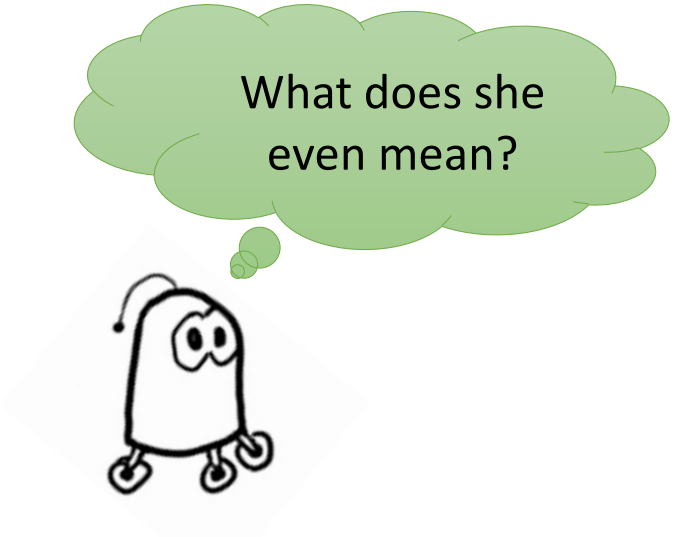
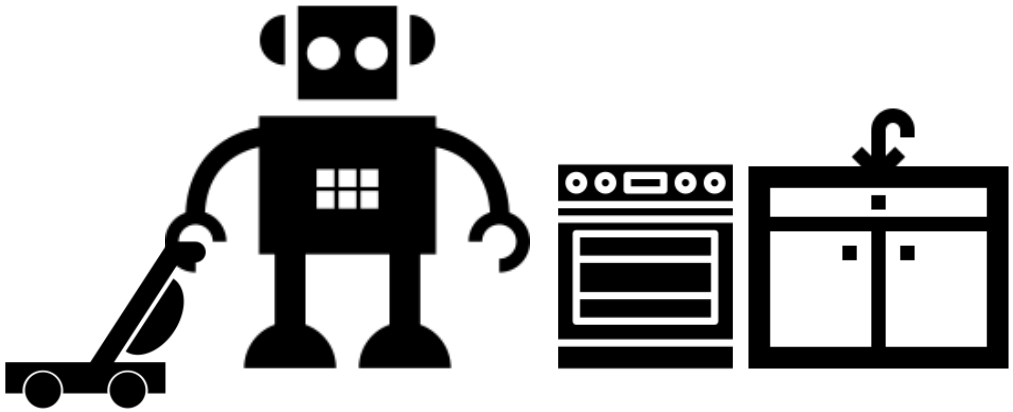
Rationally (Re)constructing Rational Robots



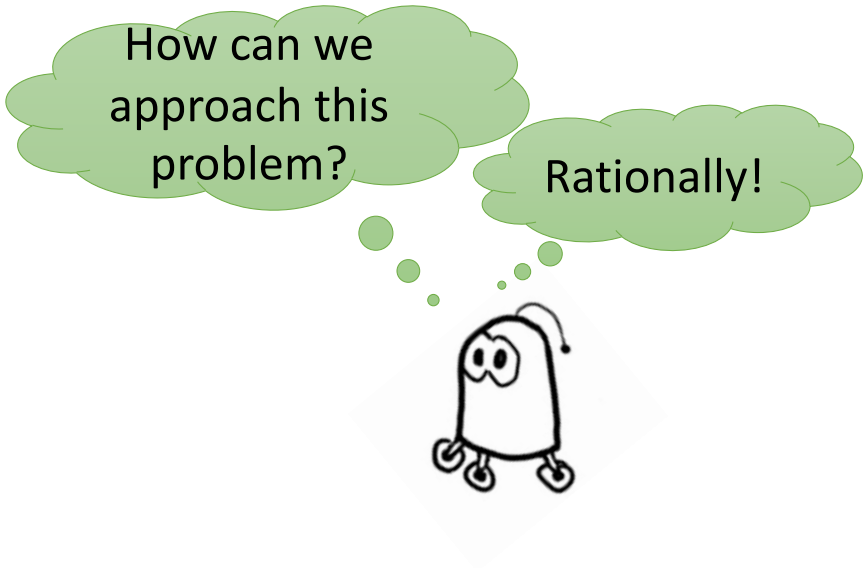
Leslie Pack Kaelbling

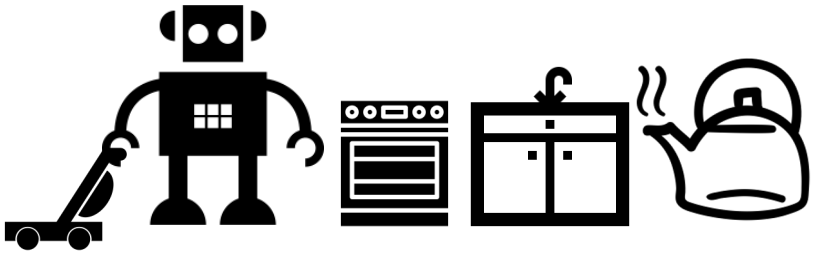
Research goal:
understand the computational mechanisms
necessary to make
a general-purpose intelligent robot



What makes this hard?

