CPSC 322
Introduction to Artificial Intelligence

November 26, 2004

Things...

Term project is due Monday

Finding the plan

Some of what follows is a repeat from the previoius lecture, but I added some new material in the middle of what's repeated. So to make the new stuff more understandable, the slides from last time that surround the new material are presented again.

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:
ontable(b)
clear(a)
armempty
on(a,b)

NOTE: clear = cleartop...I just forgot to type the whole word

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:
ontable(b)
clear(a)
armempty
on(a,b)

if a goal is fulfilled in the current state, then don't worry about it

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:
ontable(b)
clear(a)
armempty
on(a,b)

if a goal is fulfilled in the current state, then don't worry about it

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

on(a,b)

if a goal is fulfilled in the current state, then don't worry about it

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

on(a,b)

for remaining goals, reduce difference between goal and current state

Means-ends analysis

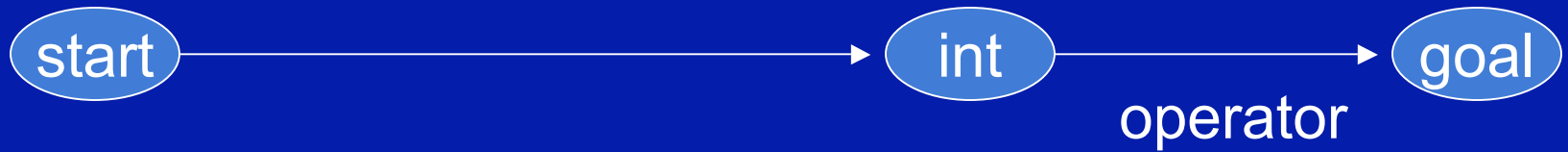
Detecting differences between the start state and goal state, then finding an operator to reduce those differences (thereby solving part of the problem) is often called means-ends analysis. Chapter 8 uses it, but doesn't mention its name.

Means-ends analysis

how to reduce this difference?

one way might be to find an operator that gets you from some intermediate state to the goal state...

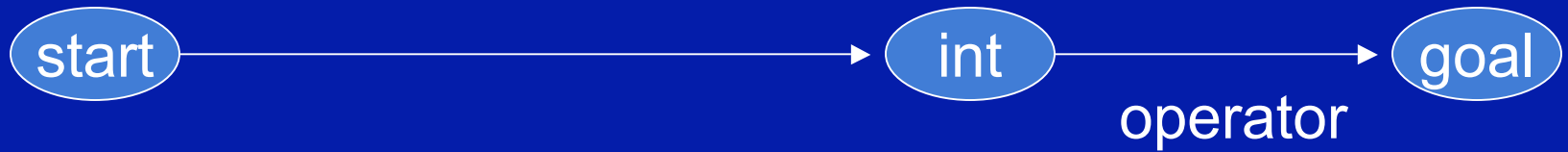
Means-ends analysis



how to reduce this difference?

one way might be to find an operator that gets you from some intermediate state to the goal state...

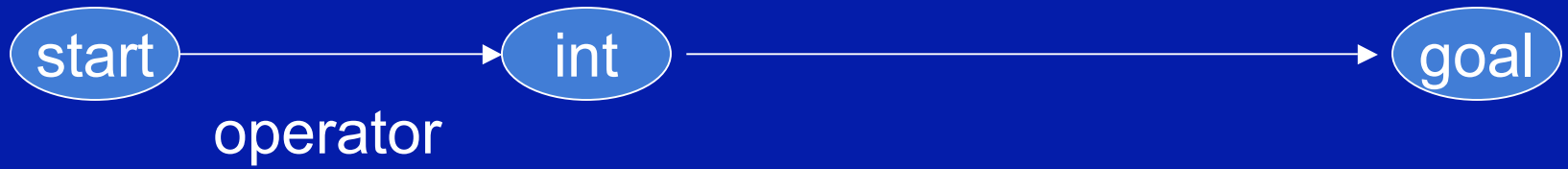
Means-ends analysis



then apply means-ends analysis recursively to reduce the distance between start and intermediate states

This is what the STRIPS planner we're talking about now is doing. But you could also work from the start state toward the goal (forward chaining)...

Means-ends analysis



then apply means-ends analysis recursively to reduce the distance between start and intermediate states

This is what the STRIPS planner we're talking about now is doing. But you could also work from the start state toward the goal (forward chaining)...and then apply means-ends analysis on the distance between the intermediate and goal states

Means-ends analysis

or you could find an operator that gets you from one intermediate state to another, and apply MEA to both of the remaining distances.

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

on(a,b)

for remaining goals, reduce difference between goal and current state

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

on(a,b)

find an action with add list that contains goal...

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

clear(b)
holding(a)

post that action's preconditions as new goals...

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

stack(a,b)

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

clear(b)
holding(a)

post that action as a step in the plan...

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

stack(a,b)

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

clear(b)
holding(a)

if a goal is fulfilled in the current state, then don't worry about it

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

stack(a,b)

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

clear(b)
holding(a)

if a goal is fulfilled in the current state, then don't worry about it

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

stack(a,b)

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

holding(a)

if a goal is fulfilled in the current state, then don't worry about it

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

stack(a,b)

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

holding(a)

for remaining goals, reduce difference between goal and current state

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

stack(a,b)

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

holding(a)

find an action with add list that contains goal...

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

stack(a,b)

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

clear(a)
ontable(a)
armempty

post that action's preconditions as new goals...

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

clear(a)
ontable(a)
armempty

post that action as a step in the plan...

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

clear(a)
ontable(a)
armempty

if a goal is fulfilled in the current state, then don't worry about it

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

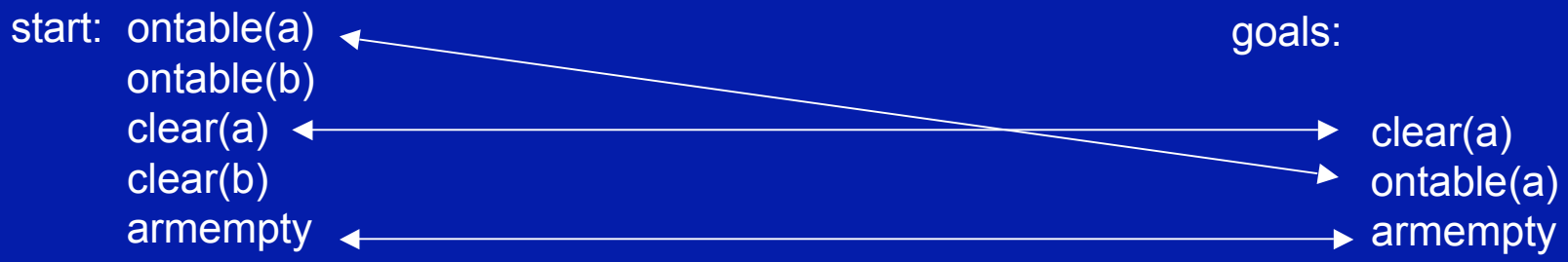
unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)



if a goal is fulfilled in the current state, then don't worry about it

Finding the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

all the goals have been fulfilled...we now have our plan

Executing the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

are the preconditions for the first step true? yes

Executing the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start: ontable(a)
ontable(b)
clear(a)
clear(b)
armempty

goals:

then do what the delete list says to do...

Executing the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start:

ontable(b)
clear(a)
clear(b)
armempty

goals:

then do what the delete list says to do...

Executing the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start:

ontable(b)
clear(a)
clear(b)

goals:

then do what the delete list says to do...

Executing the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start:

ontable(b)
clear(a)
clear(b)

goals:

and then do what the add list says to do...

Executing the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start: holding(a)
ontable(b)
clear(a)
clear(b)

goals:

and then do what the add list says to do...

Executing the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start: holding(a)
ontable(b)
clear(a)
clear(b)

goals:

are the preconditions for the second step true? yes

Executing the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start: holding(a)
ontable(b)
clear(a)
clear(b)

goals:

then do what the delete list says to do...

Executing the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start: holding(a)
ontable(b)
clear(a)

goals:

then do what the delete list says to do...

Executing the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start:

ontable(b)
clear(a)

goals:

then do what the delete list says to do...

Executing the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start:

ontable(b)
clear(a)

goals:

and then do what the add list says to do...

Executing the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start: armempty
ontable(b)
clear(a)

goals:

and then do what the add list says to do...

Executing the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start: armempty
ontable(b)
clear(a)
on(a,b)

goals:

and then do what the add list says to do...

Executing the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start: armempty
ontable(b)
clear(a)
on(a,b)

goals:

the plan is finished...how does the resulting state compare to original goal?

Executing the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)

start: armempty
ontable(b)
clear(a)
on(a,b)

goals:

ontable(b)
clear(a)
armempty
on(a,b)

the plan is finished...how does the resulting state compare to original goal?

Executing the plan

stack(X,Y) - preconds: clear(Y) & holding(X)
delete: clear(Y) & holding(X)
add: armempty & on(X,Y)

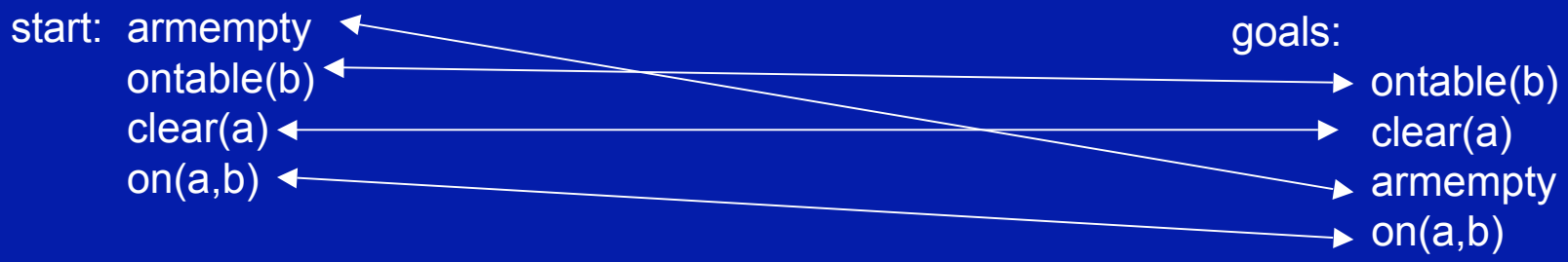
unstack(X,Y) - preconds: on(X,Y) & clear(X) & armempty
delete: on(X,Y) & armempty
add: holding(X) & clear(Y)

pickup(X) - preconds: clear(X) & ontable(X) & armempty
delete: ontable(X) & armempty
add: holding(X)

putdown(X) - preconds: holding(X)
delete: holding(X)
add: ontable(X) & armempty

plan:

pickup(a)
stack(a,b)



give your robot a cookie

How about some tasty CILOG code?

```
/*
This is an implementation of the simple STRIPS planner
shown on page 302 of the Computational Intelligence text.

launch it with the query:
cilog: ask goals(G) & achieve_all(G,init,Plan).

Note that the add list is denoted here by "achieves" instead of
"add", and that the add and delete lists aren't exactly lists.
*/

/* stack action */
preconditions(stack(X,Y),[cleartop(Y),holding(X)]).
achieves(stack(X,Y),armempty).
achieves(stack(X,Y),on(X,Y)).
deletes(stack(X,Y),cleartop(Y)).
deletes(stack(X,Y),holding(X)).
```

