CS 1571 Introduction to AI
Lecture 17

Planning

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

Planning

Planning problem:
- find a sequence of actions that achieves some goal
- an instance of a search problem
- the state description is typically very complex and relies on a logic-based representation

Methods for modeling and solving a planning problem:
- State space search
- Situation calculus based on FOL
- STRIPS – state-space search algorithm
- **Partial-order planning algorithms**

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

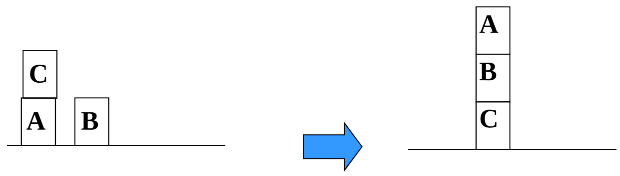
In planning we would like to satisfy a set of goals

- **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

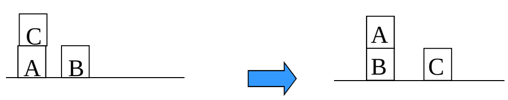


Initial state **Goal**

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

Sussman's anomaly

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

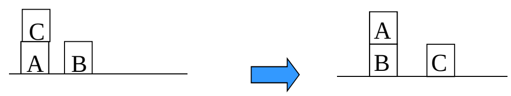


Initial state

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

Sussman's anomaly

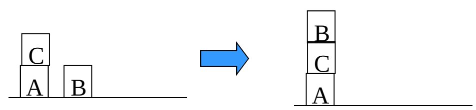
1. Assume we want to satisfy $On(A,B)$ first



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

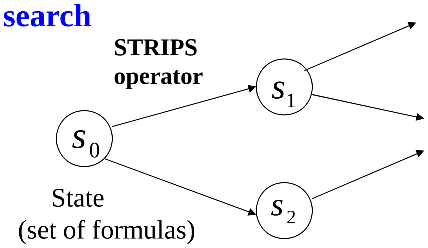
But now we cannot satisfy $On(A,B)$ without undoing $On(B,C)$

State space vs. plan space search

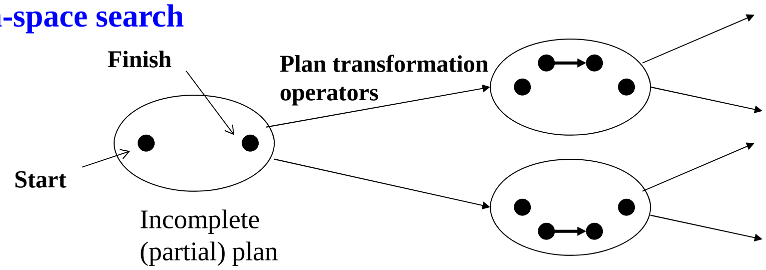
- An alternative to planning algorithms that search states (configurations of world)
- **Plan:** Defines a sequence of operators to be performed
- **Partial plan:**
  - plan that is not complete
    - Some plan steps are missing
  - some orderings of operators are not finalized
    - Only relative order 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



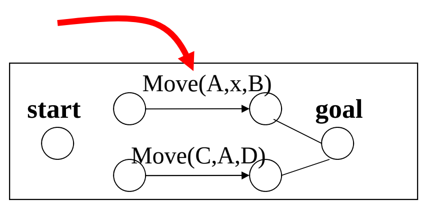
Plan-space search



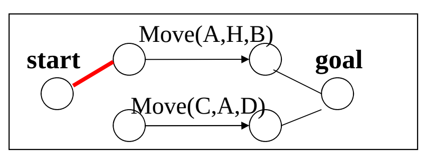
Plan transformation operators

Examples of :

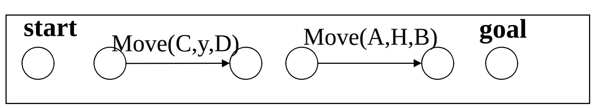
- **Add an operator to a plan so that it satisfies some open condition**



- **Add link (+ instantiate)**



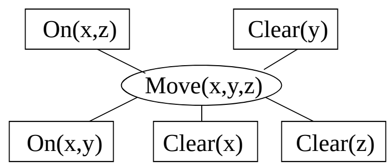
- **Order (reorder) operators**



Partial-order planners (POP)

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

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



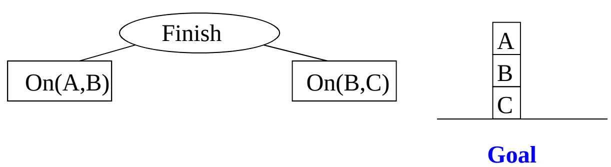
add list

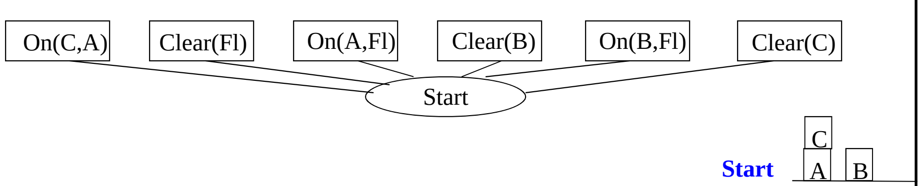
preconditions

Delete list is not shown !!!