- Enormous variability in task demands and environments (countries, houses, families, jobs)
- Enormous state space (miles, millions of objects)
- Long horizon (years)

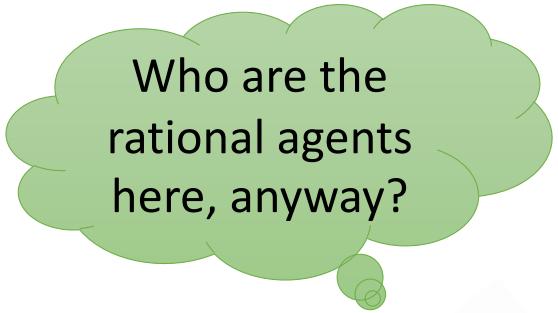




Rational agents

An agent is **rational** if it **selects actions** that

- in expectation,
- given the information it has about the state of the environment,
- will maximize its expected utility

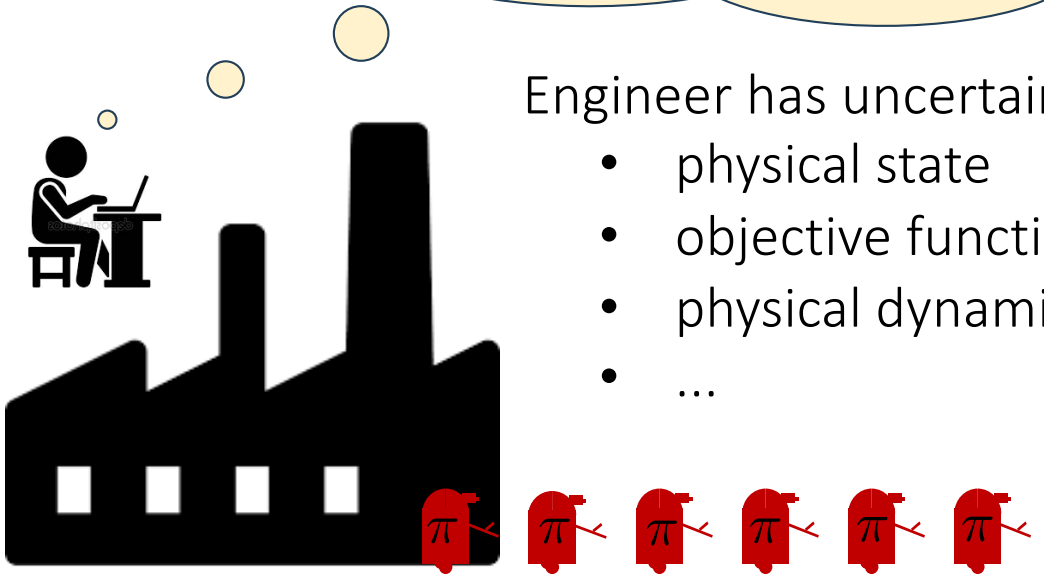


Realistically, only ever “bounded”:
subject to the agent’s cognitive/hardware/energy/time constraints

Rational (robot factory)

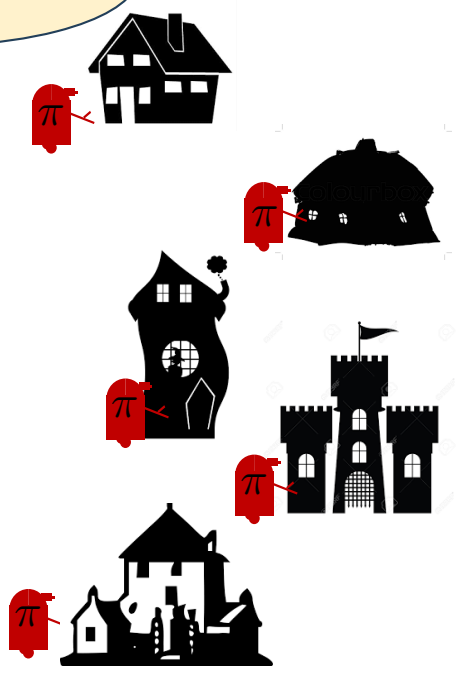
Find a mapping $\pi : (o, a)^* \rightarrow a$ that optimizes expected future reward

$$E_{\text{Dom}}\left[\sum_{t=0}^{T_{\text{dom}}} R_{\text{dom}}(s_t, a_t) \mid \pi, W_{\text{dom}}\right]$$



