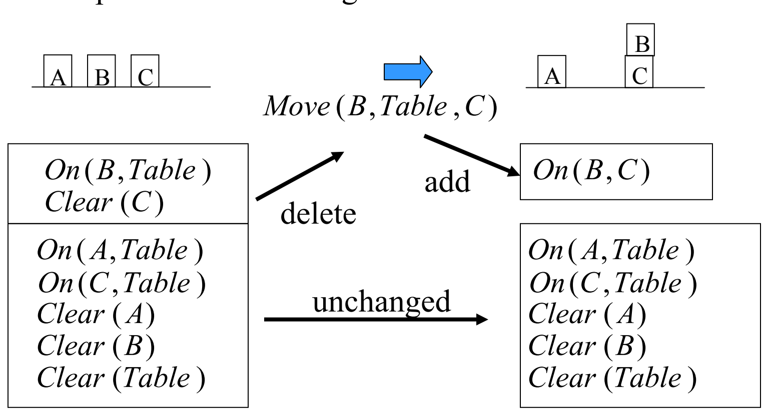
Find a sequence of operators (a plan) from the initial state to the state satisfying the goal

Two approaches to build a plan:

- **Forward state space search (goal progression)**
  - Start from what is known in the initial state and apply operators in the order they are applied
- **Backward state space search (goal regression)**
  - Start from the description of the goal and identify actions that help to reach the goal

Forward search (goal progression)

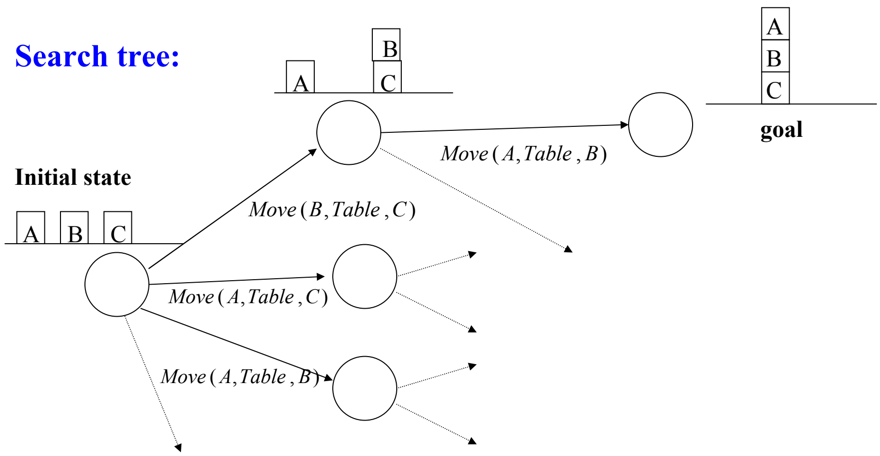
- **Idea:** Given a state *s*
  - Unify the preconditions of some operator *a* with *s*
  - Add and delete sentences from the add and delete list of an operator *a* from *s* to get a new state



Forward search (goal progression)

- Use operators to generate new states to search
- Check new states whether they satisfy the goal

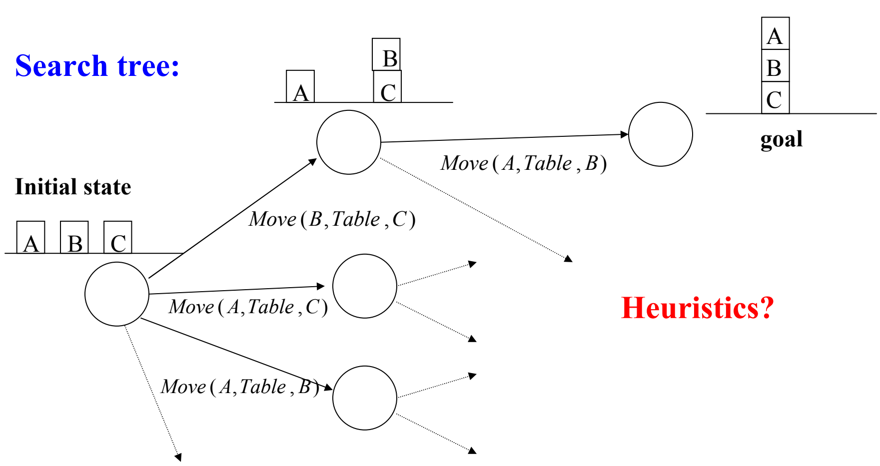
Search tree:



Forward search (goal progression)

- Use operators to generate new states to search
- Check new states whether they satisfy the goal

Search tree:

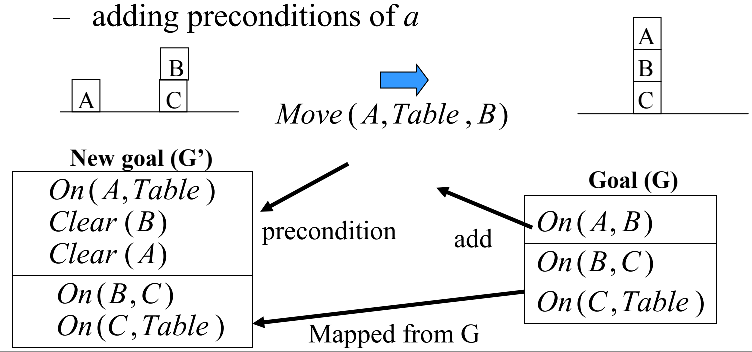


Heuristics?

