16-350
Planning Techniques for Robotics

Search Algorithms:
Planning on Symbolic Representations

Alex LaGrassa (Guest lecturer)

Robotics Institute

Carnegie Mellon University

Adapted from slides by Maxim Likhachev

Personal intro

About me:

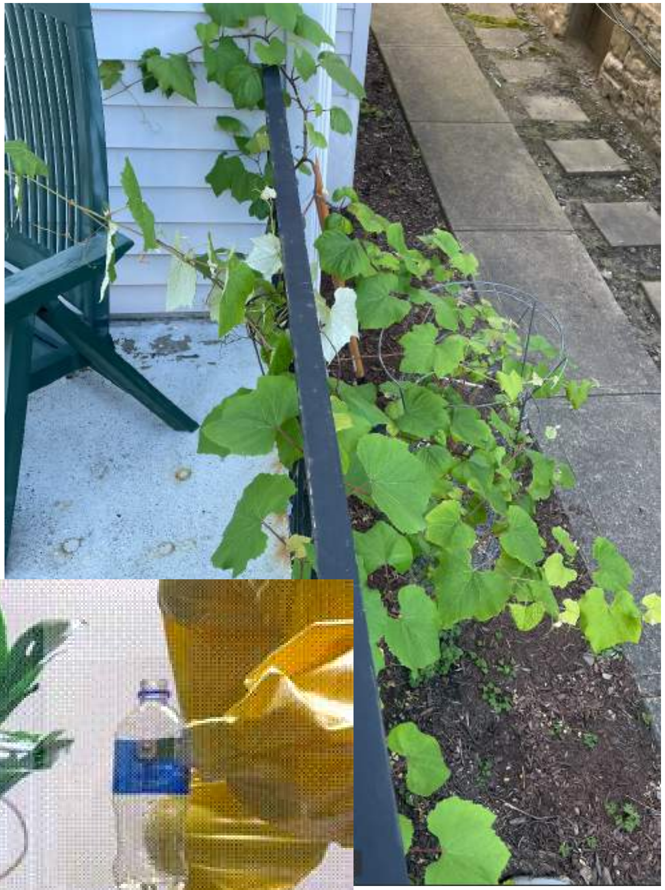
4th year PhD Student (Candidate!)
IAM lab, advised by Oliver Kroemer

Research interests:

Planning with inaccurate models
deformable object manipulation

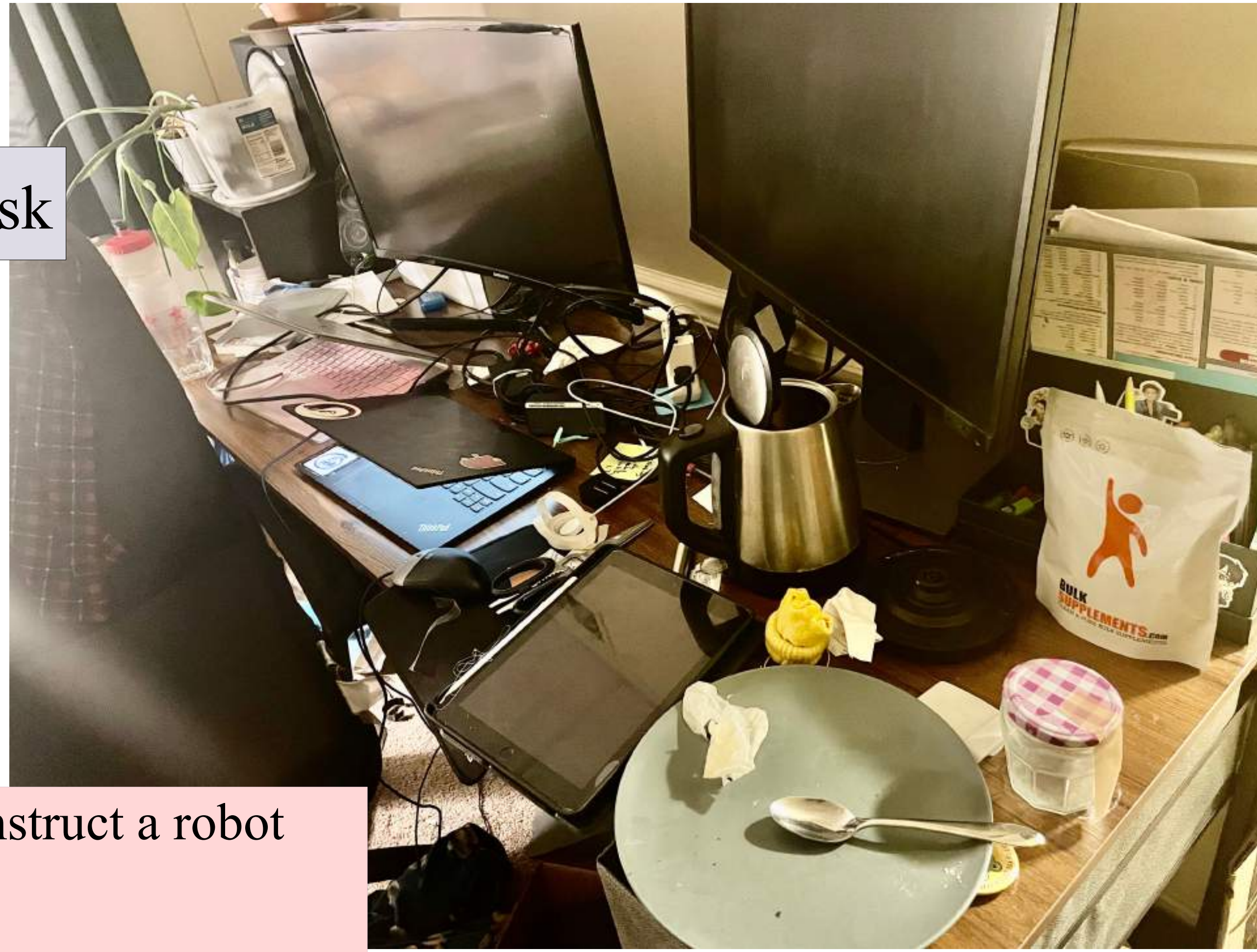
Other interests:

DEI in Robotics/AI research
concord grape vines



Review: Motivating example

Goal: tidy the desk



1) How would you instruct a robot to tidy this scene?
Assume
- grasp/place actions
- robot knows where each object goes

2) What symbols can we use here?