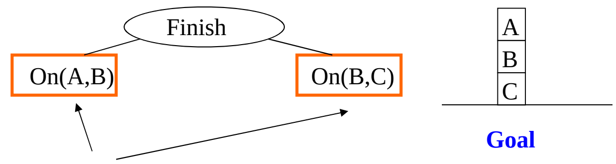
Illustration of a POP on the Sussman's anomaly case

Partial order planning. Start and finish.

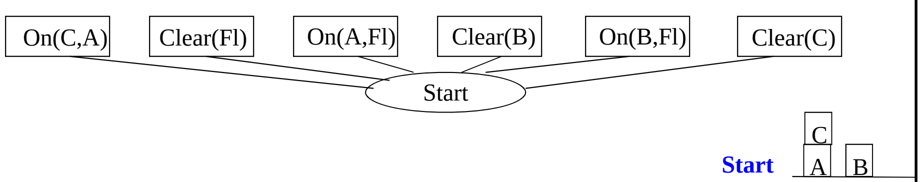




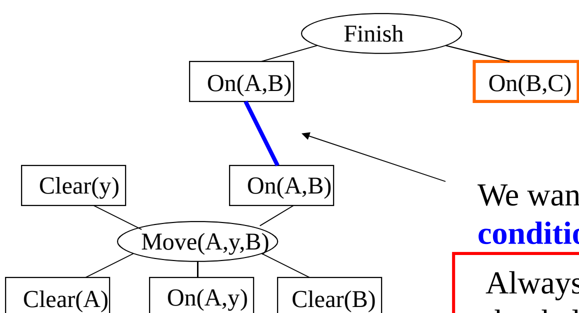
Partial order planning. Start and finish.

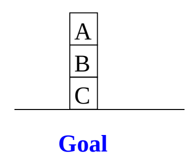


Open conditions: conditions yet to be satisfied



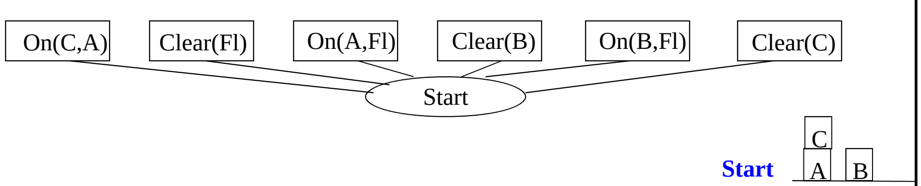
Partial order planning. Add operator.



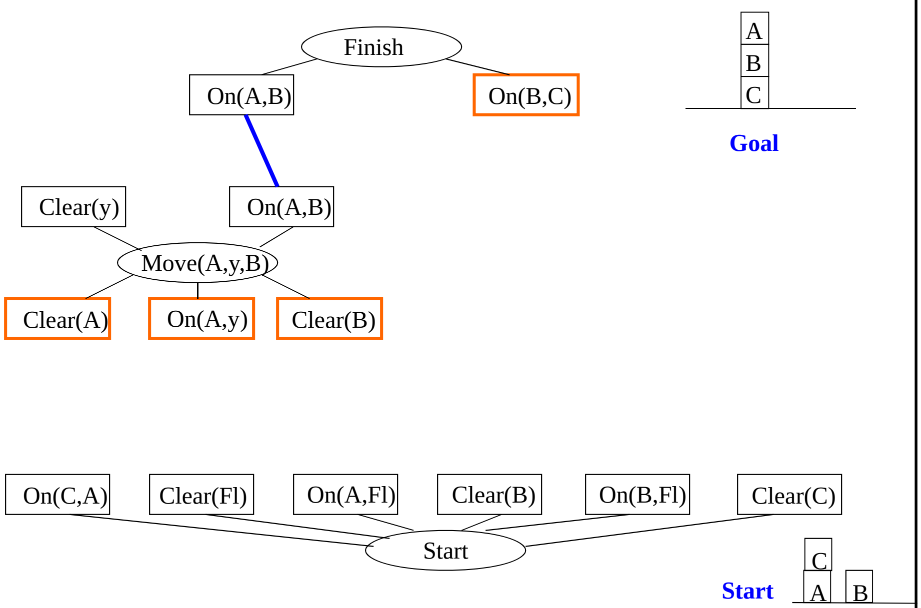


We want to satisfy **an open condition**

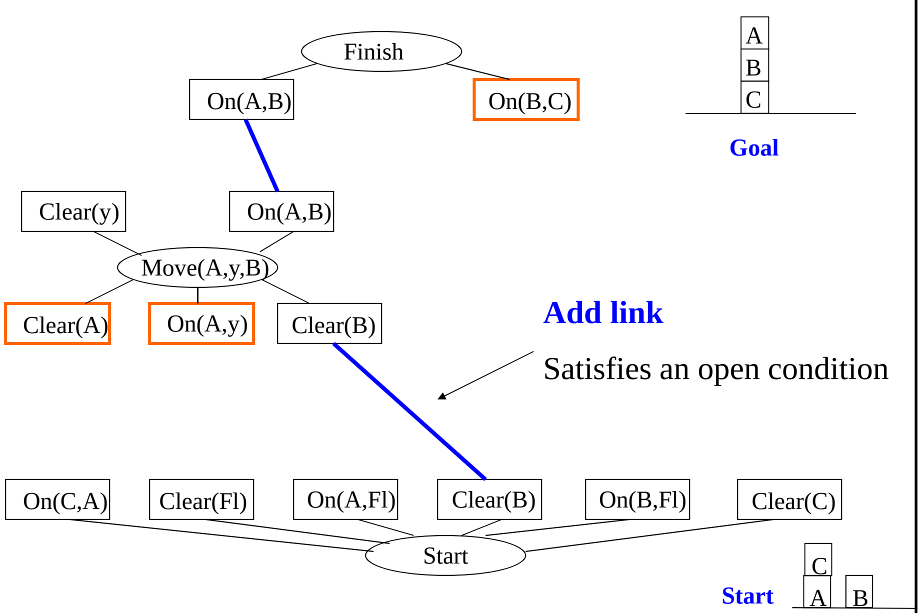
Always select an operator that helps to satisfy one of the open conditions