Backward search (goal regression)

Idea: Given a goal *G*

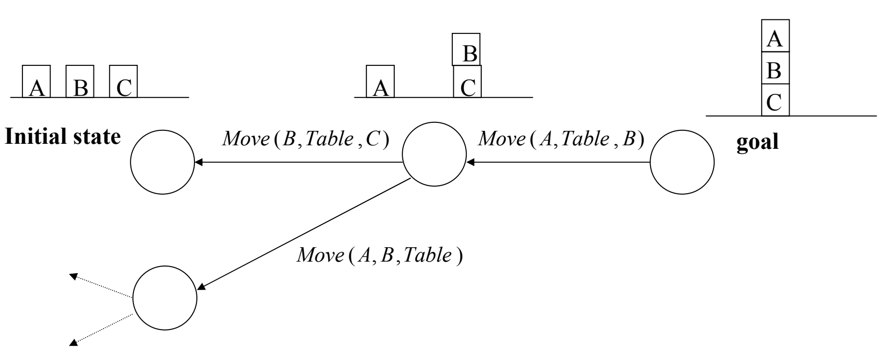
- Unify the add list of some operator *a* with a subset of *G*
- If the delete list of *a* does not remove elements of *G*, then the goal regresses to a new goal *G'* that is obtained from *G* by:
  - deleting add list of *a*
  - adding preconditions of *a*



Backward search (goal regression)

- Use operators to generate new goals
- Check whether the initial state satisfies the goal

Search tree:



State-space search

- **Forward and backward state-space planning approaches:**
  – Work with strictly linear sequences of actions

- **Disadvantages:**
  – They cannot take advantage of the **problem decompositions** in which the goal we want to reach consists of a set of independent or nearly independent sub-goals
  – Action sequences cannot be **built from the middle**
  – No mechanism to represent **least commitment** in terms of the action ordering

Divide and conquer

- **Divide and conquer strategy**:
  – divide the problem to a set of smaller sub-problems,
  – solve each sub-problem independently
  – combine the results to form the solution

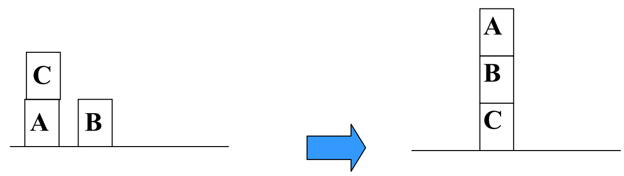
In planning we would like to satisfy a set of goals

- **Divide and conquer in planning:**
  – Divide the planning goals along individual goals
  – Solve (find a plan for) each of them independently
  – Combine the plan solutions in the resulting plan

- Is it always safe to use divide and conquer?
  – No. There can be interacting goals.

Sussman's anomaly.

- An example from the blocks world in which the divide and conquer fails due to interacting goals

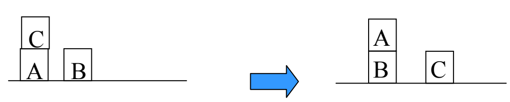


Initial state **Goal**

$On(A,B)$
$On(B,C)$

Sussman's anomaly

1. Assume we want to satisfy $On(A,B)$ first

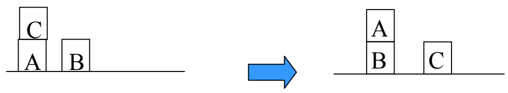


Initial state

But now we cannot satisfy $On(B,C)$ without undoing $On(A,B)$

Sussman's anomaly

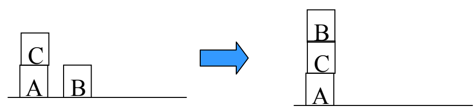
1. Assume we want to satisfy $On(A,B)$ first



Initial state

But now we cannot satisfy $On(B,C)$ without undoing $On(A,B)$

2. Assume we want to satisfy $On(B,C)$ first.



Initial state

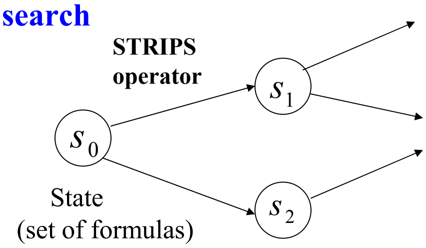
But now we cannot satisfy $On(A,B)$ without undoing $On(B,C)$

State space vs. plan space search

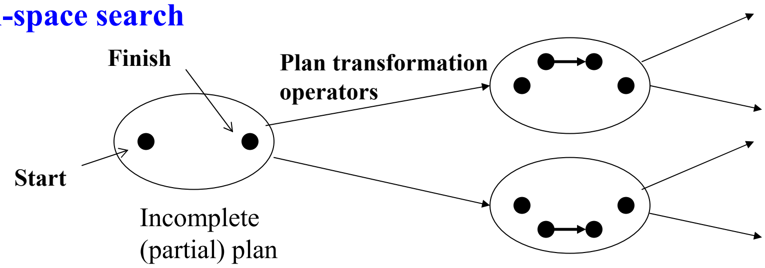
- An alternative to planning algorithms that search states (configurations of world)
- **Plan:** Defines a sequence of operators to be performed
- **Partial plan:**
  - plan that is not complete
    - Some plan steps are missing
  - some orderings of operators are not finalized
    - Only relative order is given
- **Benefits of working with partial plans:**
  - We do not have to build the sequence from the initial state or the goal
  - We do not have to commit to a specific action sequence
  - We can work on sub-goals individually (divide and conquer)

State-space vs. plan-space search

State-space search



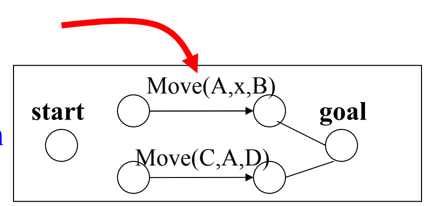
Plan-space search



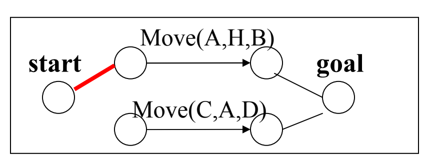
Plan transformation operators

Examples of :

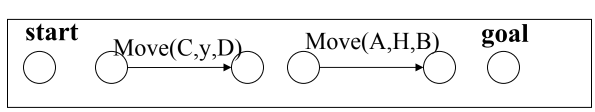
- **Add an operator to a plan so that it satisfies some open condition**



- **Add link (+ instantiate)**



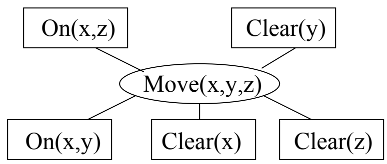
- **Order (reorder) operators**



Partial-order planners (POP)

- also called **Non-linear planners**
- Use STRIPS operators

Graphical representation of an **operator** **Move(x,y,z)**



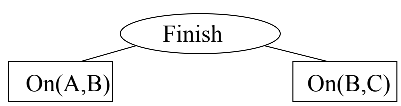
add list

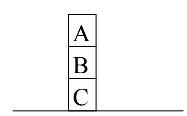
preconditions

Delete list is not shown !!!

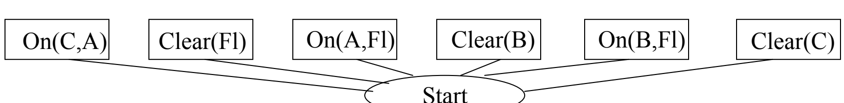
Illustration of a POP on the Sussman's anomaly case

Partial order planning. Start and finish.

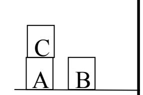




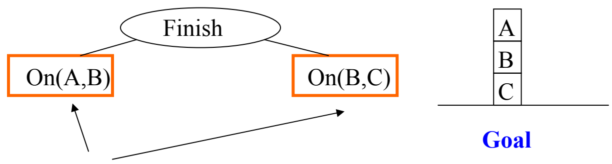
Goal



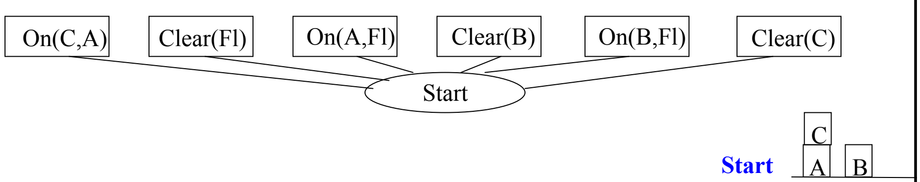
Start



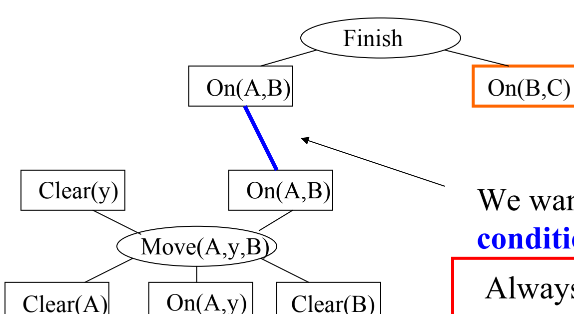
Partial order planning. Start and finish.

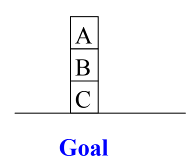


Open conditions: conditions yet to be satisfied



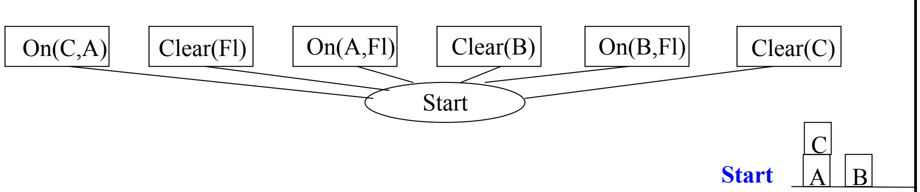
Partial order planning. Add operator.



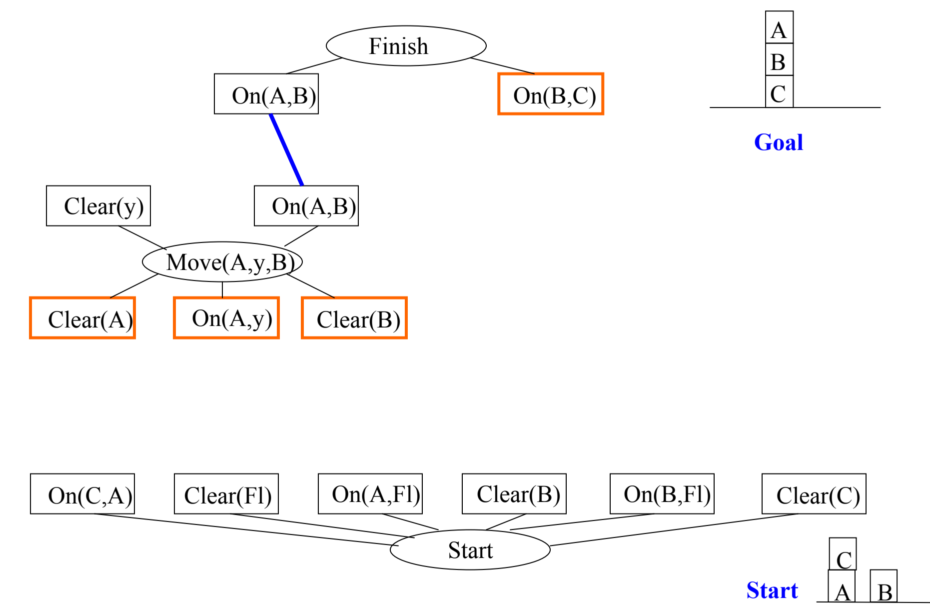


We want to satisfy **an open condition**

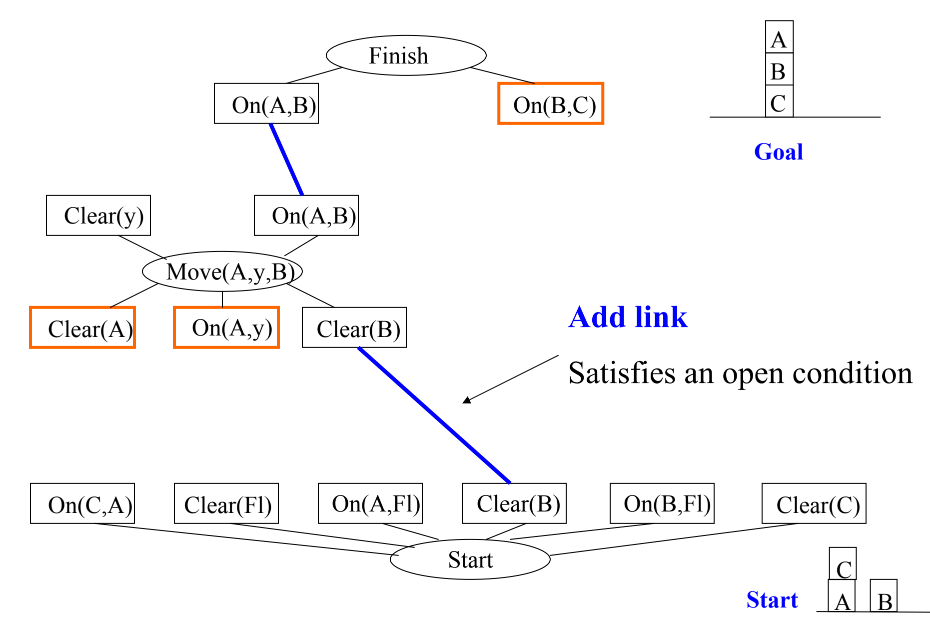
Always select an operator that helps to satisfy one of the open conditions



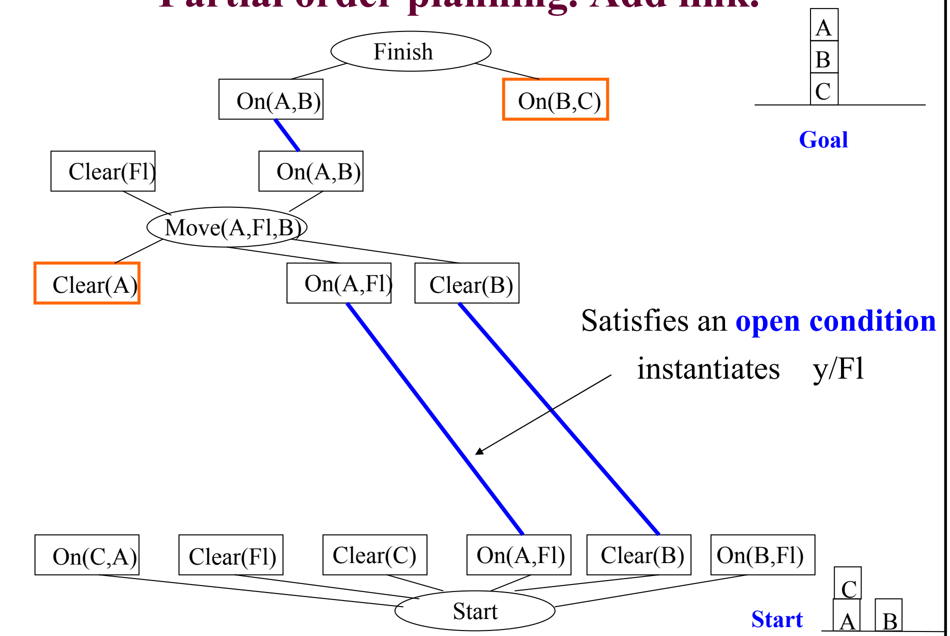
Partial order planning. Add link.