Source: desk of a “friend”

Review: Challenges in symbolic task planning

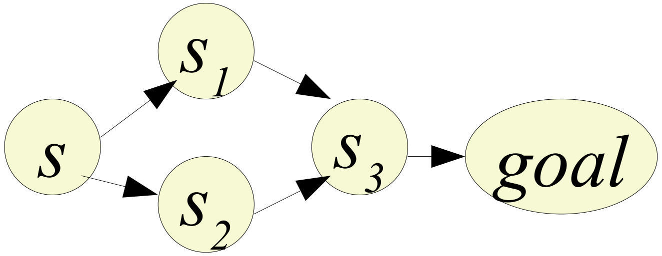
1. What are the symbols?

2. What needs to be done before what?

3. Branching factor

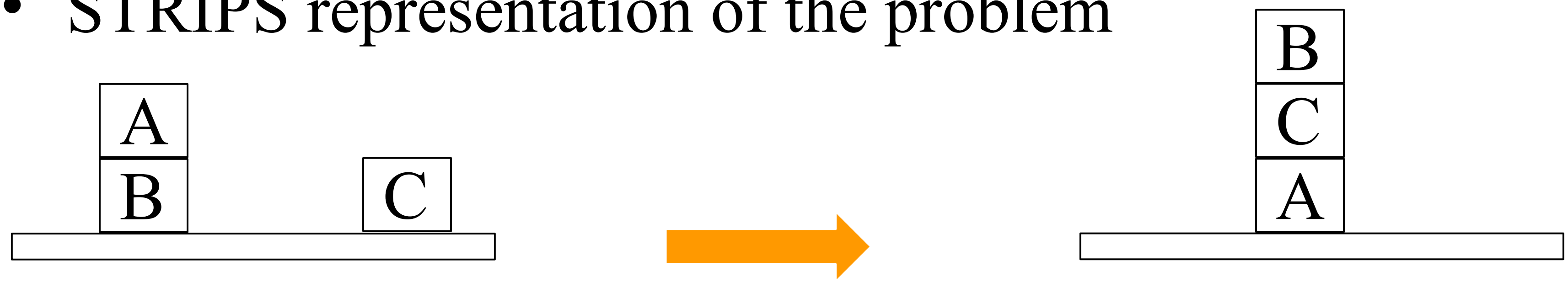
Review: Comparing symbolic with other representations

symbolic	*previous (HW1, HW2, ...)*
State representation:	
AND of literals $\text{statement}_1 \wedge \text{statement}_2$ ...	abstract, position
Action representation	
Precondition: literals Effects: literals	abstract, target position
Generating valid successors	
1. Generate combinations of inputs and action types 2. Check that preconditions are satisfied	Check constraints (ex. collisions)



Review: Defining symbolic planning problems

- STRIPS representation of the problem



Start state:

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

Goal state:

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

Actions:

<u>MoveToTable(b,x)</u>

Precond: On(b,x)^Clear(b)^Block(b)

Effect: On(b,Table)^Clear(x)^~On(b,x)

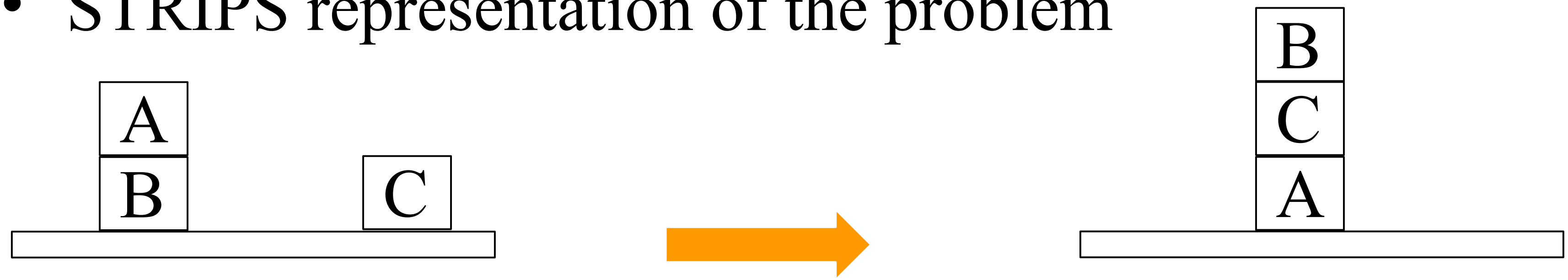
<u>Move(b,x,y)</u>

Precond: On(b,x)^Clear(b)^Clear(y)^Block(b)^Block(y)^(b~=y)

Effect: On(b,y)^Clear(x)^~On(b,x)^~Clear(y)

Review: Defining symbolic planning problems

- STRIPS representation of the problem



Start state:

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

Goal state:

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

Actions:

<u>MoveToTable(b,x)</u>

Precond: On(b,x)^Clear(b)^Block(b)

Effect: On(b,Table)^Clear(x)^~On(b,x)

<u>Move(b,x,y)</u>

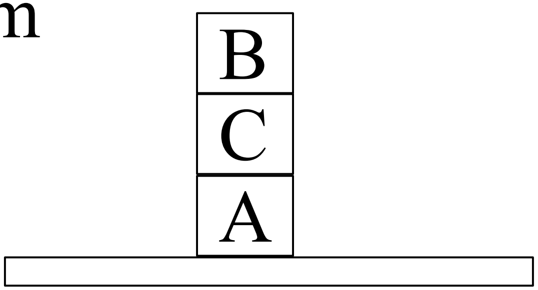
Precond: On(b,x)^Clear(b)^Clear(y)^Block(b)^Block(y)^(b~=y)