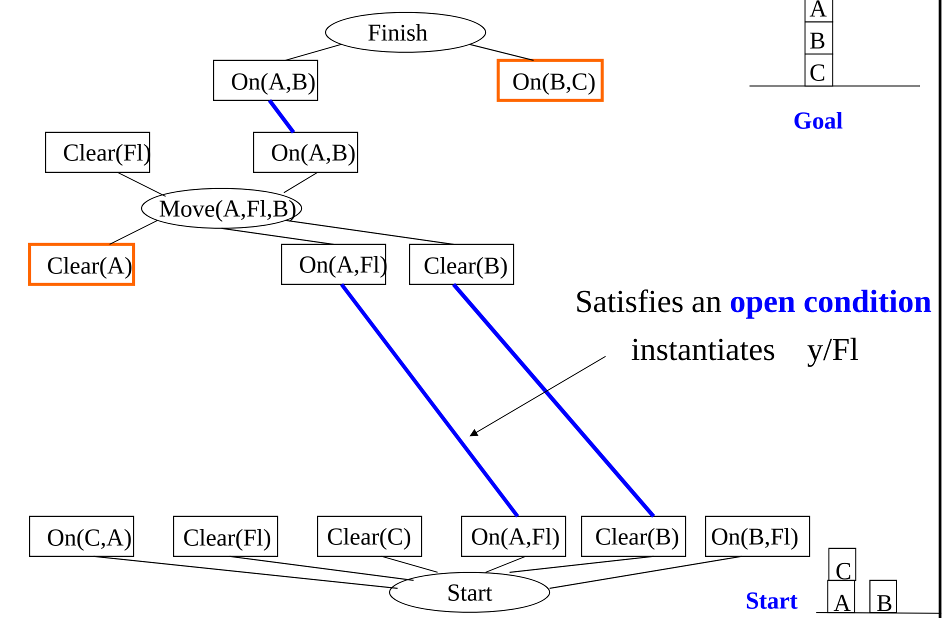
Partial order planning. Add link.



Partial order planning. Add link.



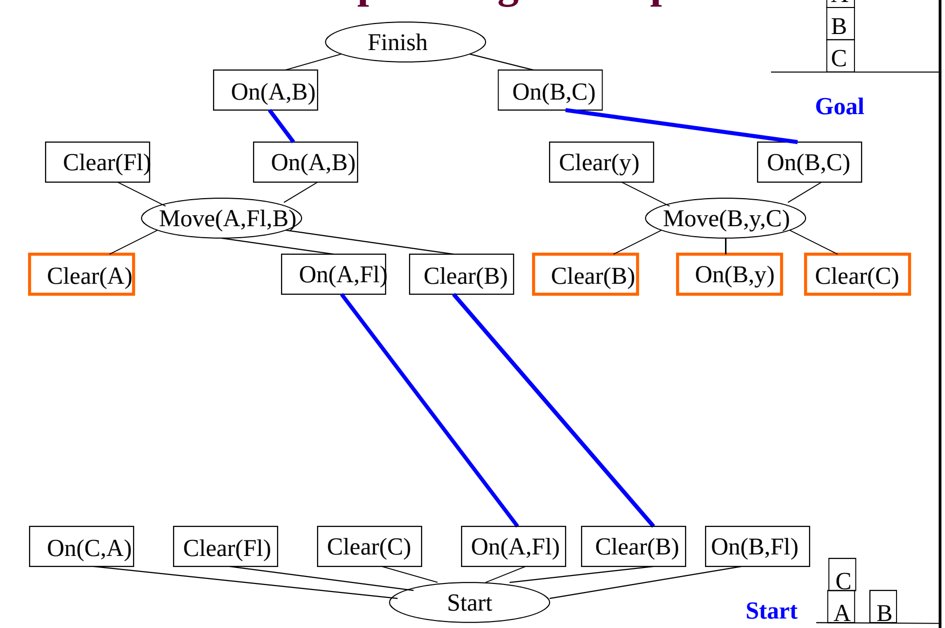
Partial order planning. Add link.