How about some tasty CILOG code?

```
/* unstack action */
preconditions(unstack(X,Y),[on(X,Y),clear(X),armempty]).
achieves(unstack(X,Y),holding(X)).
achieves(unstack(X,Y),cleartop(Y)).
deletes(unstack(X,Y),on(X,Y)).
deletes(unstack(X,Y),armempty).

/* pickup action */
preconditions(pickup(X),[cleartop(X),ontable(X),armempty]).
achieves(pickup(X),holding(X)).
deletes(pickup(X),ontable(X)).
deletes(pickup(X),armempty).

/* putdown action */
preconditions(putdown(X),[holding(X)]).
achieves(putdown(X),ontable(X)).
achieves(putdown(X),armempty).
deletes(putdown(X),holding(X)).
```

How about some tasty CILOG code?

```
/* initial situation */

holds(ontable(a),init).
holds(ontable(b),init).
holds(cleartop(a),init).
holds(cleartop(b),init).
holds(armempty,init).

achieves(init,X) <- holds(X,init).

goals([ontable(b),cleartop(a),armempty,on(a,b)]).
```

How about some tasty CILOG code?

```
/* the simple STRIPS planner */

remove(X,[X|Y],Y).

achieve_all([],W0,W0).

achieve_all(Goals,W0,W2) <-
     remove(G,Goals,Rem_Gs) &
     achieve(G,W0,W1) &
     achieve_all(Rem_Gs,W1,W2).

achieve(G,W,W) <- holds(G,W).

achieve(G,W0,do(Action,W1)) <-
     achieves(Action,G) &
     preconditions(Action,Pre) &
     achieve_all(Pre,W0,W1).
```

How about some tasty CILOG code?

```
/* the simple STRIPS planner */

remove(X,[X|Y],Y).

achieve_all([],W0,W0).

achieve_all(Goals,W0,W2) <-
     remove(G,Goals,Rem_Gs) &          % choose first goal on list
     achieve(G,W0,W1) &                % try to achieve that goal G
     achieve_all(Rem_Gs,W1,W2).        % then deal with other goals

achieve(G,W,W) <- holds(G,W).

achieve(G,W0,do(Action,W1)) <-
     achieves(Action,G) &
     preconditions(Action,Pre) &
     achieve_all(Pre,W0,W1).
```

How about some tasty CILOG code?

```
/* the simple STRIPS planner */

remove(X,[X|Y],Y).

achieve_all([],W0,W0).

achieve_all(Goals,W0,W2) <-
     remove(G,Goals,Rem_Gs) &          % choose first goal on list
     achieve(G,W0,W1) &                % try to achieve that goal G
     achieve_all(Rem_Gs,W1,W2).        % then deal with other goals

achieve(G,W,W) <- holds(G,W).          % goal G is achieved if it
                                       % holds for state W

achieve(G,W0,do(Action,W1)) <-
     achieves(Action,G) &
     preconditions(Action,Pre) &
     achieve_all(Pre,W0,W1).
```

How about some tasty CILOG code?

```
/* the simple STRIPS planner */

remove(X,[X|Y],Y).

achieve_all([],W0,W0).

achieve_all(Goals,W0,W2) <-
     remove(G,Goals,Rem_Gs) &          % choose first goal on list
     achieve(G,W0,W1) &                % try to achieve that goal G
     achieve_all(Rem_Gs,W1,W2).        % then deal with other goals

achieve(G,W,W) <- holds(G,W).          % goal G is achieved if it
                                       % holds for state W

achieve(G,W0,do(Action,W1)) <-
     achieves(Action,G) &              % G on add list for Action
     preconditions(Action,Pre) &       % get preconds for Action
     achieve_all(Pre,W0,W1).           % preconds are now goals
```

Desirable attributes of a knowledge representation approach

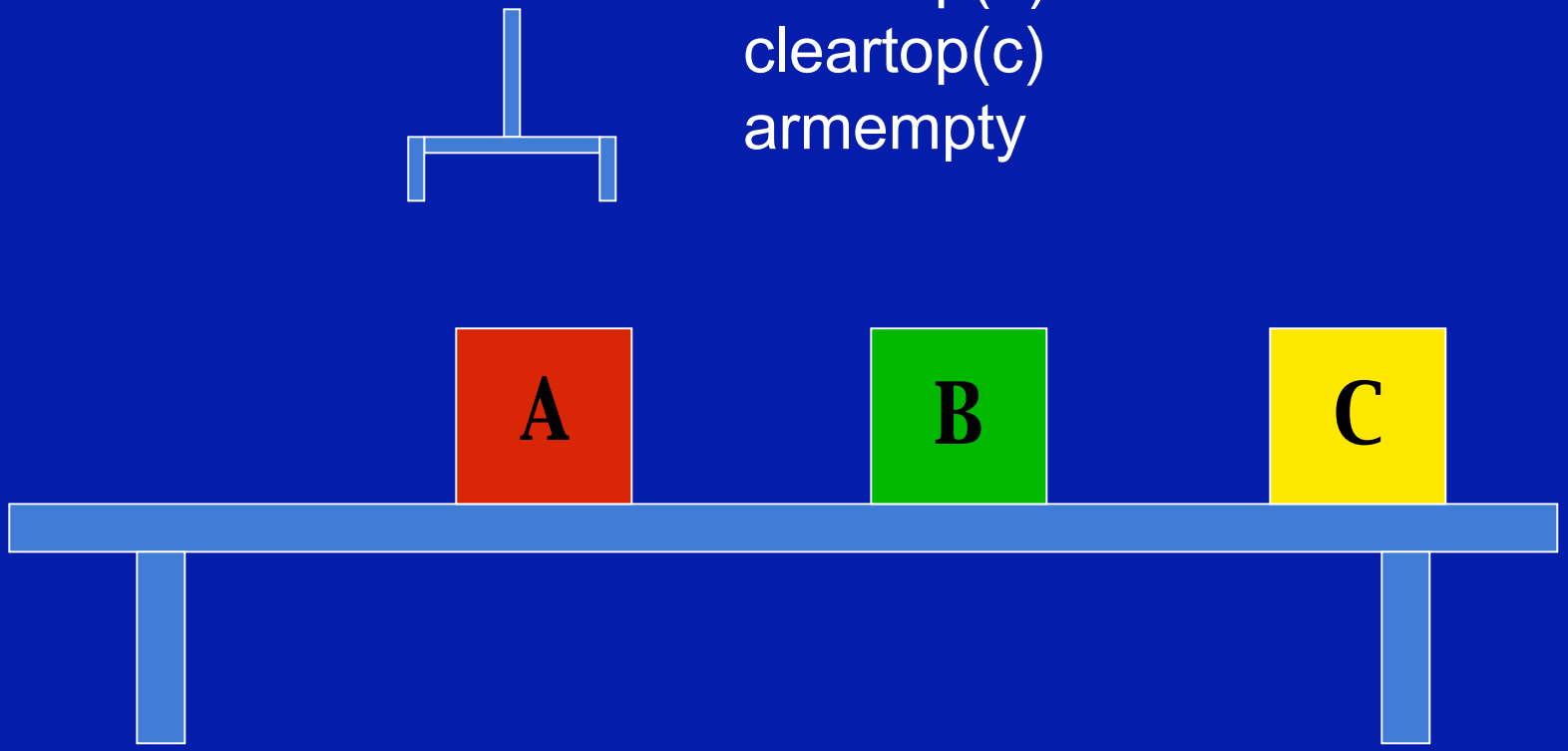
- capture generalities in the world being modeled
- easily modifiable to reflect changes so that new knowledge can be derived from old knowledge
- transparent - understandable by people who provide the knowledge as well as those who look at it later
- usable even if not entirely accurate or complete
- explicitly represent important objects and relationships
- natural constraints on how one object or relation influences another should be obvious
- irrelevant detail should be suppressed (abstracted away)
- complete -- everything that needs to be represented can be represented
- concise -- what needs to be said can be said efficiently
- fast -- you can store and retrieve information quickly
- computable -- enables reasoning to proceed easily with known procedures (doesn't rely on bizarre coding tricks)

Did choosing a good KR approach make this problem easier?

```
/* the simple STRIPS planner */

remove(X,[X|Y],Y).

achieve_all([],W0,W0).

achieve_all(Goals,W0,W2) <-
     remove(G,Goals,Rem_Gs) &
     achieve(G,W0,W1) &
     achieve_all(Rem_Gs,W1,W2).

achieve(G,W,W) <- holds(G,W).

achieve(G,W0,do(Action,W1)) <-
     achieves(Action,G) &
     preconditions(Action,Pre) &
     achieve_all(Pre,W0,W1).
```

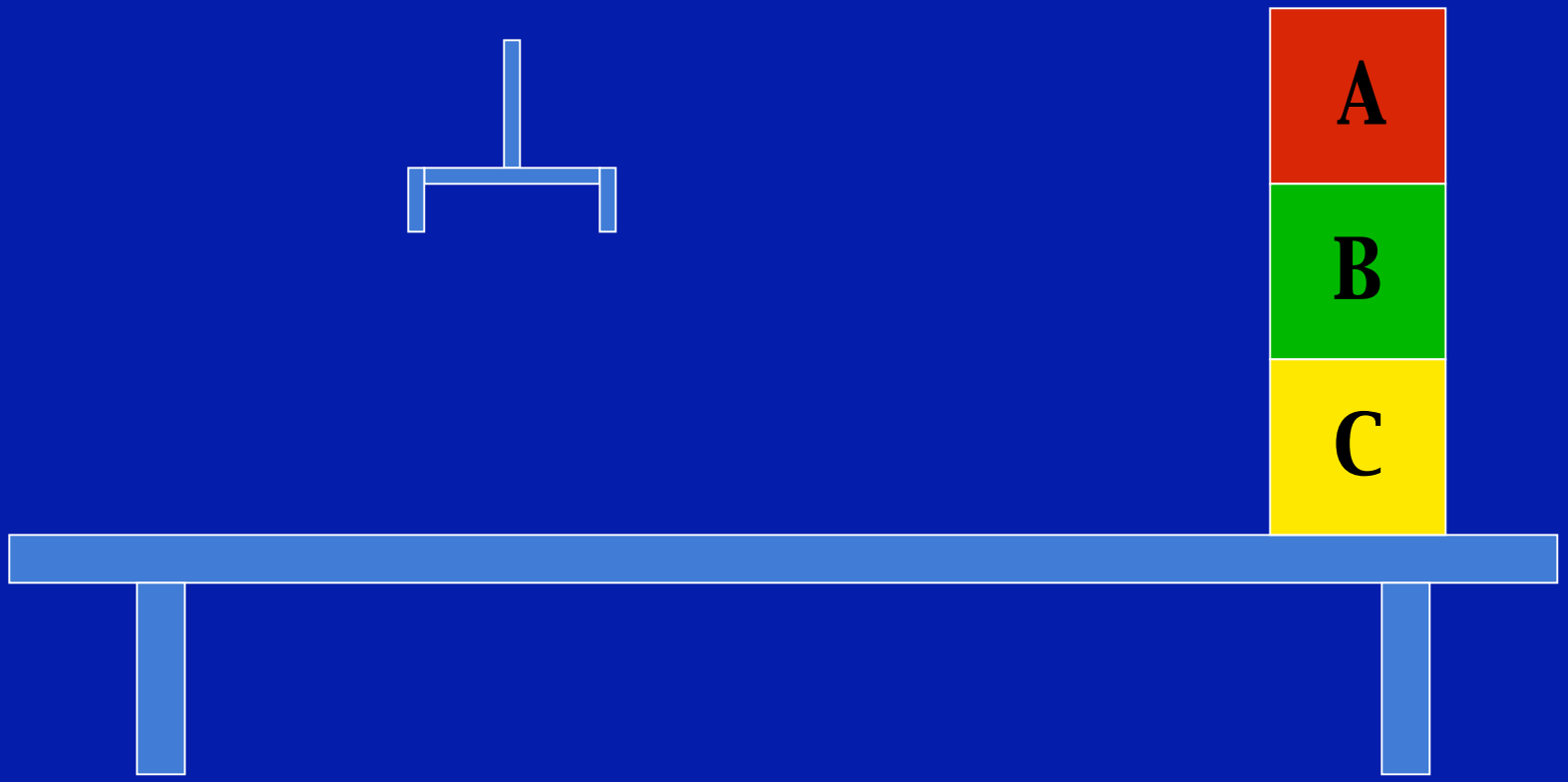
A new start state

ontable(a)
ontable(b)
ontable(c)
cleartop(a)
cleartop(b)
cleartop(c)
armempty



A new goal state

on(a,b)
on(b,c)



Will our simple planner work?

Load ‘strips01.ci’ and add the extra block

Try it on both of these:

```
ask achieve_all([on(b,c),on(a,b)],init,P).
ask achieve_all([on(a,b),on(b,c)],init,P).
```

Why it doesn't work

```
/* the simple STRIPS planner */

remove(X,[X|Y],Y).

achieve_all([],W0,W0).

achieve_all(Goals,W0,W2) <-
     remove(G,Goals,Rem_Gs) &          % choose first goal on list
     achieve(G,W0,W1) &                % try to achieve that goal G
     achieve_all(Rem_Gs,W1,W2).        % then deal with other goals

achieve(G,W,W) <- holds(G,W).          % goal G is achieved if it
                                       % holds for state W

achieve(G,W0,do(Action,W1)) <-
     achieves(Action,G) &              % G on add list for Action
     preconditions(Action,Pre) &       % get preconds for Action
     achieve_all(Pre,W0,W1).           % preconds are now goals
```

You gotta check intermediate states, not just the initial state

Will a more complex planner work?

Load ‘strips02.ci’ -- the extra block is already there

Then try it on both of these:

ask achieve_all([on(b,c),on(a,b)],init,P).

ask achieve_all([on(a,b),on(b,c)],init,P).

Here's a better planner

```
achieve_all([],W0,W0).
achieve_all(Goals,W0,W2) <-
     remove(G,Goals,Rem_Gs) &
     achieve(G,W0,W1) &
     achieve_all(Rem_Gs,W1,W2).

achieve(G,W,W) <- true_in(G,W).
achieve(A \= B,W,W) <- A \= B.

achieve(G,W0,do(Action,W1)) <-
     achieves(Action,G) &
     preconditions(Action,Pre) &
     achieve_all(Pre,W0,W1).

true_in(G,init) <-
   holds(G,init).
true_in(G,do(A,_)) <-
   achieves(A,G).
true_in(G,do(A,S)) <-
   true_in(G,S) &
   ~ deletes(A,G).
```

Here's a better planner

```
achieve_all([],W0,W0).
achieve_all(Goals,W0,W2) <-
     remove(G,Goals,Rem_Gs) &
     achieve(G,W0,W1) &
     achieve_all(Rem_Gs,W1,W2).

achieve(G,W,W) <- true_in(G,W).
achieve(A \= B,W,W) <- A \= B.        % so we don't stack a block
                                      % on itself
achieve(G,W0,do(Action,W1)) <-
     achieves(Action,G) &
     preconditions(Action,Pre) &
     achieve_all(Pre,W0,W1).

true_in(G,init) <-
   holds(G,init).
true_in(G,do(A,_)) <-
   achieves(A,G).
true_in(G,do(A,S)) <-
   true_in(G,S) &
   ~ deletes(A,G).
```

Here's a better planner

```
/* stack action */
preconditions(stack(X,Y),[cleartop(Y),X\=Y,holding(X)]).
achieves(stack(X,Y),armempty).
achieves(stack(X,Y),on(X,Y)).
deletes(stack(X,Y),cleartop(Y)).
deletes(stack(X,Y),holding(X)).

/* unstack action */
preconditions(unstack(X,Y),[on(X,Y),cleartop(X),X\=Y],armempty).
achieves(unstack(X,Y),holding(X)).
achieves(unstack(X,Y),cleartop(Y)).
deletes(unstack(X,Y),on(X,Y)).
deletes(unstack(X,Y),armempty).
```

Here's a better planner

```
achieve_all([],W0,W0).
achieve_all(Goals,W0,W2) <-
     remove(G,Goals,Rem_Gs) &
     achieve(G,W0,W1) &
     achieve_all(Rem_Gs,W1,W2).

achieve(G,W,W) <- true_in(G,W).
achieve(A \= B,W,W) <- A \= B.

achieve(G,W0,do(Action,W1)) <-
     achieves(Action,G) &
     preconditions(Action,Pre) &
     achieve_all(Pre,W0,W1).

true_in(G,init) <-
   holds(G,init).
true_in(G,do(A,_)) <-
   achieves(A,G).
true_in(G,do(A,S)) <-
   true_in(G,S) &
   ~ deletes(A,G).
```

```
% check more than init state
% so we don't stack a block
% on itself
```

Here's a better planner

```
achieve_all([],W0,W0).
achieve_all(Goals,W0,W2) <-
     remove(G,Goals,Rem_Gs) &
     achieve(G,W0,W1) &
     achieve_all(Rem_Gs,W1,W2).

achieve(G,W,W) <- true_in(G,W).          % check more than init state
achieve(A \= B,W,W) <- A \= B.           % so we don't stack a block
                                         % on itself
achieve(G,W0,do(Action,W1)) <-
     achieves(Action,G) &
     preconditions(Action,Pre) &
     achieve_all(Pre,W0,W1).

true_in(G,init) <-                       % G achieved if it holds in
   holds(G,init).                        % initial state
true_in(G,do(A,_)) <-
   achieves(A,G).
true_in(G,do(A,S)) <-
   true_in(G,S) &
   ~ deletes(A,G).
```

Here's a better planner

```
achieve_all([],W0,W0).
achieve_all(Goals,W0,W2) <-
     remove(G,Goals,Rem_Gs) &
     achieve(G,W0,W1) &
     achieve_all(Rem_Gs,W1,W2).

achieve(G,W,W) <- true_in(G,W).          % check more than init state
achieve(A \= B,W,W) <- A \= B.           % so we don't stack a block
                                         % on itself
achieve(G,W0,do(Action,W1)) <-
     achieves(Action,G) &
     preconditions(Action,Pre) &
     achieve_all(Pre,W0,W1).

true_in(G,init) <-                       % G achieved if it holds in
   holds(G,init).                        % initial state
true_in(G,do(A,_)) <-                    % or if it's on the add list
   achieves(A,G).                        % of the action that's the
true_in(G,do(A,S)) <-                    % most recent plan step
   true_in(G,S) &
   ~ deletes(A,G).
```

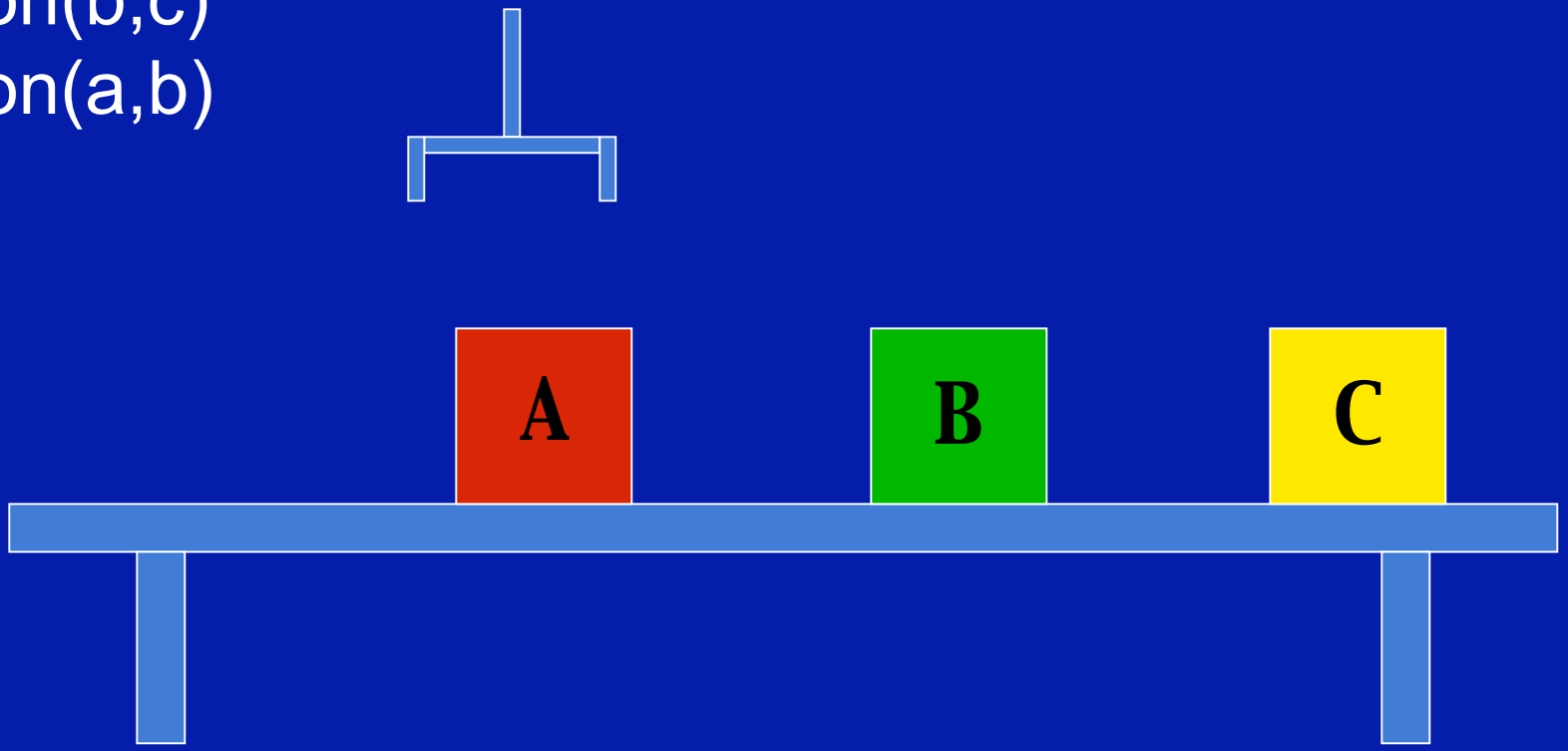
Here's a better planner

```
achieve_all([],W0,W0).
achieve_all(Goals,W0,W2) <-
     remove(G,Goals,Rem_Gs) &
     achieve(G,W0,W1) &
     achieve_all(Rem_Gs,W1,W2).

achieve(G,W,W) <- true_in(G,W).          % check more than init state
achieve(A \= B,W,W) <- A \= B.           % so we don't stack a block
                                         % on itself
achieve(G,W0,do(Action,W1)) <-
     achieves(Action,G) &
     preconditions(Action,Pre) &
     achieve_all(Pre,W0,W1).

true_in(G,init) <-                       % G achieved if it holds in
   holds(G,init).                        % initial state
true_in(G,do(A,_)) <-                    % or if it's on the add list
   achieves(A,G).                        % of the action that's the
true_in(G,do(A,S)) <-                    % most recent plan step
   true_in(G,S) &                        % or if it's achieved in an
   ~ deletes(A,G).                       % earlier plan step and not
                                         % on delete list of A
```

Why doesn't it work always?

goals:

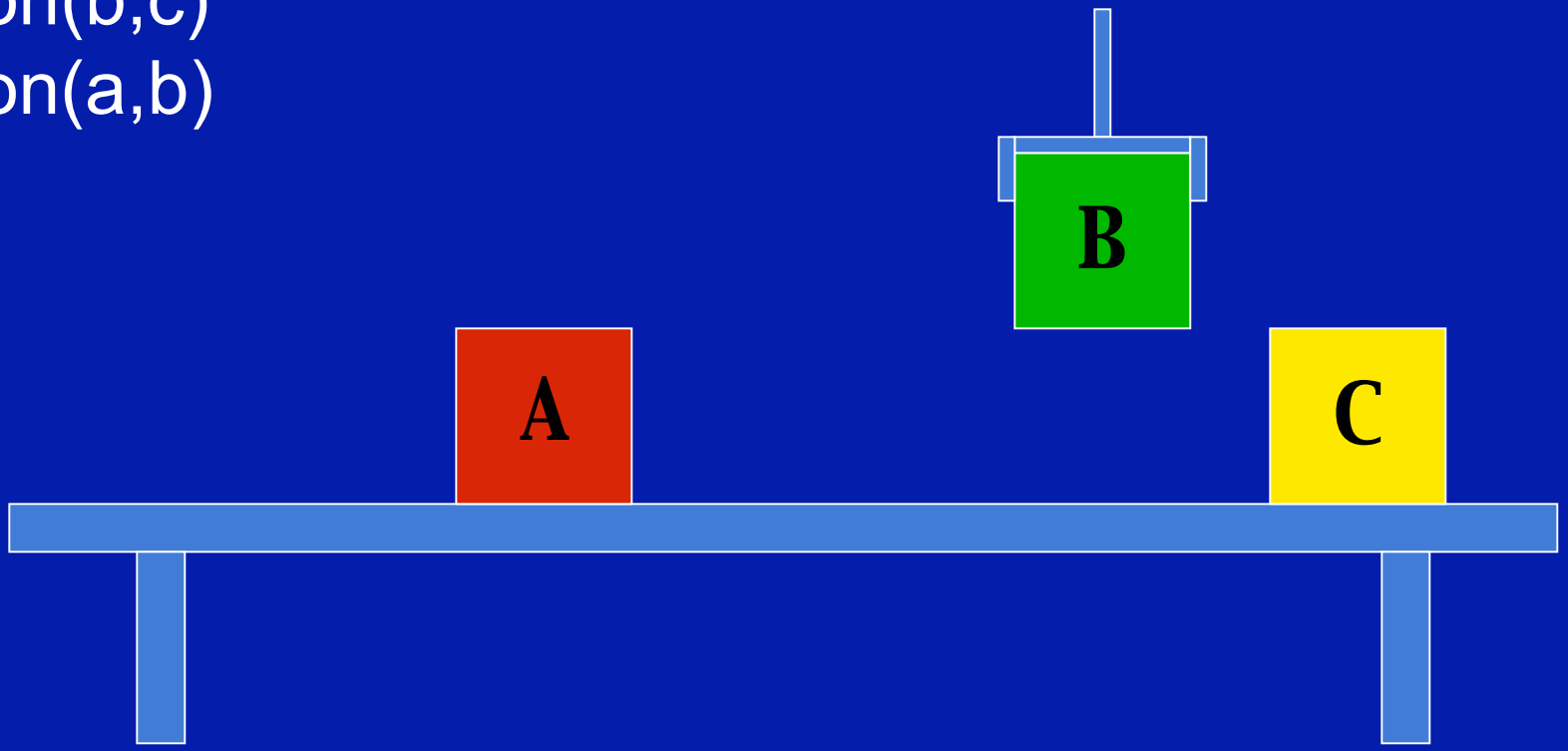
on(b,c)
on(a,b)



Why doesn’t it work always?

goals:

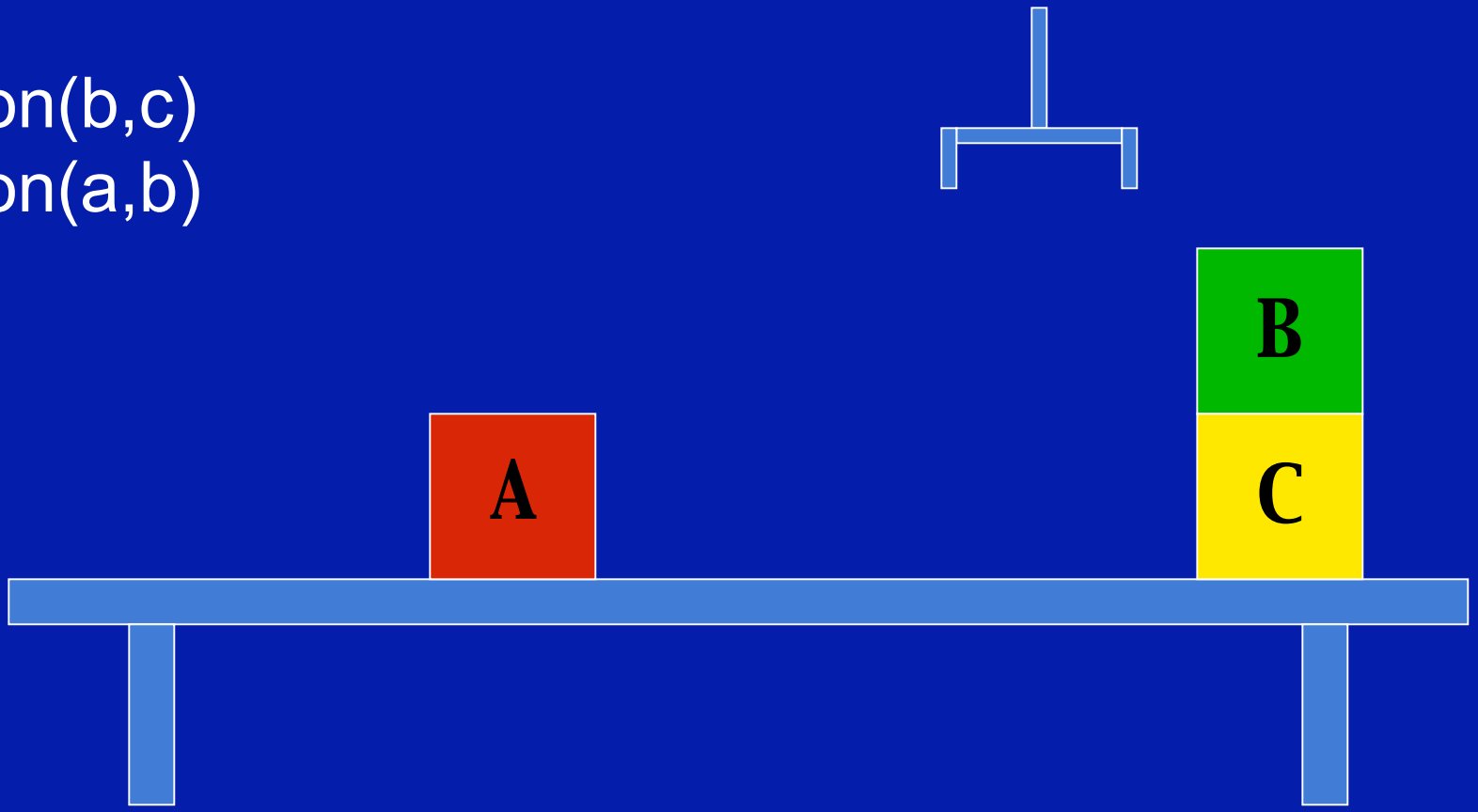
on(b,c)
on(a,b)



Why doesn't it work always?

goals:

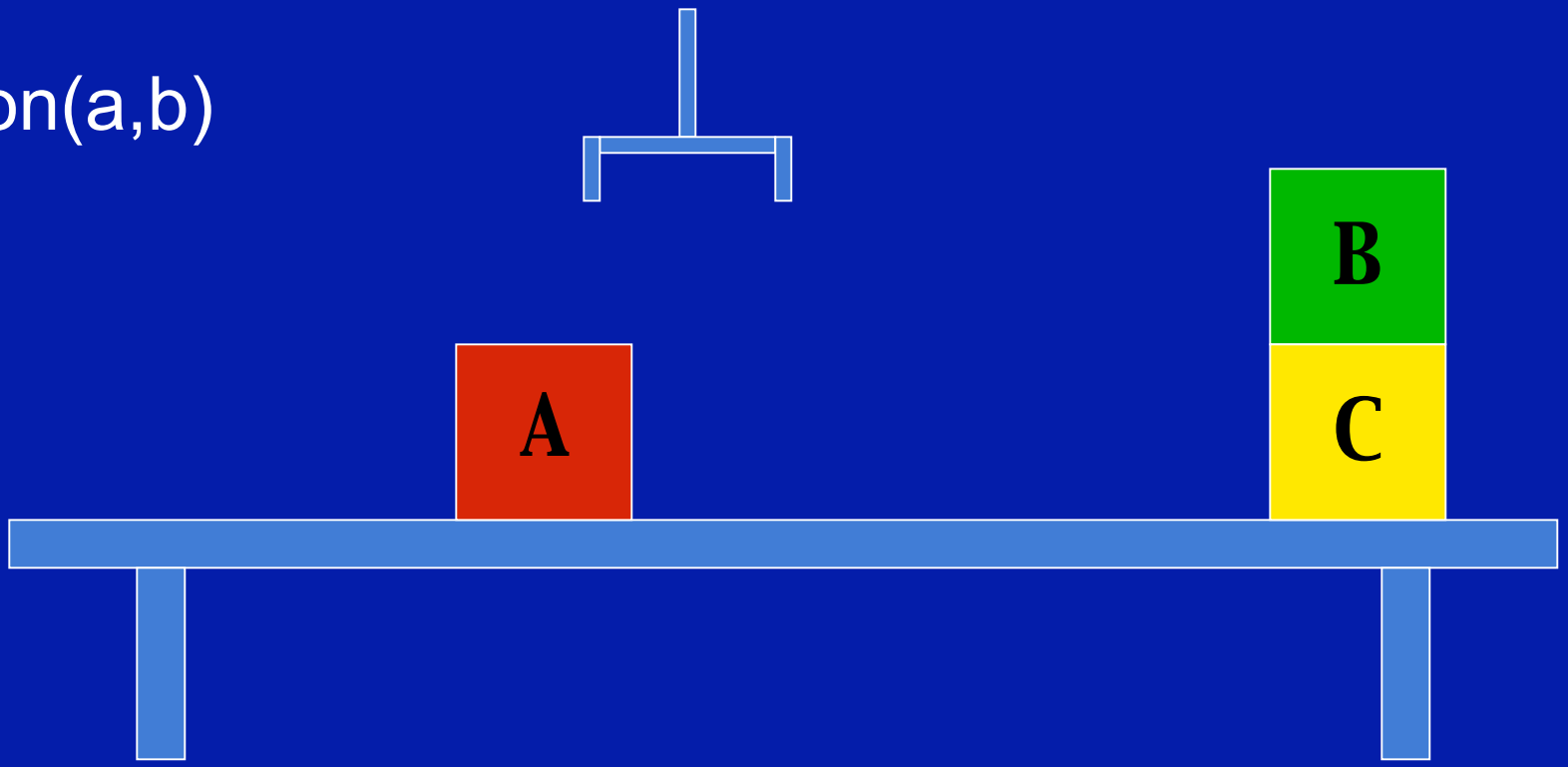
on(b,c)
on(a,b)



Why doesn’t it work always?

goals:

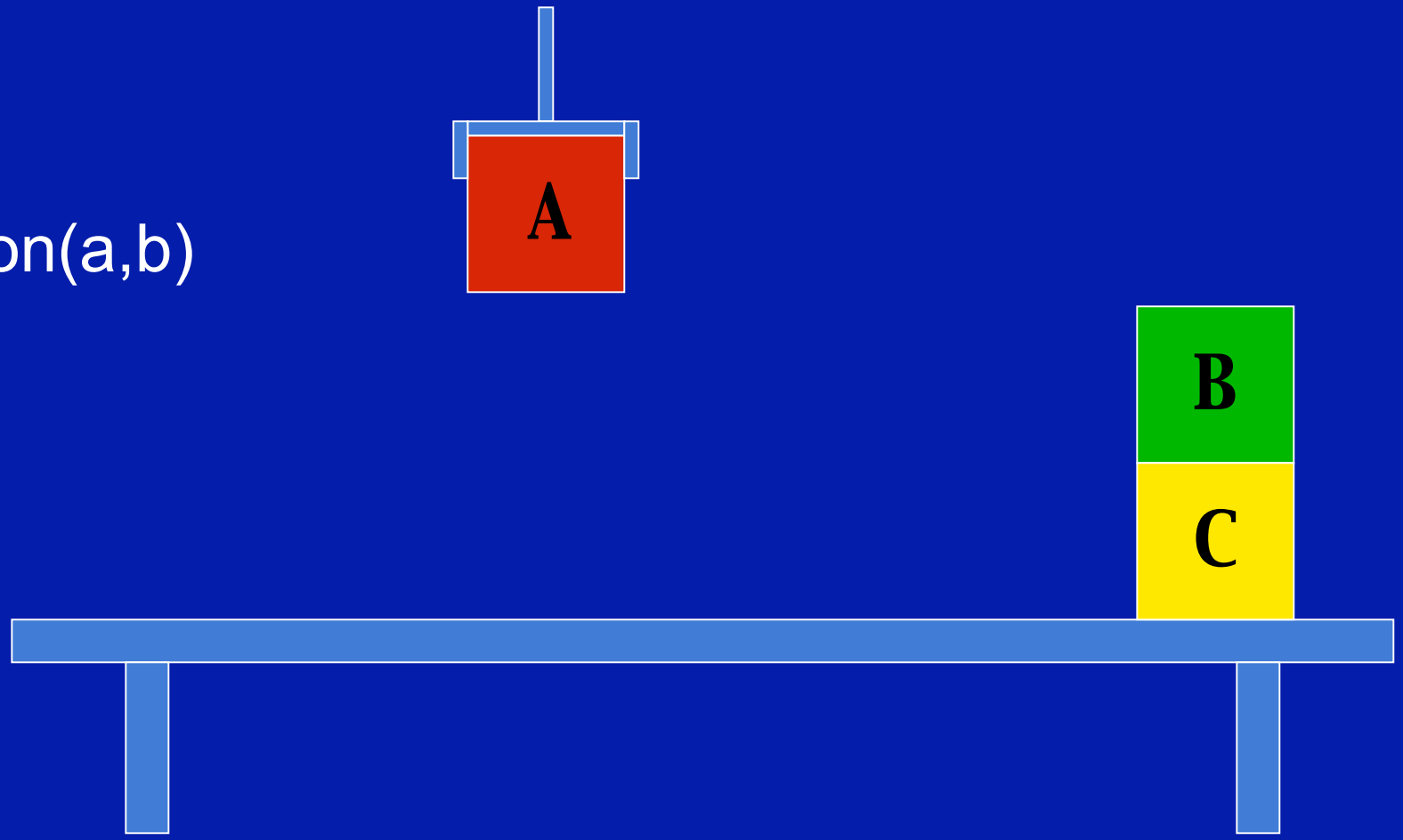
on(a,b)



Why doesn’t it work always?

goals:

on(a,b)



Why doesn’t it work always?

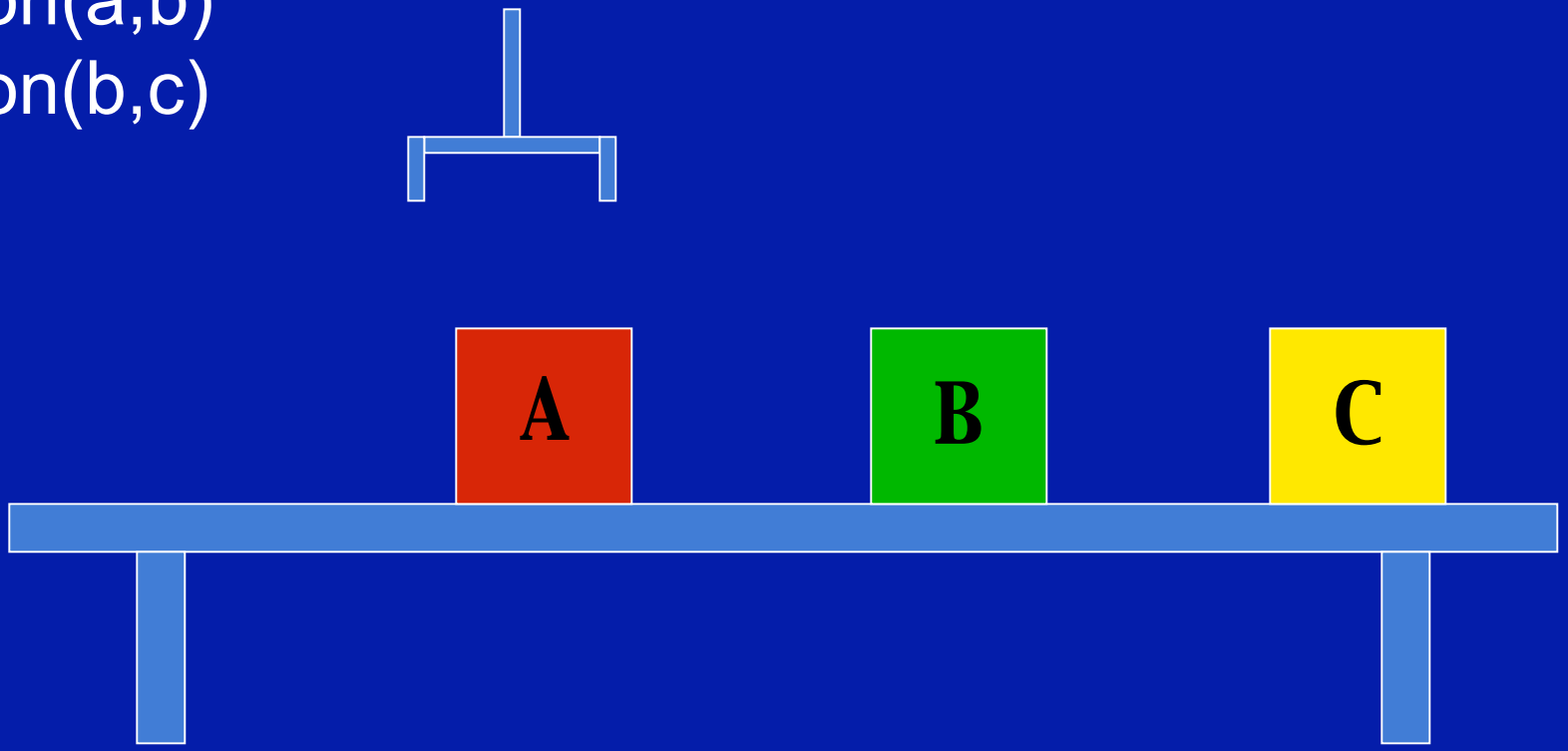
goals:

this works

Why doesn't it work always?

goals:

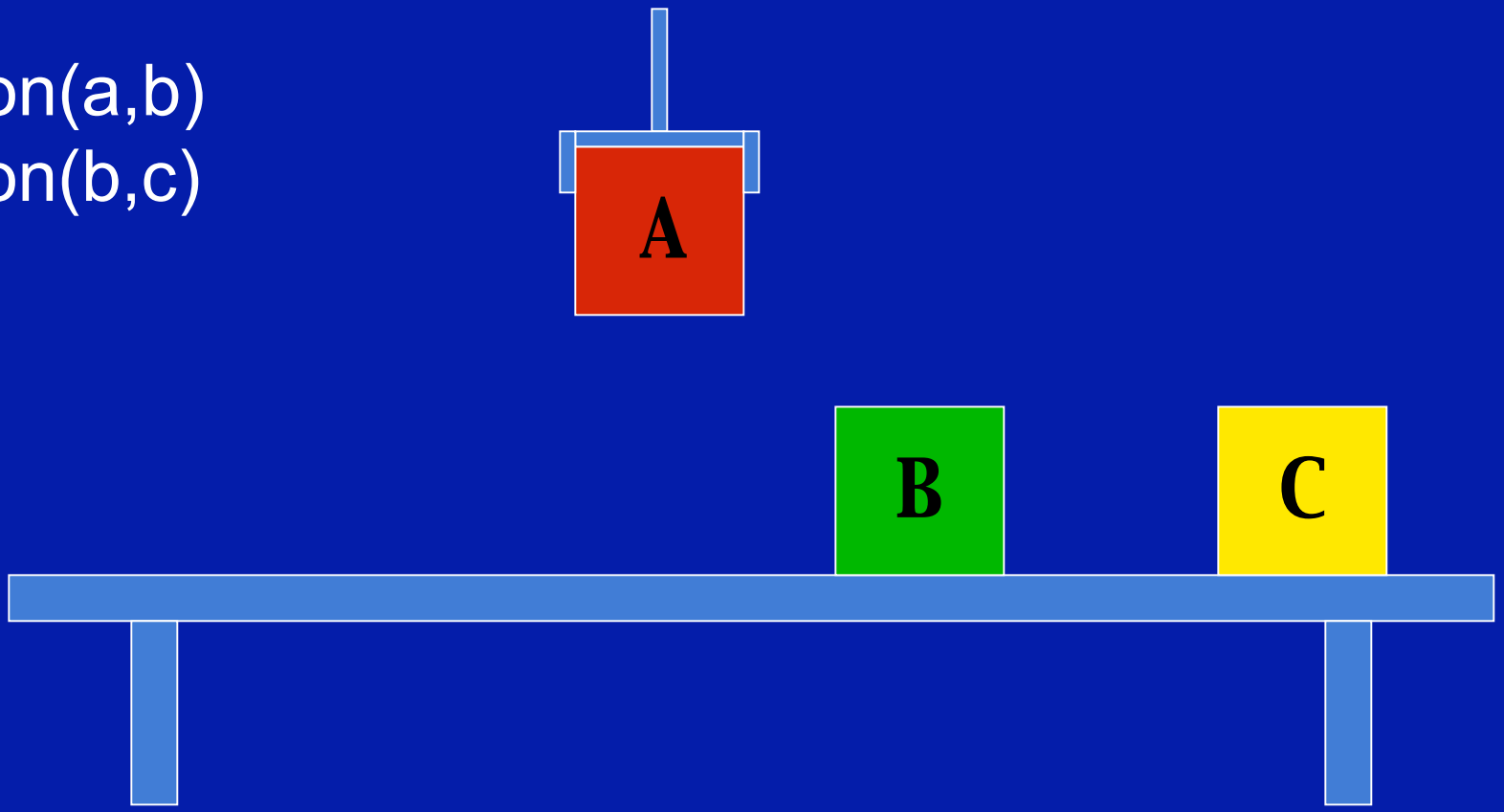
on(a,b)
on(b,c)



Why doesn't it work always?

goals:

on(a,b)
on(b,c)

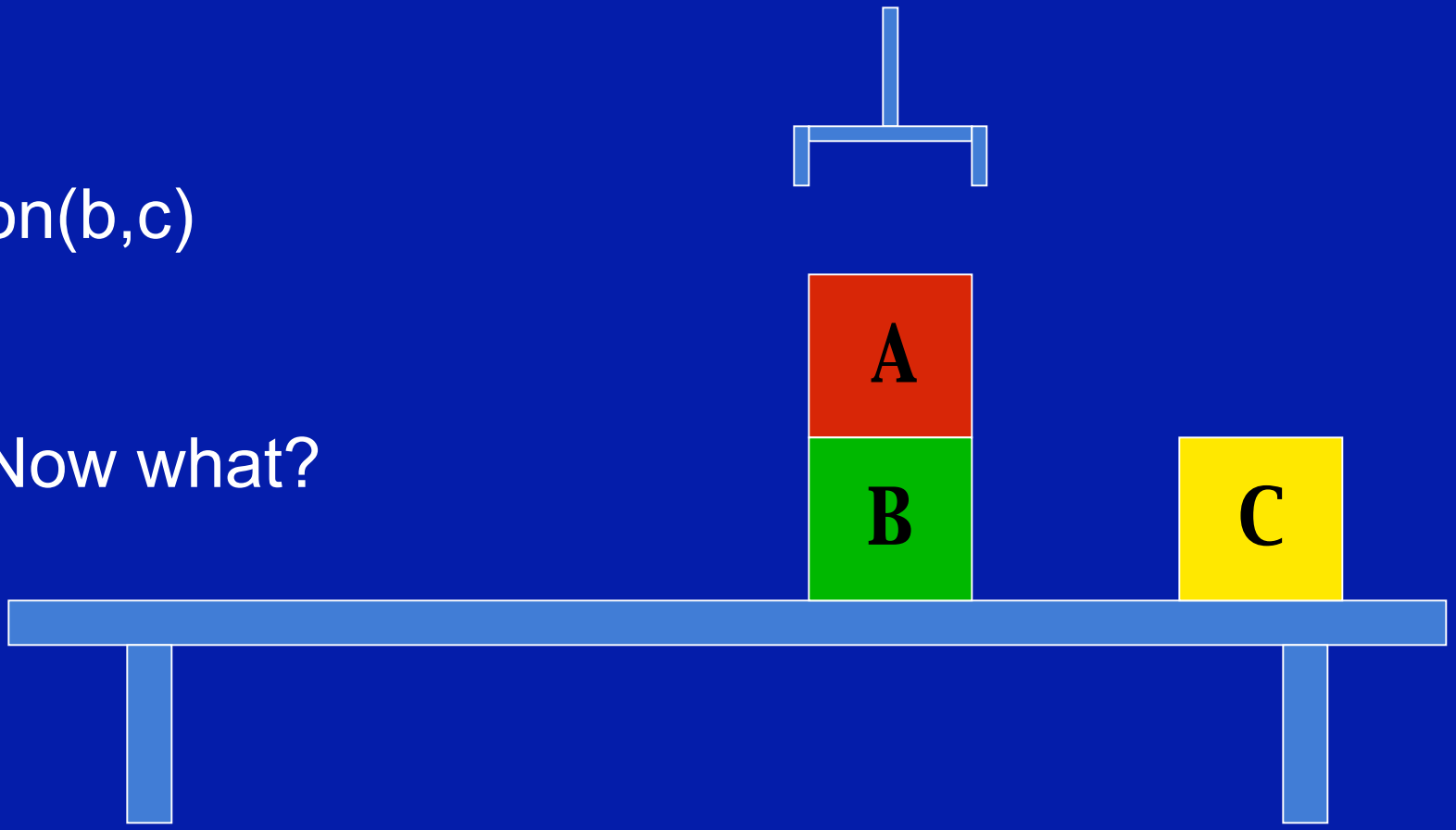


Why doesn't it work always?

goals:

on(b,c)

Now what?

