Controlling Robots via Large Language Models

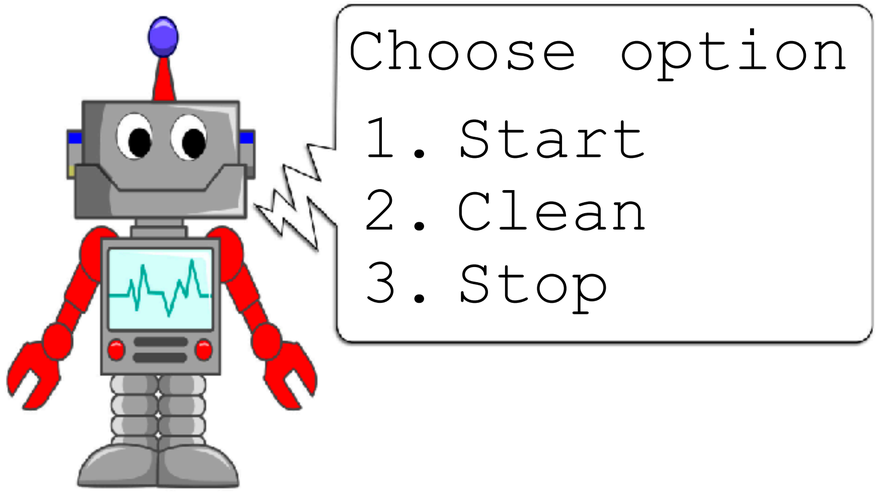
Sanjiban Choudhury

Cornell Bowers CIS
Computer Science

The Problem

The way we program robots today is ... rigid!

Engineers hand-craft behaviors



Ship robot

Frustrate users!

Cannot be flexibly re-programmed by everyday users

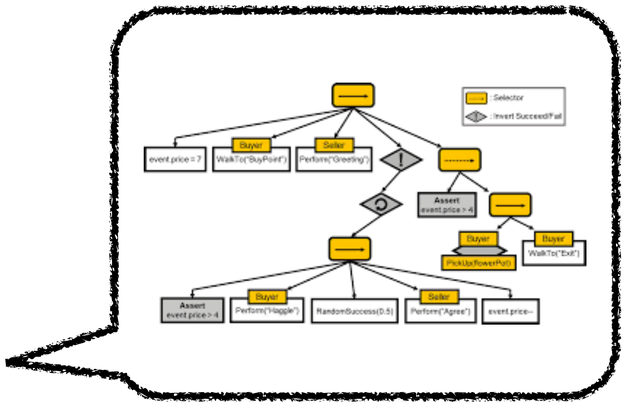
Instead of explicitly
engineering behaviors

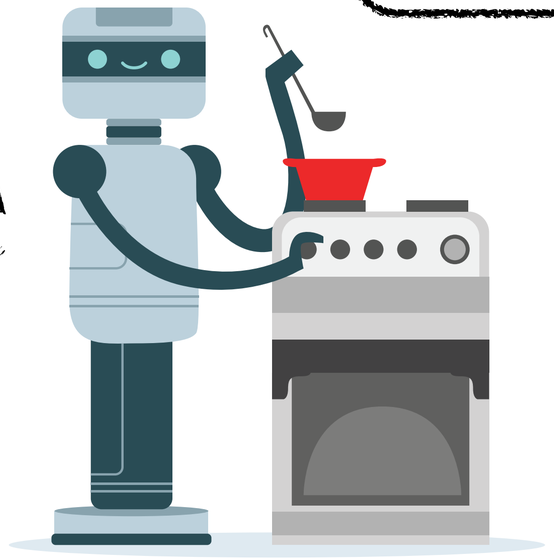
Can we implicitly program
robots via natural interactions?

Programming via natural interactions

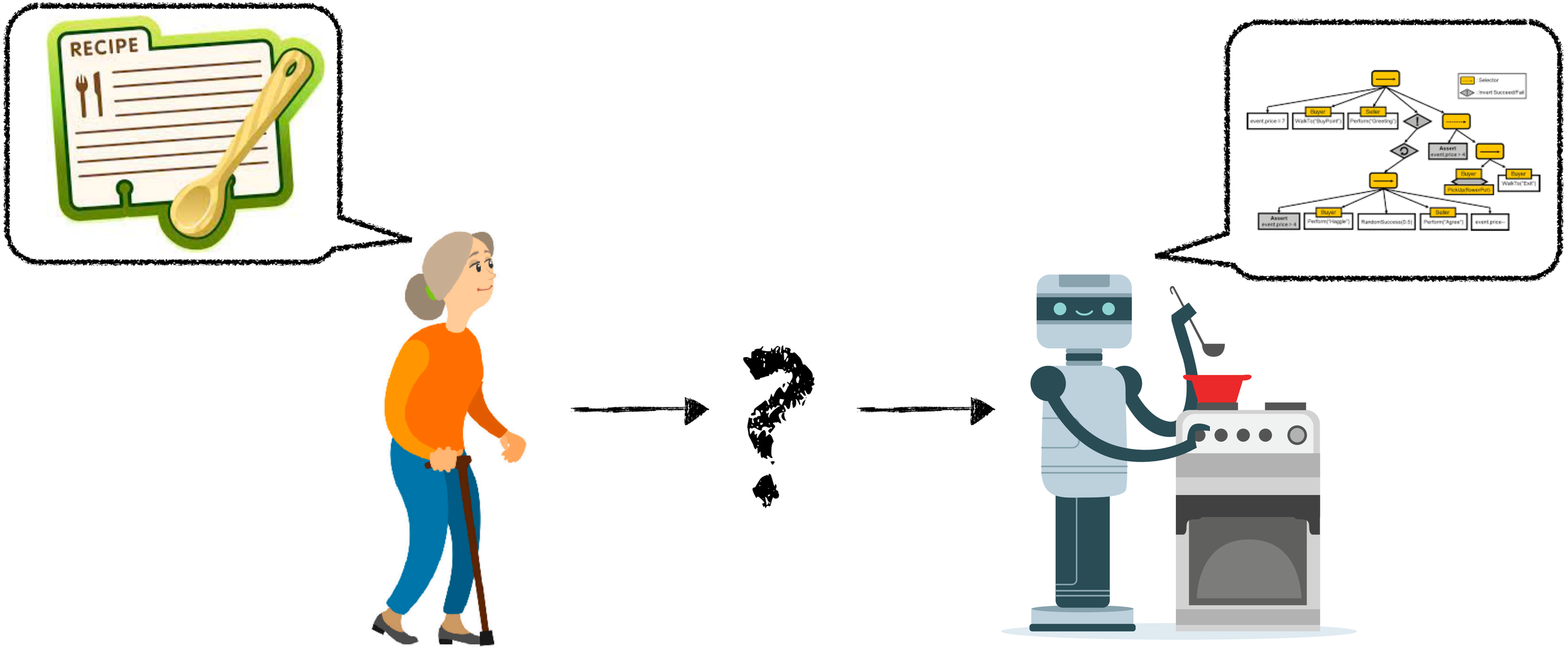
Demonstrations,
Language

Feedback,
Interactive QA

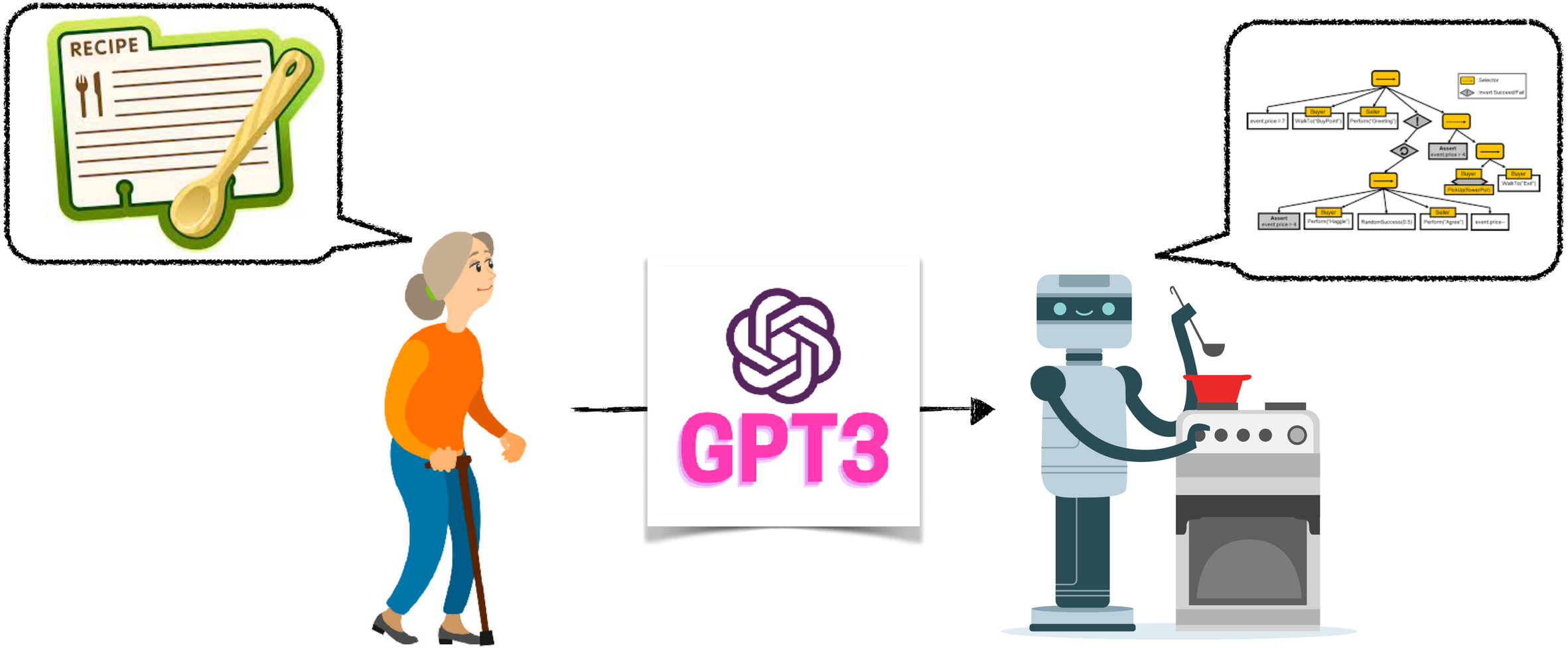




Question: How do we translate between humans and robots?



Large Language Models to the rescue!



LLMs parse language and plan tasks

Prompt: Here we have a book, 9 eggs, a laptop, a bottle and a nail. Please tell me how to stack them onto each other in a stable manner.

LLMs parse language and plan tasks

Prompt: Here we have a book, 9 eggs, a laptop, a bottle and a nail. Please tell me how to stack them onto each other in a stable manner.