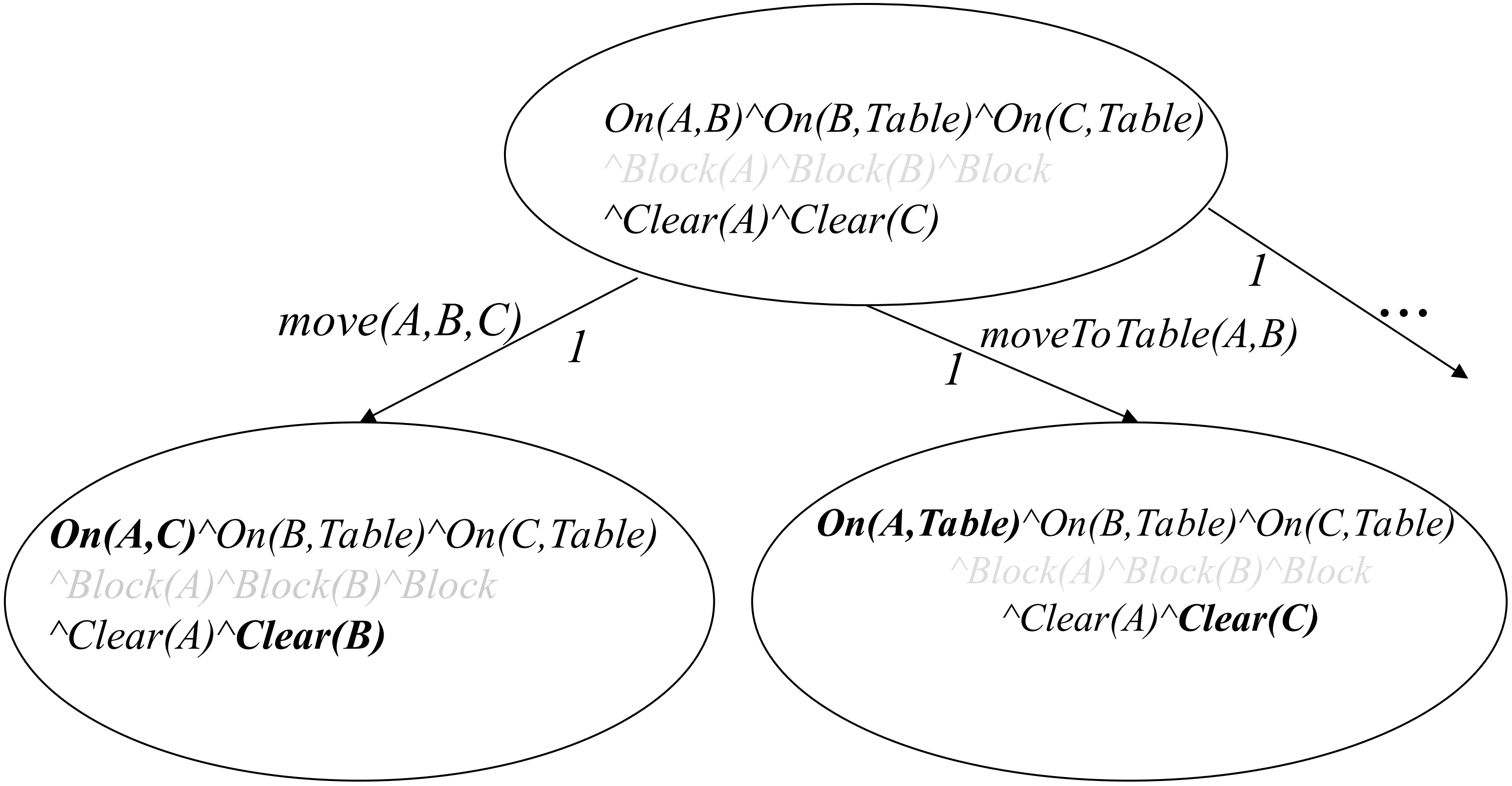
Effect: On(b,y)^Clear(x)^~On(b,x)^~Clear(y)

Review: Planning via graph search

- STRIPS representation of the problem

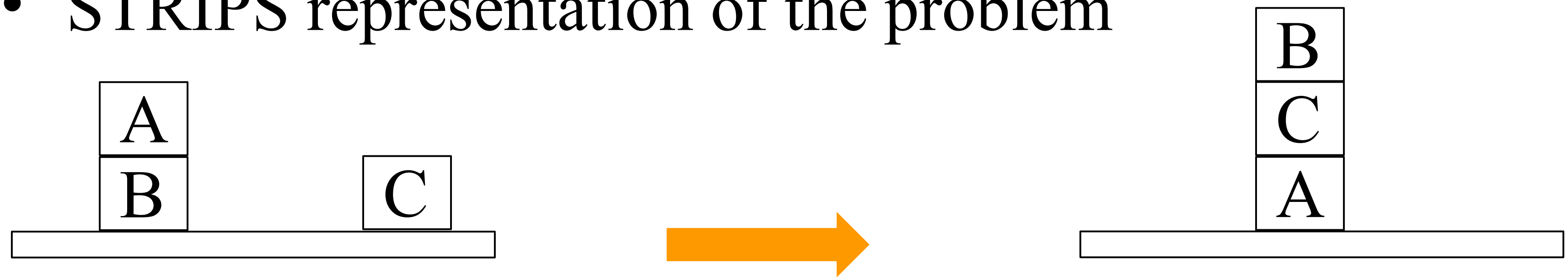
Question: How to find a path to the goal?
How to define costs?



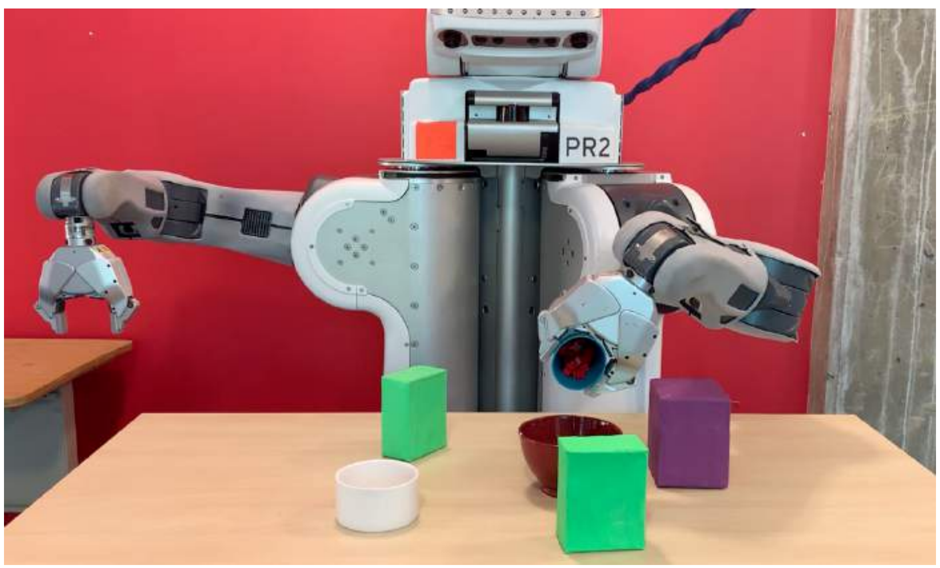


Review: Domain independent heuristics

- STRIPS representation of the problem



Question: What makes a heuristic <u>domain independent</u>?




```
pick[obj](q,p,g)
  con: Stable[obj](p), Grasp[obj](g), Kin[obj](q,p,g)
  pre: holding = None, atRob = q, at[obj] = p
  eff: holding ← obj, at[obj] ← g
place[obj](q,p,g)
  con: Stable[obj](p), Grasp[obj](g), Kin[obj](q,p,g)
  pre: holding = obj, atRob = q, at[obj] = g
  eff: holding ← None, at[obj] ← p
cook[obj](p)
  con: Stable[obj](p), OnStove[obj](p)
  pre: at[obj] = p
  eff: cooked[obj] ← True
```

Source: Wang et al. 2021

Review: Simple euclidean heuristic

$l_1 \wedge l_5 \wedge l_7$

S

$h(s)$ - ?

…

$l_1 \wedge l_3 \wedge l_5$

Goal

Question: How useful is this heuristic?
- Applicable problems
- Admissibility
- Local minima

Key idea:
h(s) = # of literals in goal not satisfied in *s*

i.e., h(s) = # of literals l_i such that $l_i(s)$=false and l_i(goal) = true

Domain independent heuristics

$l_1 \wedge l_5 \wedge l_7$

S

h(s) - ?

…

$l_2 \wedge l_3 \wedge l_5$

Goal

Question: What would be a more useful heuristic?

Hint: what are properties of a useful heuristic?

… ***the most*** *useful heuristic?*

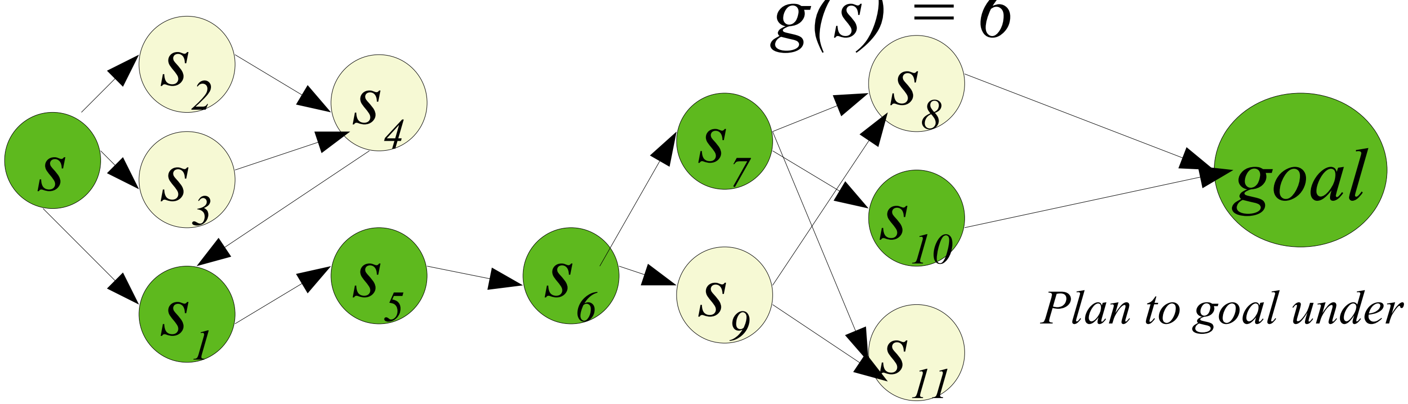
Domain independent heuristic: empty-delete-list

-

Key idea:
*1) Compute h(s) by solving a **relaxed** (simpler) problem*
*2) **empty-delete-list:** assume actions do not have any <u>negative</u> effects*

<u>MoveToTable(b,x)</u>
Precond: On(b,x)^Clear(b)^Block(b)
Effect: On(b,Table)^Clear(x)^~On(b,x)

g(s) = 6



Plan to goal under relaxed problem

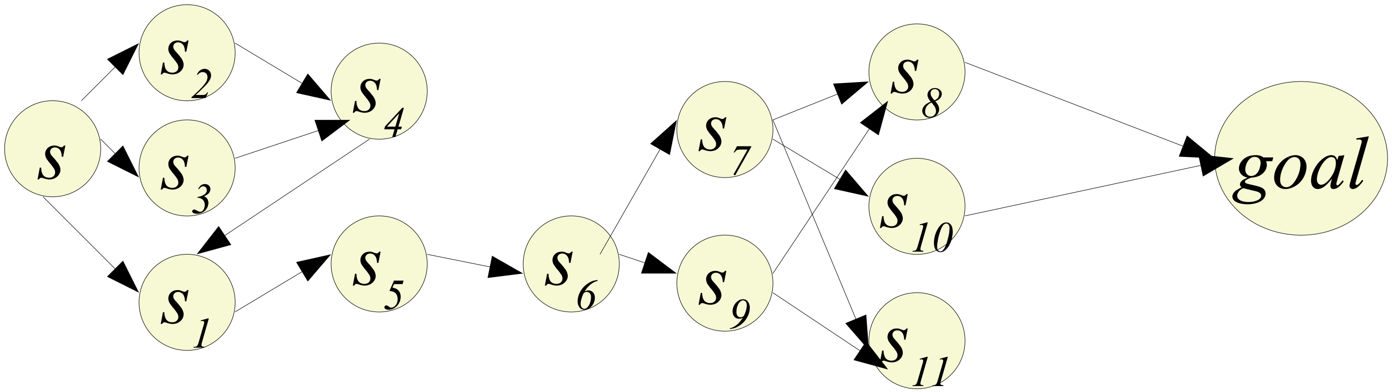
Domain independent heuristic: empty-delete-list

•

Key idea:
*1) Compute h(s) by solving a **relaxed** (simpler) problem*
*2) **empty-delete-list:** assume actions do not have any <u>negative</u> effects*

Question: How does g(s) from this search inform the planner?

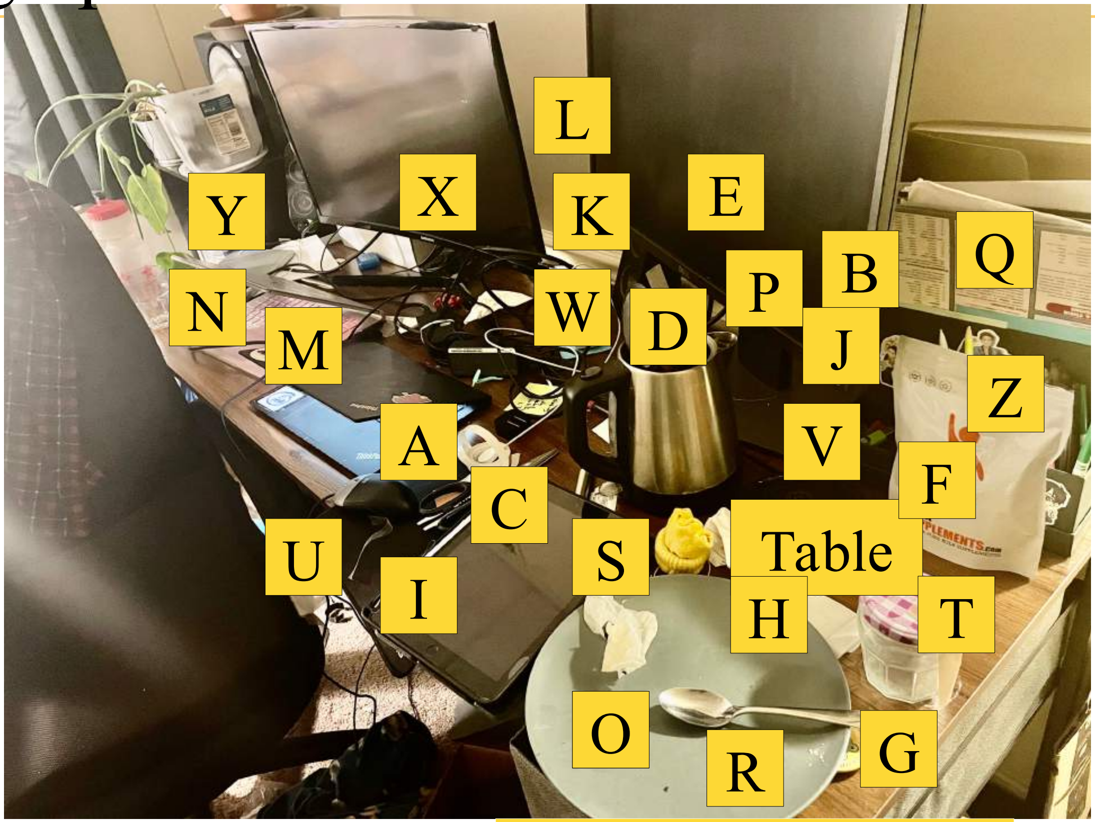
Question: What are the downsides to this heuristic?



Challenges in graph search formulation

Goal: clean table

Add action:
<u>wipe(table)</u>
Precond: Clear(table)
^ Dirty(table)
Effects: Clean(surface)
^ ~Dirty(table)



Source: desk of a “friend”

Question: How to generate all successors for s?

Question: Is a complete list of actions necessary?

Question: What needs to be done in a particular order?

s= On(A, Table)^On(B, Table)^On(C,Table)^On(H,G)….

Intuition: Partial-Order Planning (POP)

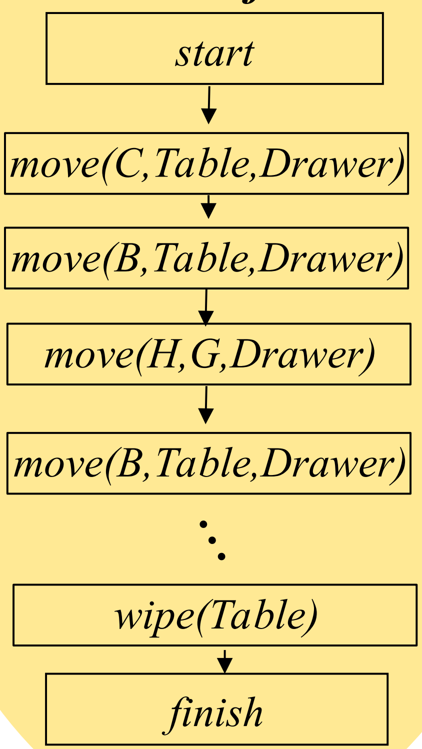
- Search space of *plans*

Question: What does it mean to search in the space of plans?

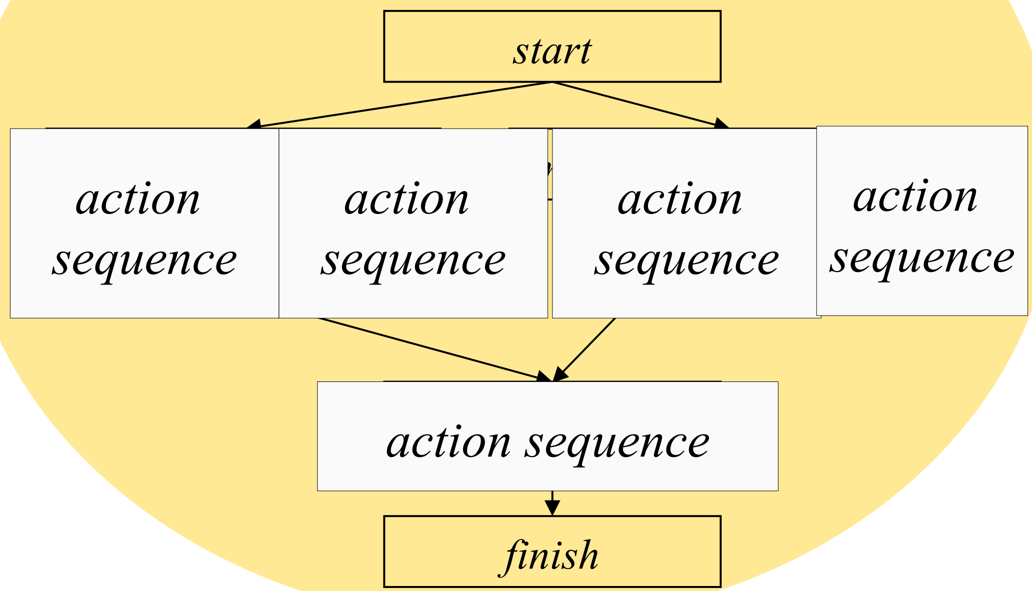
Intuition: Partial-Order Planning (POP)

Question: What needs to be kept track of in the action sequence?

The plan we find is a
total order of actions*:*



POP aims to compute
a partial order of actions*:*



Formulation: Partial-Order Planning (POP)

- Search space of plans
- State in partial-order planning is a plan

action sequence

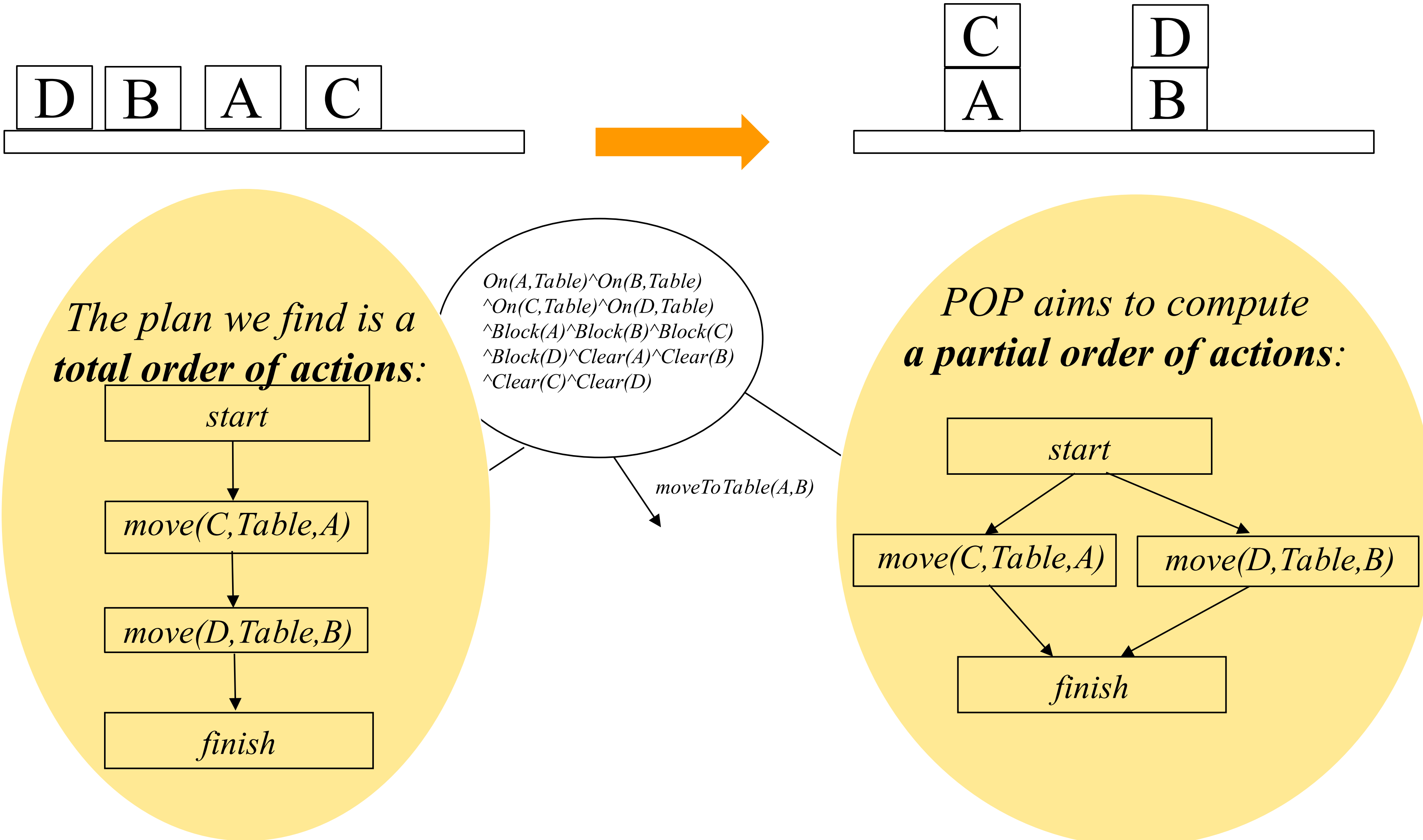
Question: Can there be cycles in the constraints?

action set
constraints: action ordering: form of A < B (A before B)
causal links: how preconds are satisfied
by actions of form $A \xrightarrow{p} B$
(action A achieves precondition p required by action B)

Example on board

Partial-Order Planning (POP)

- Total vs. partial ordering of actions



Partial-Order Planning (POP)

Example on board

Actions
Preconditions:
Effects:

Start
Preconds: {}
Effects: start state

Finish
Preconds: goal state
Effects: {}

Start
Preconds: {}
Effects: On(A,T)^On(,T)^On(C,T)^On(B,T)
^ Cl(A)^Cl(B)^Cl(C)^Cl(D)

Finish
Preconds: On(C,A)^On(D,B)
^Cl(D)^**Cl(C)**^On(A,T)^On(B,T)
Effects: {}

Partial-Order Planning (POP)

Example on board

Start
Preconds: {}
Effects: start state

Finish
Preconds: goal state
Effects: {}

Start state

Actions: *{Start, Finish}*
Constraints: *{Start < Finish}*
Causal links: *{}*

Algorithm: Partial-Order Planning (POP)

- How do we compute successors of a state *s*?
- Show on board (can use this for reference)

1. Pick action *B*
where at least one precondition *p* is not satisfied in s

↓ *p, B*

2. Pick action A either in *s* or a new action
that satisfies *p*

s'
Actions: include *A*
(if not already present)
Constraints: add $A < B$,
$Start < A$,
$A < Finish$
Causal links: Add $A \xrightarrow{p} B$

Question: What if … ?

If any other action *C* in *s'* removes *p* → Then add $C<A$ or $B<C$ as constraints

If A removes p' used in link $D \rightarrow p' \rightarrow F$ → Then add $A<D$ or $F<A$ as constraints

s' is invalid if there is a constraint cycle

Generating successors in (POP)

Example on board

Start
Preconds: {}
Effects: On(A,T)^On(,T)^On(C,T)^On(B,T) ^ Cl(A)^Cl(B)^Cl(C)^Cl(D)

Finish
Preconds: On(C,A)^On(D,B) ^Cl(D)^Cl(C)^On(A,T)^On(B,T)
Effects: {}

*Pick **Cl(C)** to satisfy*

Actions: *{Start, Finish}*
Constraints:
{Start < Finish}
Causal links:
{}

Actions:
{Start, Finish}
Constraints:
{Start < Finish}
Causal links:
Start $\xrightarrow{Cl(C)}$ *Finish*

Question: How do we find which preconditions are satisfied or not?

Generating successors in (POP)

Example on board

Start
Preconds: {}
Effects: On(A,T)^On(,T)^On(C,T)^On(B,T) ^ Cl(A)^Cl(B)^Cl(C)^Cl(D)

Finish
Preconds: On(C,A)^On(D,B) ^Cl(D)^Cl(C)^On(A,T)^On(B,T)
Effects: {}

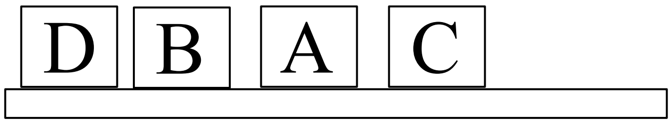
Pick ***On(C,A)*** *to satisfy*

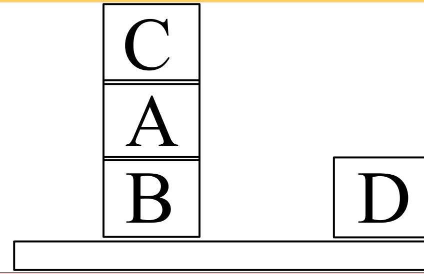
Actions: *{Start, Finish}*
Constraints:
{Start < Finish}
Causal links:
{}

Actions:
{Start, Finish, Move(C,T,A)}
Constraints: *{Start < Finish,*
Start < Move(C,T,A),
Move(C,T,A) < Finish}
Causal links:
Move(C,T,A) $\xrightarrow{On(C,A)}$ *Finish*

Preconditions violated in POP

Example on board





Goal changed! for explanatory purposes

*Suppose we pick **Move(A,T,B)***

<u>This action removes Cl(A) precondition of Move(A,T,B)!</u>

Question: is it possible to add this action to the plan?

Actions:
{Start, Finish, Move(C,T,A)}
Constraints: *{Start < Finish, Start < Move(C,T,A), Move(C,T,A) < Finish}*
Causal links:
Move(C,T,A) $\xrightarrow{On(C,A)}$ *Finish*

Actions:
{Start, Finish, Move(C,T,A), …,Move(A,T,B)}
Constraints: *{Start < Finish, Start < Move(C,T,A), …,*
Move(A,T,B) < Move(C,T,A)
Move(C,T,A) < Finish}
Causal links:
Move(C,T,A) $\xrightarrow{On(C,A)}$ *Finish, …,*

Partial-Order Planning (POP)

- Searches the space of "plans"
  - Terminate the search as soon as a state where all actions have all their preconditions met is reached (e.g., a goal state of the search)

Question: How should we decide which actions to add?

Actions: Start, Finish
Order constraints: Start < Finish

Start state

Actions: Start, Finish, A, C, ...
Order constraints: Start < Finish,

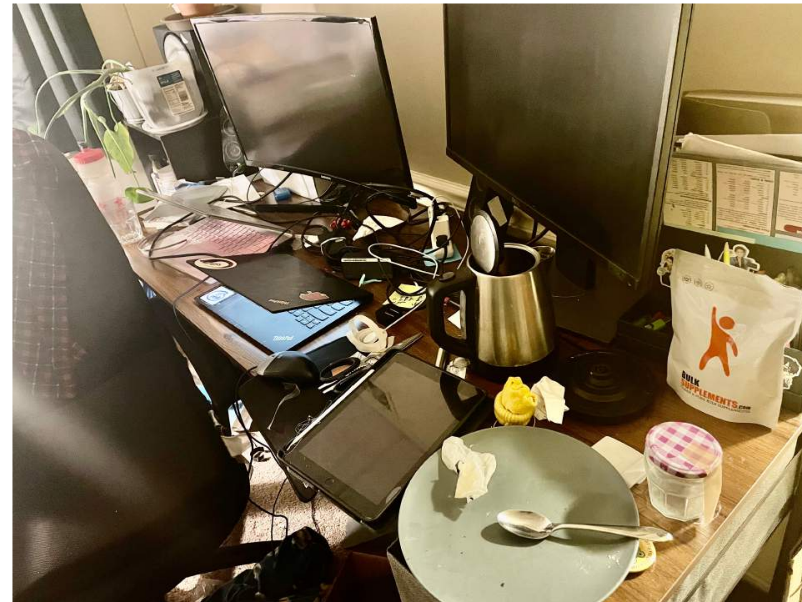
Goal state

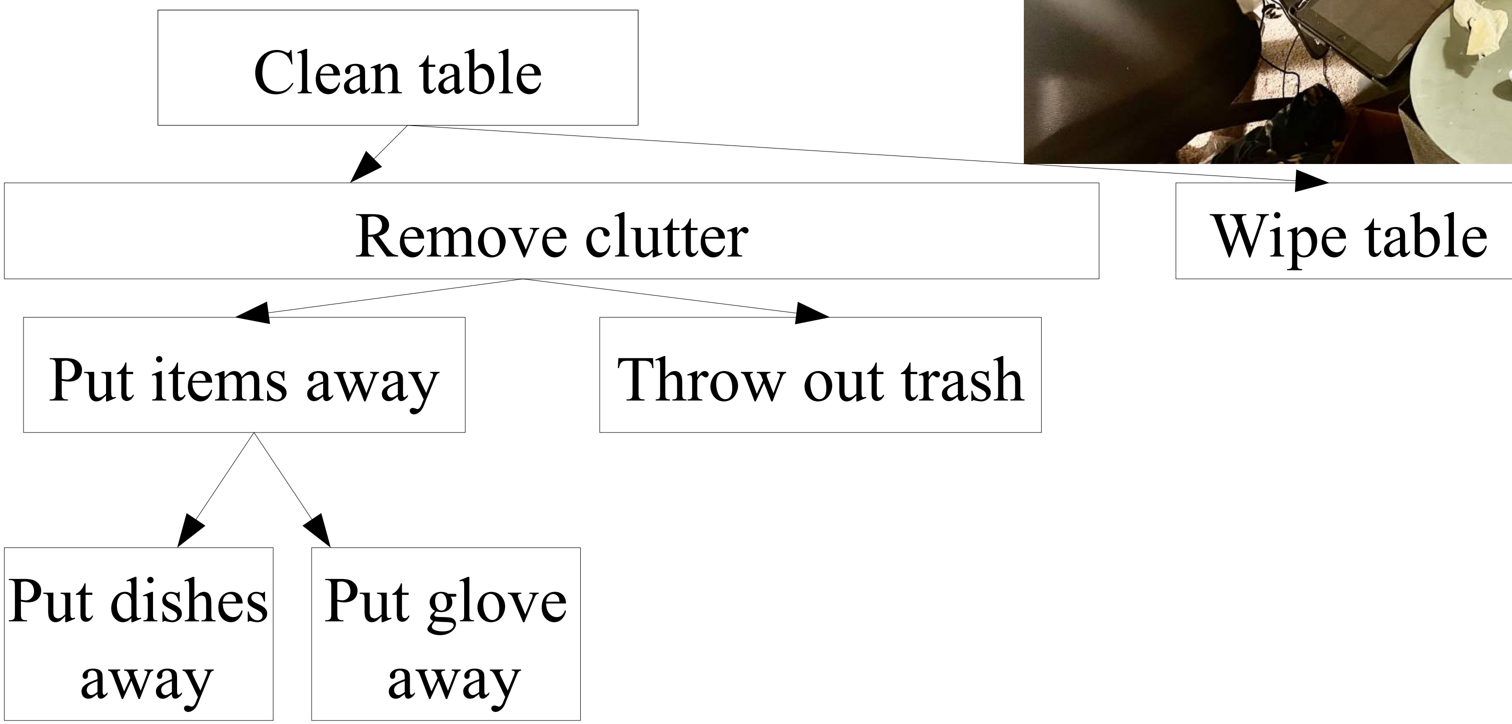
This gives us an implicit graph
that is typically searched by Depth-First Search
for any feasible solution to the goal state

Hierarchical planning

Key idea:
Not every action needs to be fully planned out from the beginning!

1. Plan at a high level
2. Work out details as needed





Kaelbling and Lozano-Pérez 2013

Discussion questions

1. How does partial order planning differ from other planning techniques we've discussed? In what scenarios might partial-order planning be particularly effective?

2. Think of an everyday example when you formulate a problem as symbols. Write down a problem using literals. You can make up predicates as you need.

3. How would you solve this problem? Can it be solved with the planning techniques we've learned? How can you organize the problem into a hierarchy?

What You Should Know…

- How to compute domain-independent heuristics
- Advantages of partial-order planning: avoid needing to compute total order
- The general idea behind how Partial-order Planning works

Please give Alex feedback so they can improve!
bit.ly/alex_lecturer_feedback