CS 1571 Introduction to AI
Lecture 14

STRIPS planning

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

Administration

- **Midterm:**
  - **Tuesday, October 15, 2002**
  - **In-class**
  - **Closed book**
  - **All material covered (including) today**
  - **Last year midterm is on the web**
- **Problem sets:**
  - **PS 1-3** graded and available for pick up
  - **PS 4-5** will be ready by tomorrow
    (see Tomas in 5802 Sennott Square)
  - PS solutions are on the web

Planning

Assume: We want to design an intelligent agent that acts in the real world and accomplishes desired goals

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.

  Example: action of painting car12 causes:

  color(car12, white) = **true** to become

  color(car12, white) = **false**

  and *color(car12, blue)* = **true**

Planning problem:

- is a special type of a **search problem**

Planning

Search problem:

- **State space.** States of the world among which we search
- **Initial state**. A state we start from.
- **Operators**. Map states to new states.
- **Goal condition**. Test whether the goal is satisfied.

Specifics of a planning problem:

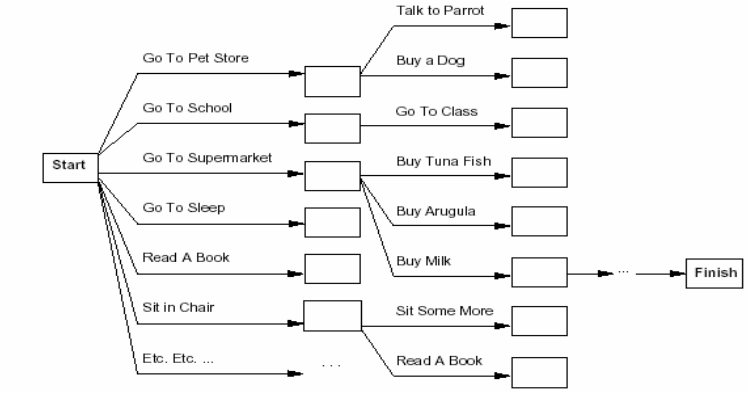
- Complex states
- Large number of actions
- Every action effects only a "small" subset of relations in the state
- Goals are defined over a "small" set of relations

This causes:

- **a large branching factor of the search tree,**
- **long action sequences (solution depth is large)**

Planning search. Example.

- Assume a simple problem of buying things:
  - **Get a quarter of milk, bananas, cordless drill**



- **A huge branch factor !!!**
- **Goals can take multiple steps to reach!!!**

Planning

Solutions to specifics of planning problems:

- **Open state, action and goal representations** to allow selection, reasoning. Make things visible and expose the structure.
  - Use FOL or its restricted subset to do the reasoning.
- **Add actions** to the plan sequence **wherever and whenever it is needed**
  - Drop the need to construct solutions sequentially from the initial state.
- **Apply divide and conquer strategies to sub-goals** if these are independent.

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 – like planners**
  - STRIPS – Stanford research institute problem solver
  - Restricted language as compared to the situation calculus
  - Allows for more efficient planning algorithms

Situation calculus

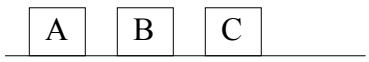
- Logic for reasoning about changes in the state of the world
- **The world is described by:**
  - Sequences of **situations** of the current state
  - Changes from one situation to another are caused by actions
- **The situation calculus allows us to:**
  - Describe the initial state and goal state
  - Build the KB that describes the effect of actions (operators)
  - Prove that the KB implies the goal state
    - and thereby allow us to extract a plan

Situation calculus

Language:

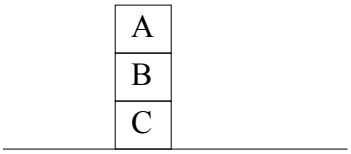
- **Special variables:** *s,a* – objects of type situation and action
- **Action functions** that return actions.
  - E.g. *Move(A, TABLE, B)* represents a move action
  - *Move(x,y,z)* represents an action schema
- **Two special function symbols of type situation**
  - s_0 – initial situation
  - *DO(a,s)* – denotes the situation obtained after performing an action a in situation s
- **Situation-dependent functions and relations** (also called fluents)
  - **Relation:** *On(x,y,s)* – object x is on object y in situation s;
  - **Function:** Above(x,s) – object that is above x in situation s.

Situation calculus. Blocks world example.



Initial state

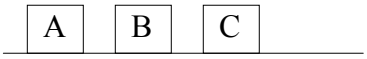
On(A,Table, s_0)
On(B,Table, s_0)
On(C,Table, s_0)
Clear(A, s_0)
Clear(B, s_0)
Clear(C, s_0)
Clear(Table, s_0)



Goal

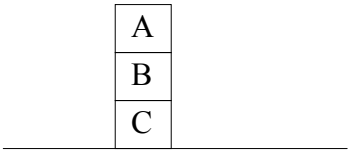
On(A,B, s)
On(B,C, s)
On(C,Table, s)

Blocks world example.



Initial state

On(A,Table, s_0)
On(B,Table, s_0)
On(C,Table, s_0)
Clear(A, s_0)
Clear(B, s_0)
Clear(C, s_0)
Clear(Table, s_0)



Goal

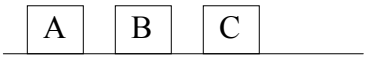
On(A,B, s)
On(B,C, s)
On(C,Table, s)

Note: It is not necessary that the goal describes all relations

Clear(A, s)

Blocks world example.

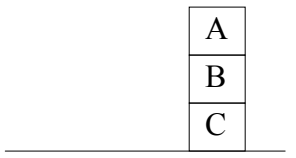
Assume a simpler goal *On(A,B, s)*

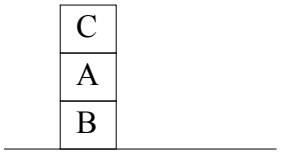


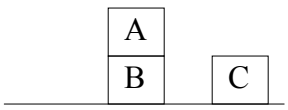
Initial state

On(A,Table, s_0)
On(B,Table, s_0)
On(C,Table, s_0)
Clear(A, s_0)
Clear(B, s_0)
Clear(C, s_0)
Clear(Table, s_0)

3 possible goal configurations







Goal *On(A,B, s)*

Knowledge about the world. Axioms.

Knowledge in the KB

- represents changes in the world due to actions.

Two types of axioms:

- **Effect axioms**
  - changes in situations that result from actions
- **Frame axioms**
  - things preserved from the previous situation

Blocks world example. Effect axioms.

Effect axioms:

Moving x from y to z. $MOVE(x, y, z)$

Effect of move changes on **On** relations

$$On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \rightarrow On(x, z, DO(MOVE(x, y, z), s))$$

$$On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \rightarrow \neg On(x, y, DO(MOVE(x, y, z), s))$$

Effect of move changes on **Clear** relations

$$On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \rightarrow Clear(y, DO(MOVE(x, y, z), s))$$

$$On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \wedge (z \neq Table) \rightarrow \neg Clear(z, DO(MOVE(x, y, z), s))$$

Blocks world example. Frame axioms.

- **Frame axioms.**
  - Represent things that remain unchanged after an action.

On relations:

$$On(u,v,s) \wedge (u \neq x) \wedge (v \neq y) \rightarrow On(u,v,DO(MOVE(x,y,z),s))$$

Clear relations:

$$Clear(u,s) \wedge (u \neq z) \rightarrow Clear(u,DO(MOVE(x,y,z),s))$$

Planning in situation calculus.

Planning problem:

- find a sequence of actions that lead to a goal

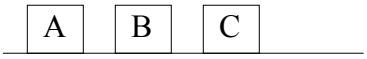
Planning in situation calculus is converted to theorem proving.

Goal state:

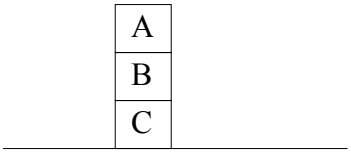
$$\exists s\ On(A,B,s) \wedge On(B,C,s) \wedge On(C,Table,s)$$

- Possible inference approaches:
  - **Inference rule approach**
  - **Conversion to SAT**
- **Plan** (solution) is a byproduct of theorem proving.
- **Example:** blocks world

Planning in a blocks world.



Initial state

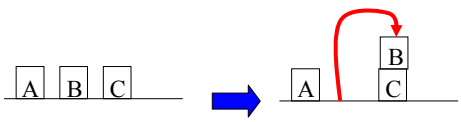


Goal

On(A,Table, s_0)
On(B,Table, s_0)
On(C,Table, s_0)
Clear(A, s_0)
Clear(B, s_0)
Clear(C, s_0)
Clear(Table, s_0)

On(A,B, s)
On(B,C, s)
On(C,Table, s)

Planning in the blocks world.



Initial state (s0) **s1**

$s_0 =$

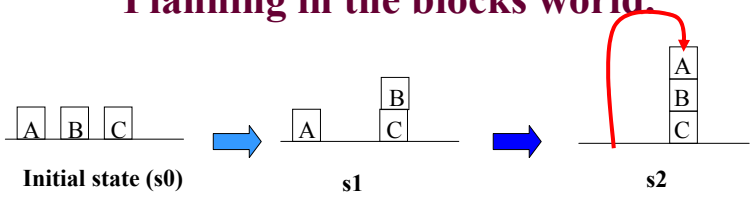
$On(A, Table, s_0)$ $Clear(A, s_0)$ $Clear(Table, s_0)$
$On(B, Table, s_0)$ $Clear(B, s_0)$
$On(C, Table, s_0)$ $Clear(C, s_0)$

Action: $MOVE(B, Table, C)$

$s_1 = DO(MOVE(B, Table, C), s_0)$

$On(A, Table, s_1)$ $Clear(A, s_1)$ $Clear(Table, s_1)$
$On(B, C, s_1)$ $Clear(B, s_1)$
$\neg On(B, Table, s_1)$ $\neg Clear(C, s_1)$
$On(C, Table, s_1)$

Planning in the blocks world.



Initial state (s0) **s1** **s2**

$$s_1 = DO(MOVE(B, Table, C), s_0)$$

$$On(A, Table, s_1)$$
$$On(B, C, s_1)$$
$$\neg On(B, Table, s_1)$$
$$On(C, Table, s_1)$$

$$Clear(A, s_1)$$
$$Clear(B, s_1)$$
$$\neg Clear(C, s_1)$$

$$Clear(Table, s_1)$$

Action: $MOVE(A, Table, B)$

$$s_2 = DO(MOVE(A, Table, B), s_1)$$
$$= DO(MOVE(A, Table, B), DO(MOVE(B, Table, C), s_0))$$

$$On(A, B, s_2)$$
$$On(B, C, s_2)$$
$$On(C, Table, s_2)$$

$$\neg On(A, Table, s_2)$$
$$\neg On(B, Table, s_2)$$
$$Clear(A, s_2)$$

$$\neg Clear(B, s_2)$$
$$\neg Clear(C, s_2)$$
$$Clear(Table, s_2)$$

Planning in situation calculus.

Planning problem:

- find a sequence of actions that lead to a goal
- Planning in situation calculus is converted to theorem proving.
- **Problems:**
  - Large search space
  - Large number of axioms to be defined for one action
  - Proof may not lead to the best (shortest) plan.

Frame problem

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

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)*

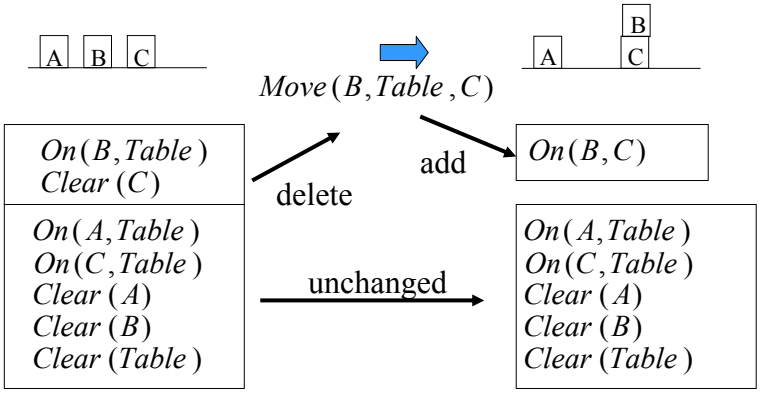
(Everything else is unaffected)

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

Application of operators

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)*



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:

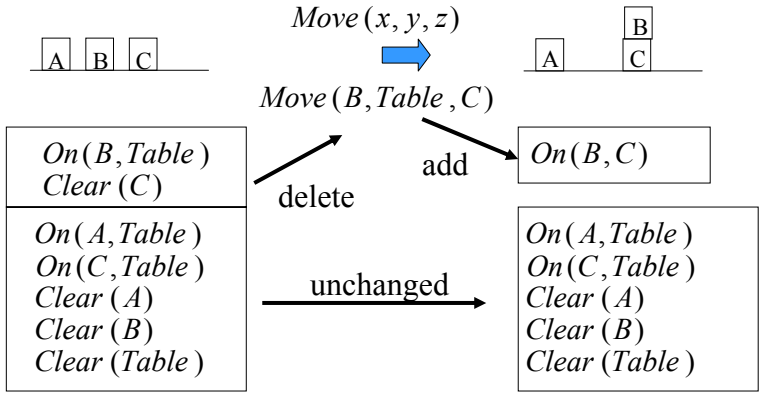
- **Forward search** (goal progression)
  - From the initial state try to reach the goal
- **Backward search** (goal regression)
  - Start from the goal and try to project it to the initial state

More complex planning method:

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

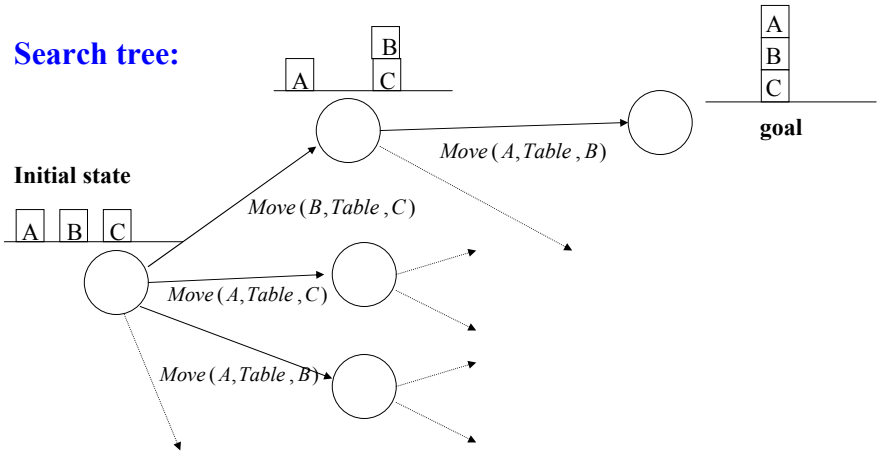
Forward search (goal progression)

- **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)

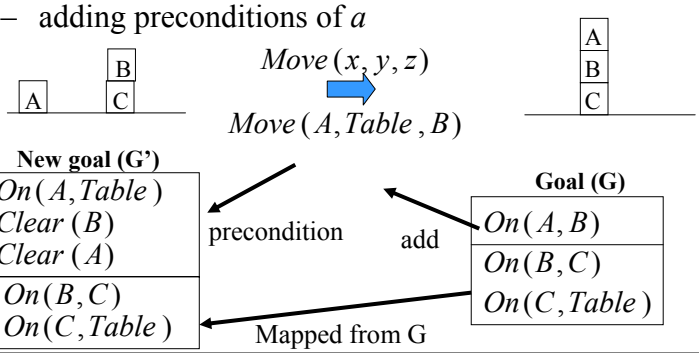
- Use operators to generate new states to explore
- Check new states whether they satisfy the goal



Backward search (goal regression)

Idea: Given a goal G

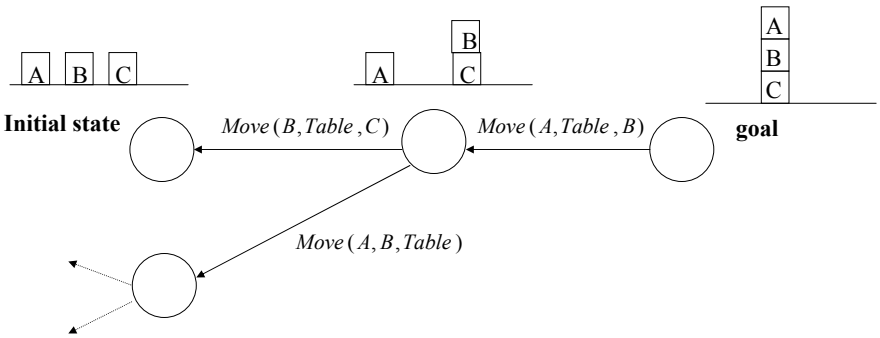
- Unify the add list of some operator a with a subset of G
- If the delete list of a does not remove elements of G, then the goal regresses to a new goal G' that is obtained from G by:
  - deleting add list of a
  - adding preconditions of a



Backward search (goal regression)

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

Search tree:



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

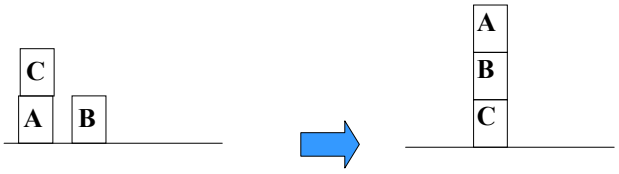
In planning we would like to satisfy a set of goals

- **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 divide and conquer fails
- Interacting goals

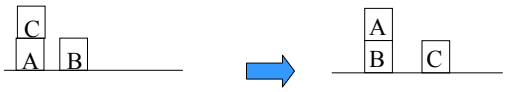


Initial state **Goal**

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

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 $On(A,B)$

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

Initial state

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

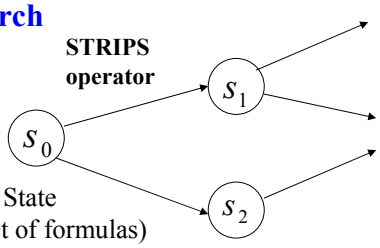
State space vs. plan space

- 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 plans:**
  - plans that are not complete
  - Some orderings of operators are not finalized

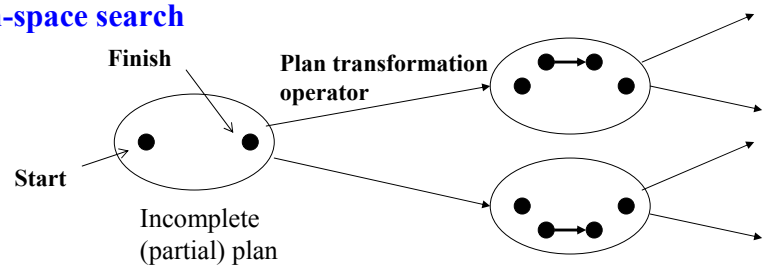
- **State-space vs Plan-space search:**
  - State-space search – a node is a configuration of the world
  - Plan-space search – a node is a partial plan

State-space vs. plan-space search

State-space search

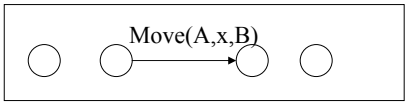


Plan-space search

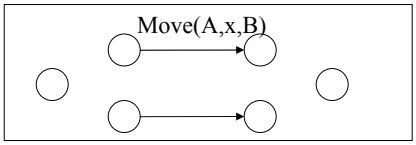


Plan transformation operators

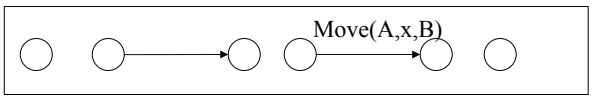
Examples:



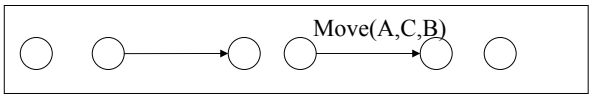
- **Add an operator to a plan**



- **Order (reorder) operators**



- **Instantiate an operator**

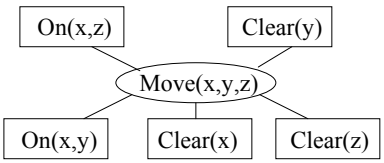


Partial-order planners (POP)

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

Illustration of POP on Sussman's anomaly case

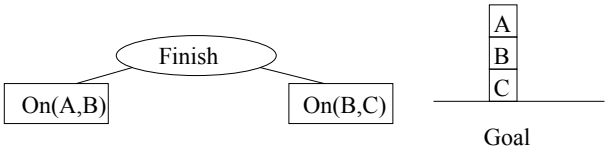
Graphical representation of an operator

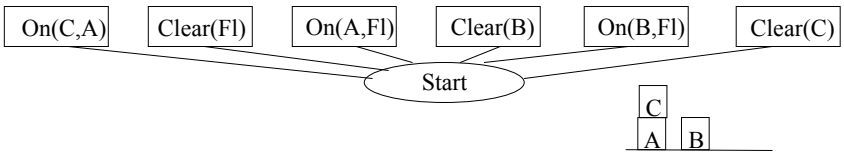


add list

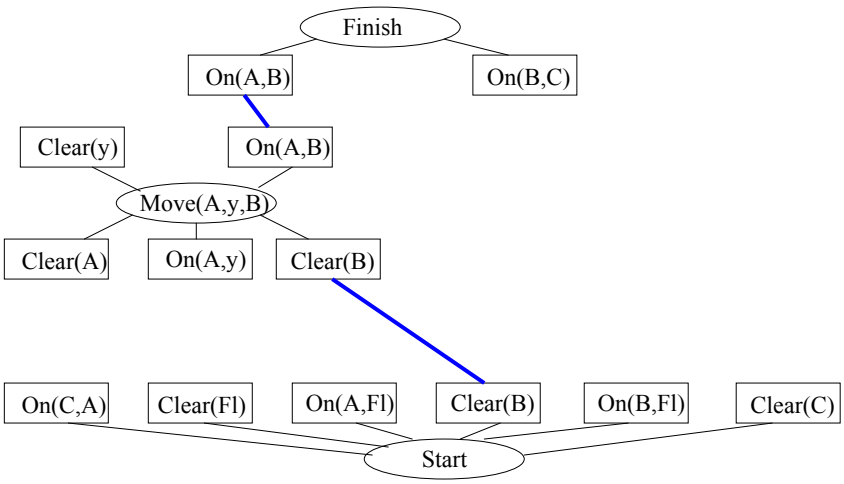
preconditions

Partial order planning. Start and finish.

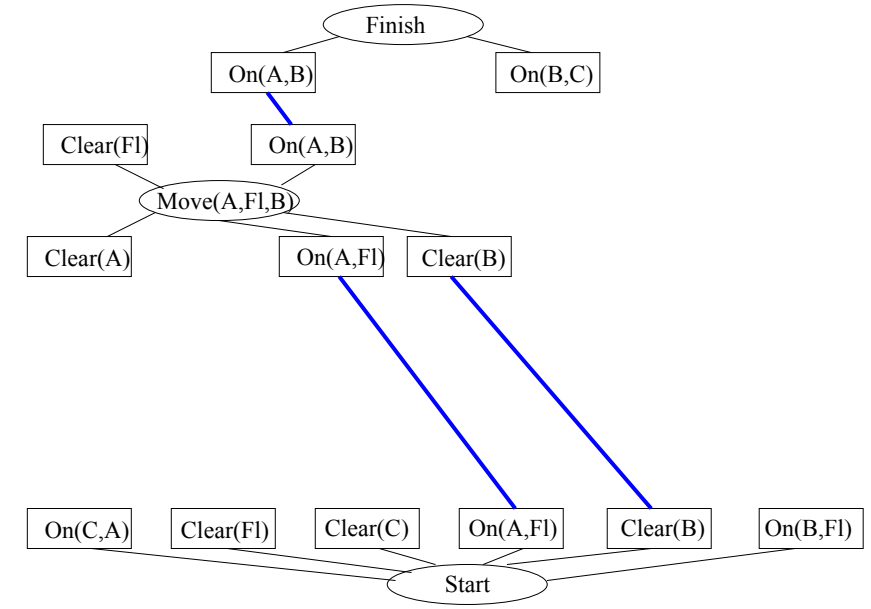




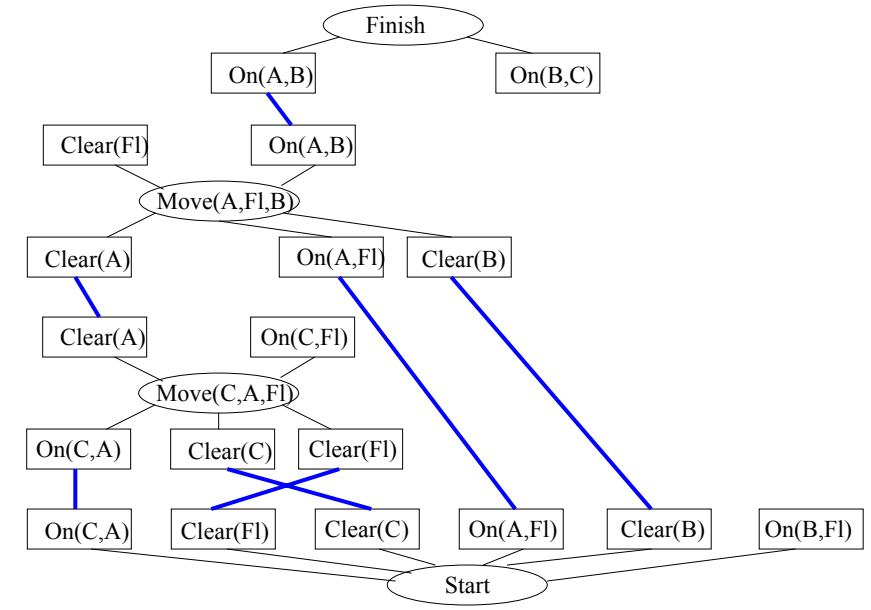
Partial order planning.



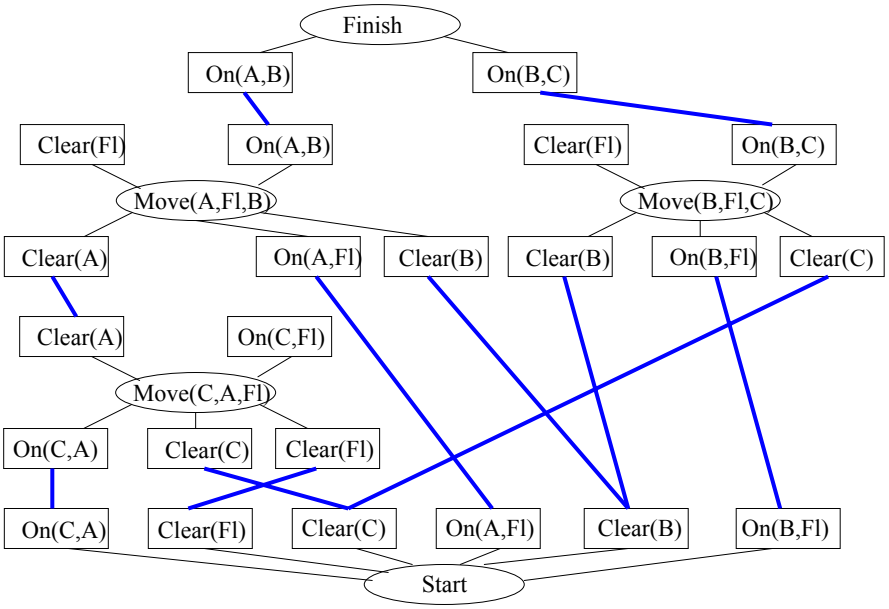
Partial order planning.



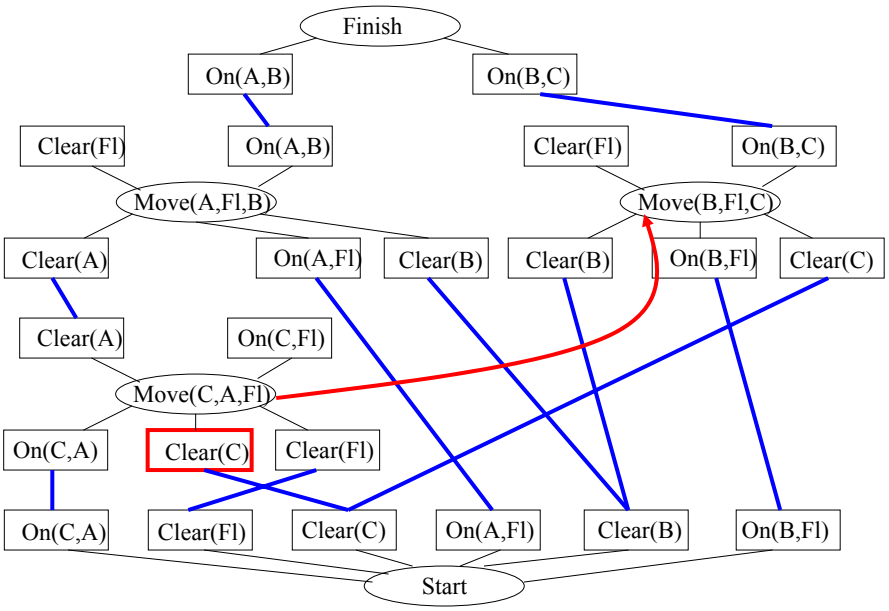
Partial order planning.



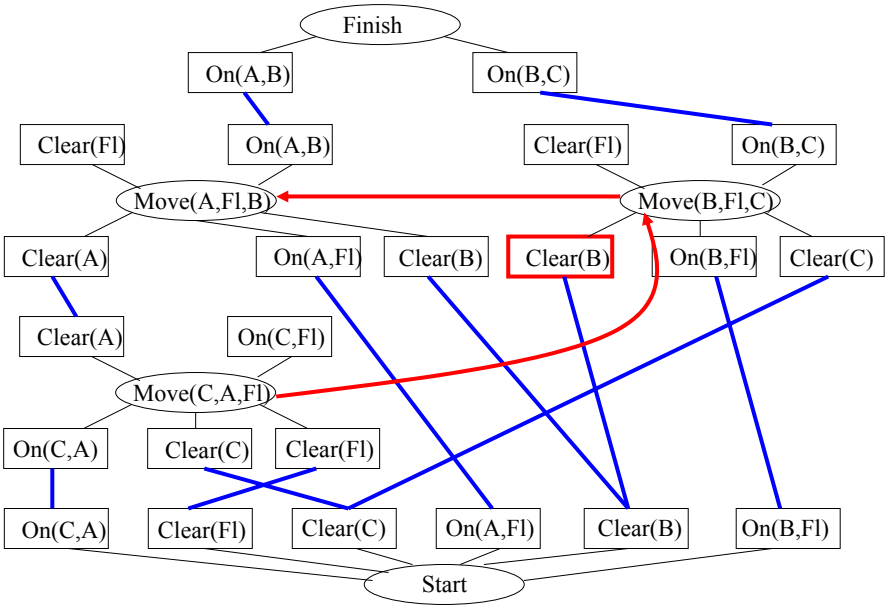
Partial order planning.



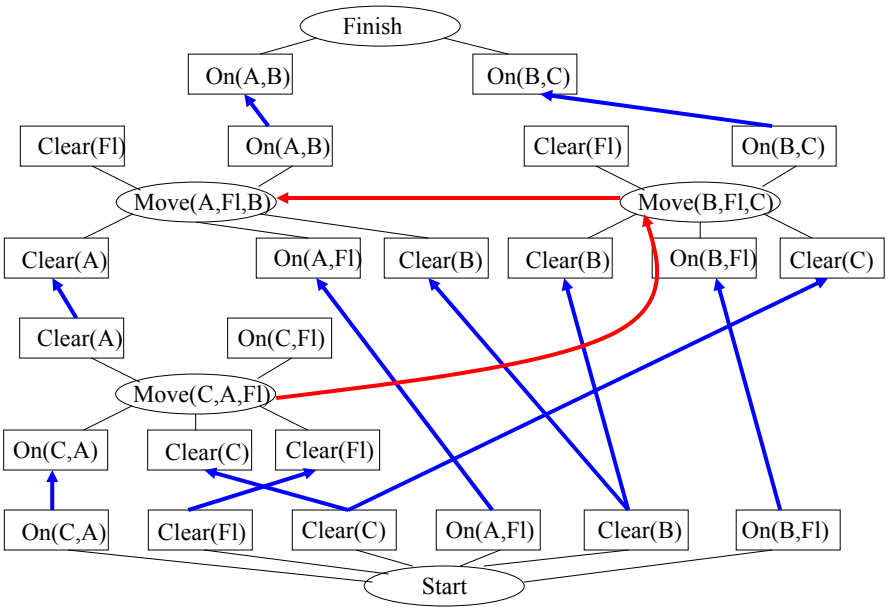
Partial order planning. Threats.



Partial order planning. Threats.

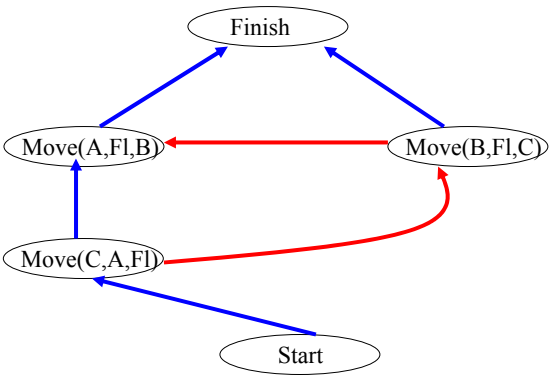


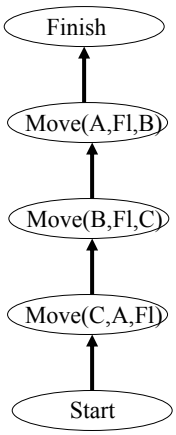
Partial order planning. Directions.



Partial order planning. Result plan.

Plan: a topological sort of a graph





Partial order planning.

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