CS 1571 Introduction to AI
Lecture 17

Partial order planning

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

Planning

Planning problem:
- find a sequence of actions that lead to a goal
- this requires to model and reason about effects of agent's actions on the real-world.

Planning problem:
  - is a special type of a **search problem**
- **State space:** states of the world.
- **Initial state**: A world state we start from.
- **Operators**. Application of actions that change the state.
- **Goal condition**. Desired state of the world.

Planning

Specifics of planning problems:

– Complex description of world states
– Large number of actions
– Every action effects only a "small" subset of relations in the state
– Goals are defined over a "small" set of relations

Challenges:

– Build a representation language for modeling action and change
– Design of special search algorithms for a given representation

Planning systems design.

Two planning systems designs:

- **Situation calculus**
  – based on first-order logic,
  – a situation variable models new states of the world
  – use inference methods developed for FOL to do the reasoning
- **STRIPS –planners**
  – STRIPS – Stanford research institute problem solver
  – Restricted language as compared to the situation calculus
  – Allows for more efficient planning algorithms

STRIPS representation.

- More restricted representation language as compared to the situation calculus
- **States:**
  - represent facts that are true at a specific point in time conjunction of literals, e.g. *On(A,B), On(B,Table), Clear(A)*
- **Actions (operators):**

Operator: *Move (x,y,z)*
- **Preconditions:** *On(x,y), Clear(x), Clear(z)*
- **Effect lists:**
  - **Add list:** *On(x,z), Clear(y)*
  - **Delete list:** *On(x,y), Clear(z)*
    (Everything else is unaffected)

- **Goals:** conjunctions of literals, e.g. *On(A,B), On(B,C),*

STRIPS representation. Benefits.

Benefits:
- States, actions and goals have structure
- **Action representation**:
  - Leads to more intuitive and compact description of actions (no need to write many axioms !!!)
  - Avoids the frame problem
- Restrictions lead to more efficient planning algorithms.

STRIPS planning:
- find a sequence of operators from the initial state to the goal
- Search problem definition in STRIPS looks much like the standard search problem definition

STRIPS planning.

STRIPS planning problem:

- Find a sequence of actions that lead to a goal
- States and goals are defined by a conjunctions of literals

Two basic search methods:

- **Forward search** (goal progression)
  - From the initial state try to reach the goal
- **Backward search** (goal regression)
  - Start from the goal and try to project it to the initial state

More complex planning method:

- **Partial-order planning (POP)**
  - Search the space of partially build plans

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

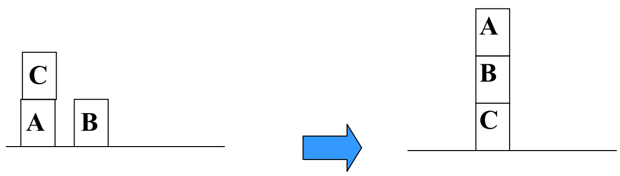
In planning we would like to satisfy a set of goals

- **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 in planning?
  - No. There can be interacting goals.

Sussman's anomaly.

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

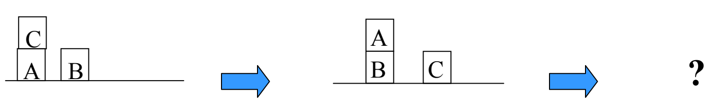


Initial state **Goal**

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

Sussman's anomaly

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

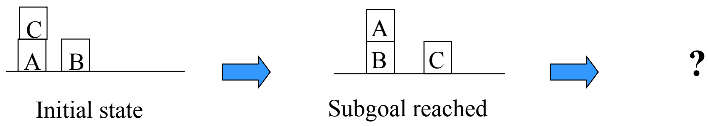


Initial state Subgoal reached

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

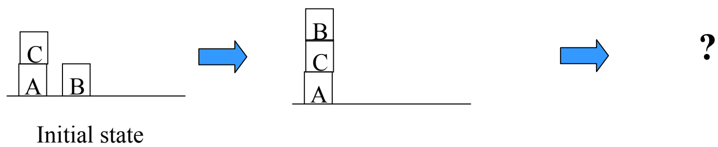
Sussman's anomaly

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



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

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



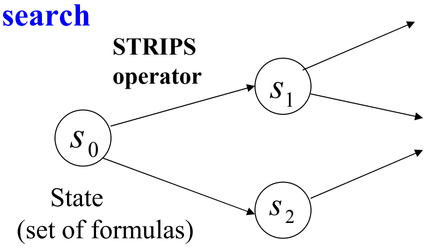
But now we cannot satisfy $On(A,B)$ without undoing $On(B,C)$

State space vs. plan space

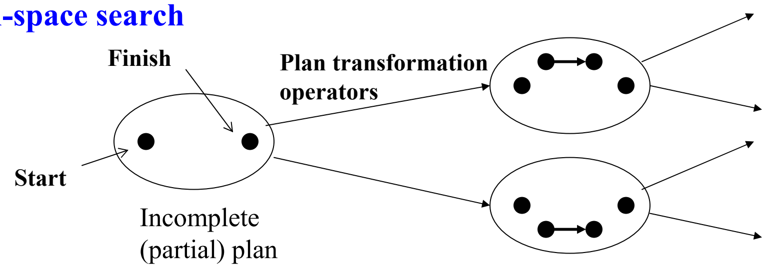
- An alternative to planning algorithms that search states (configurations of world) is to search the space of plans
- **Plan:** Defines sequences of operators to be performed
- **Partial plan:**
  - Some plan steps are missing
  - Some orderings of operators are not finalized
    - Only relative order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