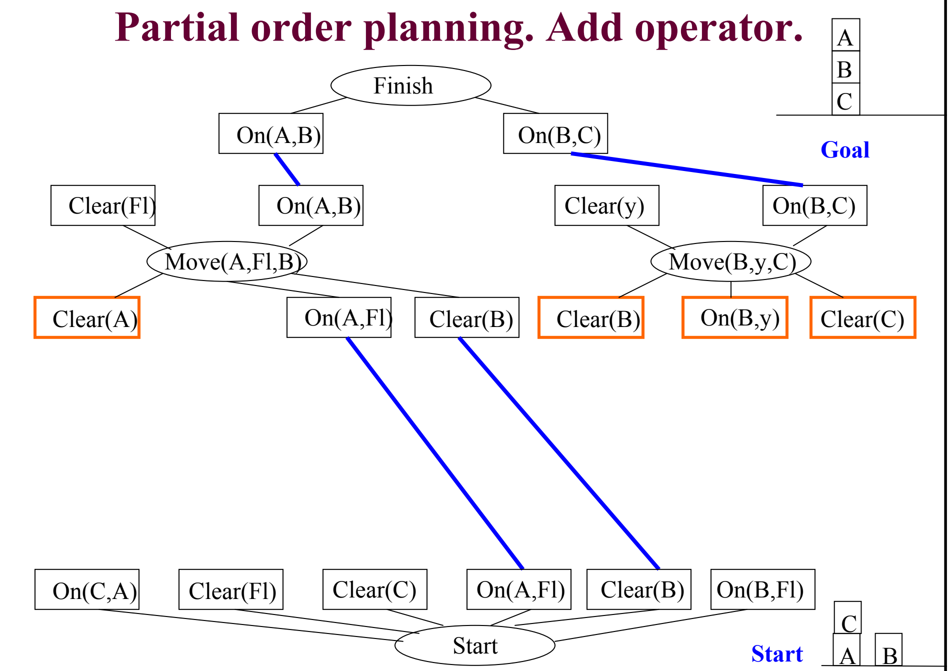
Partial order planning. Add link.



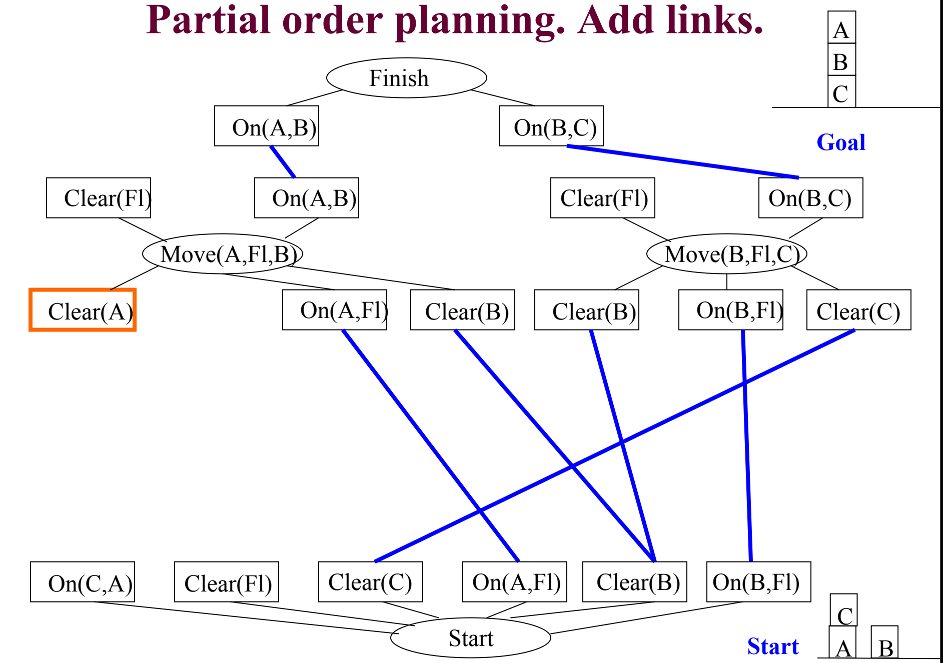
Partial order planning. Add link.



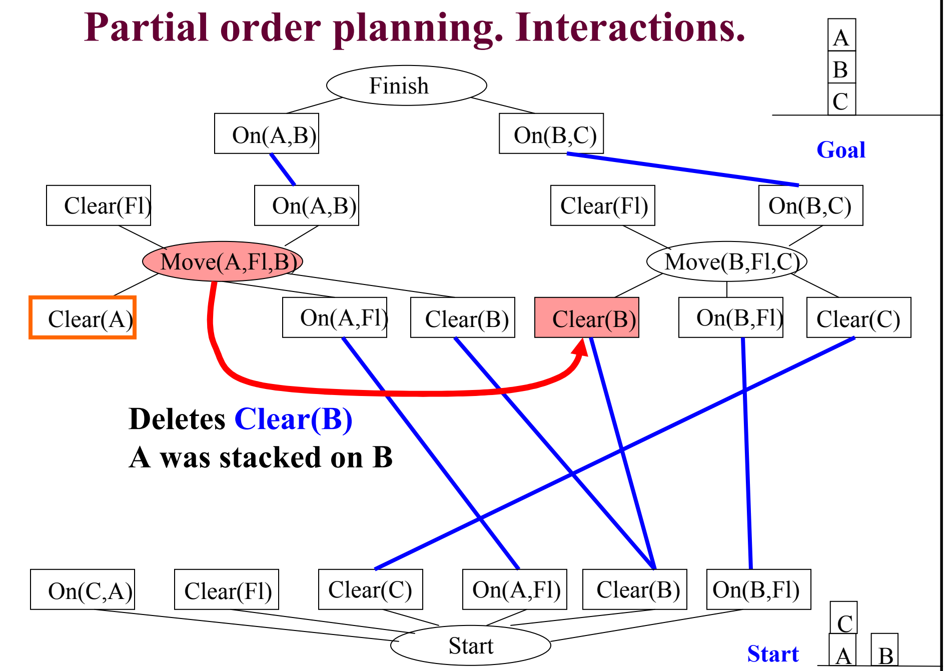
Partial order planning. Add operator.