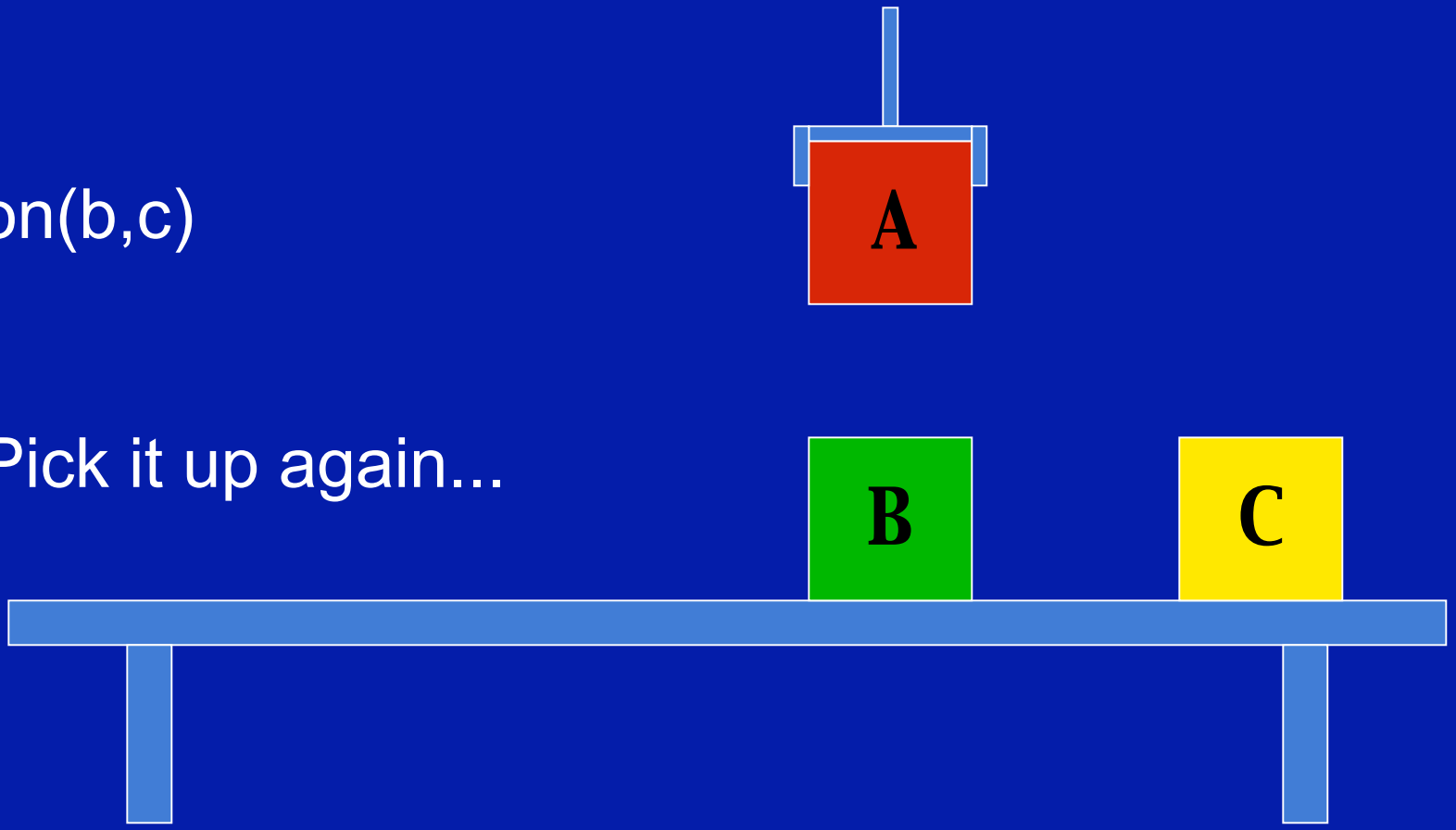


Why doesn't it work always?

goals:

on(b,c)

Pick it up again...

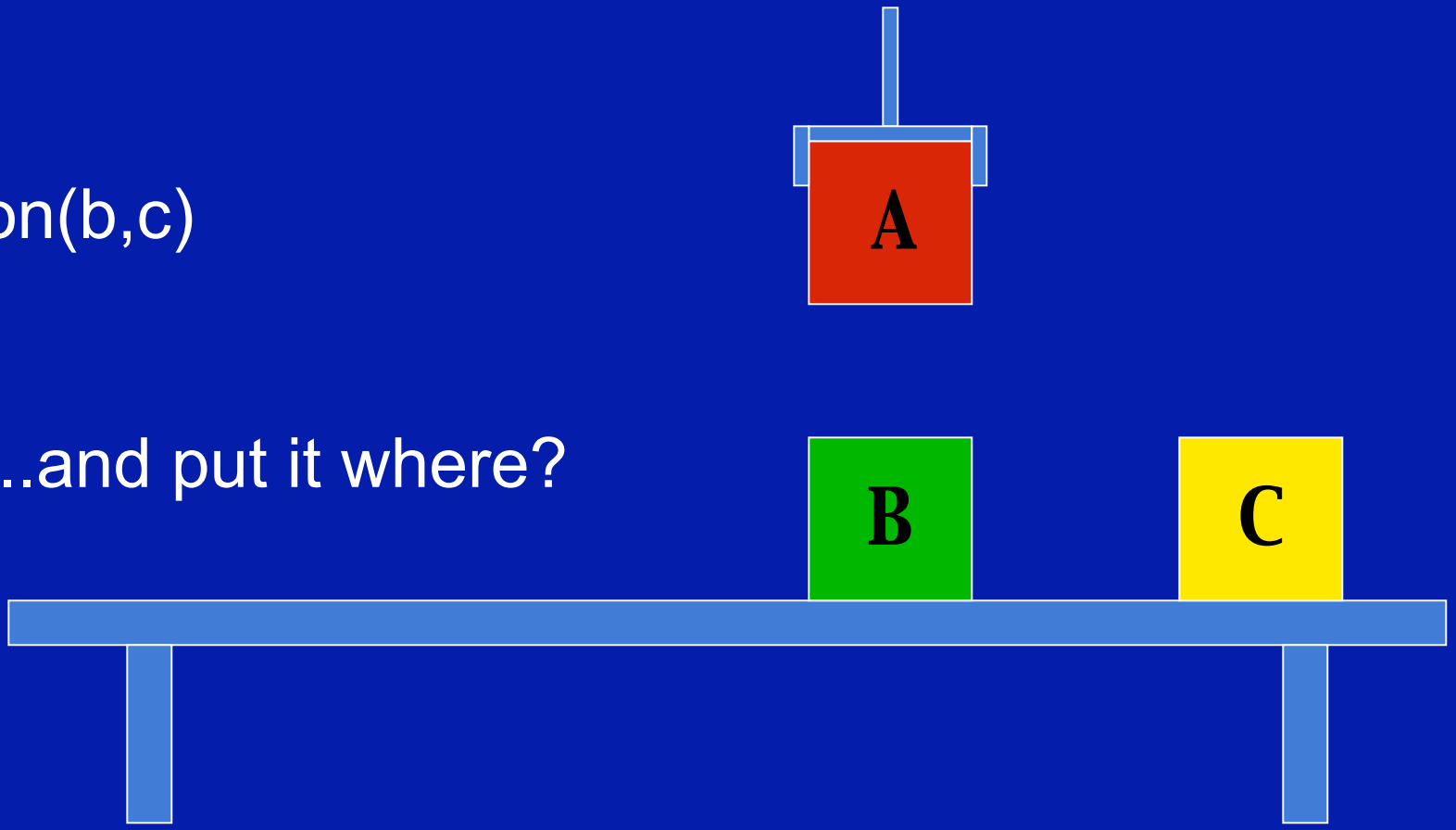


Why doesn't it work always?

goals:

on(b,c)

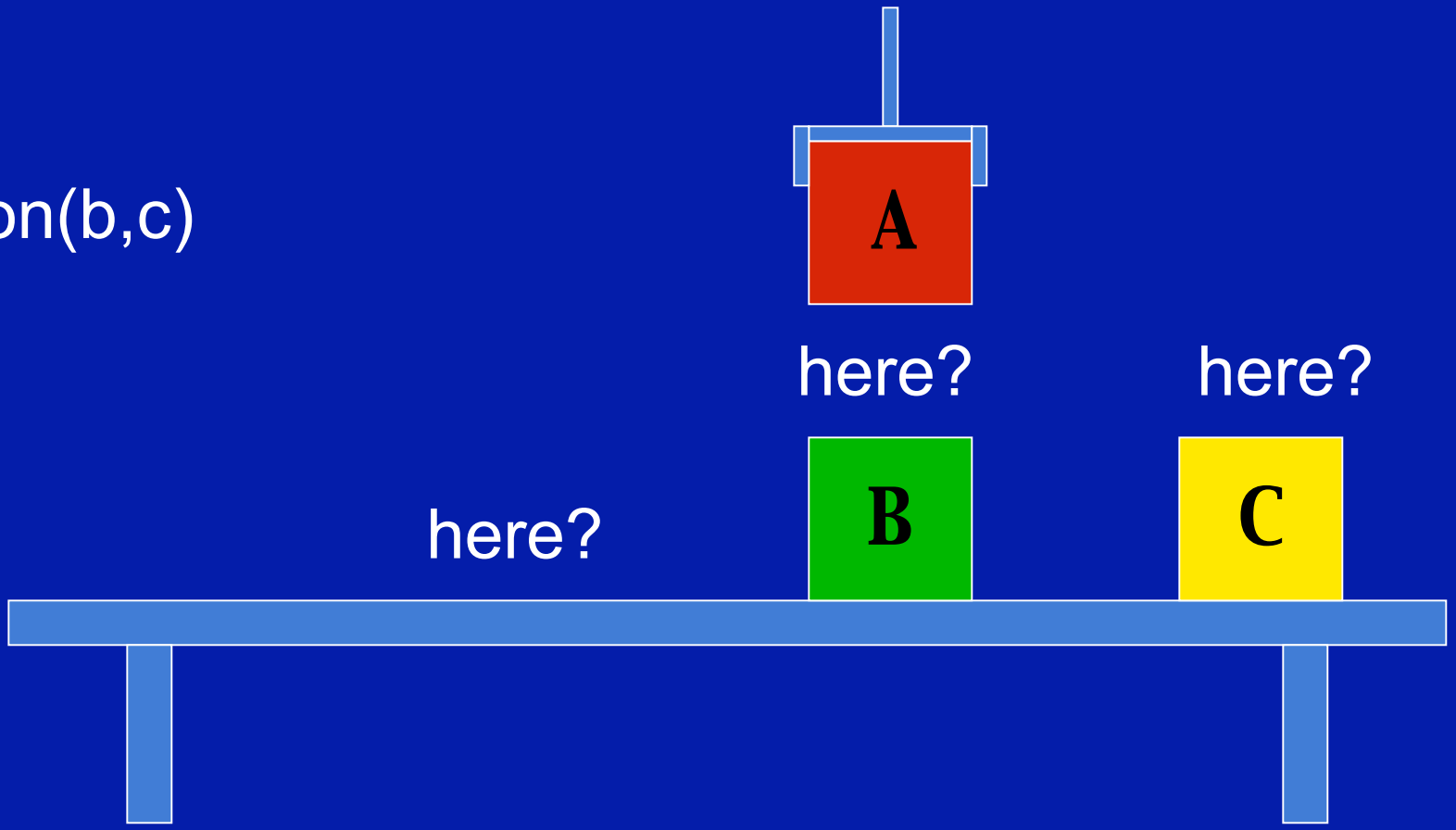
...and put it where?



Why doesn't it work always?

goals:

on(b,c)



Why doesn’t it work always?

Because satisfying one goal sometimes interferes with another.

So we employ heuristics to resolve the conflicts.

An example

We have two goals:

```
goals:      on(a,b)          on(b,c)
```

An example

means-ends analysis says choose two operators whose results are on(a,b) and on(b,c)

```
goals:      on(a,b)          on(b,c)
```

An example

those would be stack(a,b) and stack(b,c), with their associated preconditions/postconditions

pre:	clear(b)	clear(c)
	holding(a)	holding(b)
op:	stack(a,b)	stack(b,c)
post:	armempty	armempty
	on(a,b)	on(b,c)
	~clear(b)	~clear(c)
	~holding(a)	~holding(b)
goals:	on(a,b)	on(b,c)

An example

but two of the preconditions remain unsatisfied: holding(a) and holding(b)...

pre:	clear(b)	clear(c)
	holding(a)	holding(b)
op:	stack(a,b)	stack(b,c)
post:	armempty	armempty
	on(a,b)	on(b,c)
	~clear(b)	~clear(c)
	~holding(a)	~holding(b)
goals:	on(a,b)	on(b,c)

An example

...so means-ends analysis says choose operators whose results are holding(a) and holding(b)...

pre:	clear(b)	clear(c)
	holding(a)	holding(b)
op:	stack(a,b)	stack(b,c)
post:	armempty	armempty
	on(a,b)	on(b,c)
	~clear(b)	~clear(c)
	~holding(a)	~holding(b)
goals:	on(a,b)	on(b,c)

An example

...and those would be pickup(a) and pickup(b)

pre:	clear(a)	clear(b)
	ontable(a)	ontable(b)
	armempty	armempty
op:	pickup(a)	pickup(b)
post:	~ontable(a)	~ontable(b)
	~armempty	~armempty
	holding(a)	holding(b)
pre:	clear(b)	clear(c)
	holding(a)	holding(b)
op:	stack(a,b)	stack(b,c)
post:	armempty	armempty
	on(a,b)	on(b,c)
	~clear(b)	~clear(c)
	~holding(a)	~holding(b)
goals:	on(a,b)	on(b,c)

An example

we could now execute either of these plans but the one on the left prevents success of the one on the right

pre:	clear(a)	clear(b)
	ontable(a)	ontable(b)
	armempty	armempty
op:	pickup(a)	pickup(b)
post:	~ontable(a)	~ontable(b)
	~armempty	~armempty
	holding(a)	holding(b)
pre:	clear(b)	clear(c)
	holding(a)	holding(b)
op:	stack(a,b)	stack(b,c)
post:	armempty	armempty
	on(a,b)	on(b,c)
	~clear(b)	~clear(c)
	~holding(a)	~holding(b)
goals:	on(a,b)	on(b,c)

An example

but we at least know that pickup(a) comes before stack(a,b) and pickup(b) comes before stack(b,c)

pre:	clear(a)	clear(b)	pickup(a)
	ontable(a)	ontable(b)	
	armempty	armempty	stack(a,b)
op:	pickup(a)	pickup(b)	
post:	~ontable(a)	~ontable(b)	pickup(b)
	~armempty	~armempty	
	holding(a)	holding(b)	stack(b,c)
pre:	clear(b)	clear(c)	
	holding(a)	holding(b)	
op:	stack(a,b)	stack(b,c)	
post:	armempty	armempty	
	on(a,b)	on(b,c)	
	~clear(b)	~clear(c)	
	~holding(a)	~holding(b)	
goals:	on(a,b)	on(b,c)	

An example

we see that ~clear(b) on the left would prevent clear(b) on the right...

pre:	clear(a)	clear(b)	pickup(a)
	ontable(a)	ontable(b)	
	armempty	armempty	stack(a,b)
op:	pickup(a)	pickup(b)	
post:	~ontable(a)	~ontable(b)	pickup(b)
	~armempty	~armempty	
	holding(a)	holding(b)	stack(b,c)
pre:	clear(b)	clear(c)	
	holding(a)	holding(b)	
op:	stack(a,b)	stack(b,c)	
post:	armempty	armempty	
	on(a,b)	on(b,c)	
	~clear(b)	~clear(c)	
	~holding(a)	~holding(b)	
goals:	on(a,b)	on(b,c)	

An example

so stack(a,b) can't come before pickup(b)

pre:	clear(a)	clear(b)	pickup(a)
	ontable(a)	ontable(b)	
	armempty	armempty	stack(a,b)
op:	pickup(a)	pickup(b)	
post:	~ontable(a)	~ontable(b)	pickup(b)
	~armempty	~armempty	
	holding(a)	holding(b)	stack(b,c)
pre:	clear(b)	clear(c)	
	holding(a)	holding(b)	
op:	stack(a,b)	stack(b,c)	
post:	armempty	armempty	
	on(a,b)	on(b,c)	
	~clear(b)	~clear(c)	
	~holding(a)	~holding(b)	
goals:	on(a,b)	on(b,c)	

An example

so stack(a,b) can't come before pickup(b)

pre:	clear(a)	clear(b)	pickup(a)
	ontable(a)	ontable(b)	
	armempty	armempty	pickup(b)
op:	pickup(a)	pickup(b)	
post:	~ontable(a)	~ontable(b)	stack(a,b)
	~armempty	~armempty	
	holding(a)	holding(b)	stack(b,c)
pre:	clear(b)	clear(c)	
	holding(a)	holding(b)	
op:	stack(a,b)	stack(b,c)	
post:	armempty	armempty	
	on(a,b)	on(b,c)	
	~clear(b)	~clear(c)	
	~holding(a)	~holding(b)	
goals:	on(a,b)	on(b,c)	

An example

and you can't pickup(b) then stack(a,b)

pre:	clear(a)	clear(b)	pickup(a)
	ontable(a)	ontable(b)	
	armempty	armempty	pickup(b)
op:	pickup(a)	pickup(b)	
post:	~ontable(a)	~ontable(b)	stack(a,b)
	~armempty	~armempty	
	holding(a)	holding(b)	stack(b,c)
pre:	clear(b)	clear(c)	
	holding(a)	holding(b)	
op:	stack(a,b)	stack(b,c)	
post:	armempty	armempty	
	on(a,b)	on(b,c)	
	~clear(b)	~clear(c)	
	~holding(a)	~holding(b)	
goals:	on(a,b)	on(b,c)	

An example

so you can promote pickup(b) again

pre:	clear(a)	clear(b)	pickup(b)
	ontable(a)	ontable(b)	
	armempty	armempty	pickup(a)
op:	pickup(a)	pickup(b)	
post:	~ontable(a)	~ontable(b)	stack(a,b)
	~armempty	~armempty	
	holding(a)	holding(b)	stack(b,c)
pre:	clear(b)	clear(c)	
	holding(a)	holding(b)	
op:	stack(a,b)	stack(b,c)	
post:	armempty	armempty	
	on(a,b)	on(b,c)	
	~clear(b)	~clear(c)	
	~holding(a)	~holding(b)	
goals:	on(a,b)	on(b,c)	

An example

pickup(b) can't come right before pickup(a)

pre:	clear(a)	clear(b)	pickup(b)
	ontable(a)	ontable(b)	
	armempty	armempty	pickup(a)
op:	pickup(a)	pickup(b)	
post:	~ontable(a)	~ontable(b)	stack(a,b)
	~armempty	~armempty	
	holding(a)	holding(b)	stack(b,c)
pre:	clear(b)	clear(c)	
	holding(a)	holding(b)	
op:	stack(a,b)	stack(b,c)	
post:	armempty	armempty	
	on(a,b)	on(b,c)	
	~clear(b)	~clear(c)	
	~holding(a)	~holding(b)	
goals:	on(a,b)	on(b,c)	

An example

so some other action has to immediately follow pickup(b), and it can't be stack(a,b) because it must follow pickup(a)

pre:	clear(a)	clear(b)	pickup(b)
	ontable(a)	ontable(b)	
	armempty	armempty	pickup(a)
op:	pickup(a)	pickup(b)	
post:	~ontable(a)	~ontable(b)	stack(a,b)
	~armempty	~armempty	
	holding(a)	holding(b)	stack(b,c)
pre:	clear(b)	clear(c)	
	holding(a)	holding(b)	
op:	stack(a,b)	stack(b,c)	
post:	armempty	armempty	
	on(a,b)	on(b,c)	
	~clear(b)	~clear(c)	
	~holding(a)	~holding(b)	
goals:	on(a,b)	on(b,c)	

An example

the only action left is stack(b,c), so it's promoted up in front of pickup(a)...it can't go any higher

pre:	clear(a)	clear(b)	pickup(b)
	ontable(a)	ontable(b)	
	armempty	armempty	pickup(a)
op:	pickup(a)	pickup(b)	
post:	~ontable(a)	~ontable(b)	stack(a,b)
	~armempty	~armempty	
	holding(a)	holding(b)	stack(b,c)
pre:	clear(b)	clear(c)	
	holding(a)	holding(b)	
op:	stack(a,b)	stack(b,c)	
post:	armempty	armempty	
	on(a,b)	on(b,c)	
	~clear(b)	~clear(c)	
	~holding(a)	~holding(b)	
goals:	on(a,b)	on(b,c)	

An example

the only action left is stack(b,c), so it's promoted up in front of pickup(a)...it can't go any higher

pre:	clear(a)	clear(b)	pickup(b)
	ontable(a)	ontable(b)	
	armempty	armempty	stack(b,c)
op:	pickup(a)	pickup(b)	
post:	~ontable(a)	~ontable(b)	pickup(a)
	~armempty	~armempty	
	holding(a)	holding(b)	stack(a,b)
pre:	clear(b)	clear(c)	
	holding(a)	holding(b)	
op:	stack(a,b)	stack(b,c)	
post:	armempty	armempty	
	on(a,b)	on(b,c)	
	~clear(b)	~clear(c)	
	~holding(a)	~holding(b)	
goals:	on(a,b)	on(b,c)	

An example

no more conflicts...that's our plan

pre:	clear(a)	clear(b)	pickup(b)
	ontable(a)	ontable(b)	
	armempty	armempty	stack(b,c)
op:	pickup(a)	pickup(b)	
post:	~ontable(a)	~ontable(b)	pickup(a)
	~armempty	~armempty	
	holding(a)	holding(b)	stack(a,b)
pre:	clear(b)	clear(c)	
	holding(a)	holding(b)	
op:	stack(a,b)	stack(b,c)	
post:	armempty	armempty	
	on(a,b)	on(b,c)	
	~clear(b)	~clear(c)	
	~holding(a)	~holding(b)	
goals:	on(a,b)	on(b,c)	