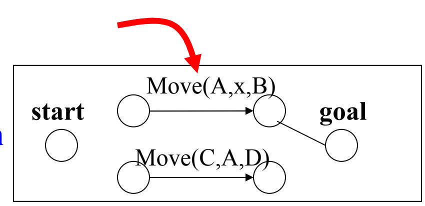
Plan-space search



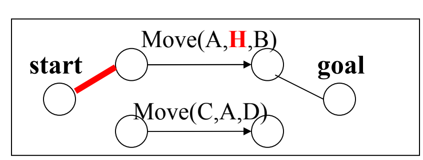
Plan transformation operators

Examples of operators:

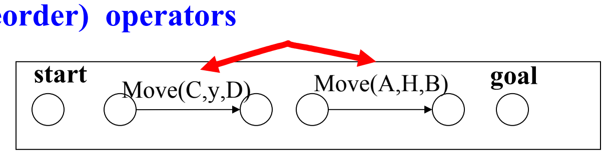
- Add an operator to a plan so that it satisfies some open condition



- Add link (+ instantiate)



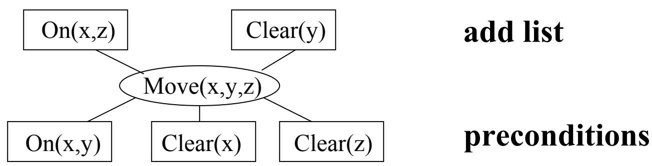
- Order (reorder) operators



Partial-order planners (POP)

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

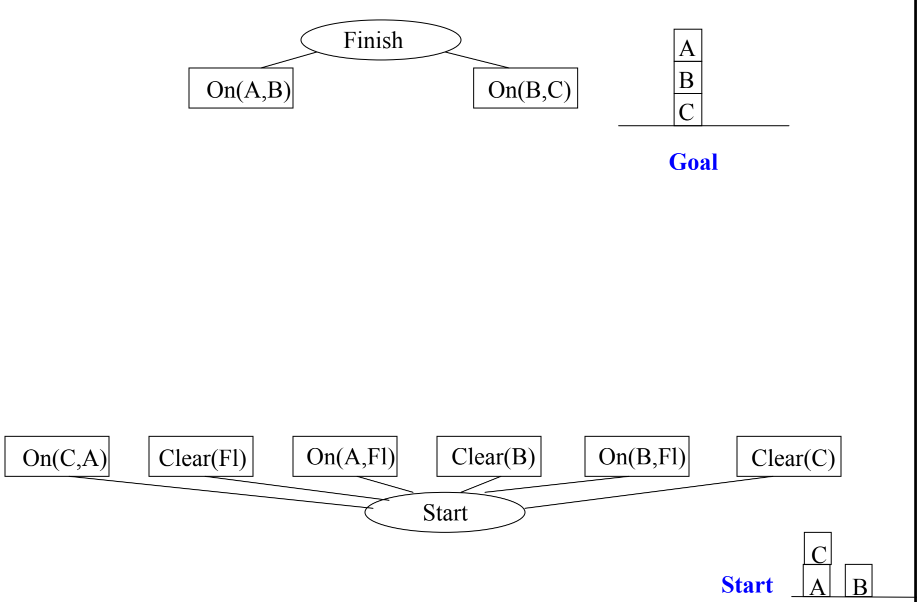
Graphical representation of an **operator** **Move(x,y,z)**



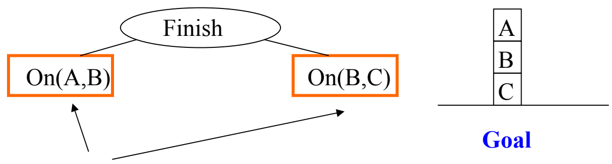
Delete list is not shown !!!

Illustration of POP on the Sussman's anomaly case

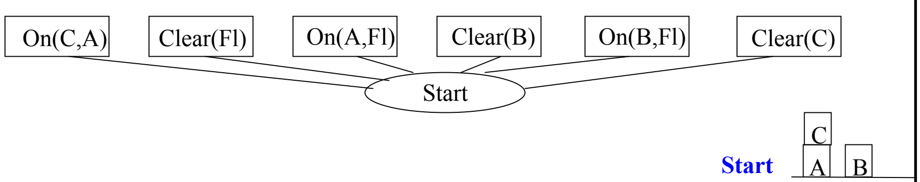
Partial order planning. Start and finish.



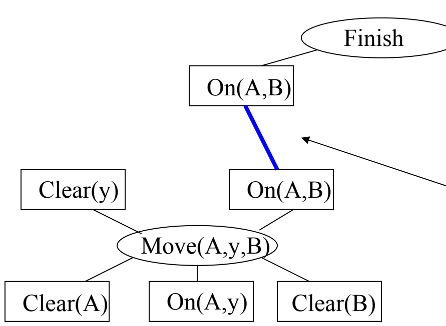
Partial order planning. Start and finish.

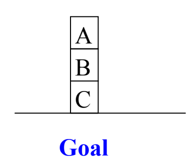


Open conditions: conditions yet to be satisfied



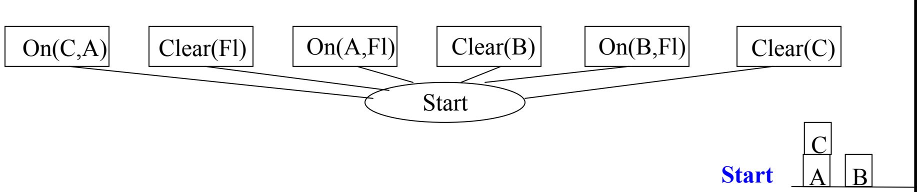
Partial order planning. Add operator.