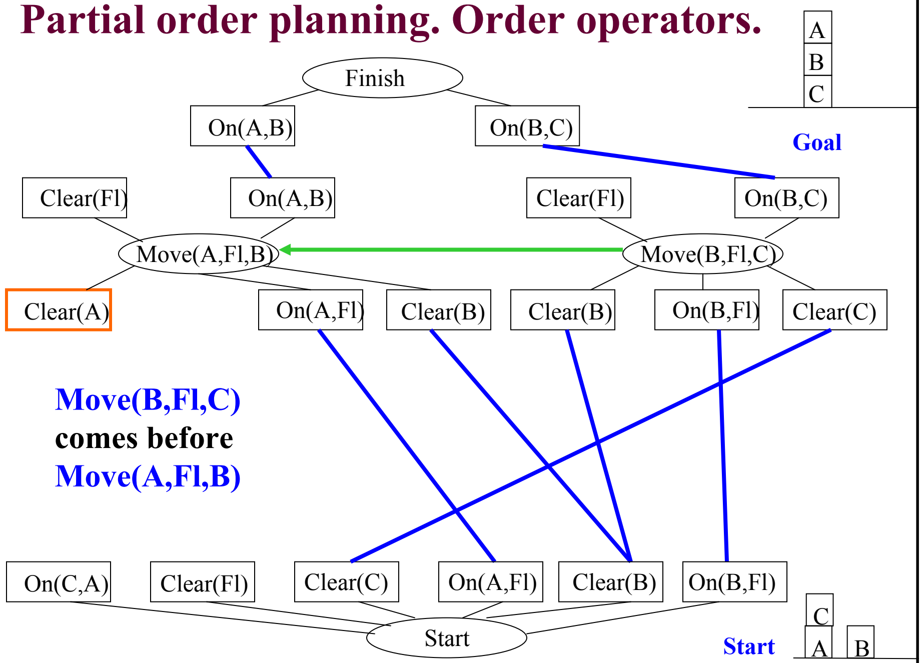
Partial order planning. Add links.



Partial order planning. Interactions.

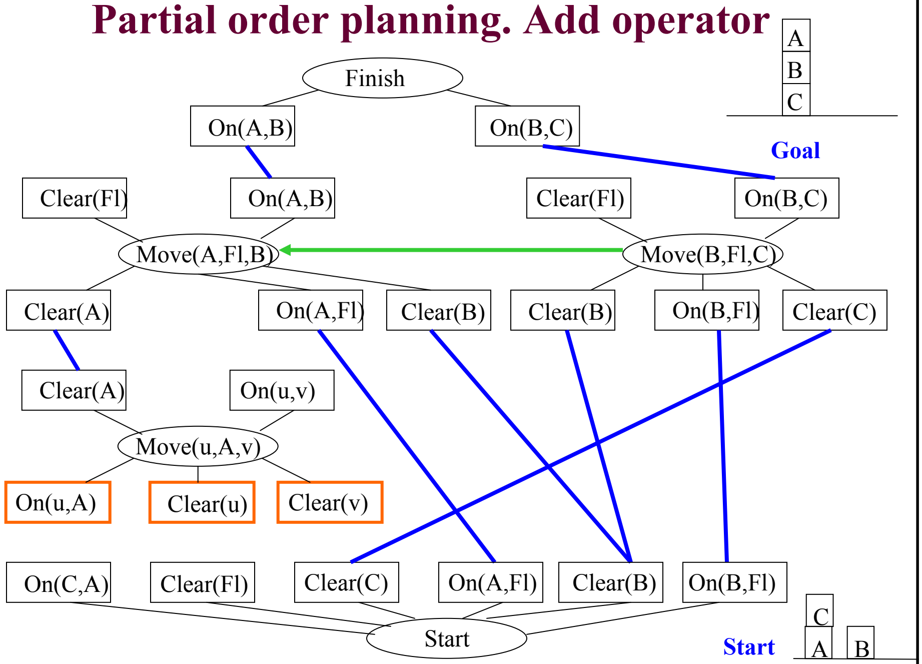


Partial order planning. Order operators.

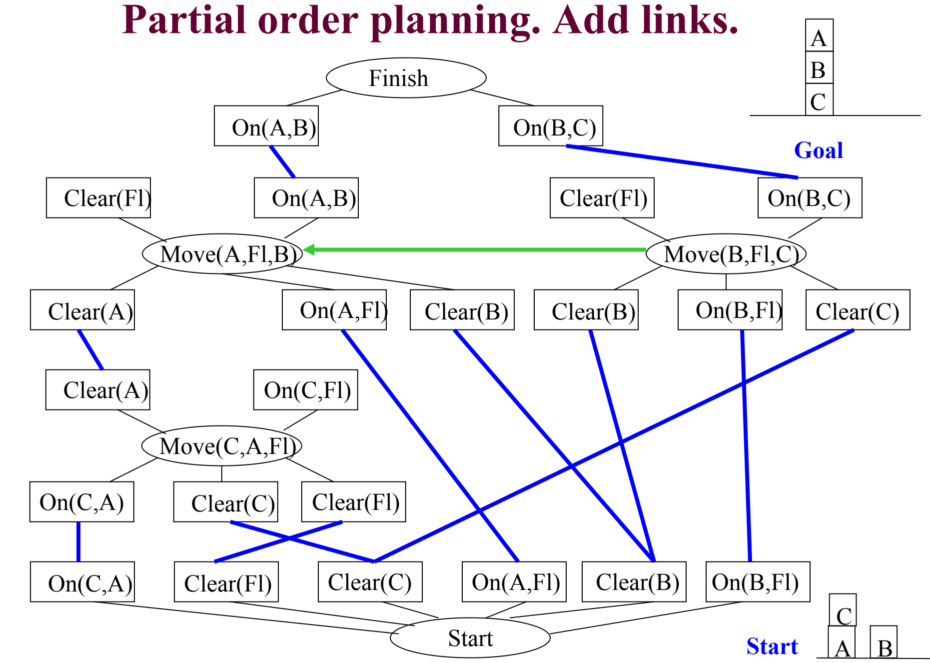


Move(B,Fl,C)
comes before
Move(A,Fl,B)

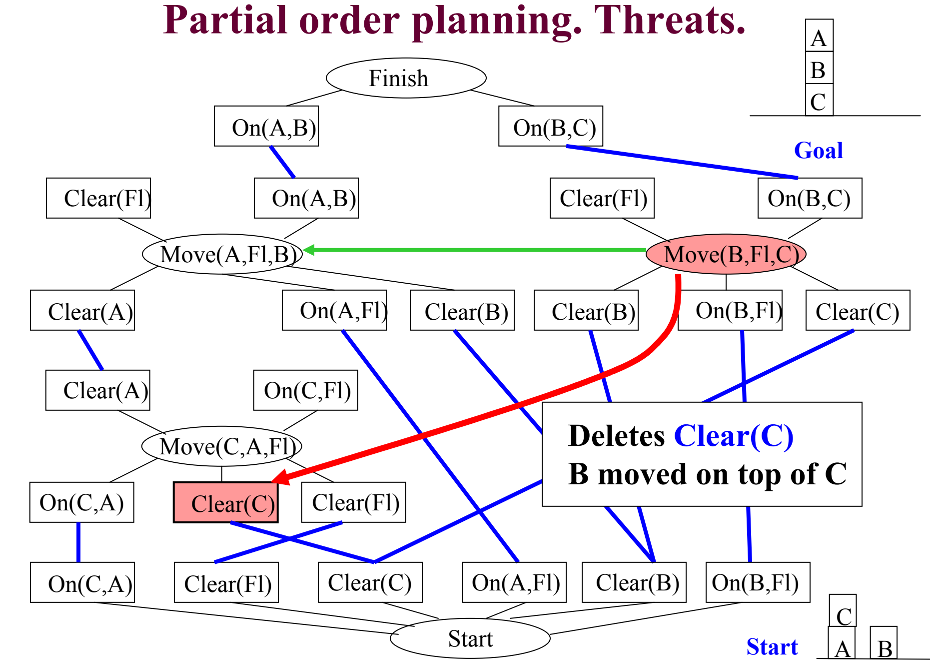
Partial order planning. Add operator



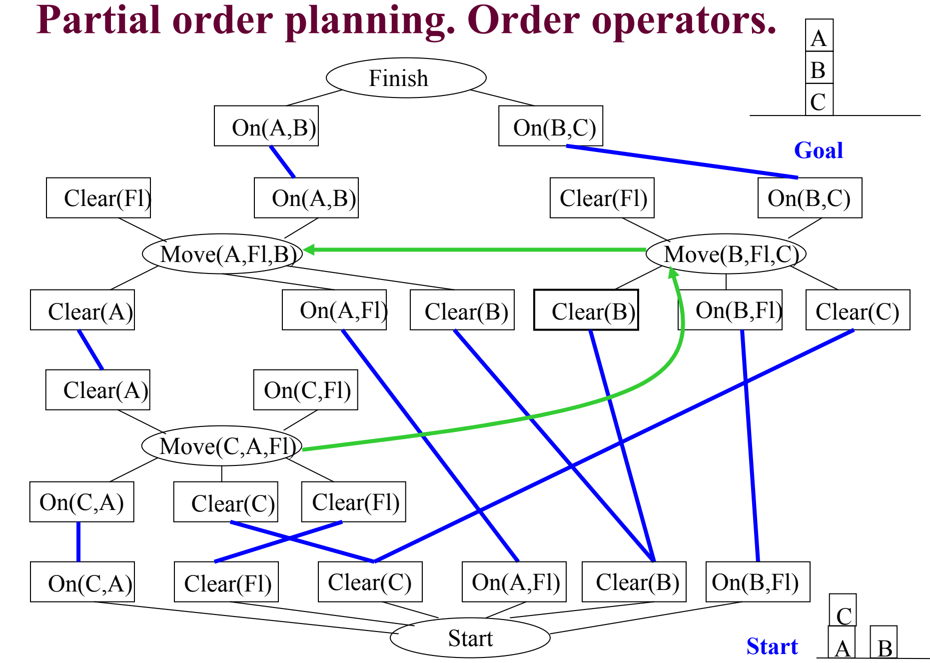
Partial order planning. Add links.



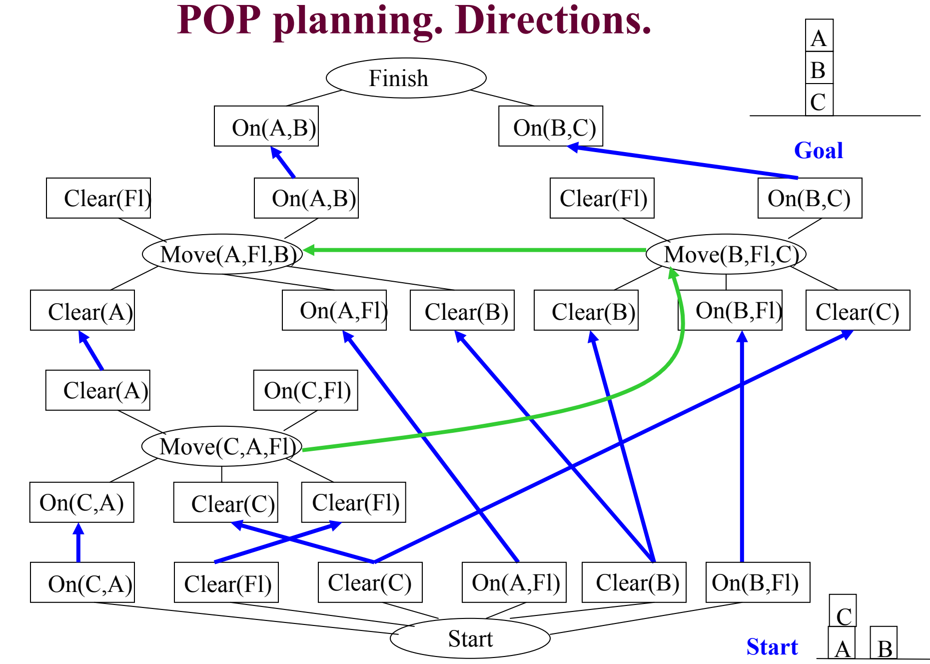
Partial order planning. Threats.



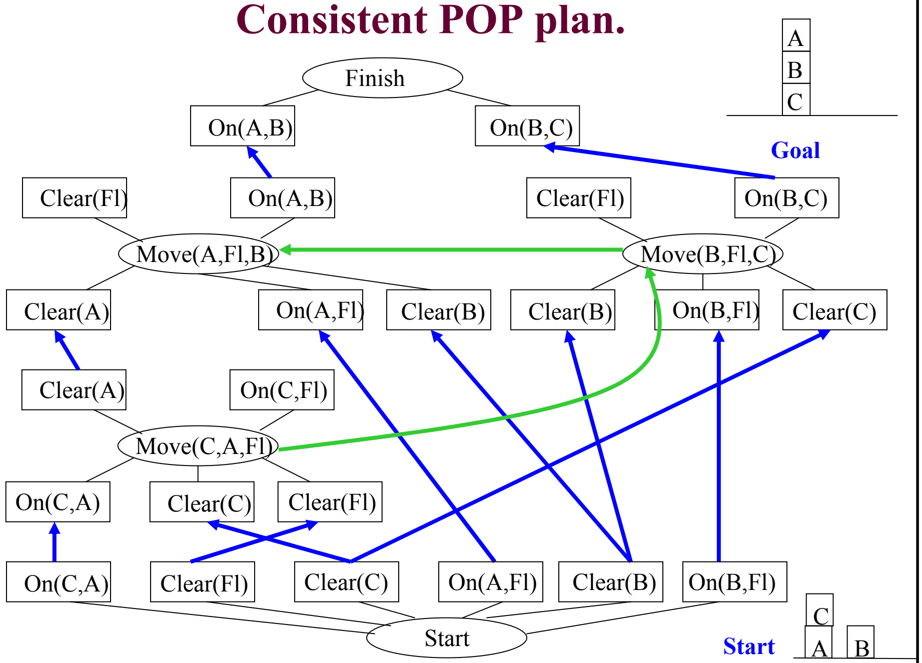
Partial order planning. Order operators.



POP planning. Directions.

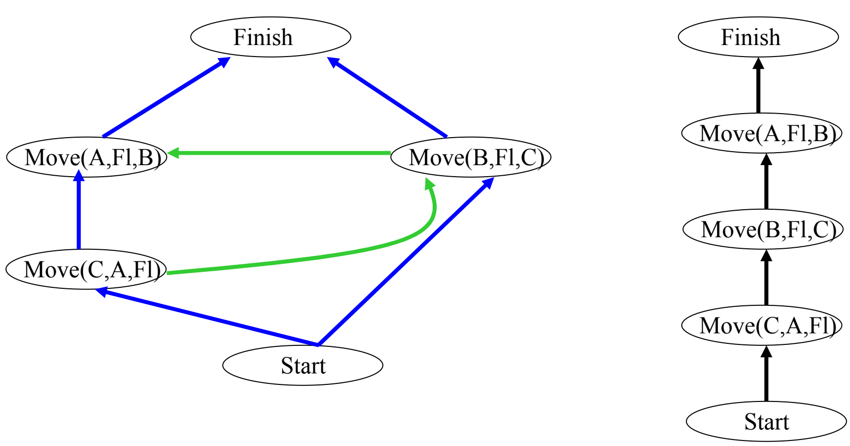


Consistent POP plan.



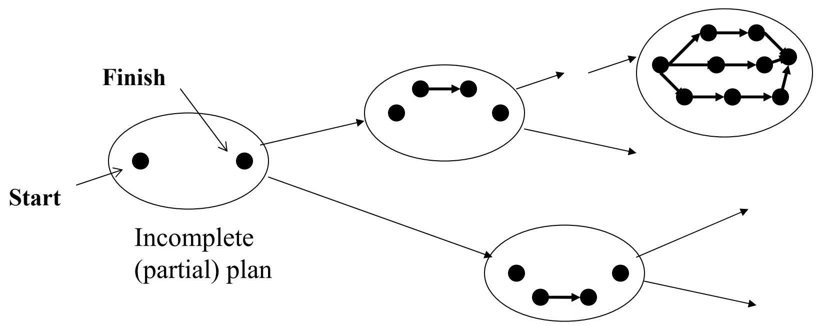
Partial order planning. Result plan.

Plan: a topological sort of a graph



Partial order planning.

- **Remember** we search the space of partial plans



- POP: **is sound and complete**