GPT-4:

One possible way to stack the objects onto each other in a stable manner is:

- Place the book flat on a level surface, such as a table or a floor. The book will serve as the base of the stack and provide a large and sturdy support.
- Arrange the 9 eggs in a 3 by 3 square on top of the book, leaving some space between them. The eggs will
form a second layer and distribute the weight evenly. Make sure the eggs are not cracked or broken, and handle them gently to avoid dropping them.
- Place the laptop on top of the eggs, with the screen facing down and the keyboard facing up. The laptop will fit snugly within the boundaries of the book and the eggs, and its flat and rigid surface will provide a stable platform for the next layer. Make sure the laptop is closed and turned off, and do not press any keys or buttons.
- Place the bottle on top of the laptop, with the cap facing up and the bottom facing down. The bottle will add some height and balance to the stack, and its cylindrical shape will reduce the contact area with the laptop. Make sure the bottle is empty and closed, and do not shake or tilt it.
- Place the nail on top of the bottle cap, with the pointy end facing up and the flat end facing down. The nail will be the final and smallest object in the stack, and its sharp tip will prevent it from rolling or sliding off the bottle. Make sure the nail is clean and straight, and do not touch or move it.

An Example

HAL

Helping Out In the Kitchen

(**H**ome **A**pprentice **L**earner)

Activity!

Think-Pair-Share!

Think (30 sec): Think of all the steps to go from what the human said to the code the robot has to execute.

Pair: Find a partner

Share (45 sec): Partners exchange ideas

Human: “Help me make vegetable soup”

Robot:

```
go_to(SALT)

pick_up_item(SALT)

go_to(TABLE)

place_item_at(TABLE)
```

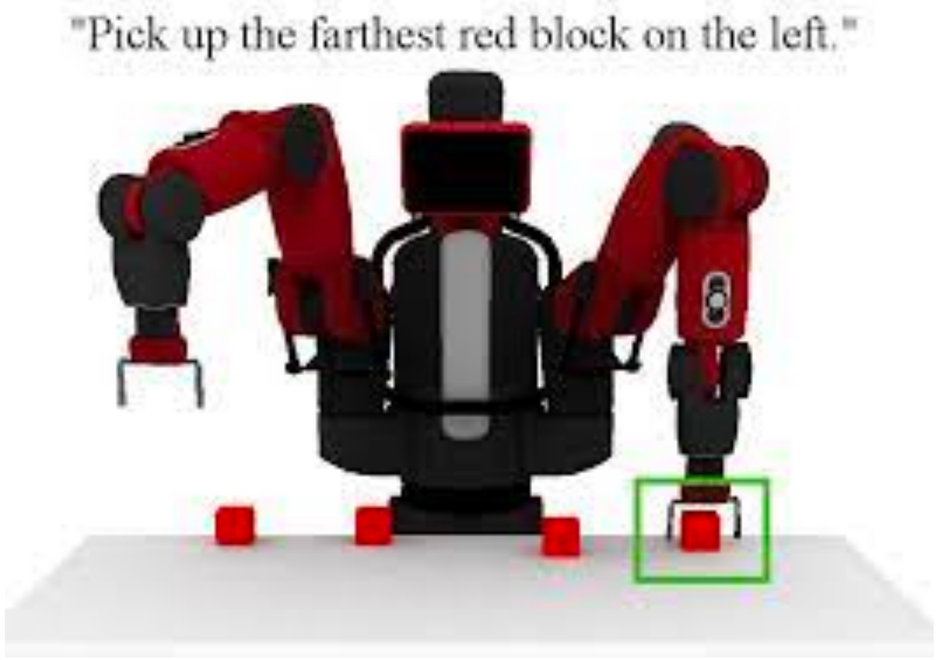
How things worked pre-LLM

Two Fundamental Challenges

Two Fundamental Challenges

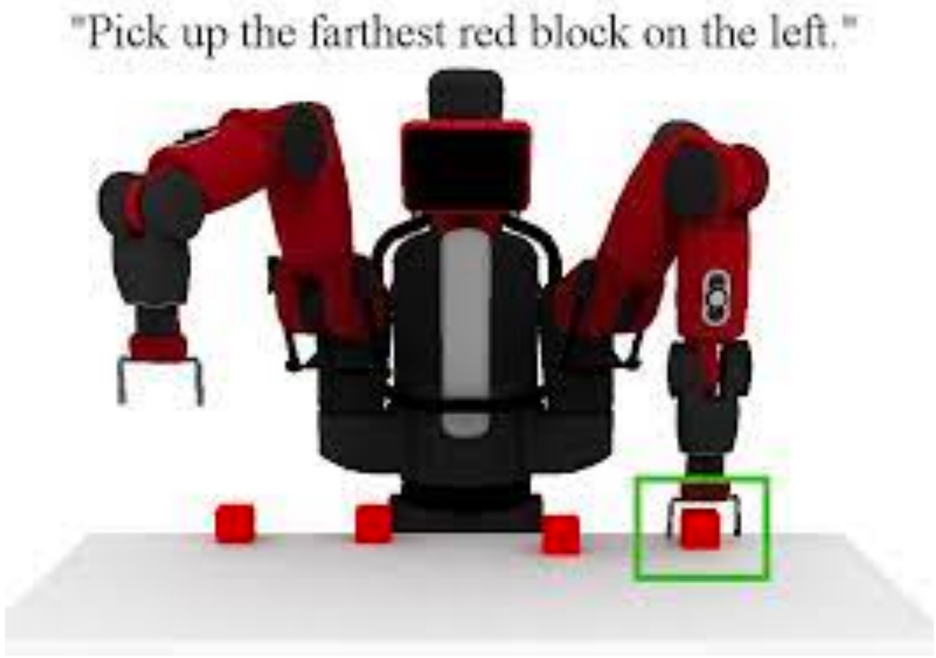
Challenge 1:

Ground natural language in robot state

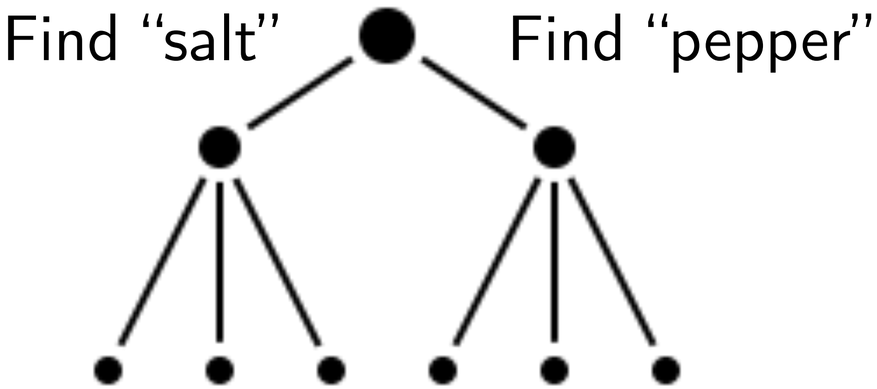


Two Fundamental Challenges

Challenge 1:
Ground natural language in robot state

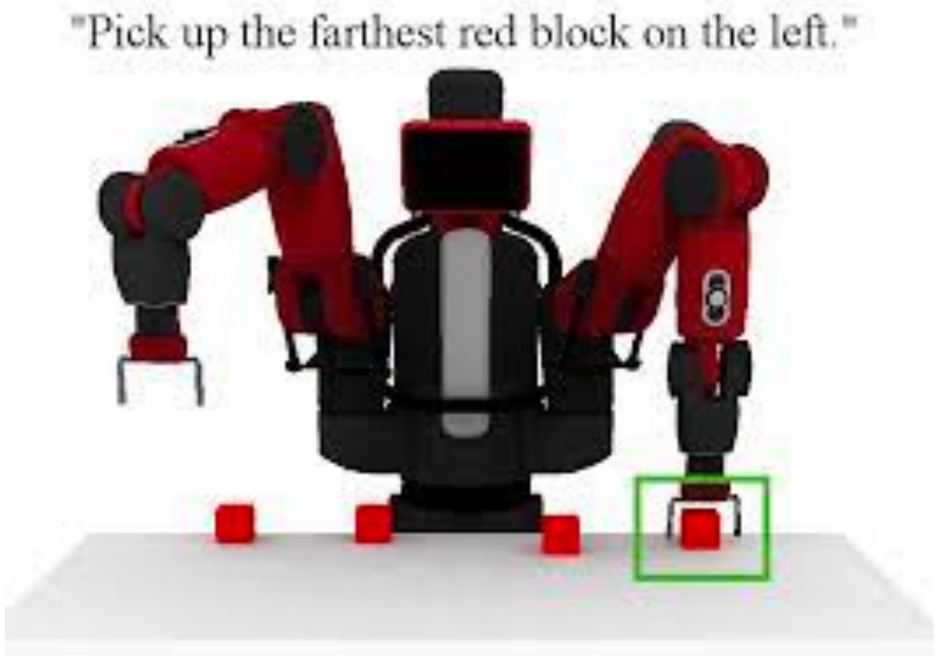


Challenge 2:
Planning actions to solve a task

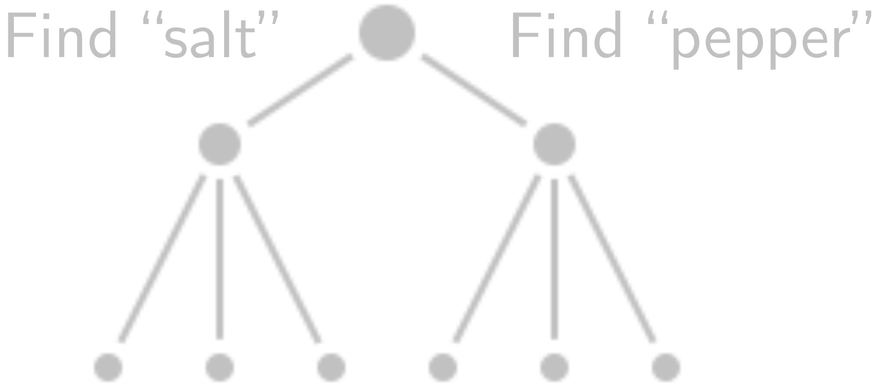


Two Fundamental Challenges

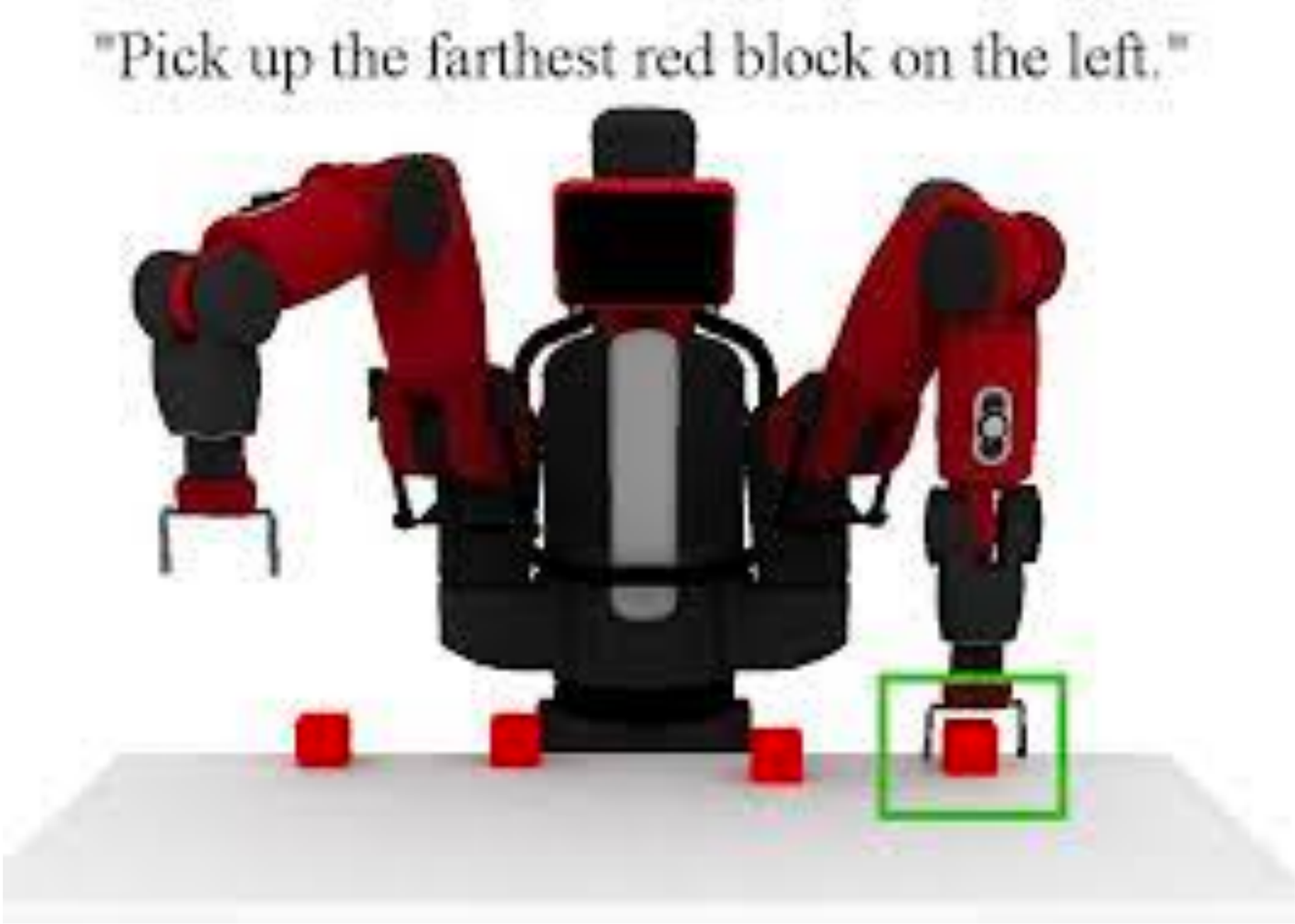
Challenge 1:
Ground natural language in robot state



Challenge 2:
Planning actions to solve a task



What is grounding? Why is it hard?

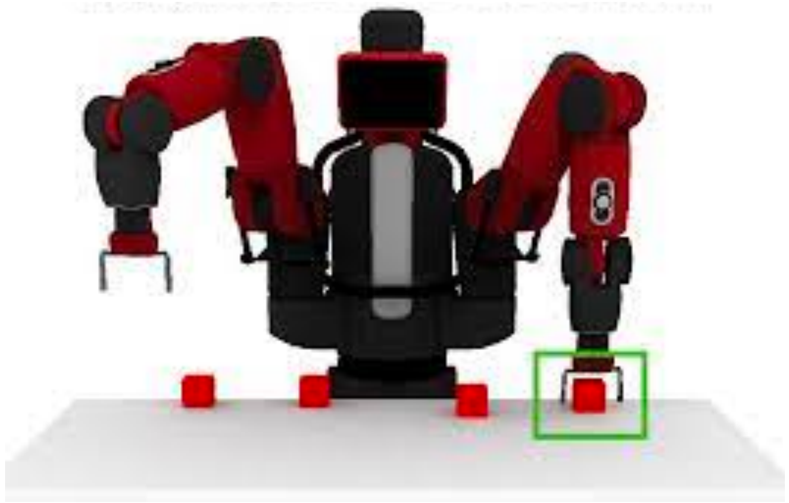


Grounding: Mapping language to robot's internal state

Natural Language → **MDP**

"Pick up the farthest red block"

$< S , A , R , \mathcal{T} >$

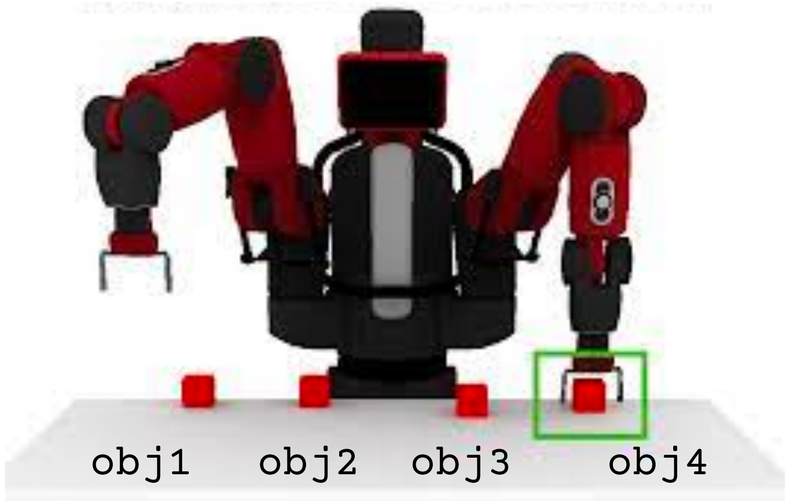


Grounding: Mapping language to robot's internal state

Natural Language → **MDP**

"Pick up the farthest red block"

$< S , A , R , \mathcal{T} >$



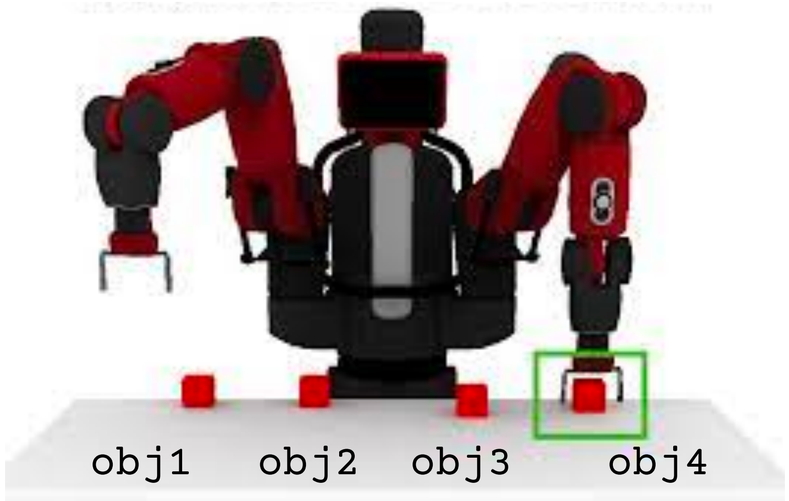

```
on('obj1','table')
on('obj2','table')
on('obj3','table')
on('obj4','table')
left('obj2','obj1')
left('obj3','obj2')
left('obj4','obj3')
...
```

Grounding: Mapping language to robot's internal state

Natural Language → **MDP**

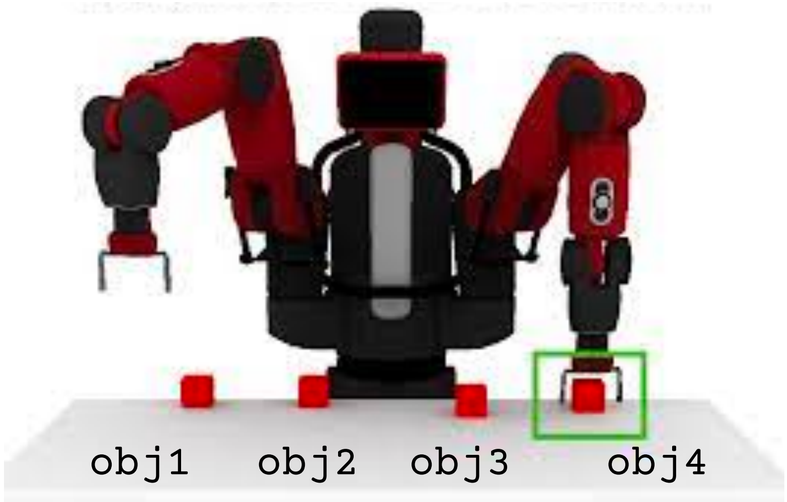
"Pick up the farthest red block"

$< S , A , R , \mathcal{T} >$



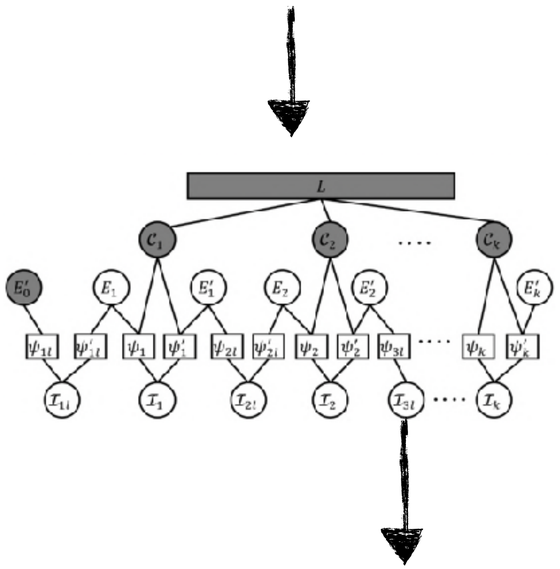
R(in(obj4, hand)) = +1

How did we solve grounding?



Complex graphical models!

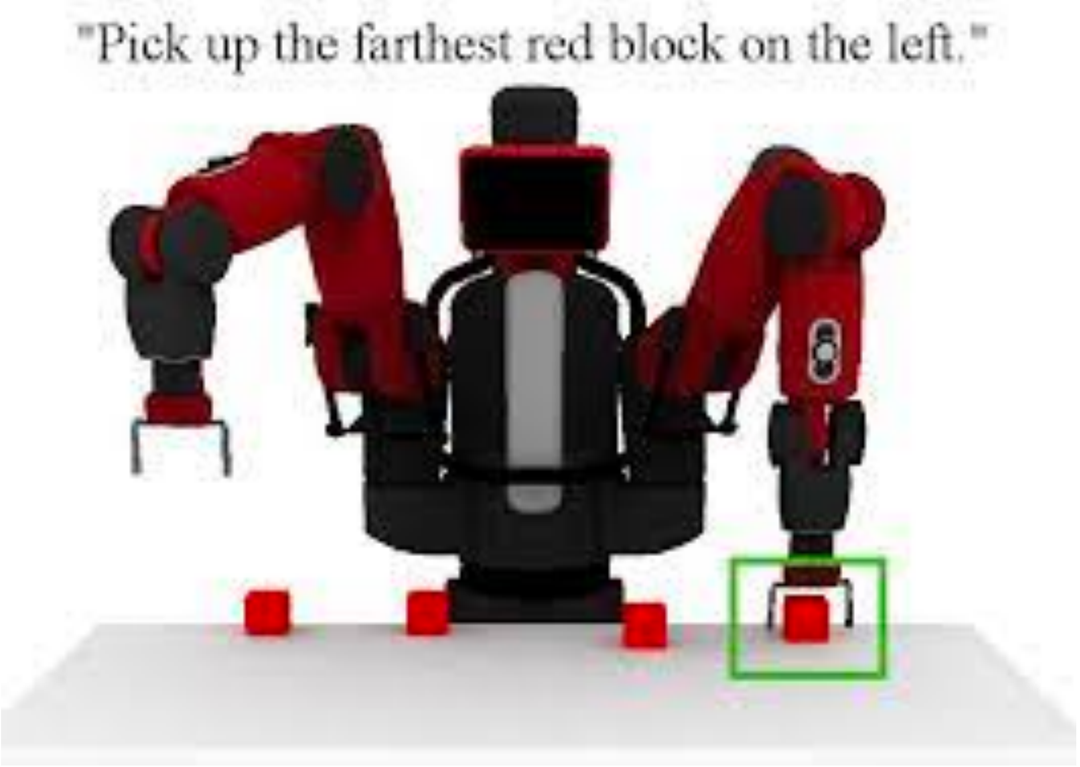
"Pick up the farthest red block"



R(in(obj4, hand)) = +1

Train this on small, custom robot datasets

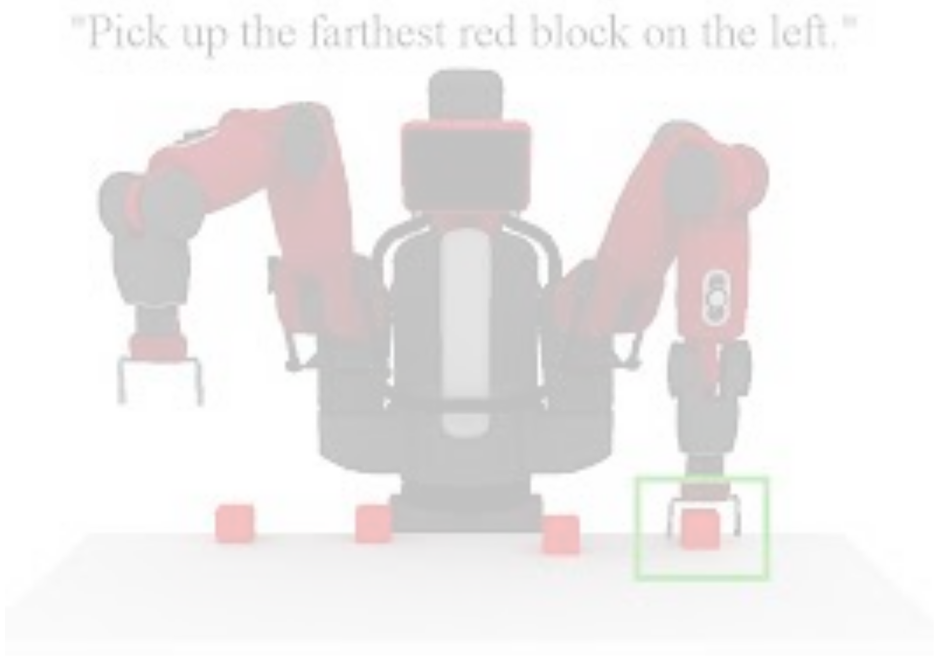
Why did this not scale?



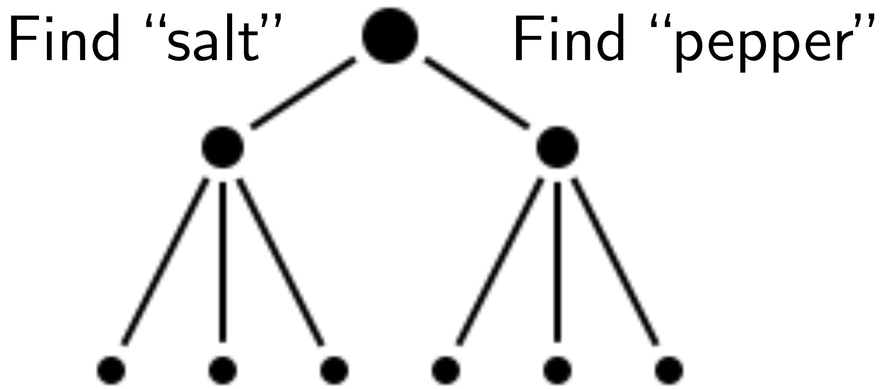
1. Failure to generalize to different human utterances
2. Failure to capture common sense
3. Failure to capture complex instructions (while loops)

Two Fundamental Challenges

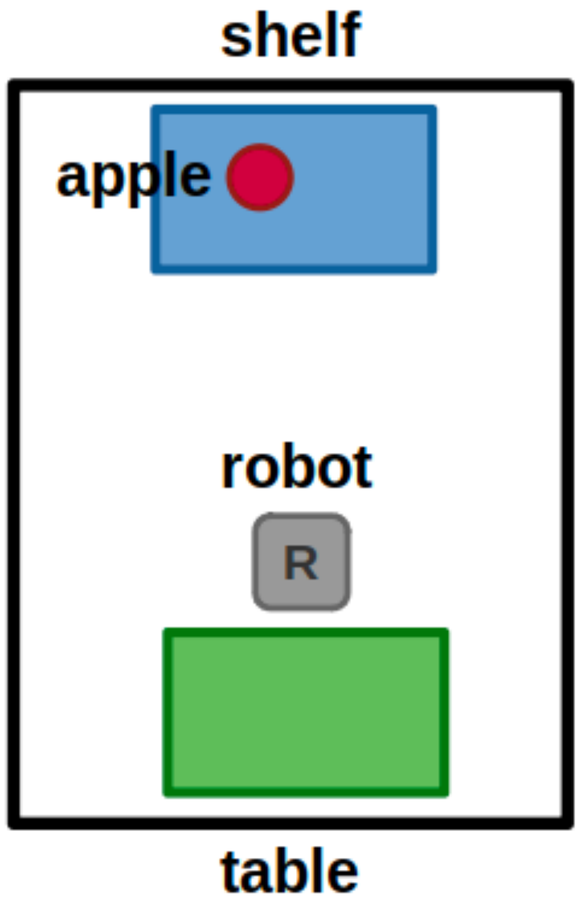
Challenge 1:
Ground natural language in robot state



Challenge 2:
Planning actions to solve a task

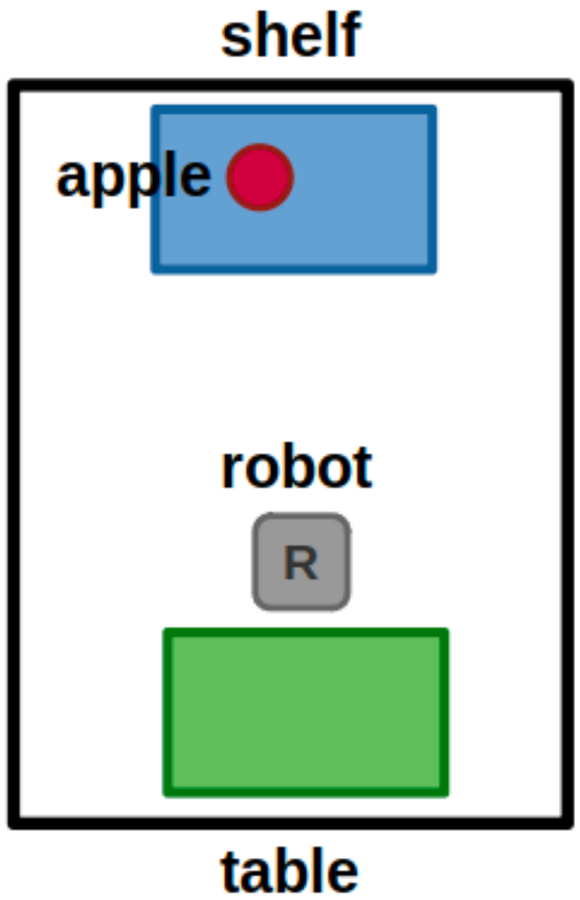


What is task planning? Why is it hard?



Take the apple from the shelf and put it on the table

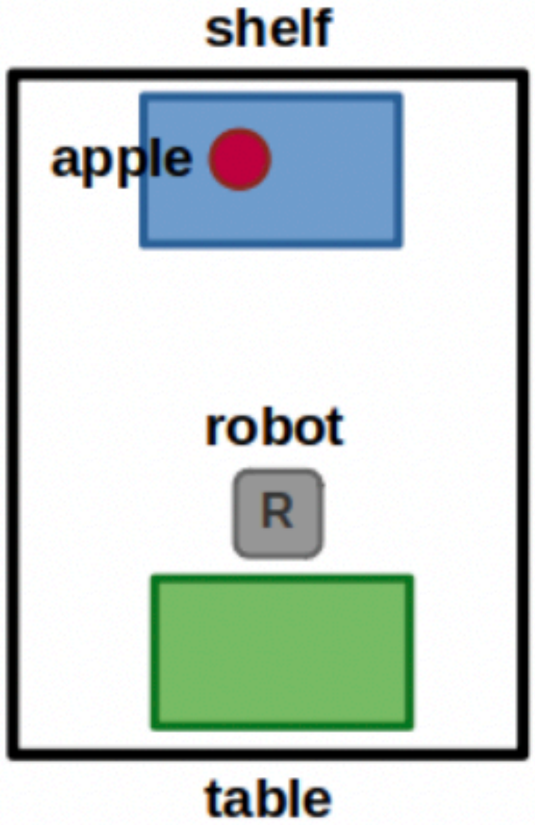
What is task planning? Why is it hard?



Take the apple from the shelf and put it on the table

1. Move to the shelf
2. Pick up the apple
3. Move back to the table
4. Place the apple

What is task planning? Why is it hard?

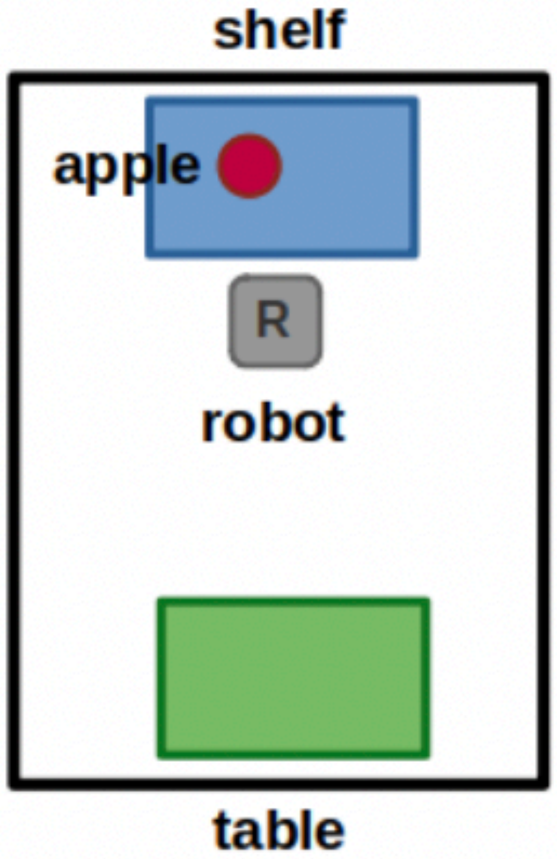


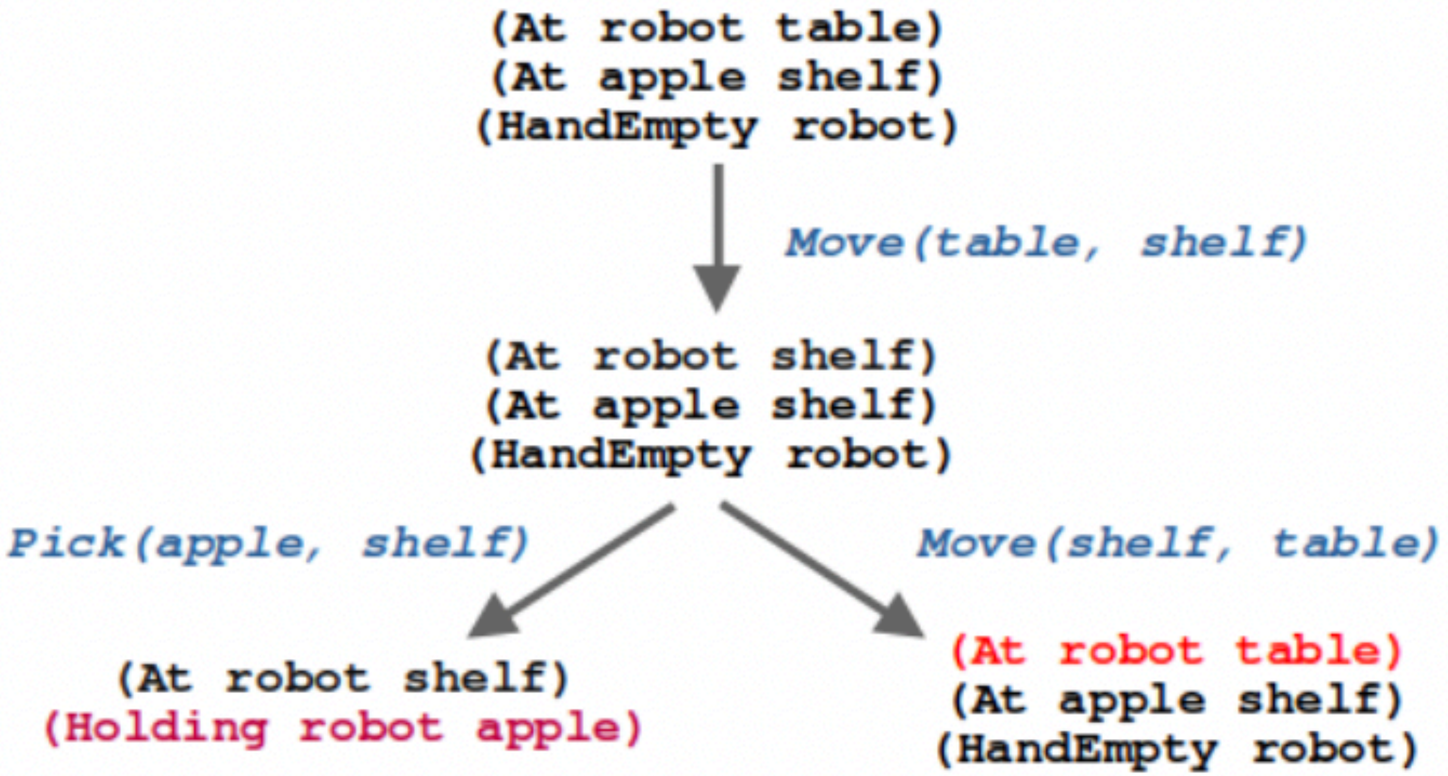
(At robot table)
(At apple shelf)
(HandEmpty robot)

Move(table, shelf)

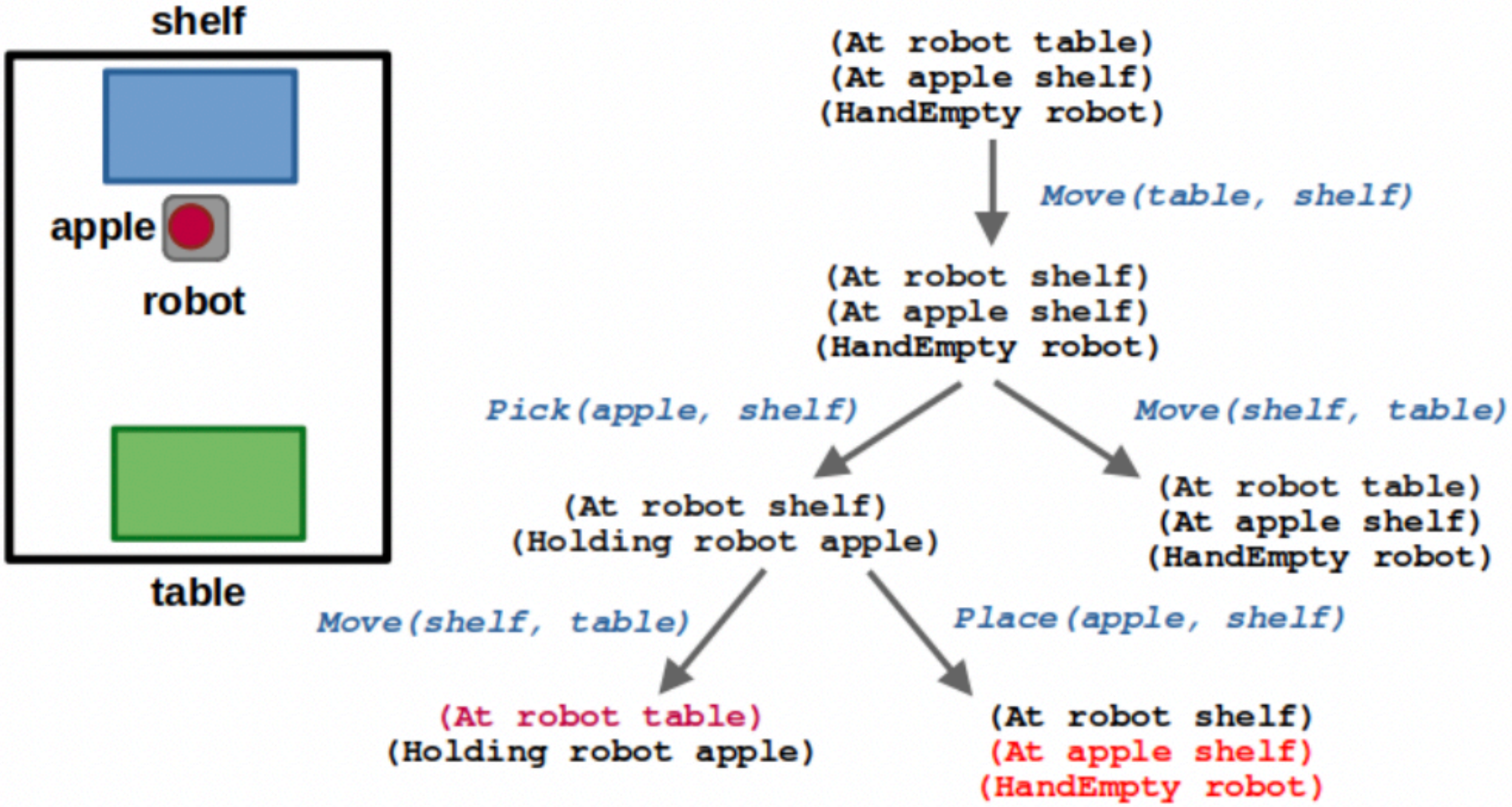
(At robot shelf)
(At apple shelf)
(HandEmpty robot)

What is task planning? Why is it hard?

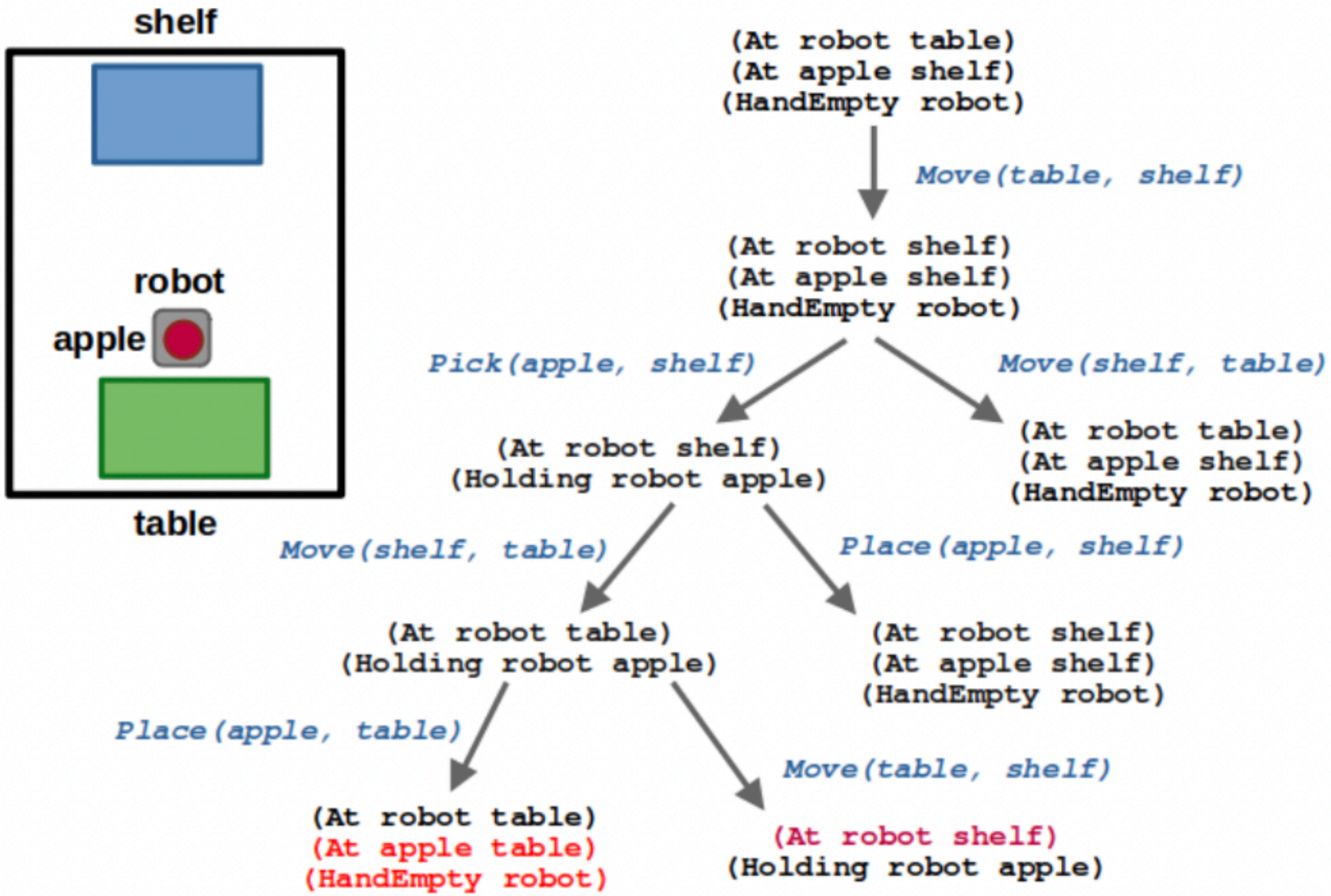




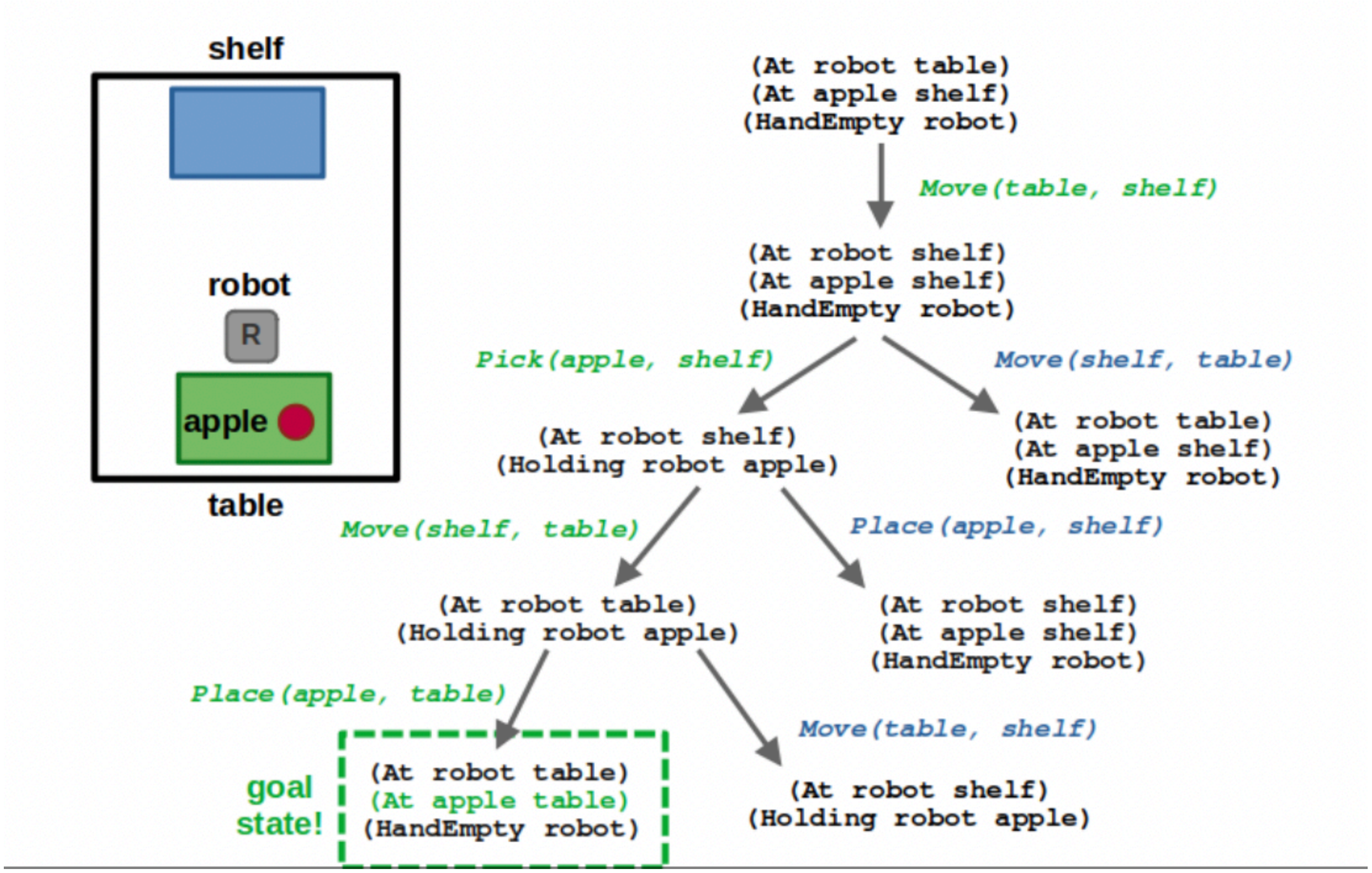
What is task planning? Why is it hard?



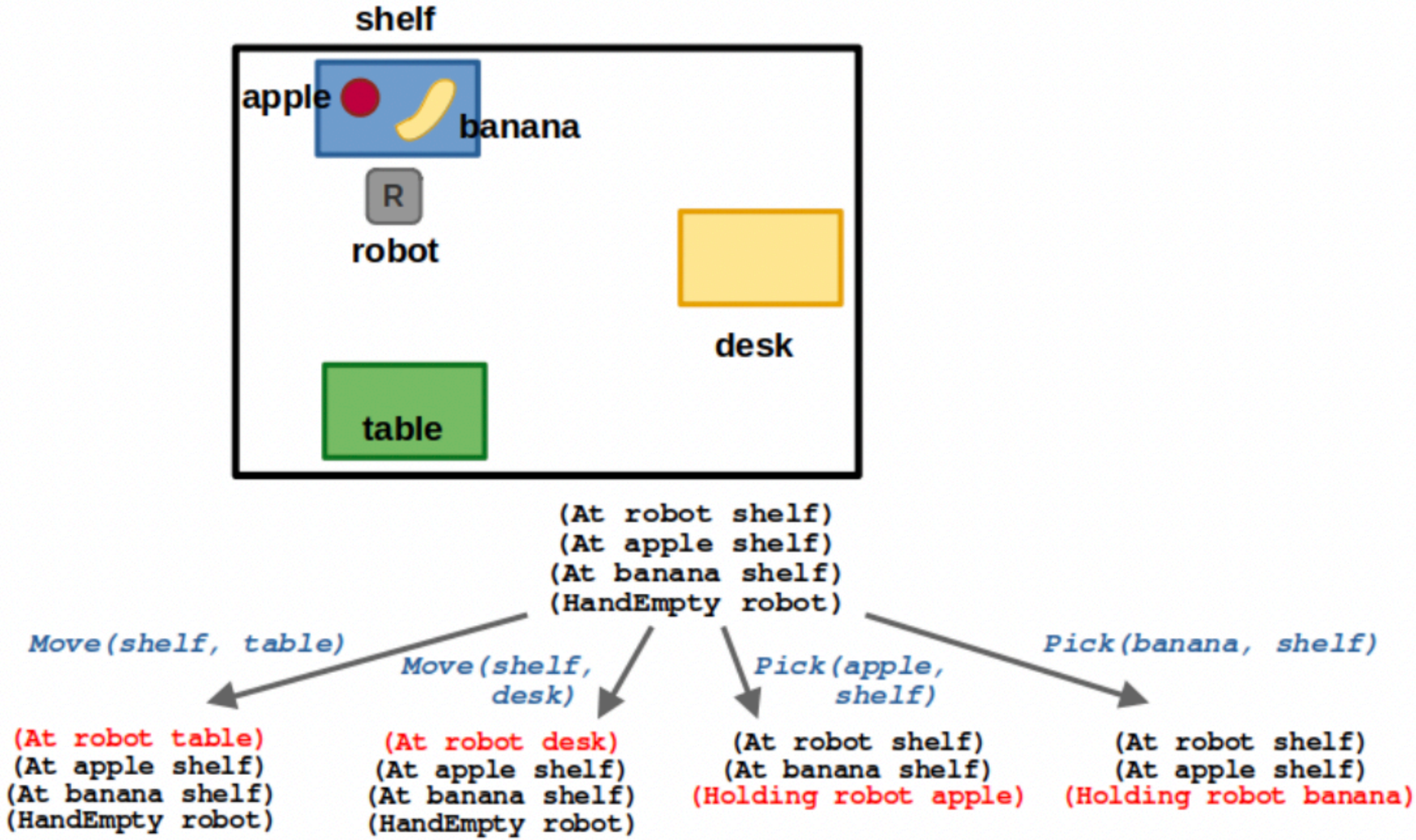
What is task planning? Why is it hard?



What is task planning? Why is it hard?



What is task planning? Why is it hard?



How did we solve it?

Good old fashioned search!

Lots of heuristics to make it real time

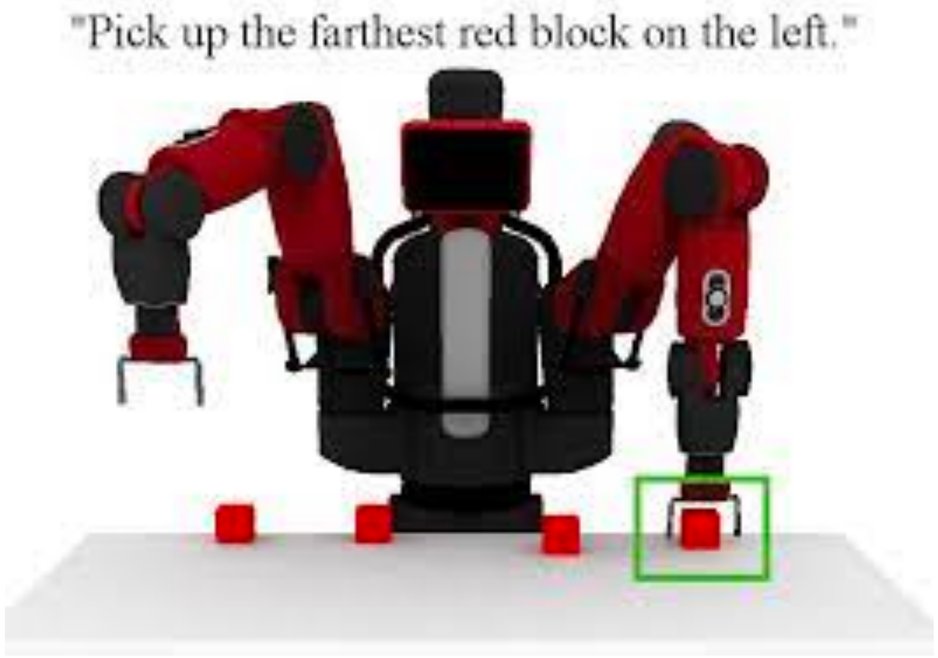
Why did it not scale?

Combinatorially large search tree

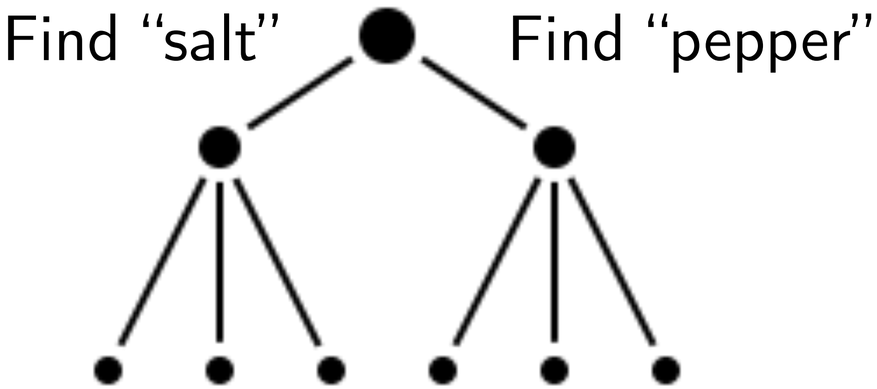
Had no notion of common sense

Two Fundamental Challenges

Challenge 1:
Ground natural language in robot state



Challenge 2:
Planning actions to solve a task



LARGE LANGUAGE MODELS

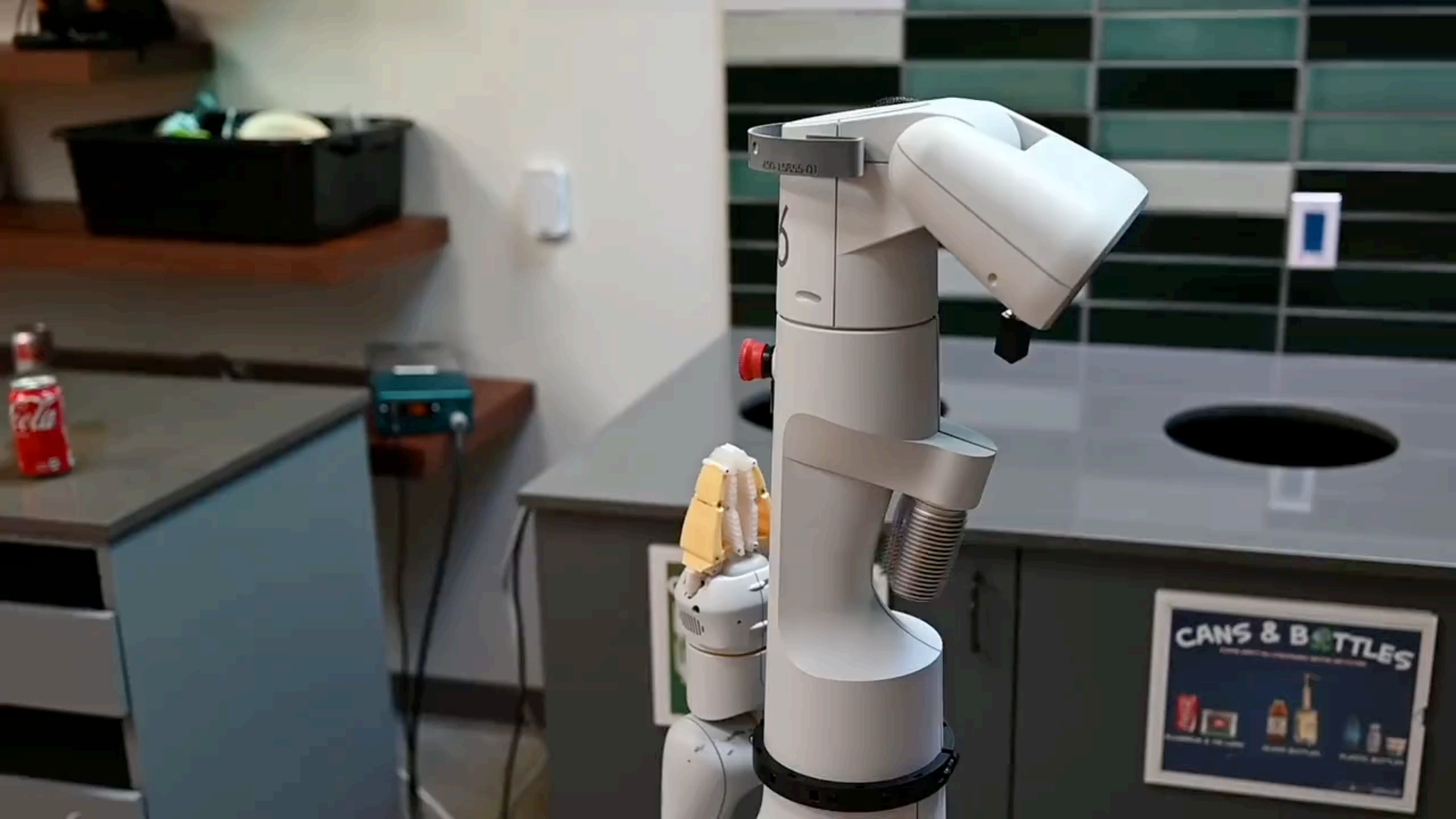
Episode IV

A NEW HOPE

Do As I Can, Not As I Say:
Grounding Language in Robotic Affordances

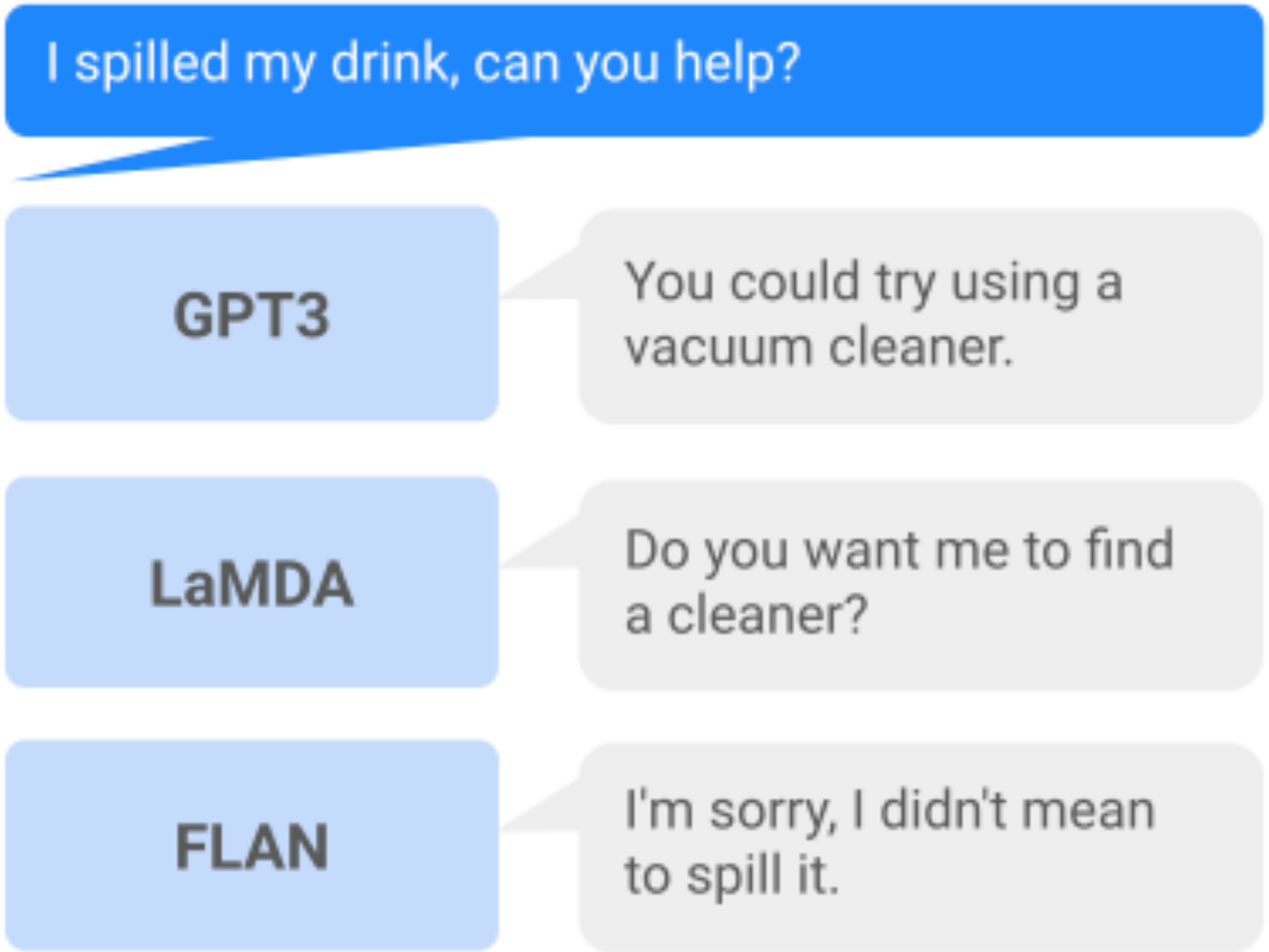
Michael Ahn* Anthony Brohan* Noah Brown* Yevgen Chebotar* Omar Cortes* Byron David* Chelsea Finn*
Chuyuan Fu* Keerthana Gopalakrishnan* Karol Hausman* Alex Herzog* Daniel Ho* Jasmine Hsu* Julian Ibarz*
Brian Ichter* Alex Irpan* Eric Jang* Rosario Jauregui Ruano* Kyle Jeffrey* Sally Jesmonth* Nikhil Joshi*
Ryan Julian* Dmitry Kalashnikov* Yuheng Kuang* Kuang-Huei Lee* Sergey Levine* Yao Lu* Linda Luu* Carolina Parada*
Peter Pastor* Jornell Quiambao* Kanishka Rao* Jarek Rettinghouse* Diego Reyes* Pierre Sermanet* Nicolas Sievers*
Clayton Tan* Alexander Toshev* Vincent Vanhoucke* Fei Xia* Ted Xiao* Peng Xu* Sichun Xu* Mengyuan Yan* Andy Zeng*

 Robotics at Google  Everyday Robots


CANS & BOTTLES

So ... we just ask an LLM to tell us what to do?

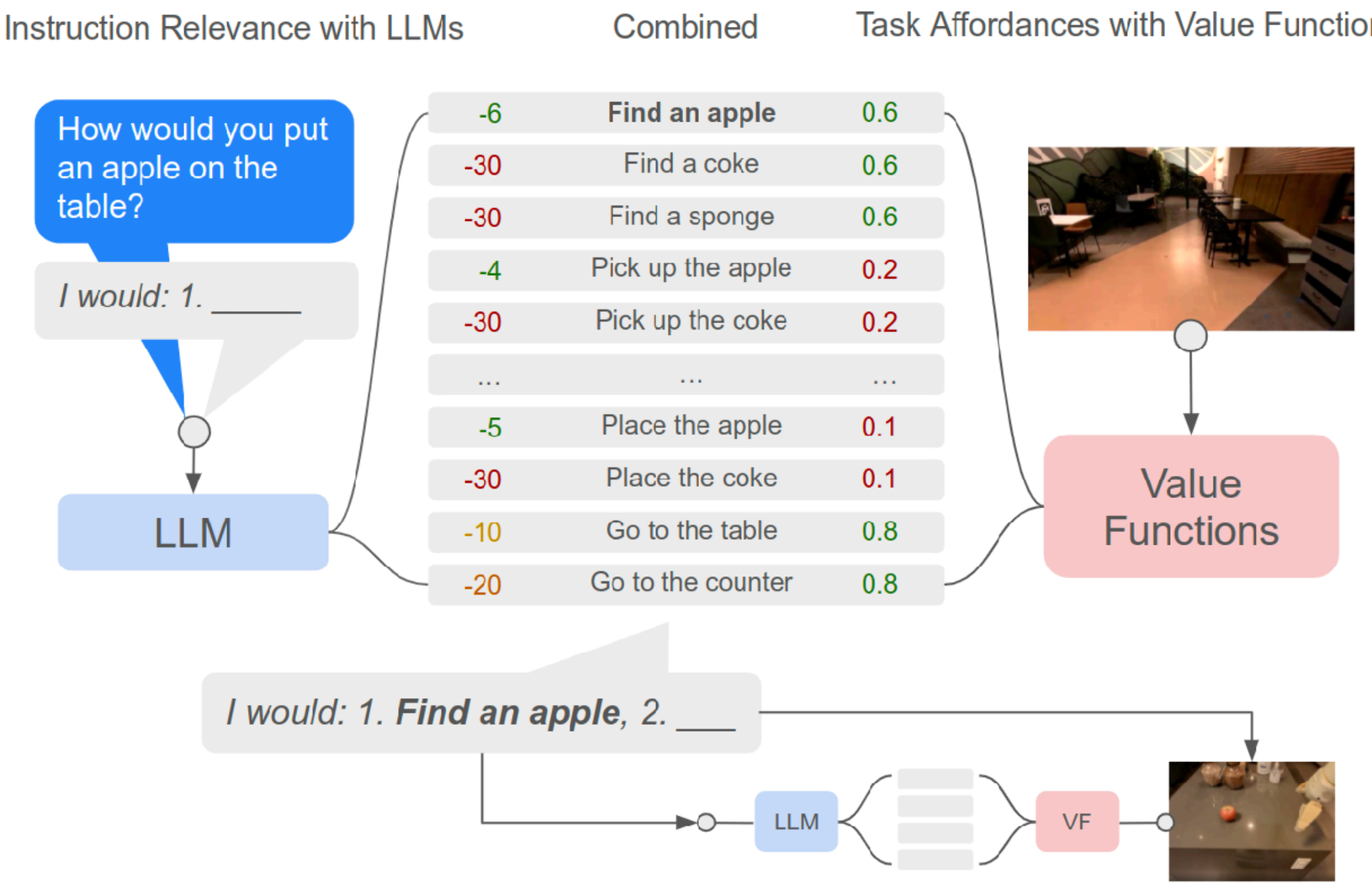
No! LLMs can say *anything* ..



Idea: Constrain LLM by what the robot can do (affordance)

Idea: Constrain LLM by what the robot can do (affordance)

The "SayCan" Approach




10x speed
COMPOSTABLES
CANS & BOTTLES

Language × Affordance

Combined Score

find a grapefruit soda

1.00

find a lime soda

0.01

find a redbull

0.00

find a 7up

0.00

go to the table

0.00

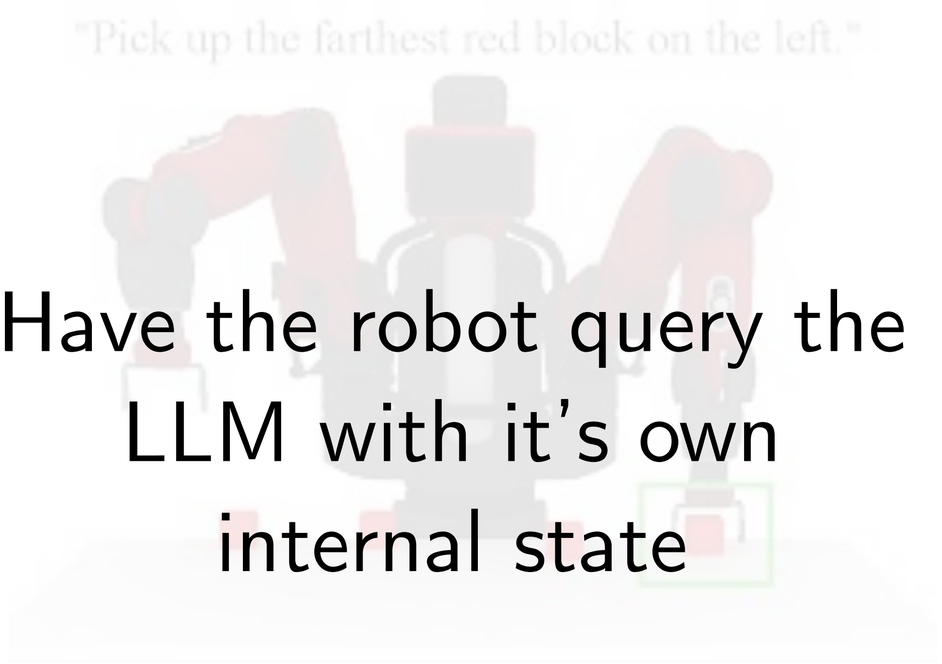
User input: Bring me a fruit flavoured drink without caffeine.

Robot: 1.

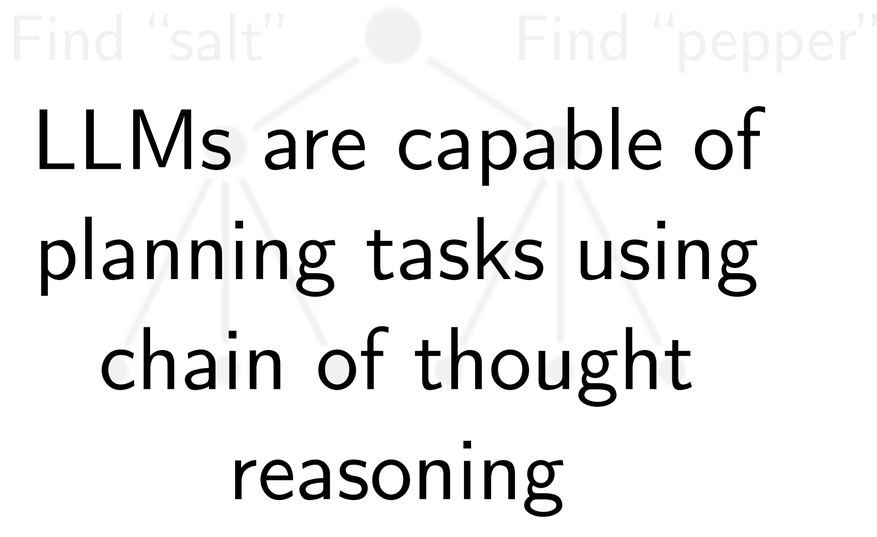
How does SayCan solve both challenges?

Challenge 1:
Ground natural language in robot state

Challenge 2:
Planning actions to solve a task



Have the robot query the LLM with it's own internal state



LLMs are capable of planning tasks using chain of thought reasoning

But there are still problems!

Problem 1: What if actions fail?

Problem 2: How do we verify correctness?

But there are still problems!

Problem 1: What if actions fail?

Idea: Close the loop on LLMs

Problem 2: How do we verify correctness?

Inner Monologue:

Embodied Reasoning through Planning with Language Models

Wenlong Huang* Fei Xia* Ted Xiao* Harris Chan Jacky Liang Pete Florence

Andy Zeng Jonathan Tompson Igor Mordatch Yevgen Chebotar Pierre Sermanet

Noah Brown Tomas Jackson Linda Luu Sergey Levine Karol Hausman Brian Ichter


Robotics at Google

But there are still problems!

Problem 1: What if actions fail?

Idea: Close the loop on LLMs

Problem 2: How do we verify correctness?

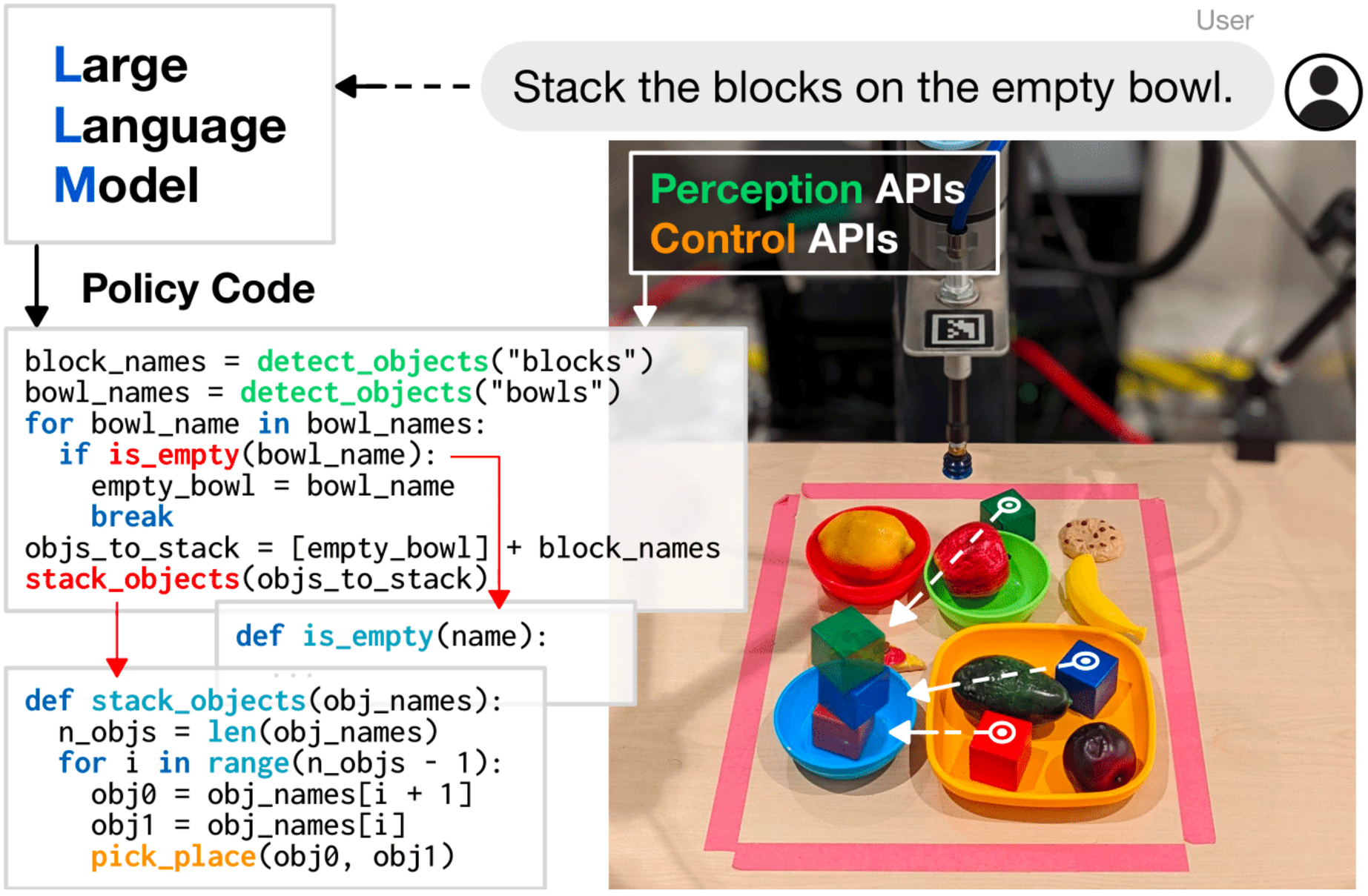
Idea: Get LLMs to generate code

Code as Policies:

Language Model Programs for Embodied Control

Jacky Liang Wenlong Huang Fei Xia Peng Xu Karol Hausman Brian Ichter Pete Florence Andy Zeng


Robotics at Google


Large Language Model
User
Stack the blocks on the empty bowl.
Policy Code
Perception APIs
Control APIs
block_names = detect_objects("blocks")
bowl_names = detect_objects("bowls")
for bowl_name in bowl_names:
if is_empty(bowl_name):
empty_bowl = bowl_name
break
objs_to_stack = [empty_bowl] + block_names
stack_objects(objs_to_stack)
def is_empty(name):
def stack_objects(obj_names):
n_objs = len(obj_names)
for i in range(n_objs - 1):
obj0 = obj_names[i + 1]
obj1 = obj_names[i]
pick_place(obj0, obj1)


It’s an exciting time for robot learning!