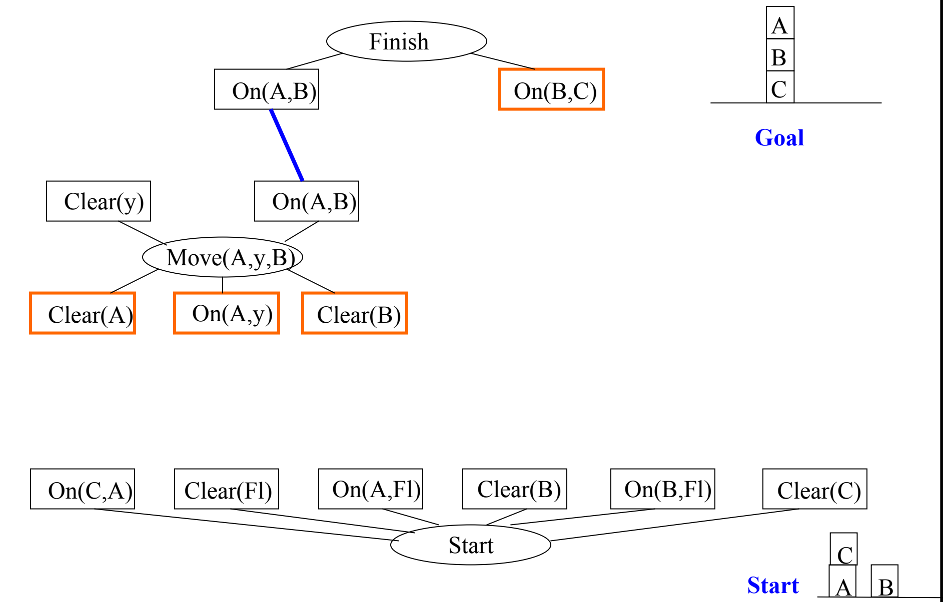


We want to satisfy **an open condition**

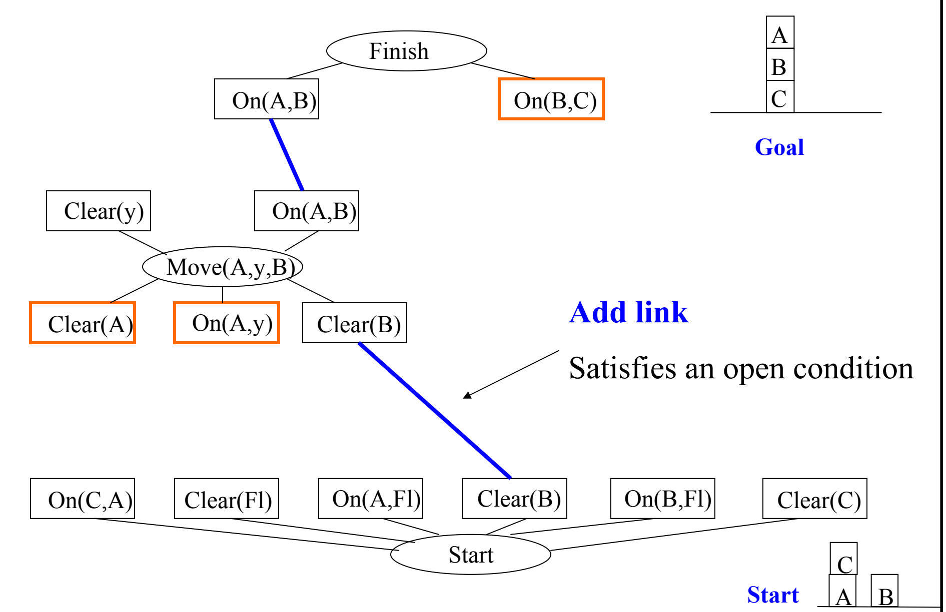
Always select an operator that helps to satisfy one of the open conditions



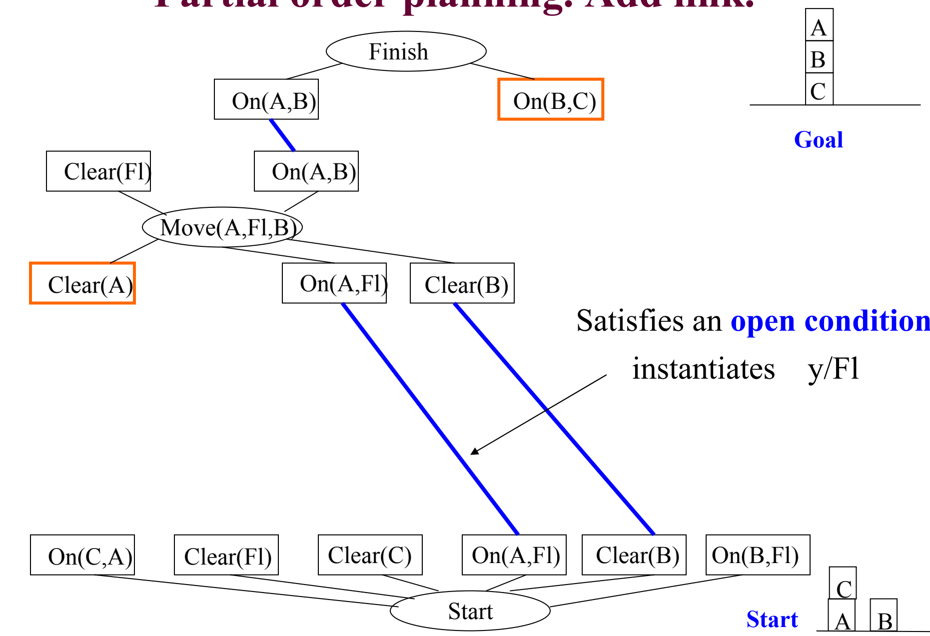
Partial order planning. Add link.