CS 1571 Intro to AI

M. Hauskrecht

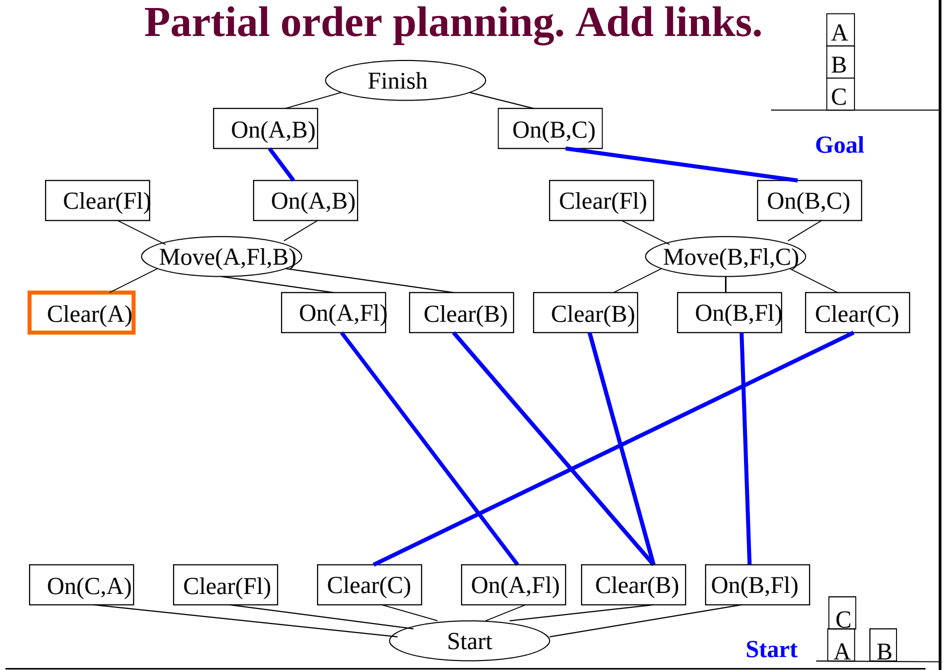
Partial order planning. Add operator.



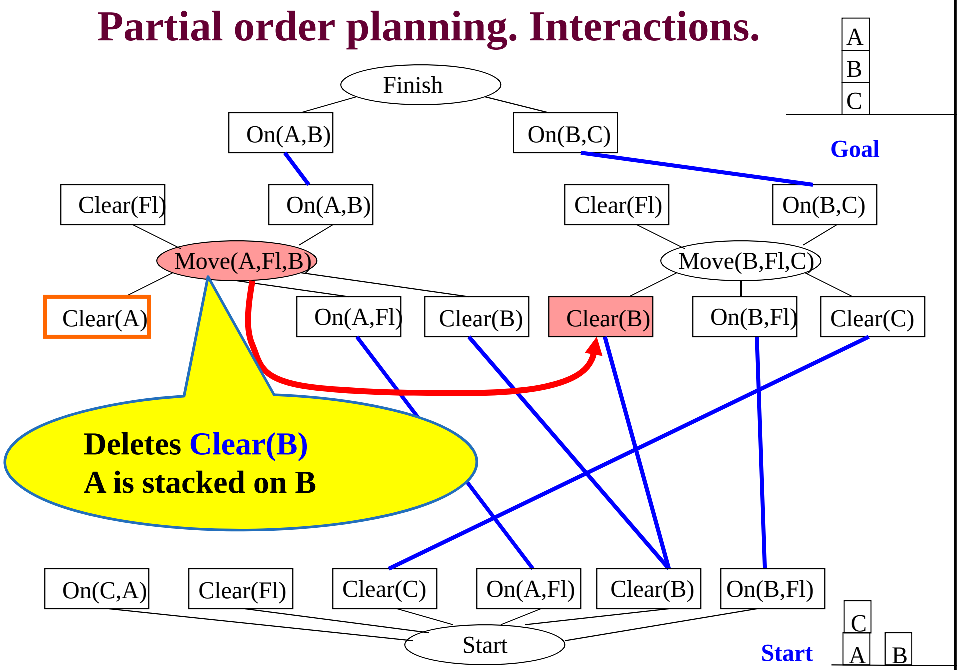
CS 1571 Intro to AI

M. Hauskrecht

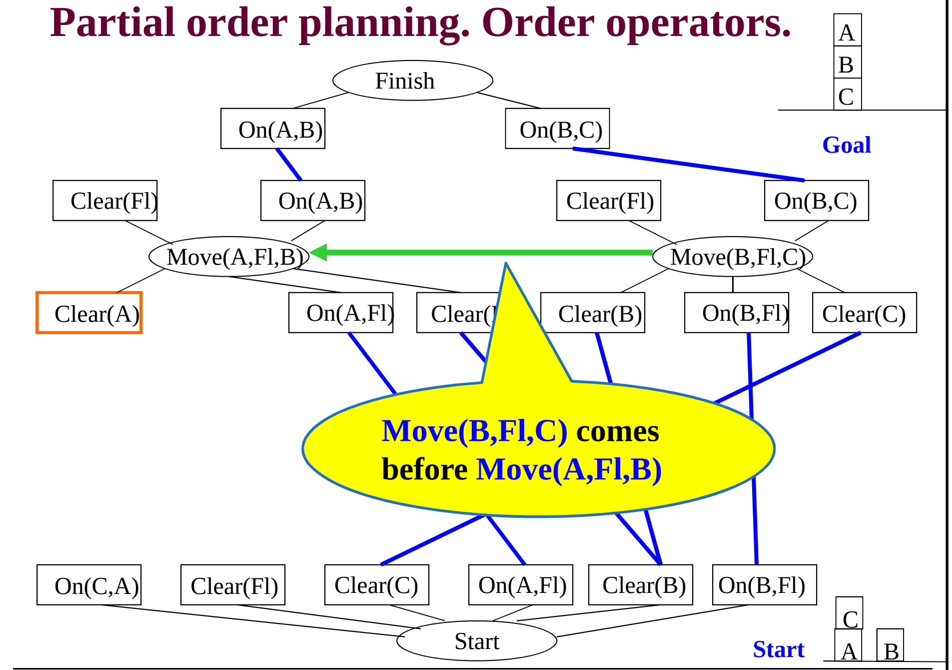
Partial order planning. Add links.



Partial order planning. Interactions.

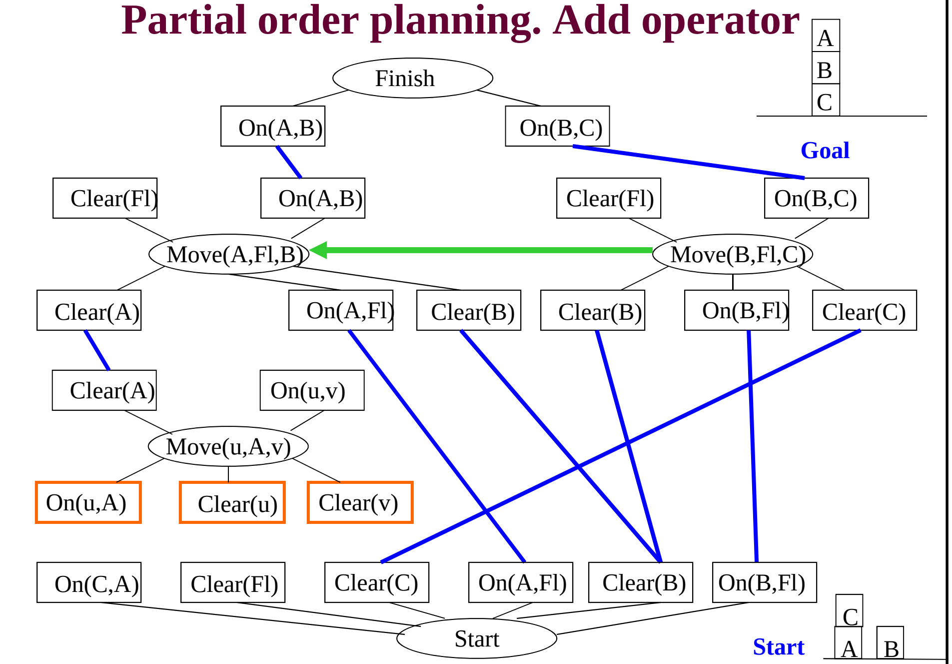


Partial order planning. Order operators.



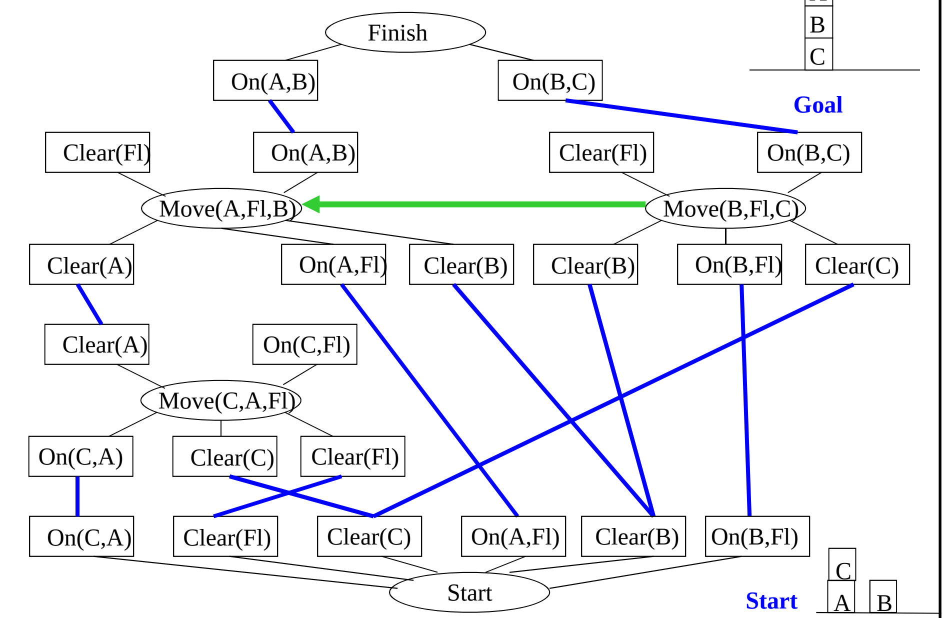
CS 1571 Intro to AI M. Hauskrecht

Partial order planning. Add operator

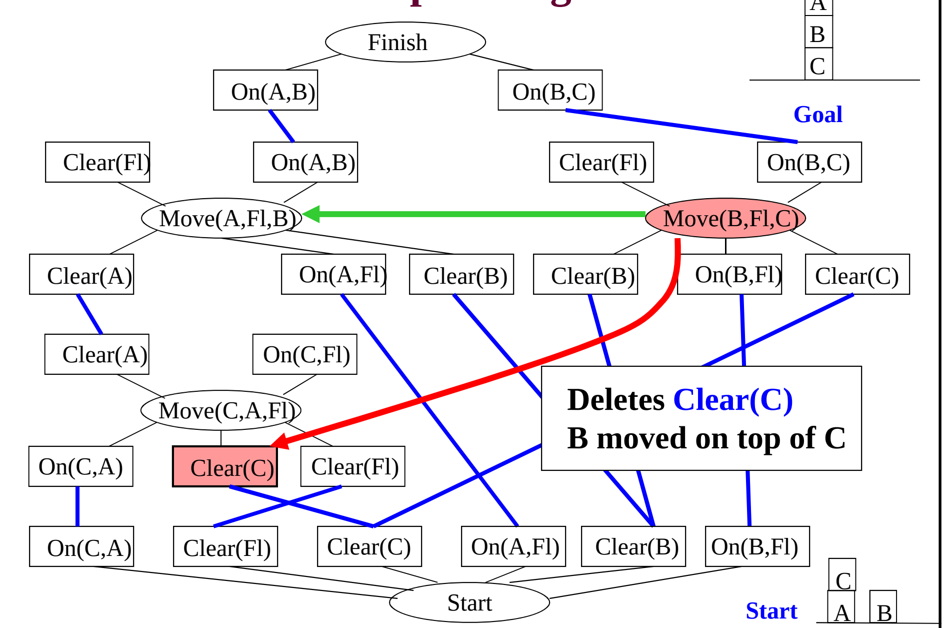


CS 1571 Intro to AI M. Hauskrecht

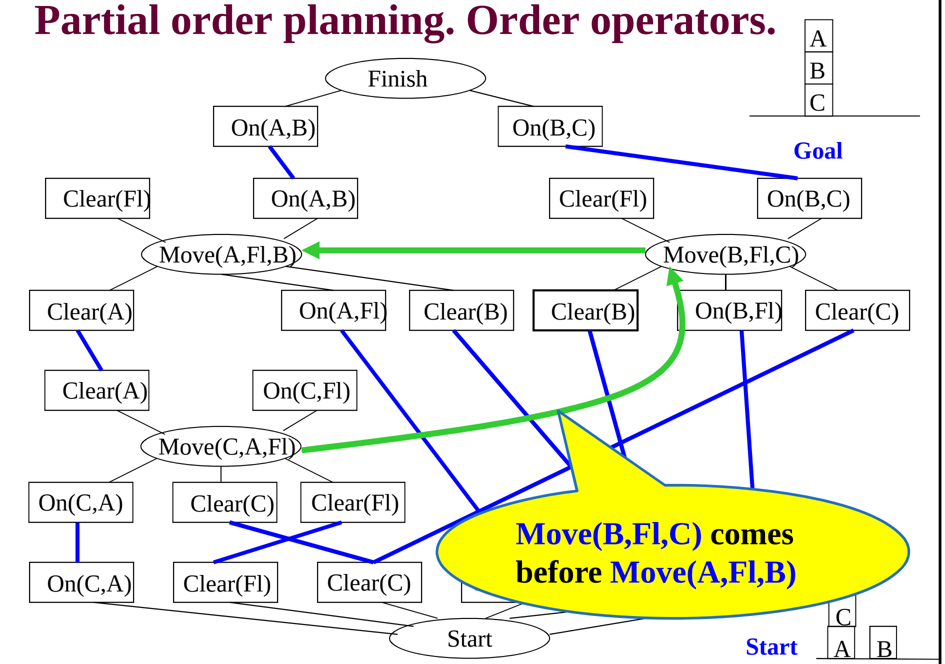
Partial order planning. Add links.



Partial order planning. Threats.



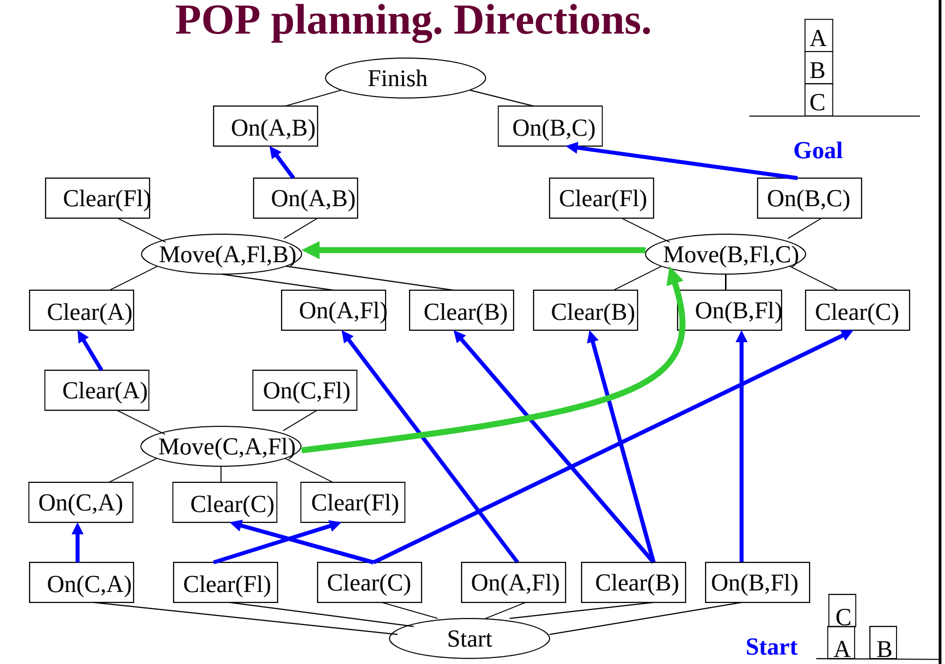
Partial order planning. Order operators.



CS 1571 Intro to AI

M. Hauskrecht

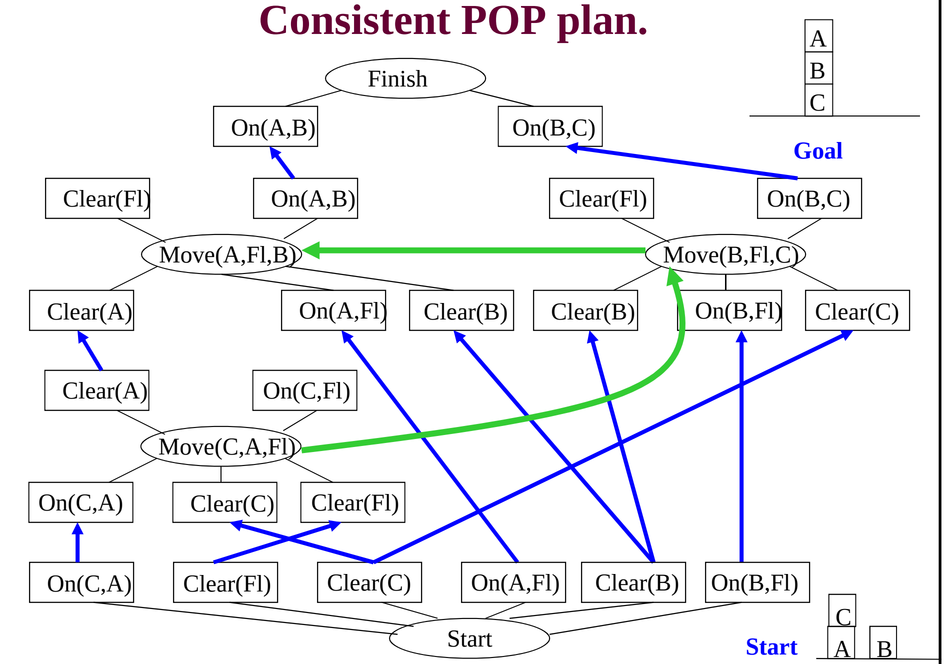
POP planning. Directions.



CS 1571 Intro to AI

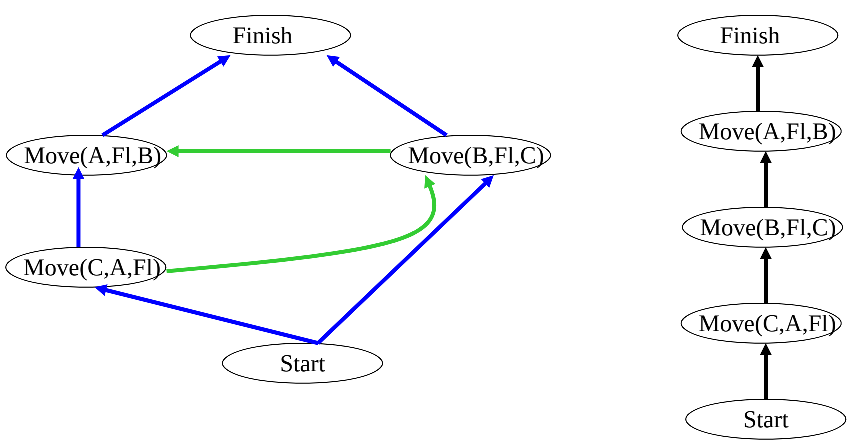
M. Hauskrecht

Consistent POP plan.



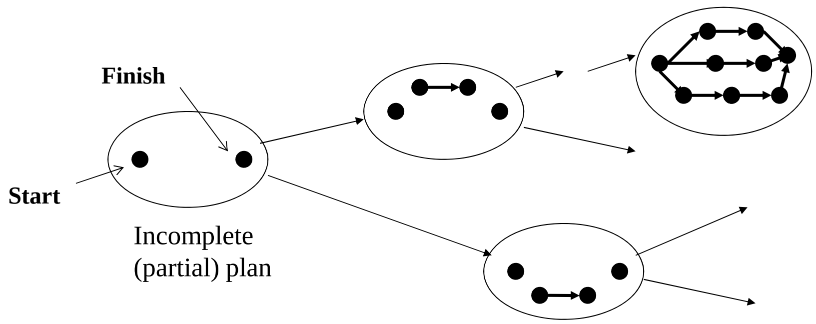
Partial order planning. Result plan.

Plan: a topological sort of a graph



Partial order planning.

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



- POP: **is sound and complete**

Hierarchical planners

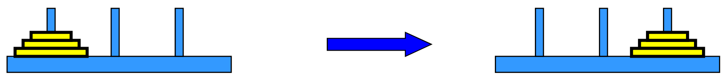
Extension of STRIPS planners.

- Example planner: ABSTRIPS.

Idea:

- Assign a **criticality level** to each conjunct in preconditions list of the operator
- Planning process refines the plan gradually based on criticality threshold, starting from the highest criticality value:
  - Develop the plan ignoring preconditions of criticality less than the criticality threshold value (assume that preconditions for lower criticality levels are true)
  - Lower the threshold value by one and repeat previous step

Towers of Hanoi



Start position   Goal position

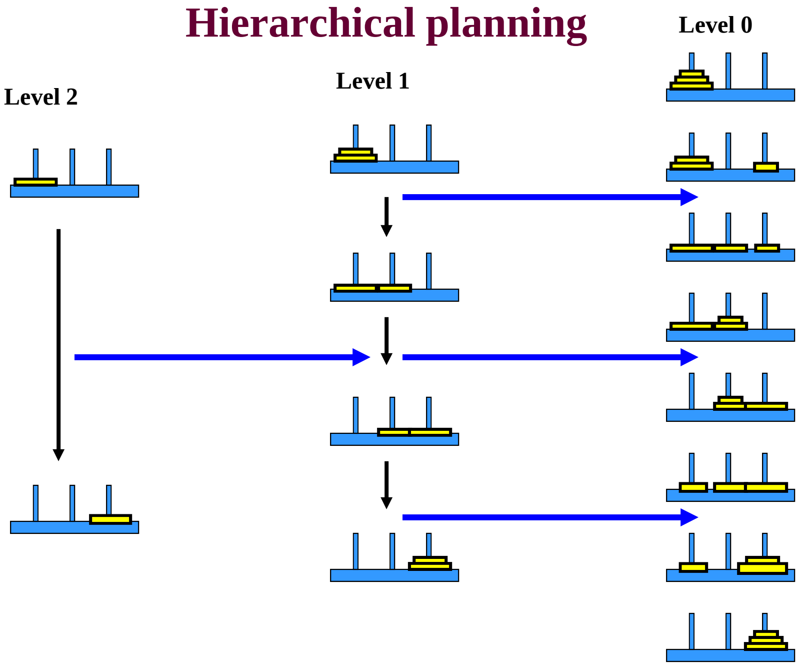
Hierarchical planning

Assume:

- the largest disk – criticality level 2
- the medium disk – criticality level 1
- the smallest disk – criticality level 0

CS 1571 Intro to AI   M. Hauskrecht

Hierarchical planning



CS 1571 Intro to AI   M. Hauskrecht

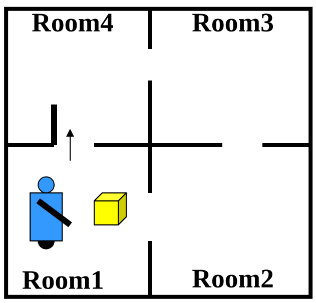
Planning with incomplete information

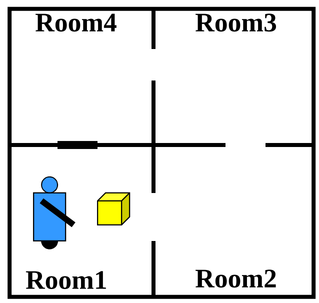
Some conditions relevant for planning can be:

- **true, false or unknown**

Example:

- Robot and the block is in Room 1
- **Goal:** get the block to Room 4
- **Problem:** The door between Room1 and 4 can be closed

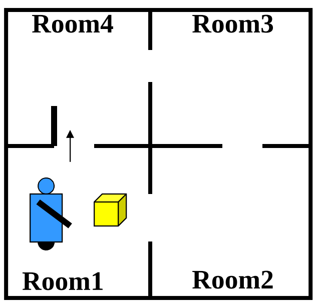


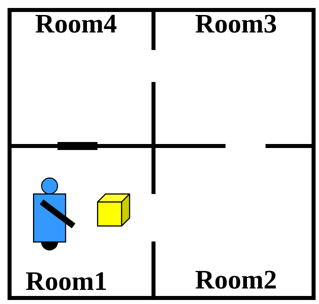


Planning with incomplete information

Initially we do not know whether the door is opened or closed:

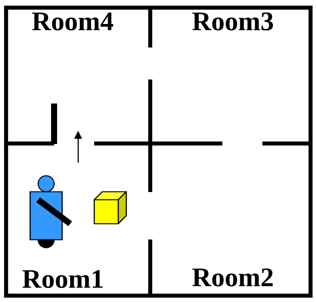
- **Different plans:**
  - **If not closed**: pick the block, go to room 4, drop the block
  - **If closed**: pick the block, go to room2, then room3 then room4 and drop the block

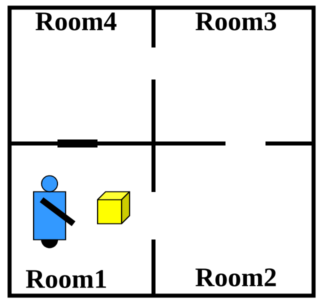




Conditional planners

- Are capable to create conditional plans that cover all possible situations (contingencies) – also called **contingency planners**
- Plan choices are applied when the missing information becomes available
- Missing information can be sought actively through actions
  - **Sensing actions**





Sensing actions

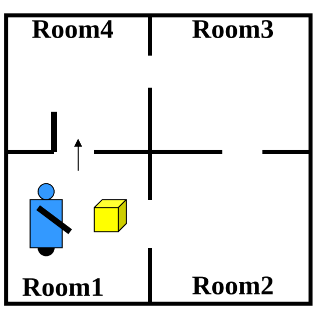
Example:

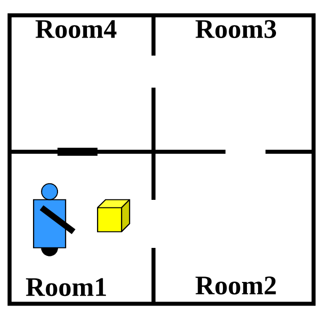
CheckDoor(d): checks the door d

Preconditions: Door(d,x,y) – one way door between x and y

& At(Robot,x)

Effect: (Closed(d) v ¬Closed(d)) - one will become true





Conditional plans

Sensing actions and conditions incorporated within the plan: