

Planning

R&N Chap. 11

(and a tiny snippet of Chap. 8-9)

Limitations of Prop. Logic

- Not very expressive: To represent the fact that a flight can originate from any of n airports, we need n symbols:
FlyFromPITT, FlyFromSFO,
FlyFromORC,...
- Instead we would like to use more expressive sentences like:
For any airport x , FlyFrom(x)
→ First order logic (FOL)

FOL (The *extremely* short version!!)

- Same as before, plus:
 - Quantifiers: \forall, \exists
 - Variables: x, y, z
 - Predicates: $P(x,y)$ = logical expression with value True/False
 - Functions: $F(x)$

$\forall x, y, z \text{ Parent}(z, x) \wedge \text{Parent}(z, y) \Rightarrow \text{Sibling}(x, y)$

FOL

- **Substitution:** Replace a part of the sentence by another one.

$\text{SUBST}(\{x/\text{John}\}, \text{Rich}(x)) \rightarrow \text{Rich}(\text{John})$

- **Unification:** Find parts of two sentences that are identical after some substitution

$\text{UNIFY}(\text{SameCountry}(F(x), y), \text{SameCountry}(\text{John}, \text{Mary})) = \{F(x)/\text{John}, y/\text{Mary}\}$

FOL Inference: Resolution

- Resolution: Resolution can be extended to FOL, but more complicated

$$\frac{l_1 \vee l_2 \quad m_1 \vee m_2}{\text{SUBST}(\theta, l_1 \vee m_1)}$$

After some substitution, l_2 and $\neg m_2$ are the same

$$\theta = \text{UNIFY}(l_2, \neg m_2)$$

$$\frac{\text{UnHappy}(x) \vee \neg \text{Rich}(x) \quad \text{Rich}(\text{John})}{\text{UnHappy}(\text{John})}$$

$$\theta = \{x / \text{John}\}$$

FOL Inference: Chaining

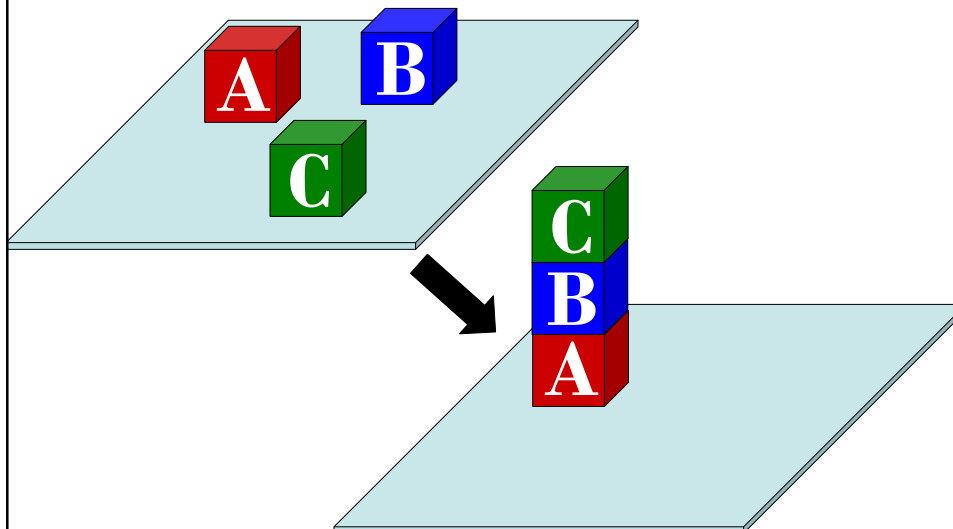
- Chaining: Forward/backward chaining idea can be extended to KBs with sentences of the form:
 - $A_1 \wedge A_2 \wedge A_3 \dots \Rightarrow B$

$$\text{WindowsLocked}(x) \wedge \text{DoorLocked}(x) \Rightarrow \text{RoomSecure}(x)$$

Summary

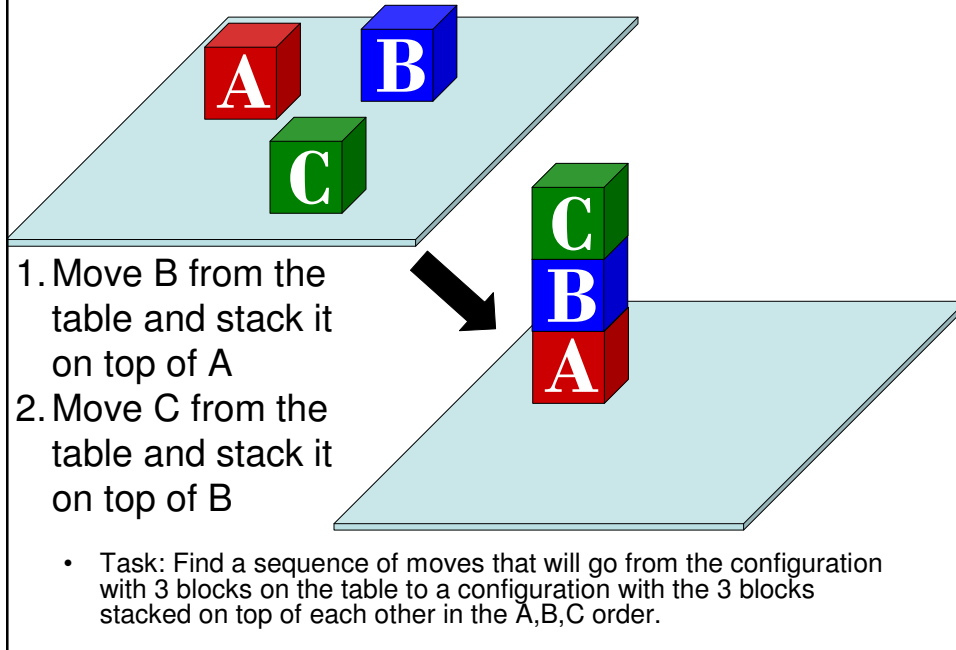
- FOL provides more compact way of representing KBs
- CNF, resolution and forward/backward chaining concepts exists in FOL
- Properties of soundness, completeness

A Simple Task



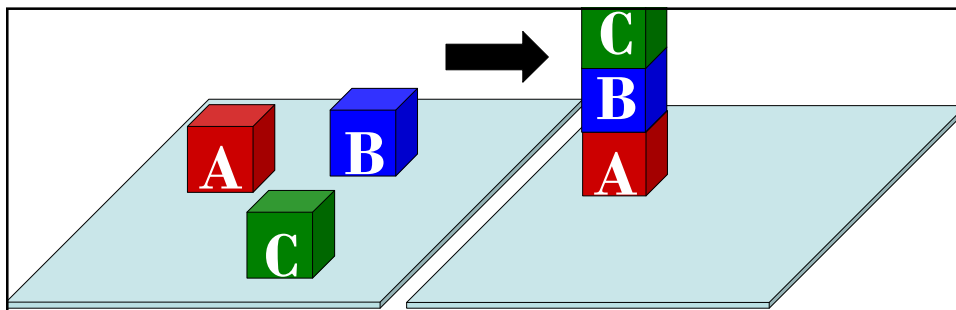
- Task: Find a sequence of moves that will go from the configuration with 3 blocks on the table to a configuration with the 3 blocks stacked on top of each other in the A,B,C order.

A Simple Task



1. Move B from the table and stack it on top of A
2. Move C from the table and stack it on top of B

- Task: Find a sequence of moves that will go from the configuration with 3 blocks on the table to a configuration with the 3 blocks stacked on top of each other in the A,B,C order.

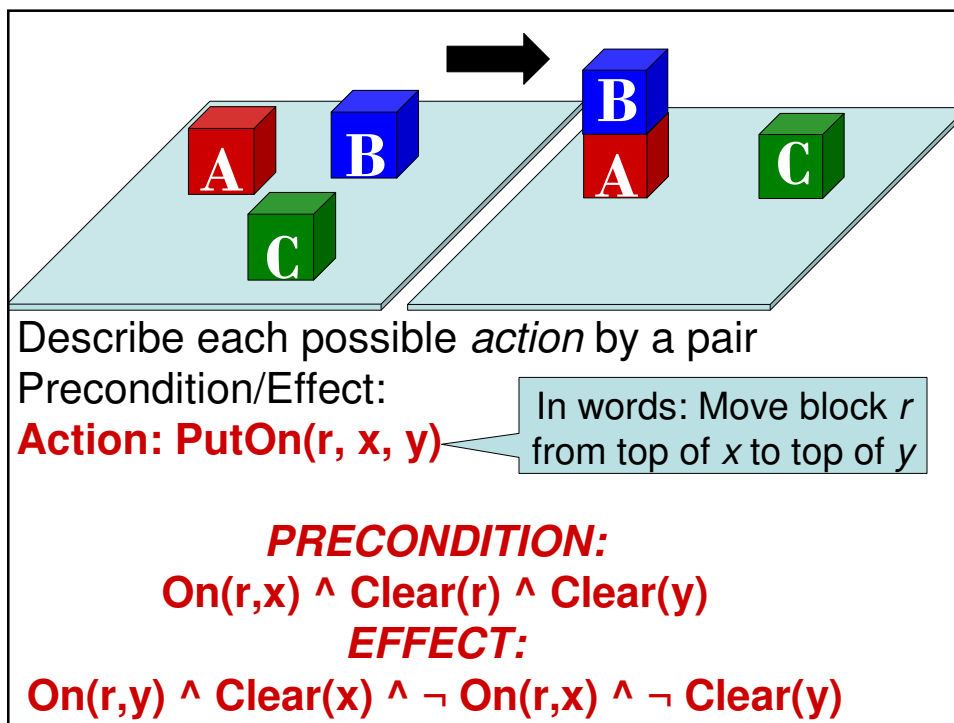
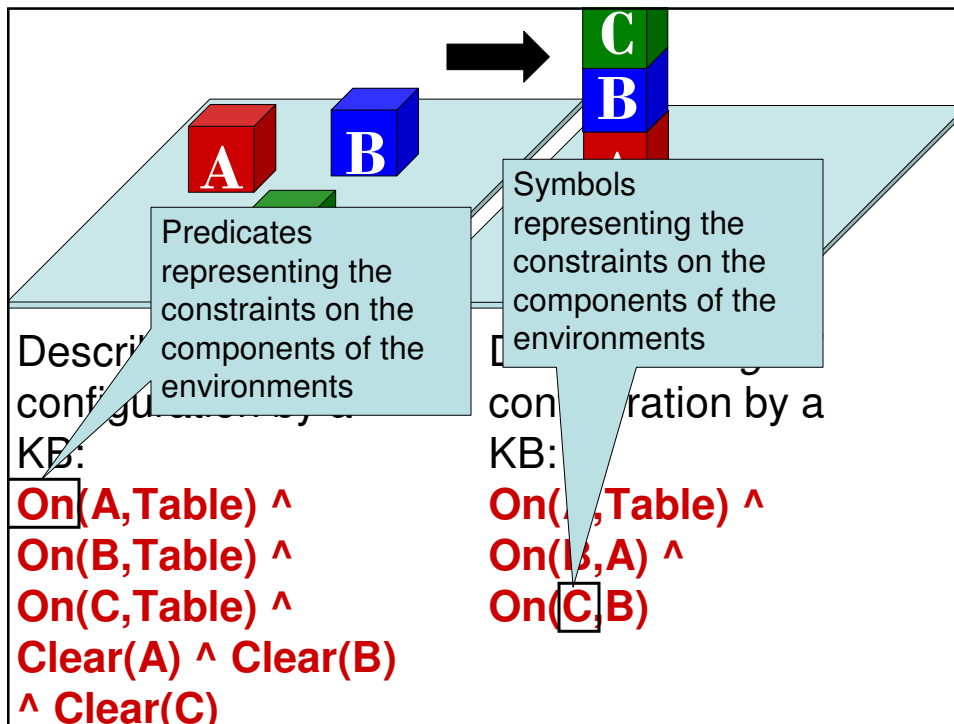


Describe the *starting* configuration by a KB:

**On(A,Table) ^
On(B,Table) ^
On(C,Table) ^
Clear(A) ^ Clear(B)
^ Clear(C)**

Describe the *goal* configuration by a KB:

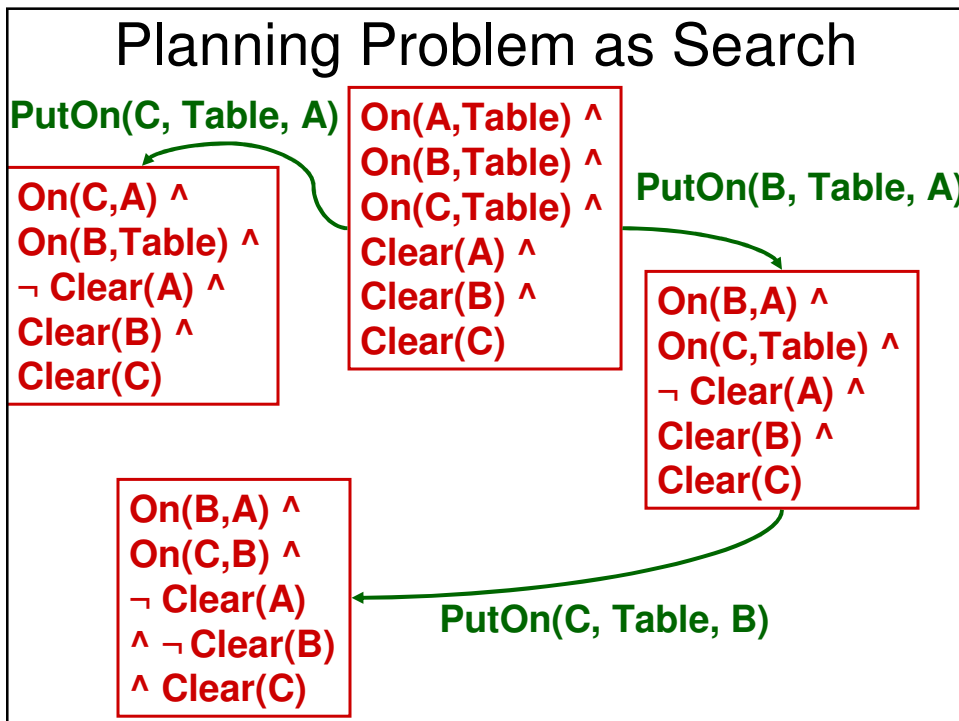
**On(A,Table) ^
On(B,A) ^
On(C,B)**



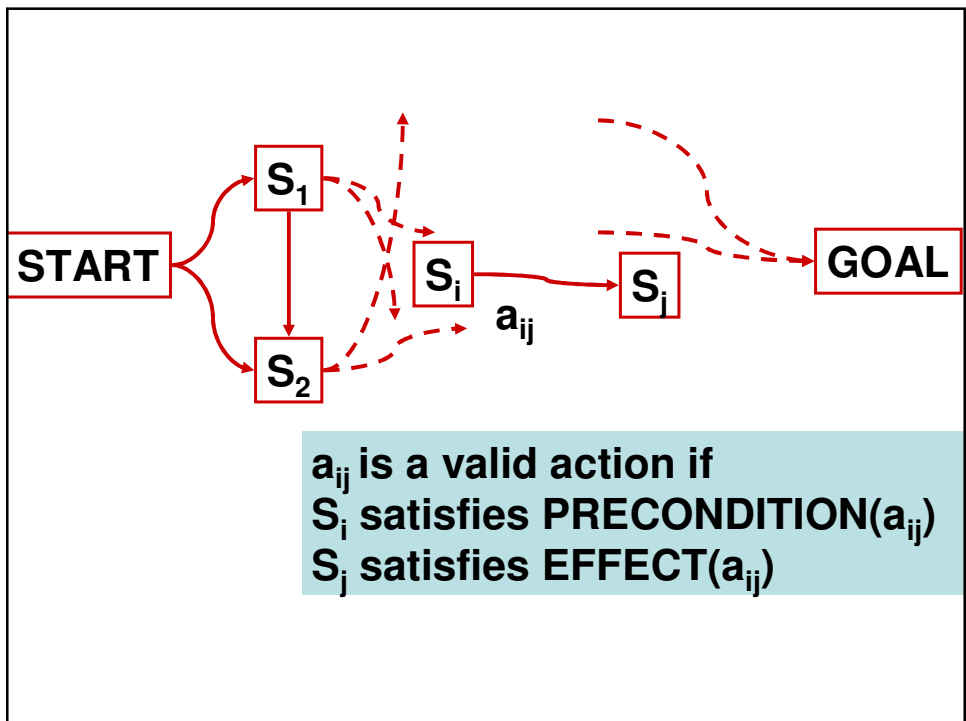
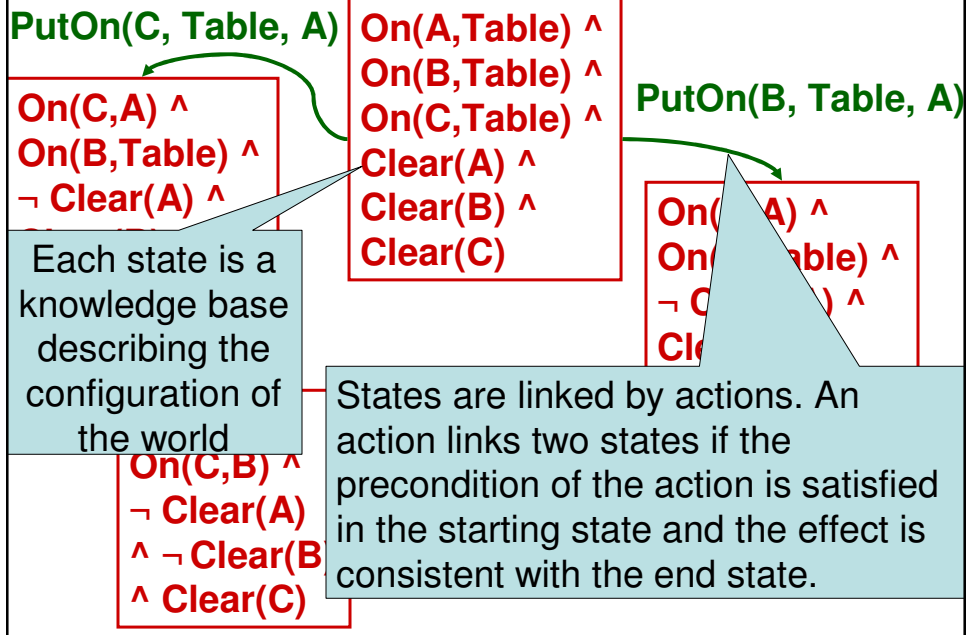
Describe each possible *action* by a pair
Precondition/Effect:

Action: PutOnTable(*r*, *x*) In words: Move block *r* from top of *x* to table

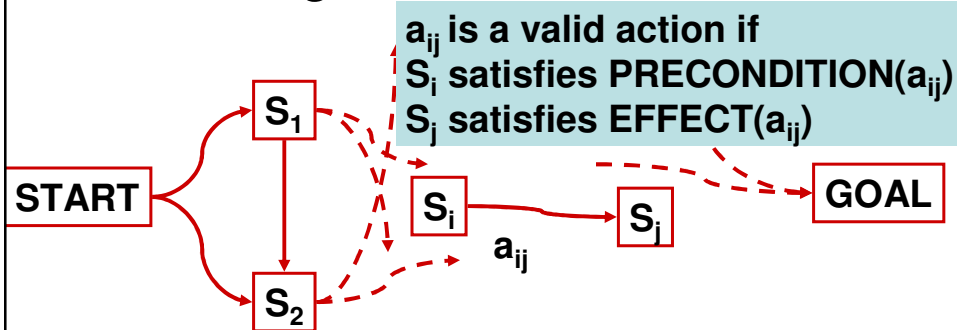
PRECONDITION: $\text{On}(r,x) \wedge \text{Clear}(r)$
EFFECT: $\text{On}(r,\text{Table}) \wedge \text{Clear}(x) \wedge \neg \text{On}(r,x)$



Planning Problem as Search



Planning Problem as Search



- *States*: KBs representing the possible configurations of the world
- *Arcs*: actions allowed between states
- Any of the previous search techniques can be used for planning in this graph (defined implicitly)
- *Forward planning*: Search from the start configuration until the goal configuration is reached
- *Backward planning*: Search backward from the goal configuration until the start configuration is reached

Notation

PutOn(r, x, y)

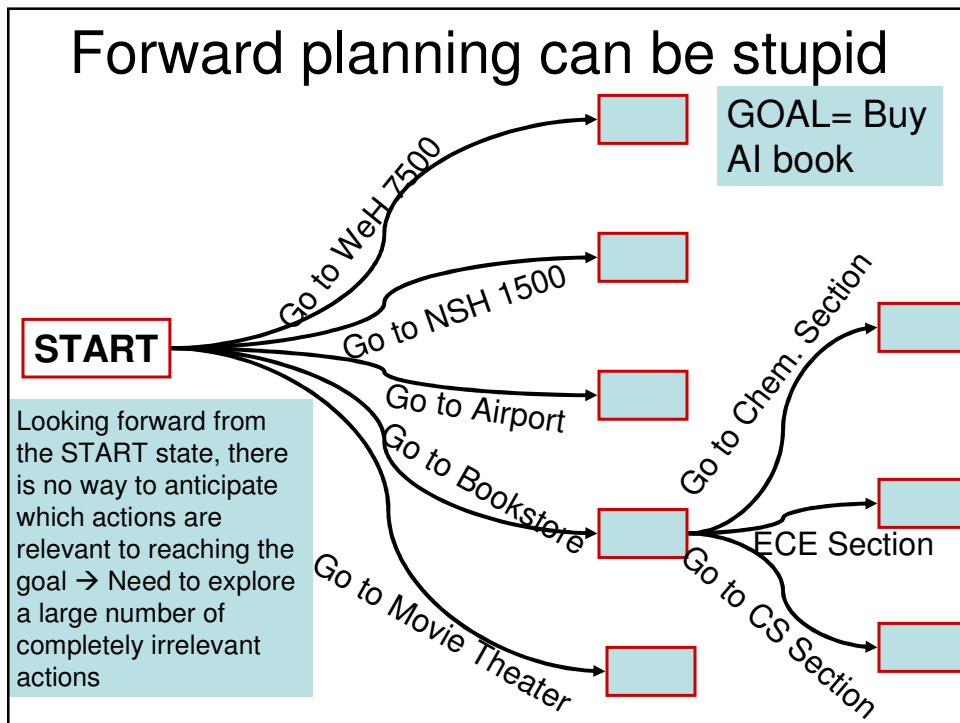
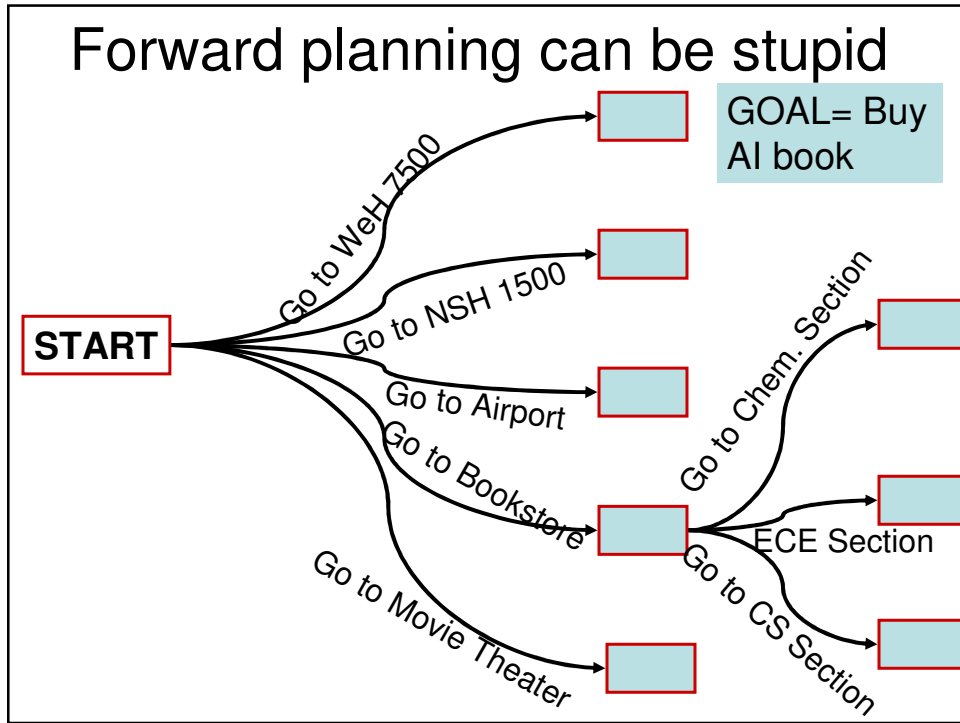
PRECONDITION:

$\text{On}(r,x) \wedge \text{Clear}(r) \wedge \text{Clear}(y)$

EFFECT:

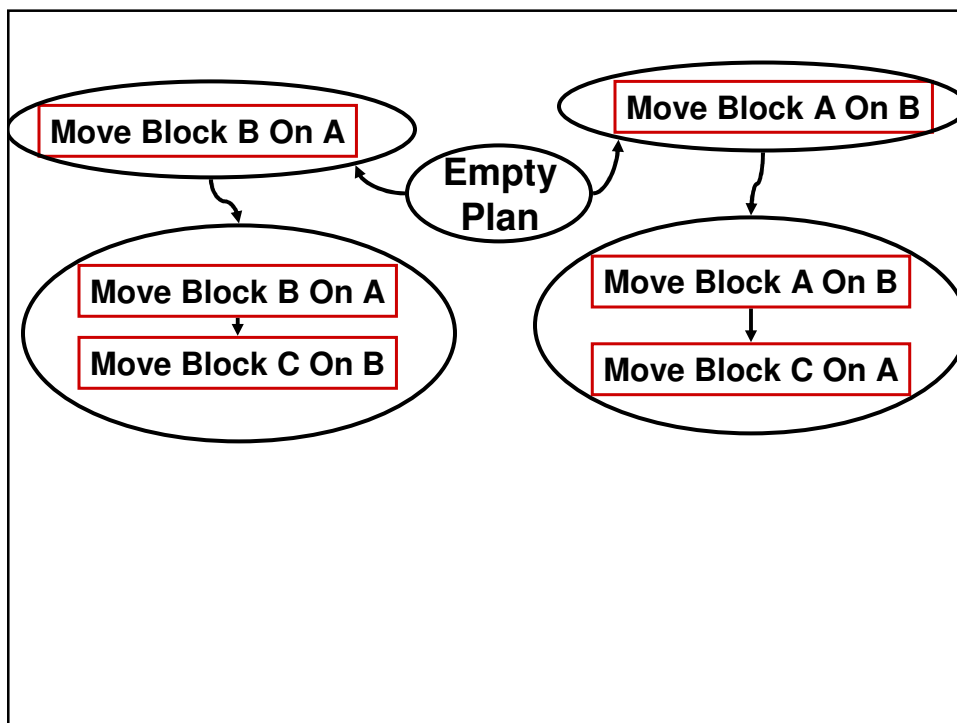
$\text{On}(r,y) \wedge \text{Clear}(x) \wedge \neg \text{On}(r,x) \wedge \neg \text{Clear}(y)$

- Describing the actions is actually quite tricky.
- *Frame problem*: Should we represent the effect of “PutOn” on the other variables? If we do, we need to enumerate explicitly all of the variables in the world!
- *One solution*: It is implicitly assumed that any symbol not mentioned in the EFFECT remains untouched.
- The particular notation used here (which uses this approach) is called the **STRIPS** notation (named after a famous Stanford system.)

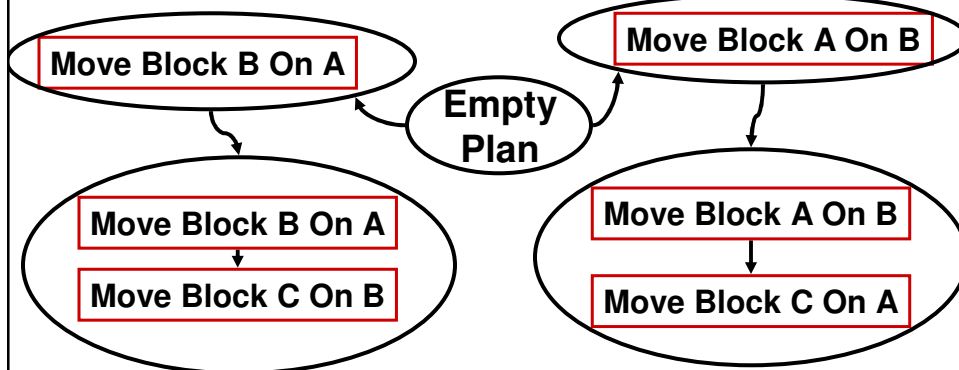


Heuristics

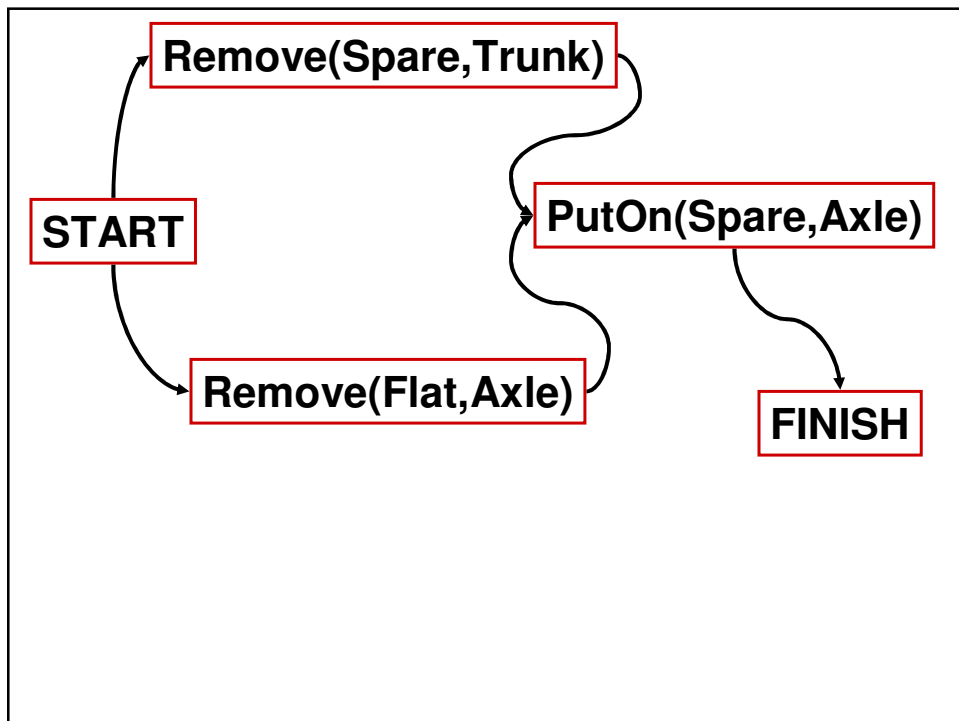
- Search is in general inefficient. Can we use heuristics to speed up the search?
- General heuristics: Try to guess a lower bound on the number of actions necessary to achieve the goal.
- Example (relaxed problems): First assume that the actions have no preconditions and find a set of the actions leading to the goal configurations (easier problem) → Provides a lower bound on the number of actions to reach the goal
- A* and related search techniques can be used to take advantage of heuristics.



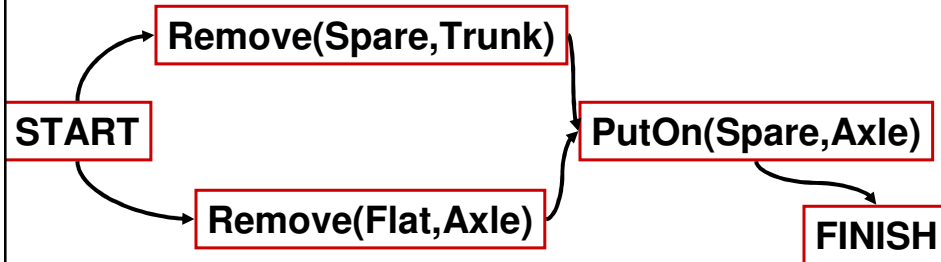
Another way to look at planning



- Instead of searching through the graph of possible world states linked by actions, we could do the opposite: Search through the set of possible plans = (informally) sequences of actions
- In fact, in many cases we can find *partial plans* that can be combined into a *complete plan* → (hopefully) more efficient search
- Formally: *Partial-Order Planning (POP)*



Example



- The nodes are now actions instead of world states
- START and FINISH are dummy nodes
- Two nodes A and A' are linked if the effect of A is a precondition for A'
- The actions are partially ordered: Some actions must occur before others
- Important: We don't need a single, totally ordered, sequence of actions

POP Algorithm

Partial plan is:

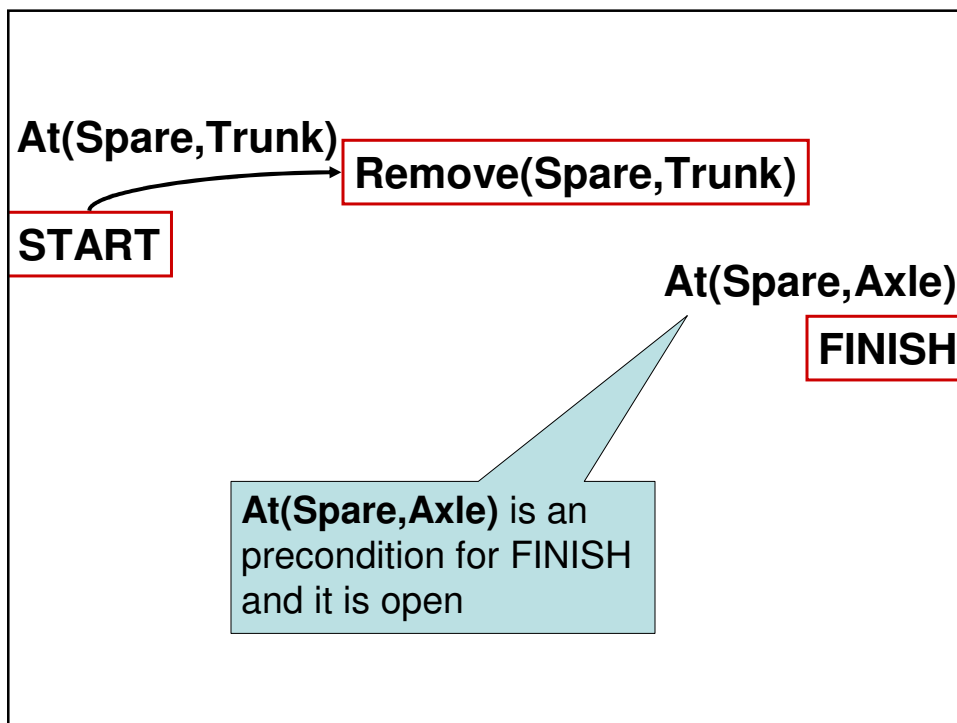
- Set of actions included in the plan
 - Example: Remove(Flat,Axle))
- Set of ordering constraints: $A < B$ means “action A must occur before action B”
- Set of links: $A \rightarrow_c B$
 - C is an effect of A
 - C is a precondition of B
 - Example:

$\text{Remove(Spare,Trunk)} \rightarrow_{\text{At(Spare,Ground)}} \text{PutOn(Spare,Axle)}$

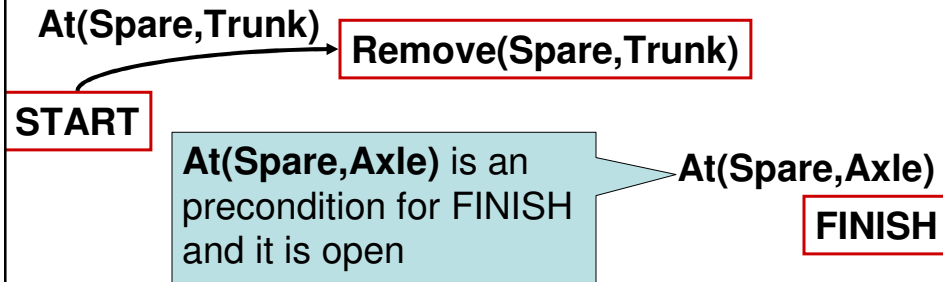
POP Algorithm

Two dummy nodes:

- **START:**
 - Precondition = None
 - Effect = Initial configuration of the world
- **FINISH:**
 - Precondition = Goal configuration of the world
 - Effect = None
- Initial plan contains only START and FINISH with the ordering START < FINISH



POP Algorithm



- *Open preconditions* = Precondition of an action in the plan that is not an effect of another action in the plan
- The plan is incomplete as long as there are open preconditions

POP Algorithm

- Initialize with {START, FINISH} nodes
- Repeat:
 - Find an open precondition C of an action B in the plan
 - Find an action A such that the effect of A meets the precondition C and add:
 - $A \rightarrow_c B$
 - $A < B$ (A must take place before B)
 - Verify that the plan is still consistent
- Until there are no open preconditions

START At(Spare,Trunk)
At(Flat,Axle)

At(Spare,Axle)

FINISH

Initialize with two actions:
START with two effects
FINISH with one precondition

START At(Spare,Trunk)
At(Flat,Axle)

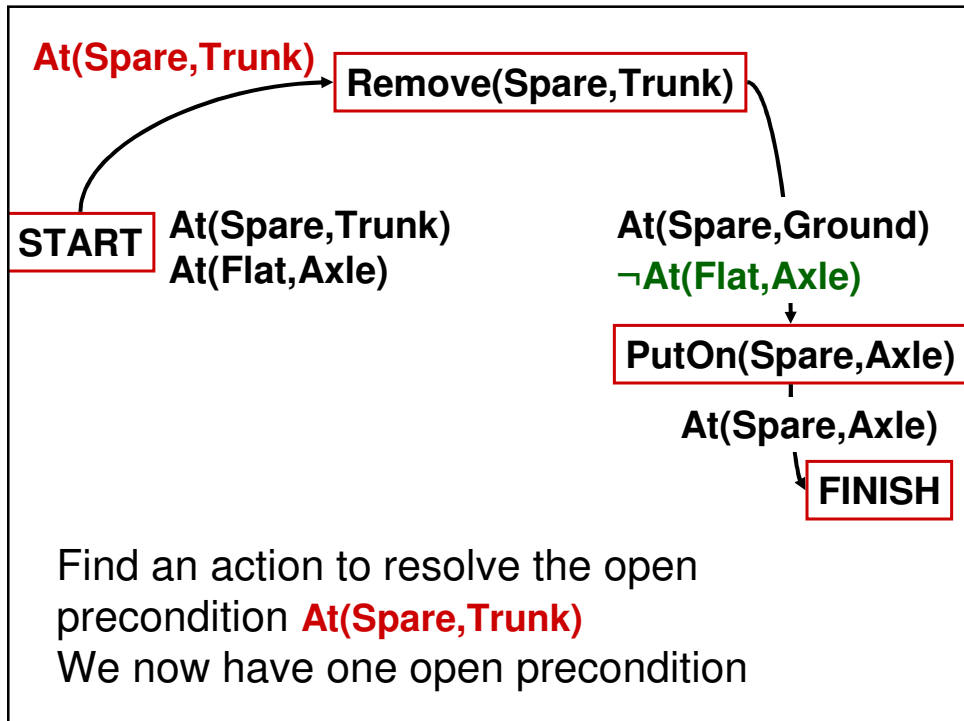
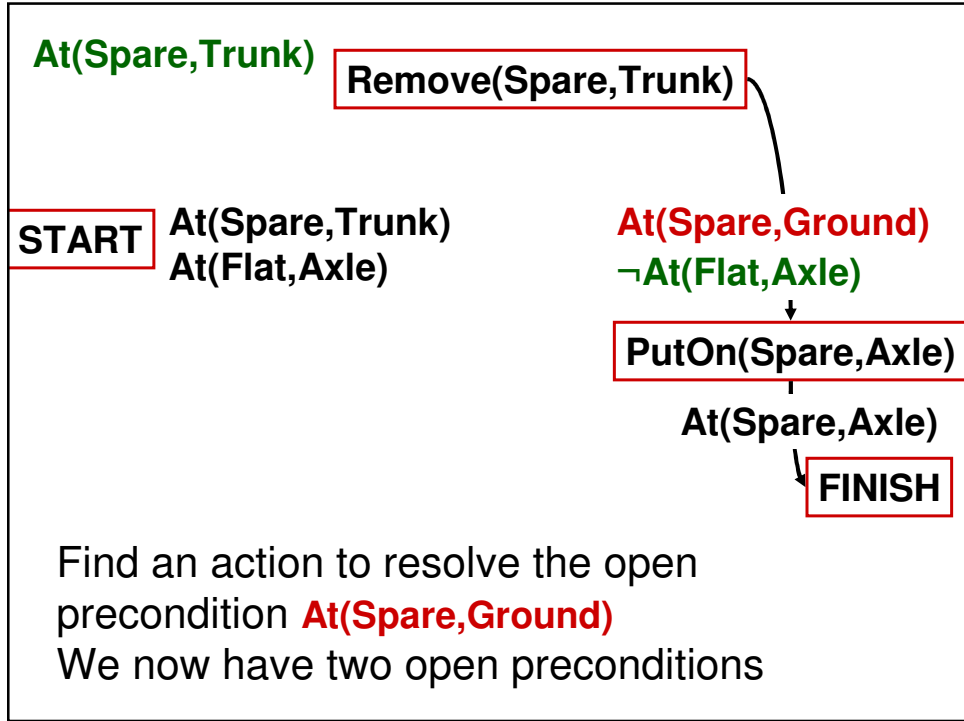
At(Spare,Ground)
 \neg At(Flat,Axle)

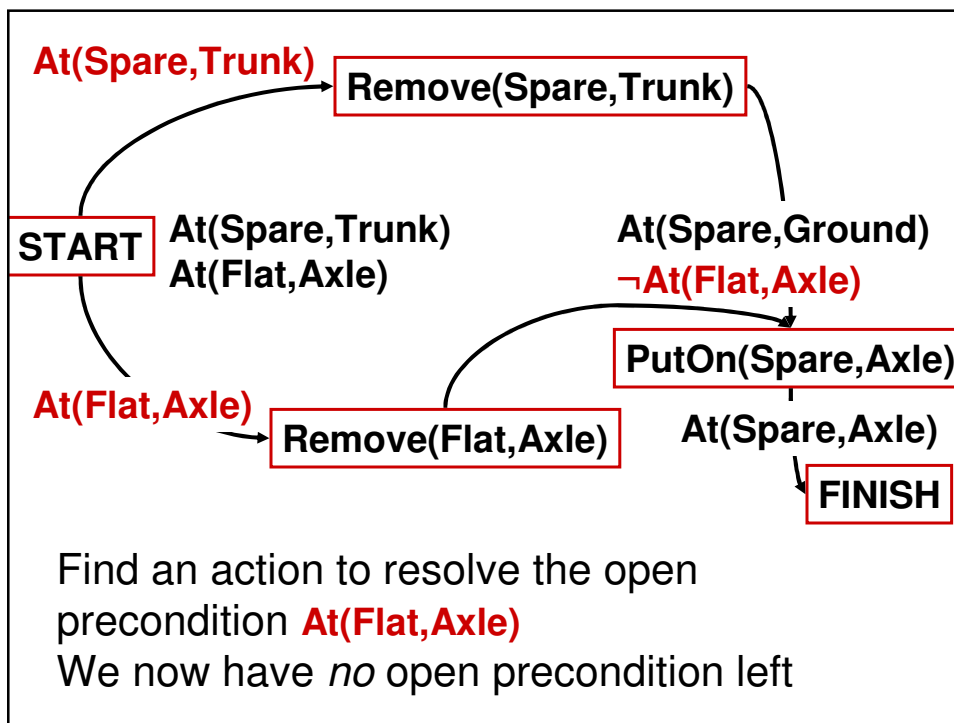
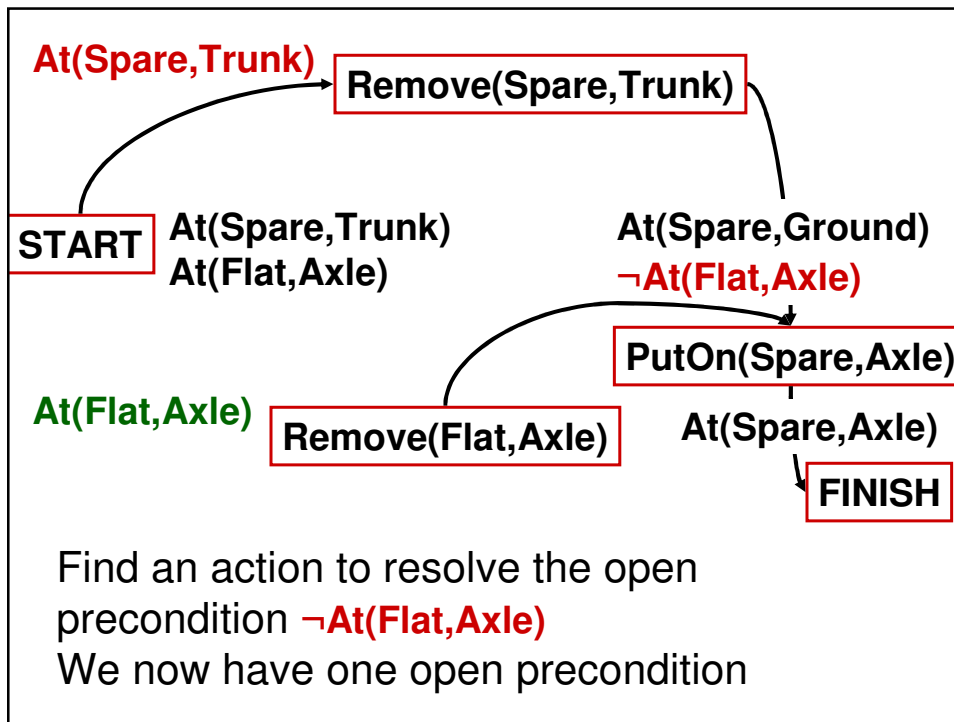
PutOn(Spare,Axle)

At(Spare,Axle)

FINISH

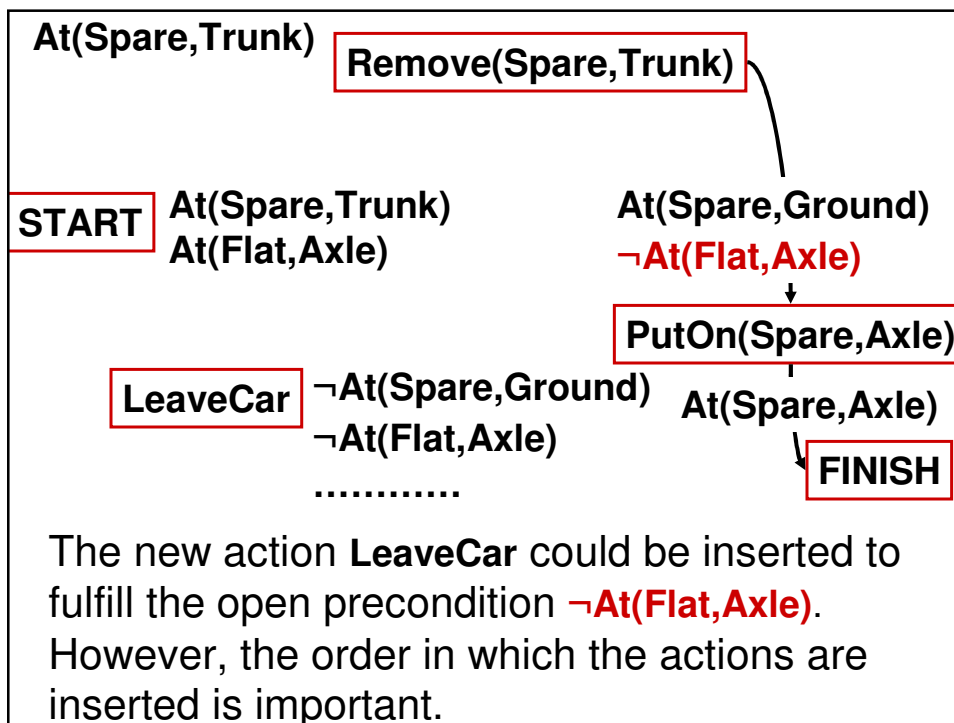
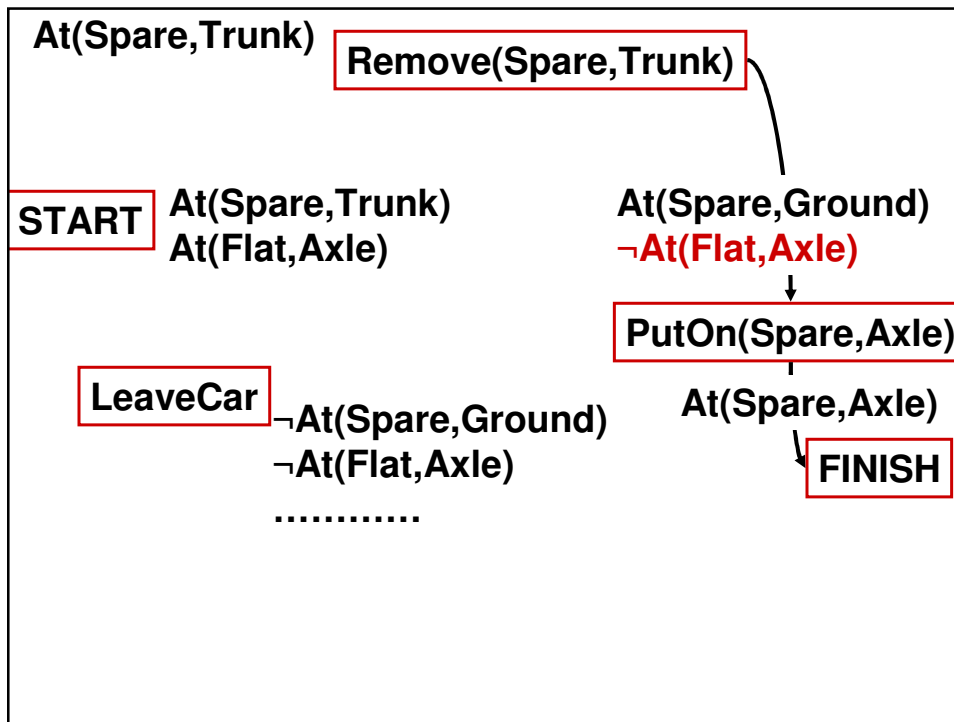
Find an action to resolve the open
precondition **At(Spare,Axle)**
We now have two more open preconditions

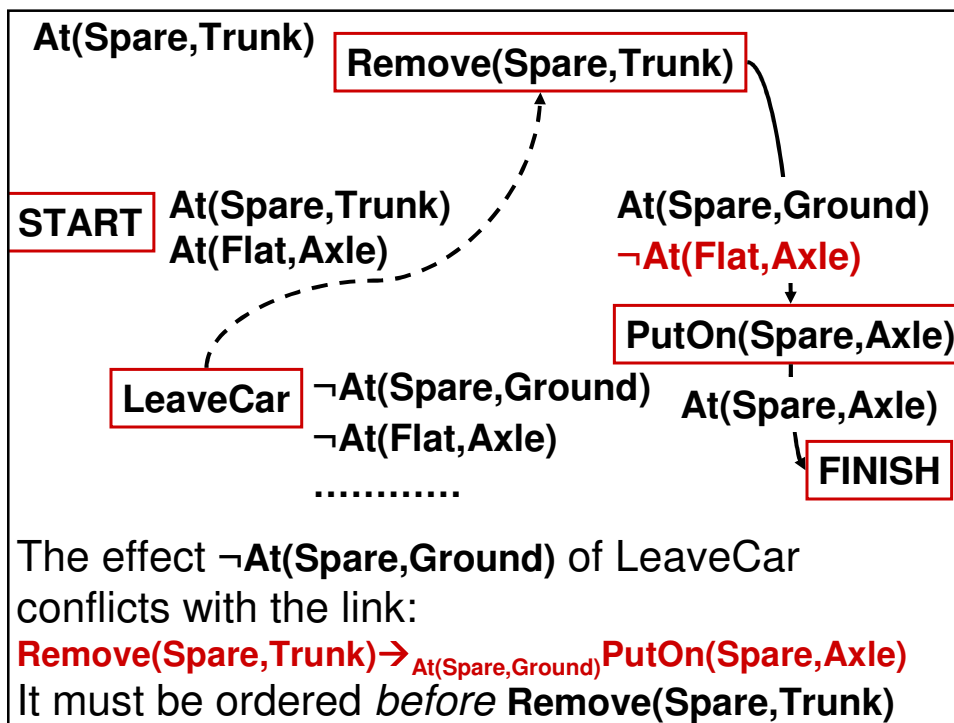
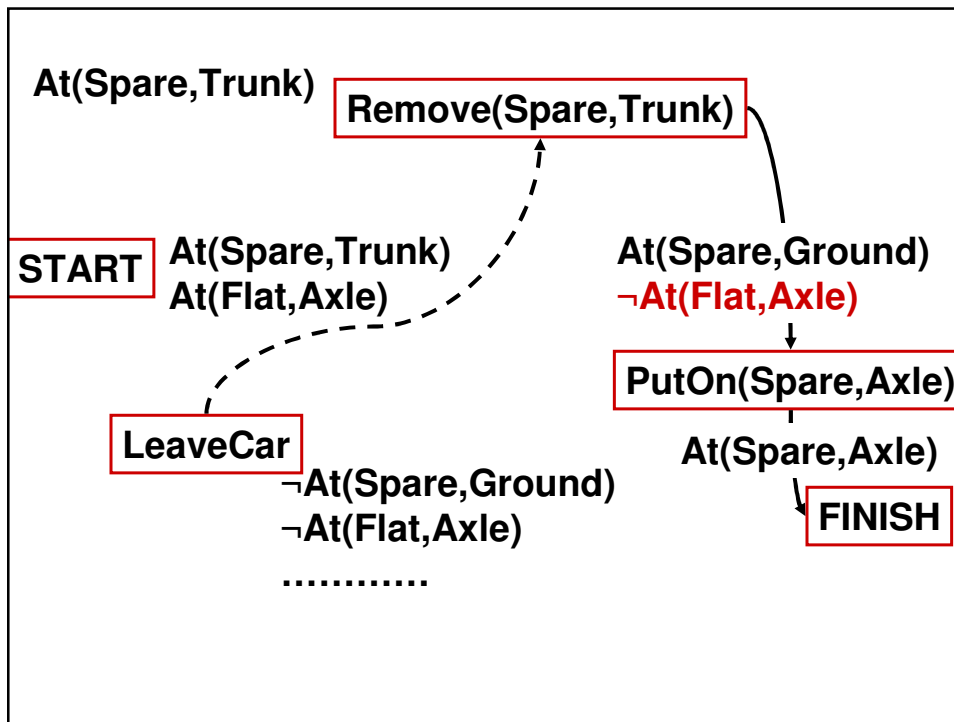




- Initialize with {START, FINISH} nodes
- Repeat:
 - Find an open precondition C of an action B in the plan
 - Find an action A such that the effect of A meets the precondition C and add:
 - $A \rightarrow_c B$
 - $A < B$ (A must take place before B)
 - Verify that the plan is still consistent
- Until there are no open preconditions

- Consistency:
 - If an existing action E in the plan conflicts with $A \rightarrow_c B$:
 - Try to add the ordering constraint $E > B$ or $A > E$
 - If no consistent ordering can be found \rightarrow Give up on adding A
 - $A \rightarrow_c B$
 - $A < B$ (A must take place before B)
 - Verify that the plan is still consistent
- Until there are no open preconditions





The POP Algorithm

- POP is particularly effective when the problem can be decomposed into subproblems → More flexibility in the search because we do not require a strictly ordered sequence of actions.
- POP is sound
- POP is complete (e.g., with breadth first search or iterative deepening)

Summary

- Configuration of the world = KB
- Actions = Preconditions + Effect
- STRIPS notation
- Planning = Find set of actions from start to goal configurations of the world
- Planning as search through the valid world configurations linked by valid actions between configurations
 - Backward search generally more effective
 - All the arsenal of heuristic search can be used
- Partial-Order Planning (POP): Planning as search through the possible plans.
 - Construct partial plans, combined by taking into account ordering constraints.
 - Takes advantages of decomposable sub-plans and sub-goals

Summary

- Configuration of the world = KB
- Actions = Preconditions + Effect
- STRIPS notation
- Planning = Find set of actions from start to goal configurations of the world
- Planning as search through the valid world configurations linked by valid actions between configurations

What is potentially a serious limitation in trying to use logic-based representations for reasoning and planning in real-world scenarios?

- Backward search generates plans, combined by taking into account ordering constraints.
- Takes advantages of decomposable sub-plans and sub-goals