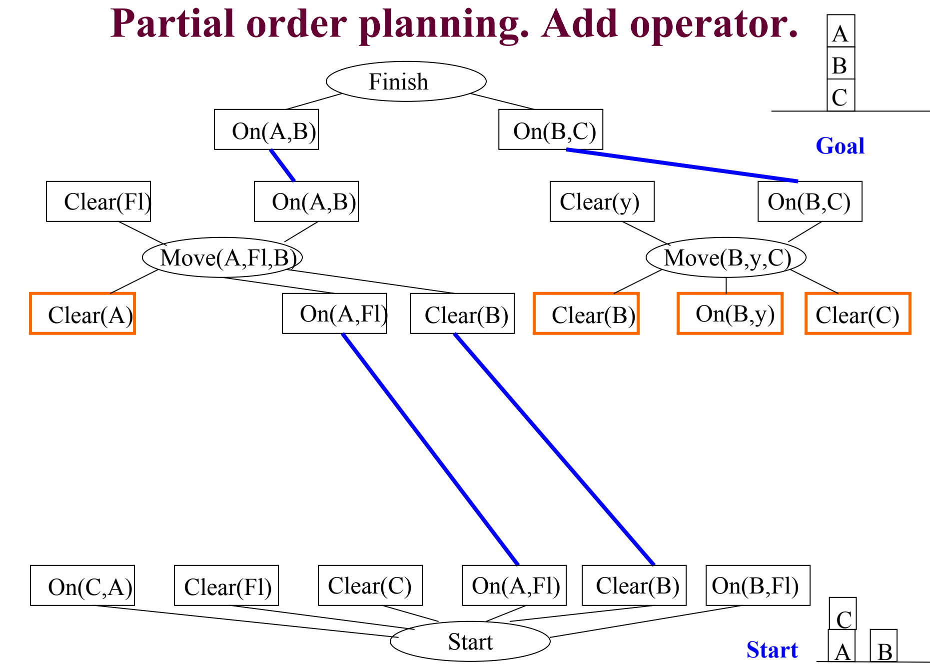
Partial order planning. Add link.



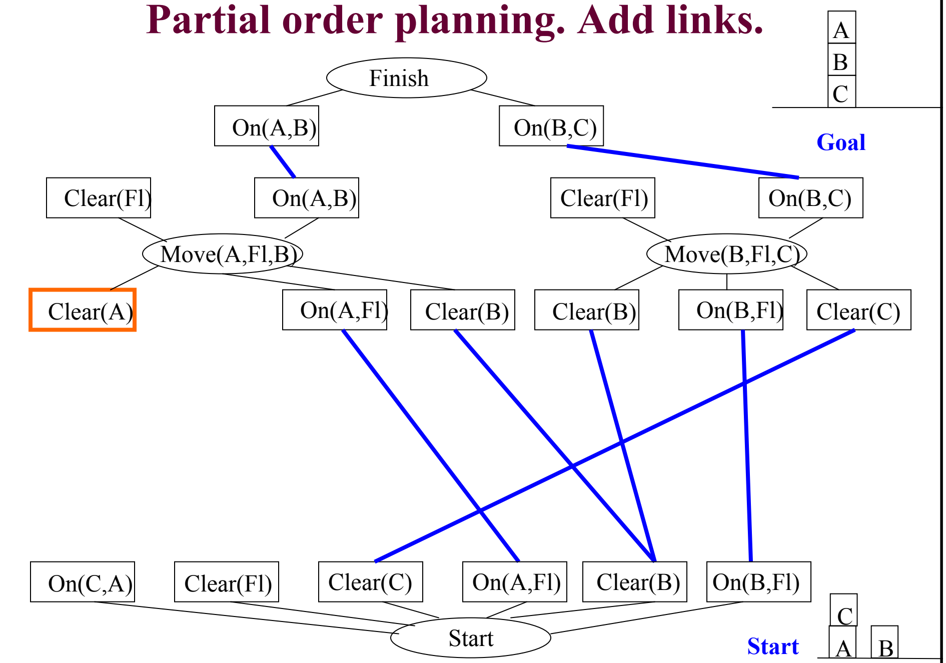
Partial order planning. Add link.



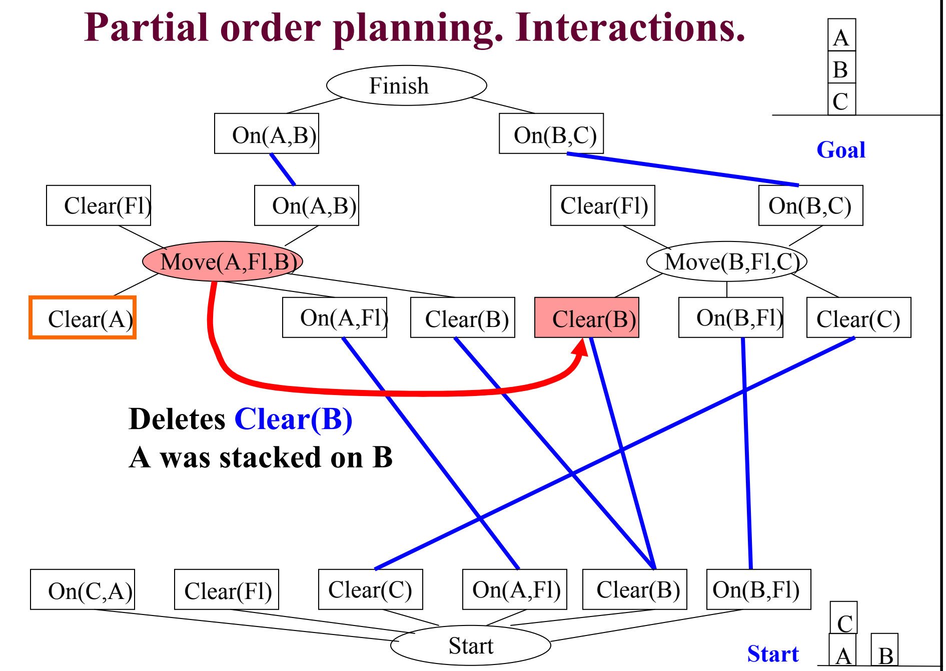
Partial order planning. Add operator.



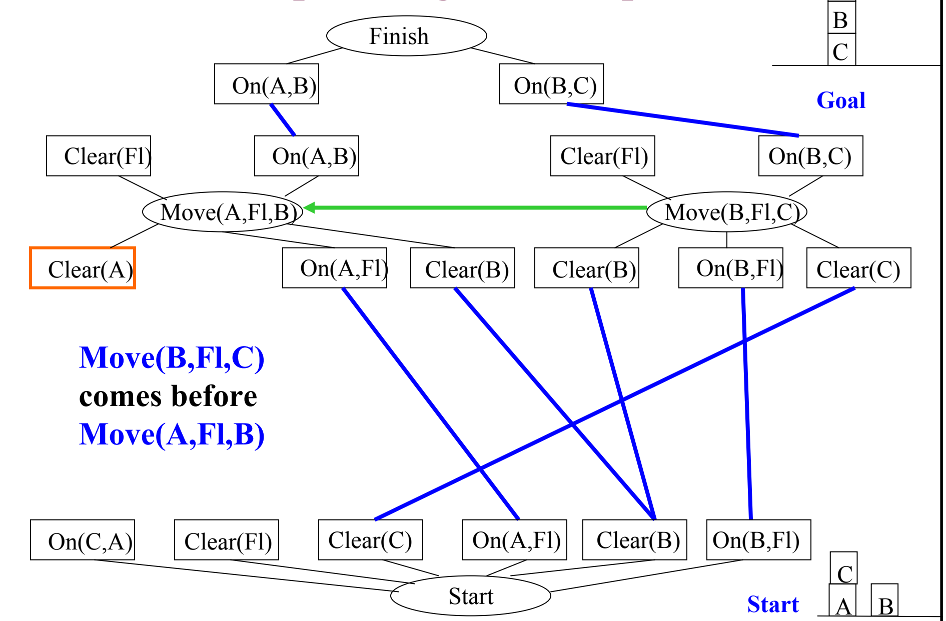
Partial order planning. Add links.



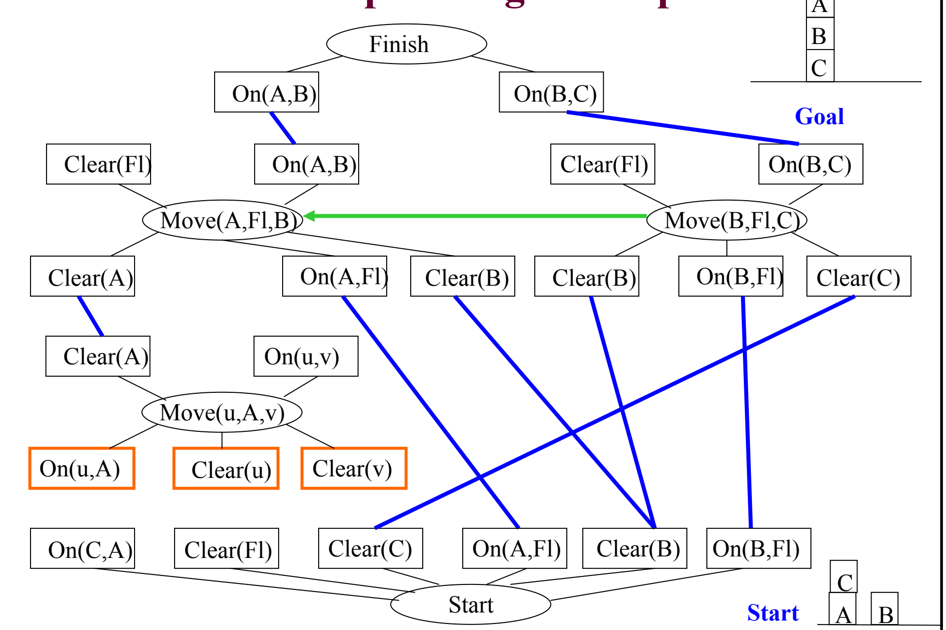
Partial order planning. Interactions.



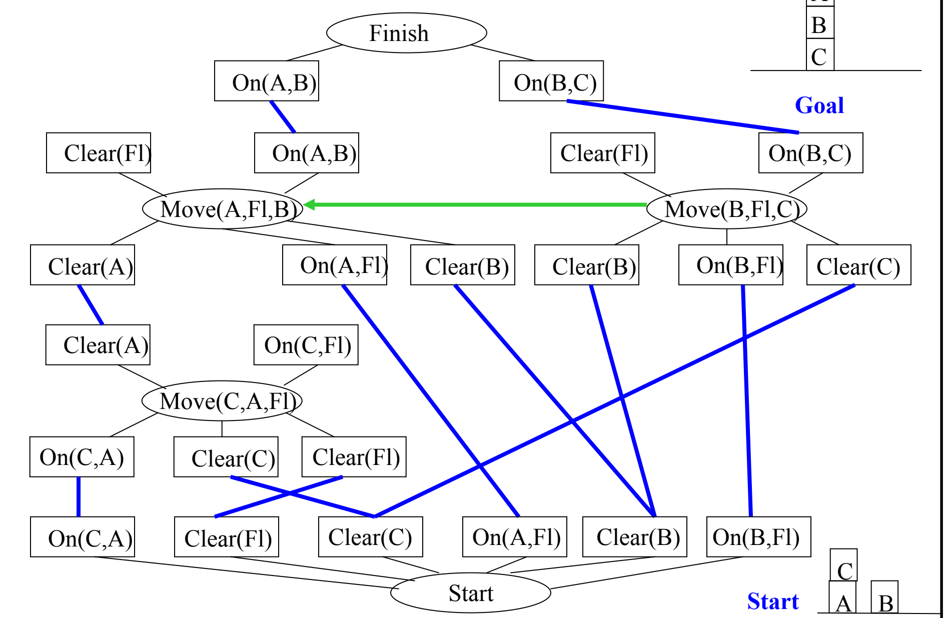
Partial order planning. Order operators.



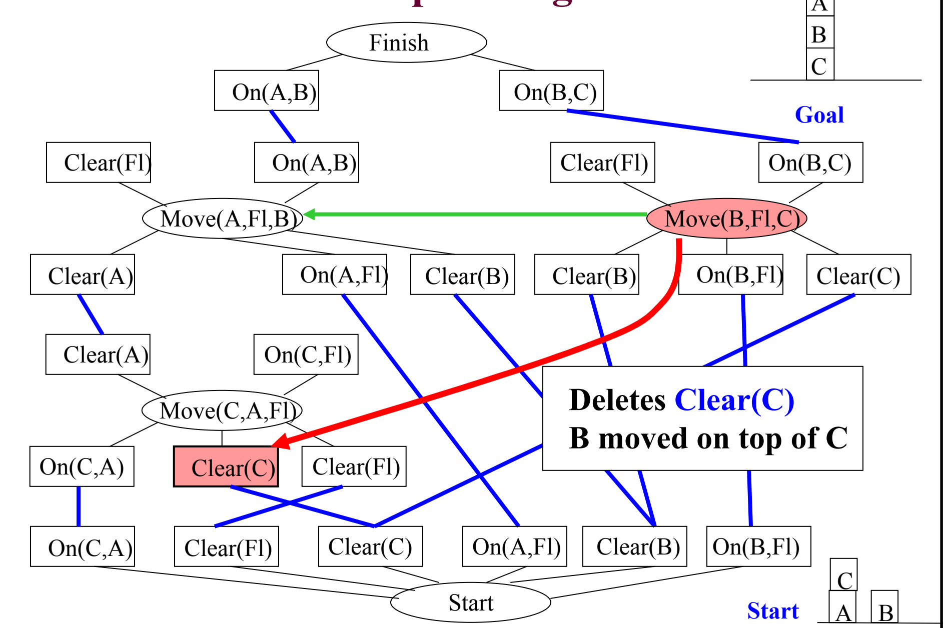
Partial order planning. Add operator



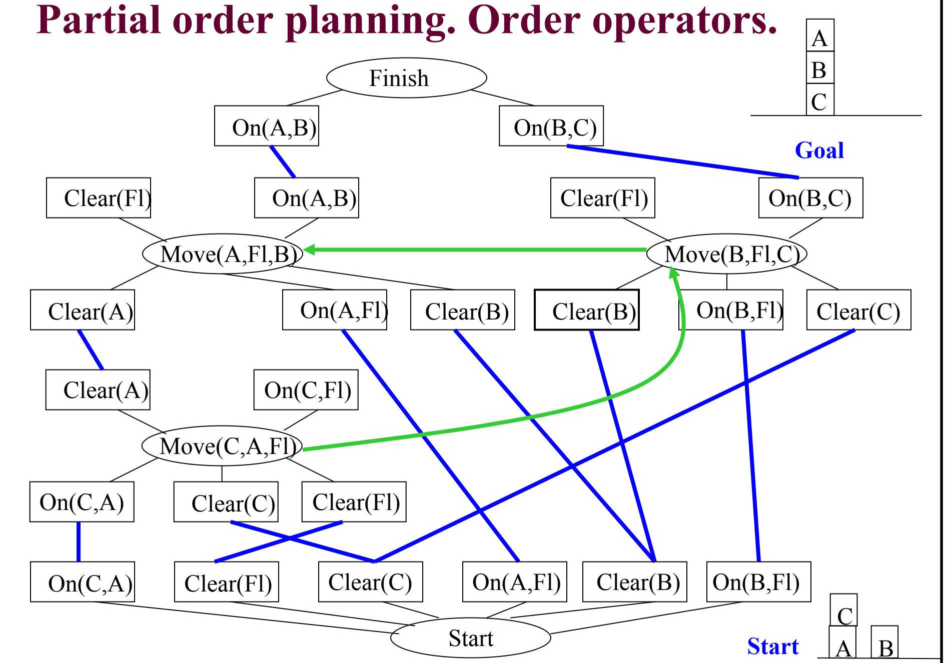
Partial order planning. Add links.



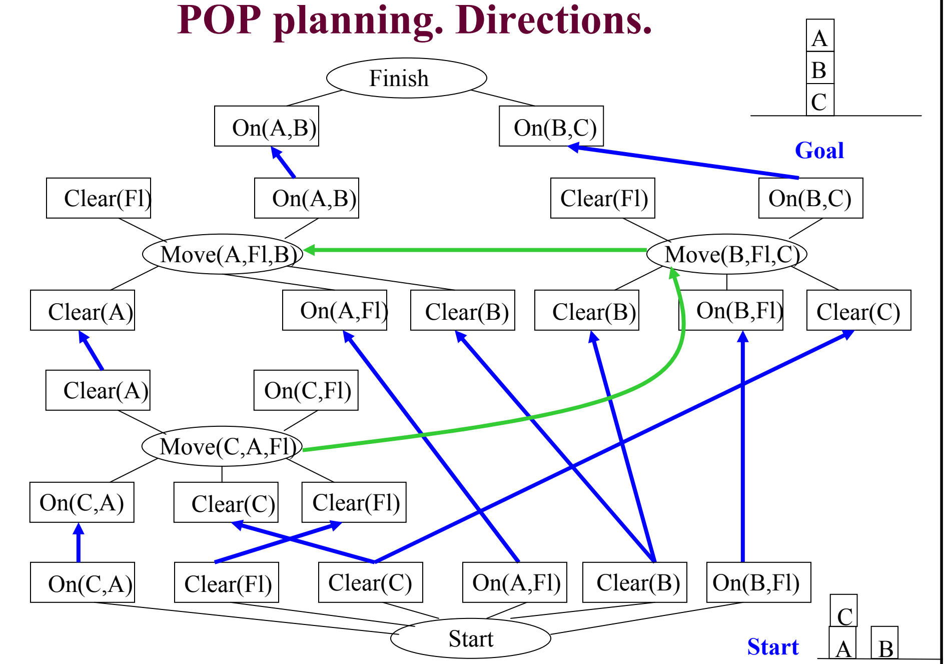
Partial order planning. Threats.



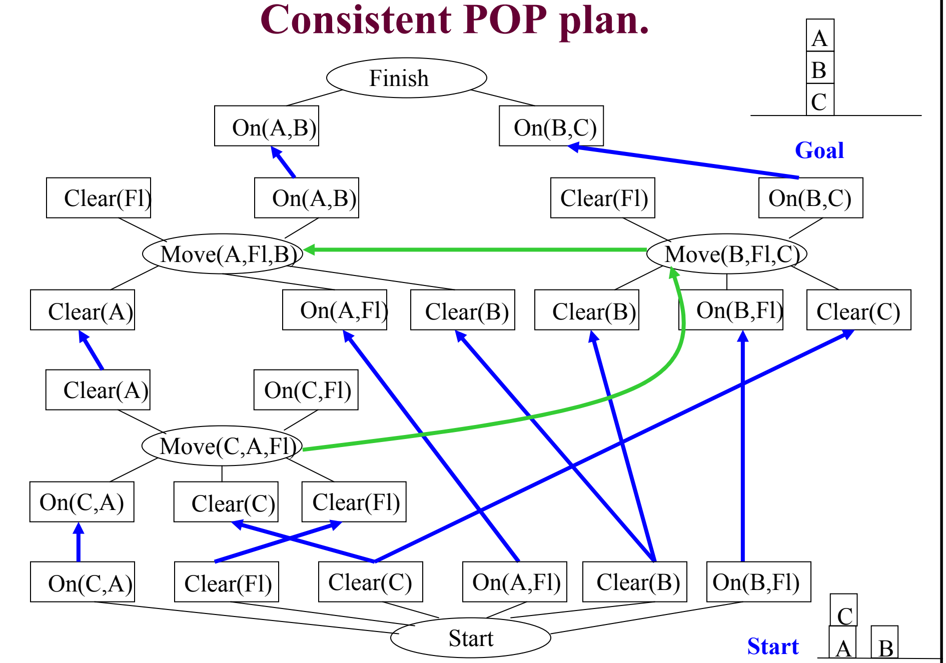
Partial order planning. Order operators.



POP planning. Directions.

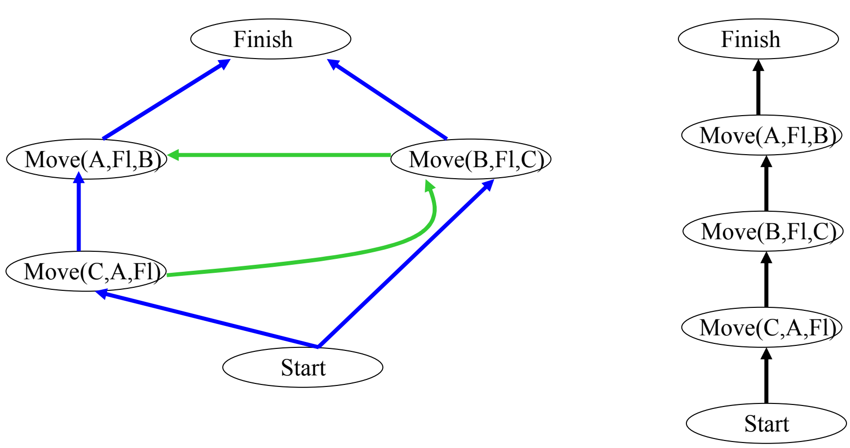


Consistent POP plan.



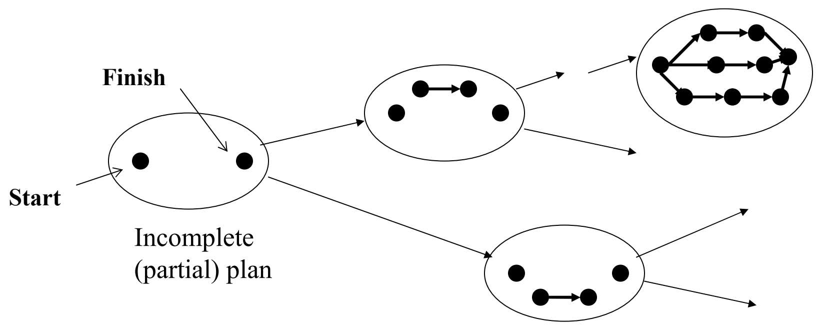
Partial order planning. Result plan.

Plan: a topological sort of a graph



Partial order planning.

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



- POP: **is sound and complete**

Hierarchical planners

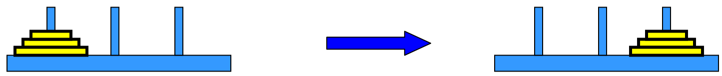
Extension of STRIPS planners.

- 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

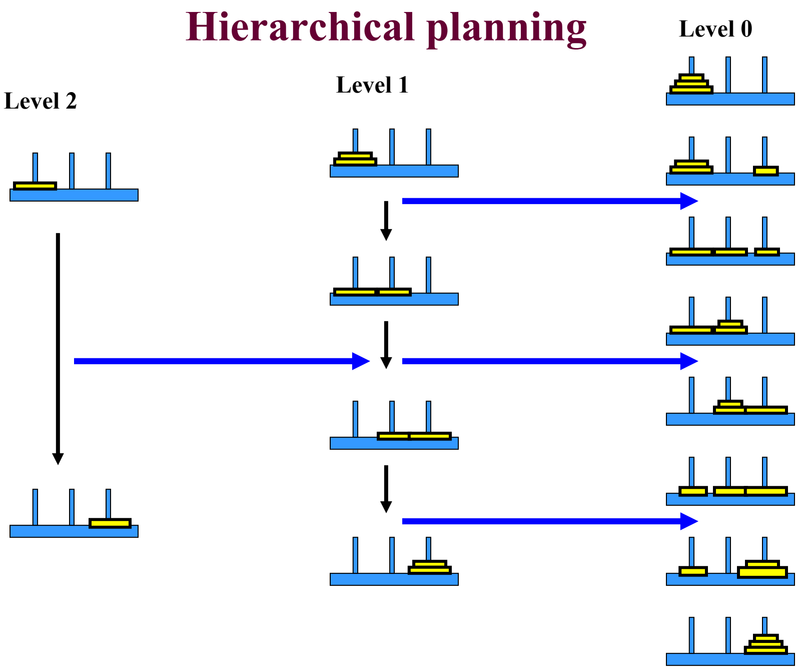
Assume:

the largest disk – criticality level 2

the medium disk – criticality level 1

the smallest disk – criticality level 0

Hierarchical planning



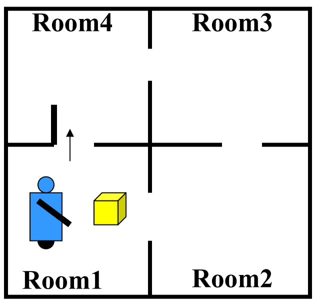
Planning with incomplete information

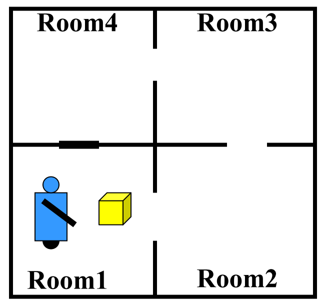
Some conditions relevant for planning can be:

– **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

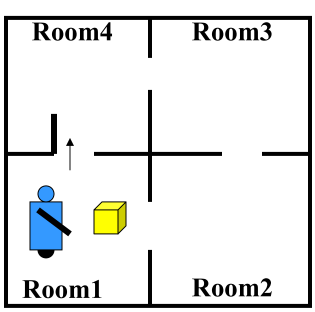


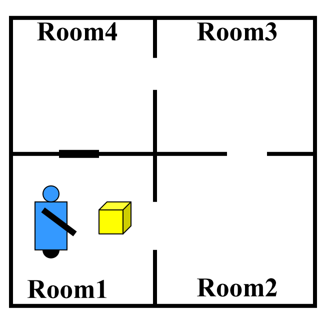


Planning with incomplete information

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

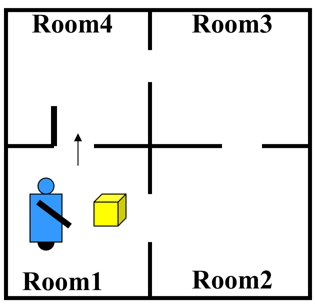
- **Different plans:**
  - **If not closed**: pick the block, go to room 4, drop the block
  - **If closed**: pick the block, go to room2, then room3 then room4 and drop the block

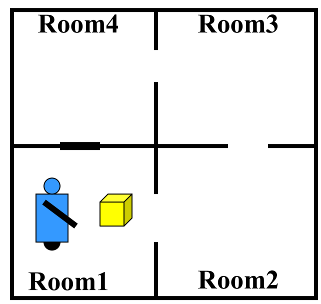




Conditional planners

- 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

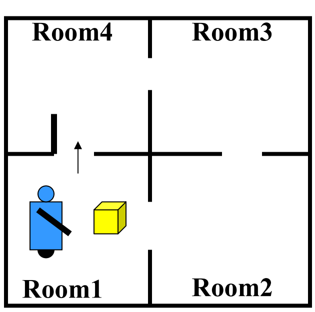
Example:

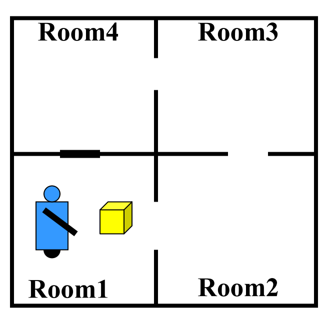
CheckDoor(d): checks the door d

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: