Planning

Lecture 2

Outline

- ♢ Planning in the plan space
- ♢ Partial-Order Planning
- ♢ RN 10.4.4, (Second edition 11.3)

Skipped

- ♢ GraphPlan (forward planning + Heuristics)
- ♢ Planning as constraint satisfaction
- ♢ Planning as propositional satisfiability

New approach to planning

Principle of **least committment**:

♢ partial ordering (instead of total)

♢ not fully instantiated plan (in the first-order case)

Change of **problem representation**:

♢ state space: node = state in the world

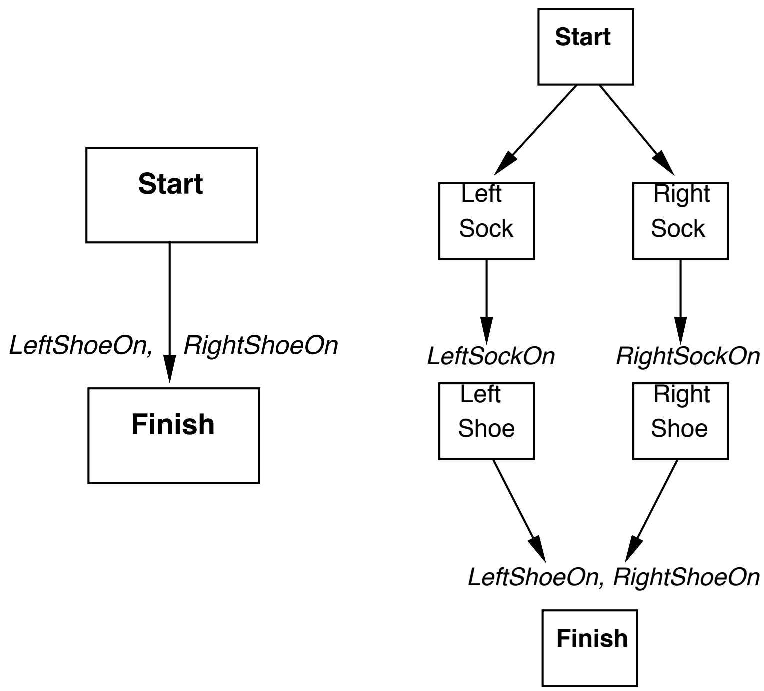
♢ plan space: node = partial plan

Dressing up

GOAL: $\{\}$
GOAL: $\{RightShoeOn, LeftShoeOn\}$

ACTION: $RightSock$, EFFECT: $RightSockOn$)
ACTION: $LeftSock$, EFFECT: $LeftSockOn$
ACTION: $RightShoe$, PRECONDITION: $RightSockOn$,
EFFECT: $RightShoeOn$
ACTION: $LeftShoe$, PRECONDITION: $LeftSockOn$,
EFFECT: $LeftShoeOn$

Example


Start
LeftShoeOn, RightShoeOn
Finish
Start
Left Sock
Right Sock
LeftSockOn
RightSockOn
Left Shoe
Right Shoe
LeftShoeOn, RightShoeOn
Finish

Partially ordered plans

Partially ordered collection of actions with

◊ $Start$ action has the initial state description as its effect

◊ $Finish$ action has the goal description as its precondition

◊ temporal ordering between pairs of actions

Two *additional elements* are needed to characterize the planning process:

◊ Open precondition = precondition of an action not yet causally linked

◊ Causal links from outcome of one action to precondition of another

Plan Representation

- ♢ set of actions
- ♢ set of ordering constraints $A \prec B$
- ♢ set of causal links $A \xrightarrow{p} B$

  A achieves p for B
- ♢ set of open preconditions

Initial State:

$$
\begin{aligned}
&Plan(\textsc{Actions:}\{Start, Finish\},\\
&\qquad \textsc{Orderings:}\ \{Start \prec Finish\},\\
&\qquad \textsc{Links:}\ \{\},\\
&\qquad \textsc{Open Preconditions:}\ \{RightShoeOn, LeftShoeOn\})
\end{aligned}
$$

Solutions in the plan space

A plan is complete iff every precondition is achieved

A precondition is achieved iff:
it is the effect of an earlier action and no possibly intervening action undoes it

Plan Representation: solution

$Plan($ACTIONS:$\{RightSock, RightShoe, LeftSock, LeftShoe,$
$Start, Finish\},$
ORDERINGS: $\{Start \prec Finish, Start \prec RightSock,$
$RightSock \prec RightShoe, RightShoe \prec Finish,$
$Start \prec LeftSock, LeftSock \prec LeftShoe,$
$LeftShoe \prec Finish\},$
LINKS: $\{RightSock \xrightarrow{RightSockOn} RightShoe,$
$RightShoe \xrightarrow{RightShoeOn} Finish,$
$LeftSock \xrightarrow{LeftSockOn} LeftShoe,$
$LeftShoe \xrightarrow{LeftShoeOn} Finish\},$
OPEN PRECONDITIONS: $\{\})$

Planning process as plan refinement

Refinements of partial plans:

- add a link from an existing action to an open condition
- add a action to fulfill an open condition
- order one action wrt another to remove possible conflicts

Gradually move from incomplete/vague plans to complete, correct plans

Backtrack if an open condition is unachievable or
if a conflict is unresolvable

The Search Procedure

1. The initial plan includes the constraints for $Start$ and $Finish$, with ordering $Start \prec Finish$;
2. The successor function
   (a) pick one open precondition p on action B
   (b) pick one action A that achieves p
   (c) add the causal link $A \xrightarrow{p} B$ and the ordering constraint $A \prec B$; if A is new add also $Start \prec A$ and $B \prec Finish$
   (d) resolve conflicts, if possible, otherwise backtrack
3. The goal test succeeds when there are no more open preconditions

Example

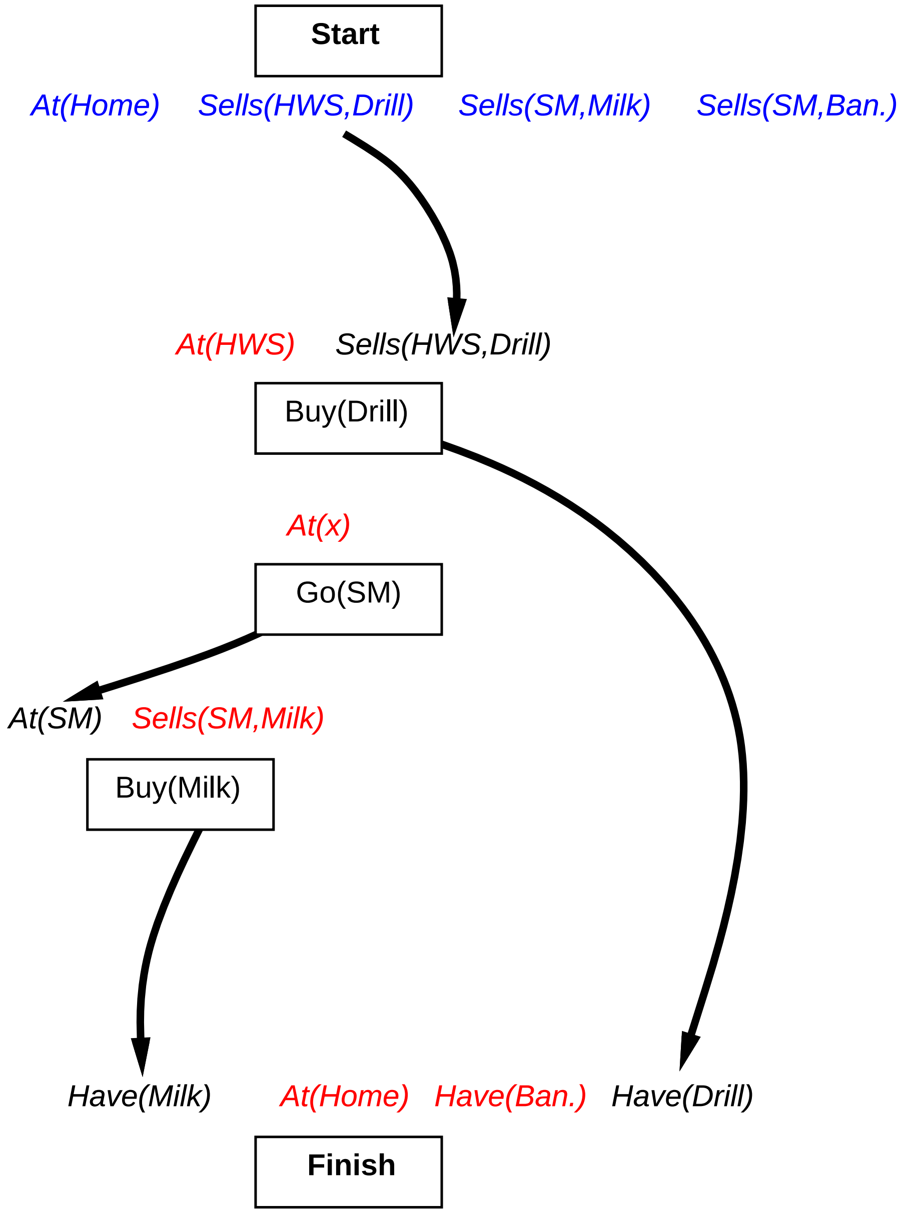

Start
At(Home) Sells(HWS,Drill) Sells(SM,Milk) Sells(SM,Ban.)
Have(Milk) At(Home) Have(Ban.) Have(Drill)
Finish

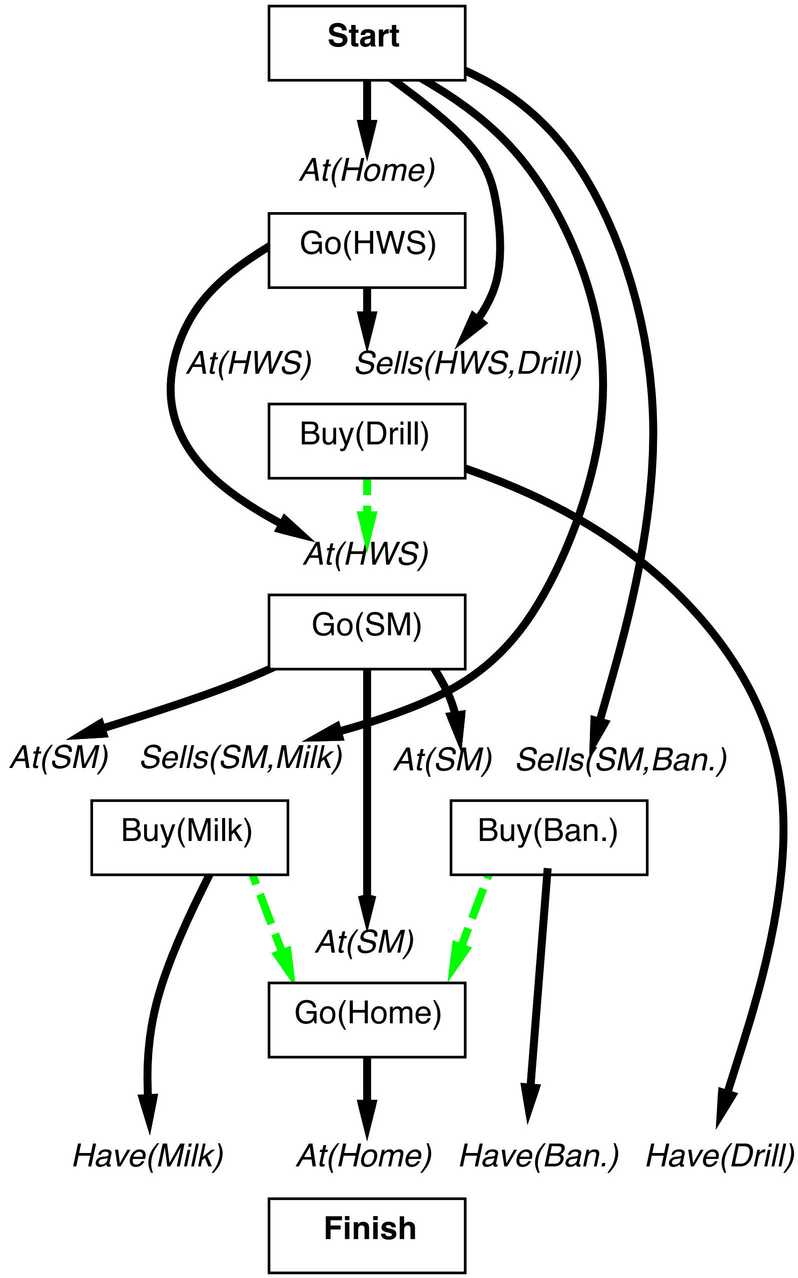
Actions for the example

ACTION: $Buy(x)$
PRECONDITION: $At(p), Sells(p, x)$
EFFECT: $Have(x)$

ACTION: $Go(x)$
PRECONDITION: $At(y)$
EFFECT: $At(x) \wedge \neg At(y)$

Objects: $Milk, Bananas, Drill, \ldots$
Places: $Home, SM, HWS, \ldots$


Start
At(Home)
Sells(HWS,Drill)
Sells(SM,Milk)
Sells(SM,Ban.)
At(HWS)
Sells(HWS,Drill)
Buy(Drill)
At(x)
Go(SM)
At(SM)
Sells(SM,Milk)
Buy(Milk)
Have(Milk)
At(Home)
Have(Ban.)
Have(Drill)
Finish


Start
At(Home)
Go(HWS)
At(HWS)
Sells(HWS,Drill)
Buy(Drill)
At(HWS)
Go(SM)
At(SM)
Sells(SM,Milk)
At(SM)
Sells(SM,Ban.)
Buy(Milk)
Buy(Ban.)
At(SM)
Go(Home)
Have(Milk)
At(Home)
Have(Ban.)
Have(Drill)
Finish

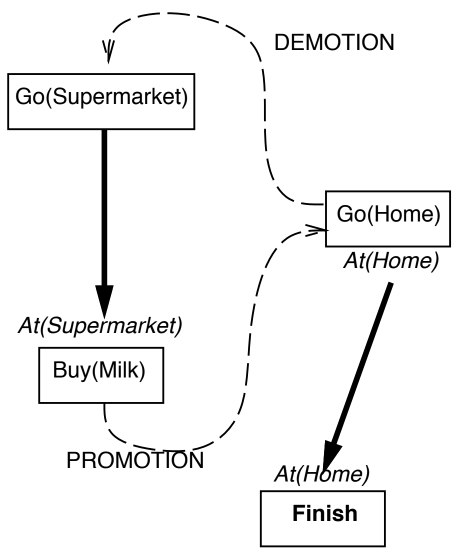
Clobbering and conflicts

A clobberer is a potentially intervening action that destroys the condition achieved by a causal link. E.g., $Go(Home)$ clobbers $At(Supermarket)$:

More specifically, a **conflict** between the causal link $A \xrightarrow{p} B$ and the action C holds when C has effect $\neg p$.

A conflict can be solved by adding:
♢ $C \prec A$ (**demotion**) or
♢ $B \prec C$ (**promotion**)

Promotion/demotion



Demotion: put before $Go(Supermarket)$

Promotion: put after $Buy(Milk)$

POP algorithm sketch

function POP(*initial, goal, operators*) **returns** *plan*

  plan ← MAKE-MINIMAL-PLAN(*initial, goal*)
  loop do
    if SOLUTION?(*plan*) **then return** *plan*
    S_{need}, c ← SELECT-OPENPRECONDITION(*plan*)
    CHOOSE-OPERATOR(*plan, operators*, S_{need}, c)
    RESOLVE-THREATS(*plan*)
  end

---

function SELECT-OPENPRECONDITION(*plan*) **returns** S_{need}, c

  pick a plan step S_{need} from ACTIONS(*plan*)
    with a precondition c that has not been achieved
  return S_{need}, c

POP algorithm contd.

procedure CHOOSE-OPERATOR($plan, operators, S_{need}, c$)

choose a step S_{add} from *operators* or ACTIONS($plan$) that has c as an effect
if there is no such step **then fail**
add the causal link $S_{add} \xrightarrow{c} S_{need}$ to LINKS($plan$)
add the ordering constraint $S_{add} \prec S_{need}$ to ORDERINGS($plan$)
if S_{add} is a newly added step from *operators* **then**
  add S_{add} to ACTIONS($plan$)
  add $Start \prec S_{add} \prec Finish$ to ORDERINGS($plan$)

Properties of POP

Nondeterministic algorithm: backtracks at choice points on failure:

– choice of action (S_{add}) to achieve open precondition (S_{need})
– choice of demotion or promotion for clobberer

Selection of open precondition (S_{need}) is irrevocable: the existence of a plan does not depend on the choice of the open preconditions.

POP is sound, and complete,

Termination? The plan space is infinite . . .

Flat tire

Action: $Remove(Spare, Trunk)$
Precondition: $At(Spare, Trunk)$
Effect: $\neg At(Spare, Trunk) \wedge At(Spare, Ground)$

Action: $Remove(Flat, Axle)$
Precondition: $At(Flat, Axle)$
Effect: $\neg At(Flat, Axle) \wedge At(Flat, Ground)$

Action: $PutOn(Spare, Axle)$
Precondition: $At(Spare, Ground) \wedge \neg At(Flat, Axle)$
Effect: $\neg At(Spare, Ground) \wedge At(Spare, Axle)$

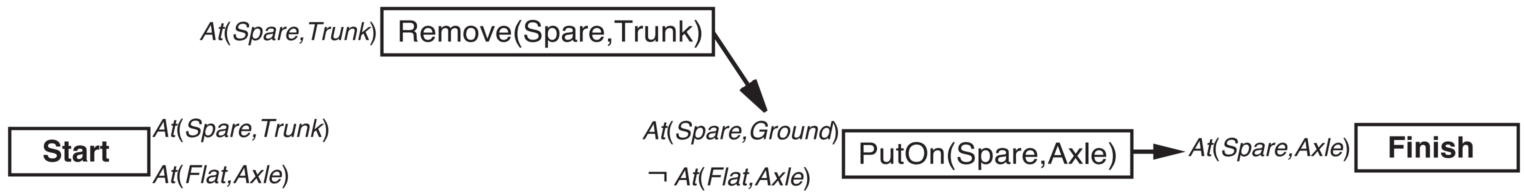
Action: $LeaveOvernight$ Precondition:
Effect: $\neg At(Spare, Ground) \wedge \neg At(Spare, Axle) \wedge$
$\neg At(Spare, Trunk) \wedge At(Flat, Ground) \wedge \neg At(Flat, Axle)$

Flat tire

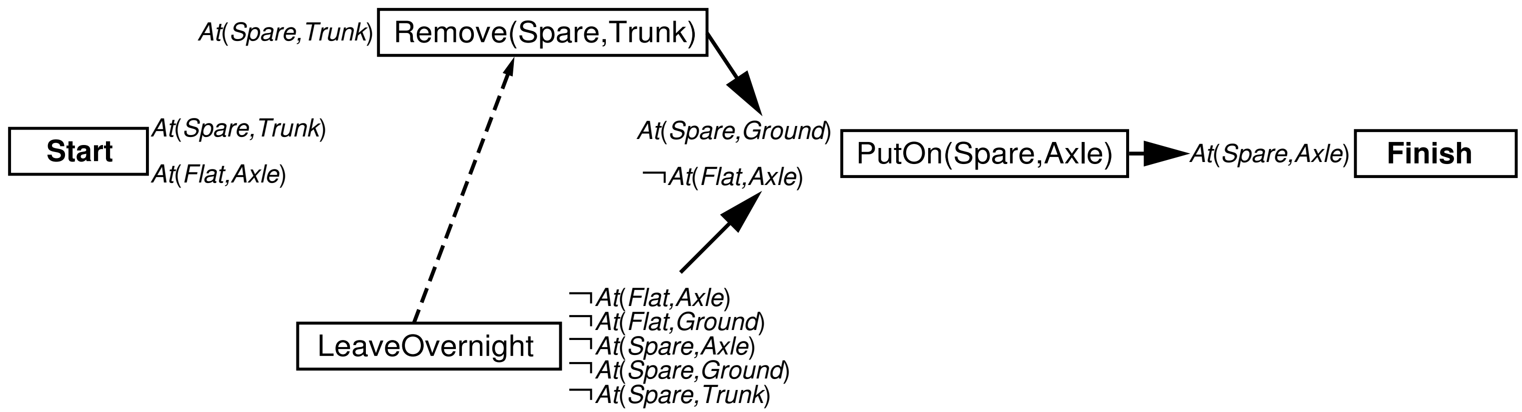
Init: $At(Flat, Axle) \wedge At(Spare, Trunk)$

Goal: $At(Spare, Axle)$

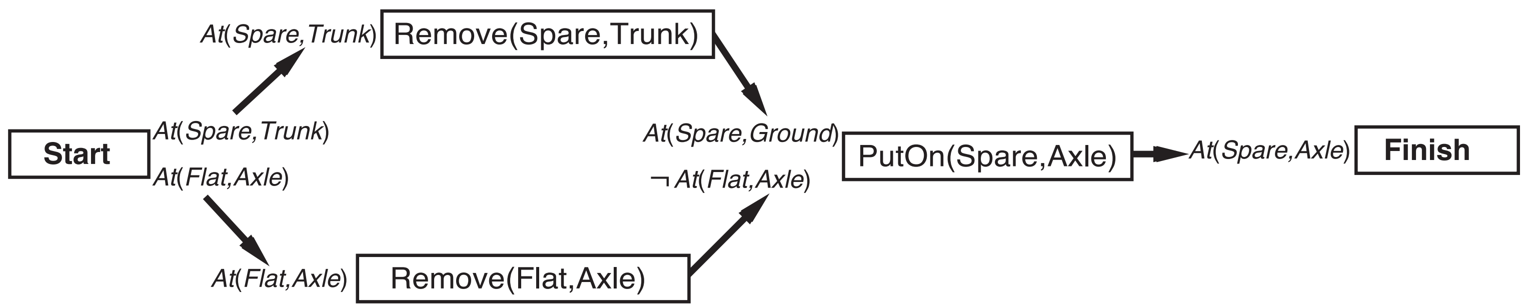
POP: Flat Tire



POP: Flat Tire



POP: Flat Tire



Extensions of POP

Handling variables: again principle of least commitment

ACTION: $Buy(x)$
PRECONDITION: $At(p), Sells(p, x)$
EFFECT: $Have(x)$

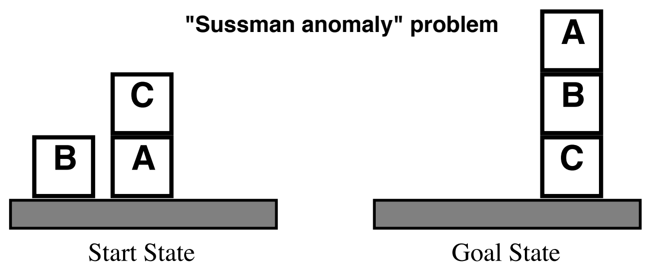
Achieving $Have(milk)$ leaves as open precondition:
$At(p), Sells(p, milk)$, which can be satisfied by any p

Equality and inequality constraints needed to handle variables

$\diamondsuit$ POP admits also extensions for disjunction, universals, negation, conditionals

Example: Blocks world

"Sussman anomaly" problem



Clear(x) On(x,z) Clear(y)	*Clear(x) On(x,z)*
PutOn(x,y)	PutOnTable(x)
~On(x,z) ~Clear(y) Clear(z) On(x,y)	*~On(x,z) Clear(z) On(x,Table)*

+ several inequality constraints

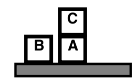
Actions in the blocks' world

$Op(PutOn(b, y),$
$\quad$ $\textsc{Precond}{:}On(b, z) \wedge Clear(b) \wedge Clear(y),$
$\quad$ $\textsc{Effect}{:}On(b, y) \wedge Clear(z) \wedge \neg On(b, z) \wedge \neg Clear(y))$

$Op(PutOnTable(b),$
$\quad$ $\textsc{Precond}{:}On(b, z) \wedge Clear(b),$
$\quad$ $\textsc{Effect}{:}On(b, Table) \wedge Clear(z) \wedge \neg On(b, z))$

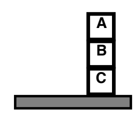
START

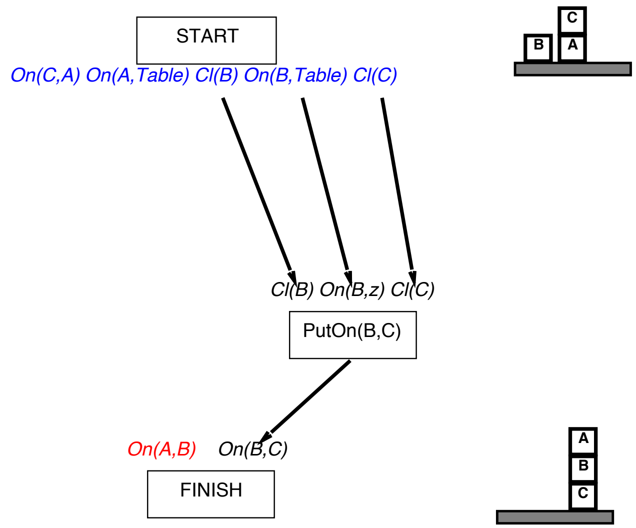
On(C,A) On(A,Table) Cl(B) On(B,Table) Cl(C)



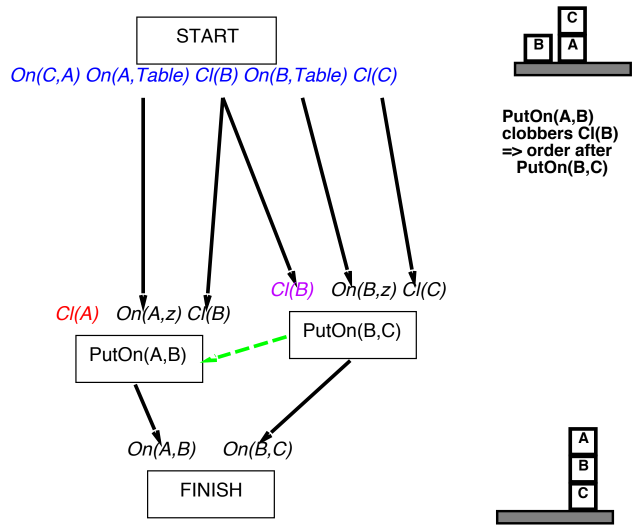
On(A,B) *On(B,C)*

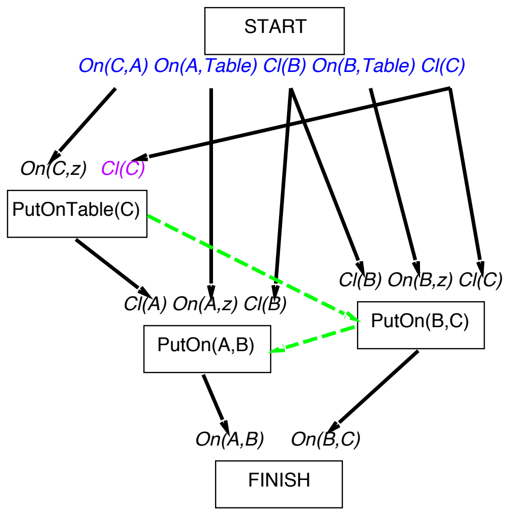
FINISH

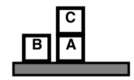




START
On(C,A) On(A,Table) Cl(B) On(B,Table) Cl(C)
Cl(B) On(B,z) Cl(C)
PutOn(B,C)
On(A,B) On(B,C)
FINISH
C
B A
A
B
C

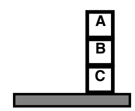

START
On(C,A) On(A,Table) Cl(B) On(B,Table) Cl(C)
Cl(A) On(A,z) Cl(B)
PutOn(A,B)
Cl(B) On(B,z) Cl(C)
PutOn(B,C)
On(A,B) On(B,C)
FINISH
C
B A
PutOn(A,B)
clobbers Cl(B)
=> order after
PutOn(B,C)
A
B
C





PutOn(A,B) clobbers Cl(B) => order after PutOn(B,C)

PutOn(B,C) clobbers Cl(C) => order after PutOnTable(C)



Heuristics for POP

General:

- ♢ number of open preconditions
- ♢ most constrained variable
  - open precond that are satisfied in fewest ways
- ♢ a special data structure: the **planning graph**

Problem Specific:

Good heuristics can be derived from problem description (by the human operator)

POP is particularly effective on problems with many loosely related subgoals

Summary

Advantages

- least commitment allows for flexible execution
- POP (sound and complete)
- very good for domains that require loose sequential constraints

Disadvantages

- infinite search space
- no representation of states
- planning is complex and difficult to devise heuristics