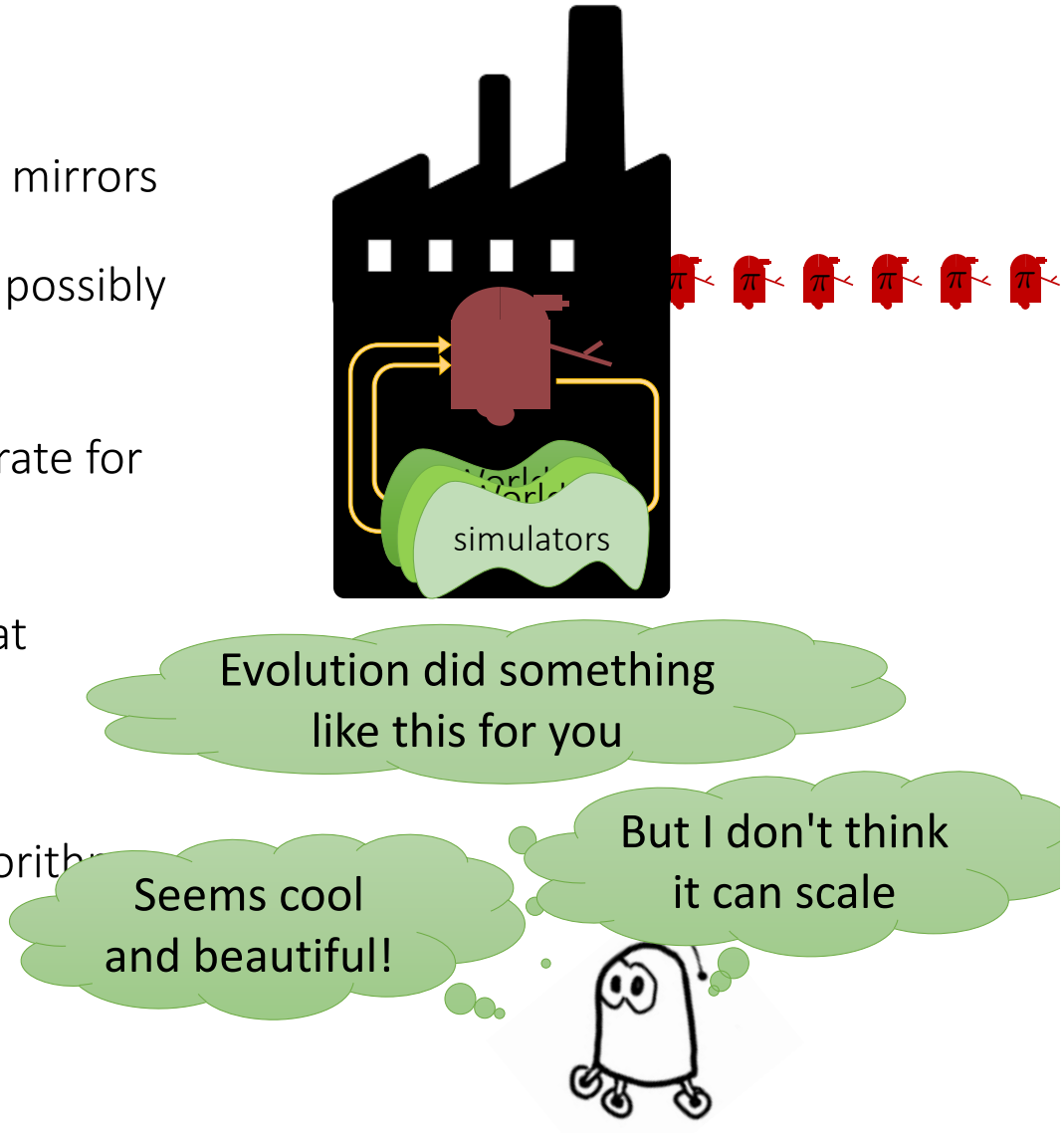
Engineer has uncertainty over

- physical state
- objective function
- physical dynamics
- …



Design-time rationality

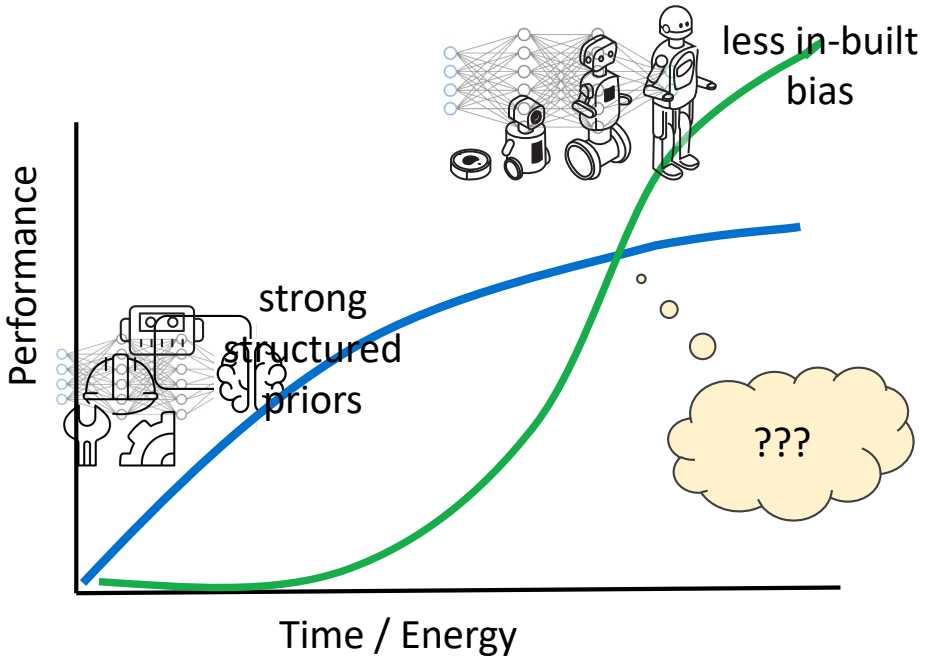
- Build a distribution of world simulators that mirrors the distribution of worlds (objects, houses, landscapes, humans) our robots could ever possibly encounter
- Design a completely general-purpose substrate for (meta) behavior learning
- Implement a **very very large** instance of that learning method
- Train it until it achieves general-purpose intelligence, inventing the best possible algorith for the problem distribution

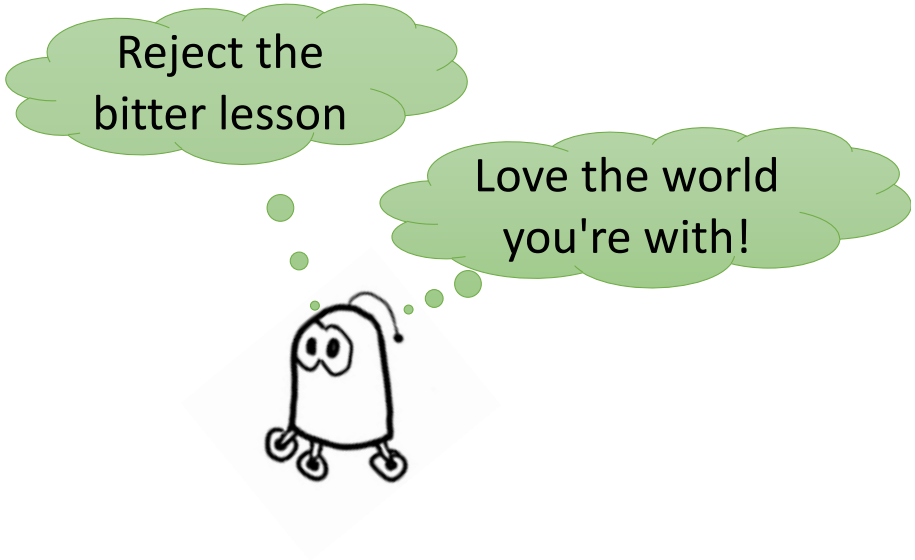


Leverage structured prior

Take all possible advantage of:

- the structure of the physical (and social) world
- algorithmic insights from computer science and cognitive science
  - abstraction, composition, inference
- our accumulated cultural knowledge





Leverage runtime inference

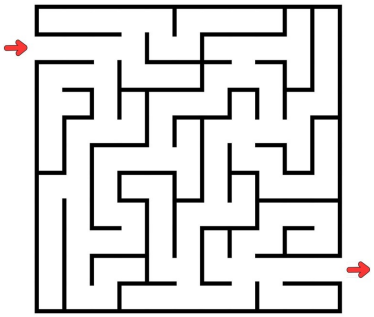
Performing computation at runtime may make action-selection slower

But it can be worthwhile

- Can reduce sample complexity for learning
  [Limits of autoregressive models and their alternatives, Lin et al, 21]
- Can make the easy cases fast!
  - circuit complexity for worst-case planning may be terrible
    [What Planning Problems Can A Relational Neural Network Solve?, Mao, et al, NeurIPS23]

Simple example: navigating in a novel 2D grid maze with n locations

- problem input: start state, goal set, maze map
- **size of a tabular goal-conditioned policy** : start x goal x map = **n x 2^n x 2^n**
  - difficult to find compact policy (if one exists)
- size of a tabular transition model: start x map = n x 2^n
  - size of structured model : **O(1)** (or O(log n) to count bits)
  - **time for action-selection is O(min(n, 4^h))** worst case

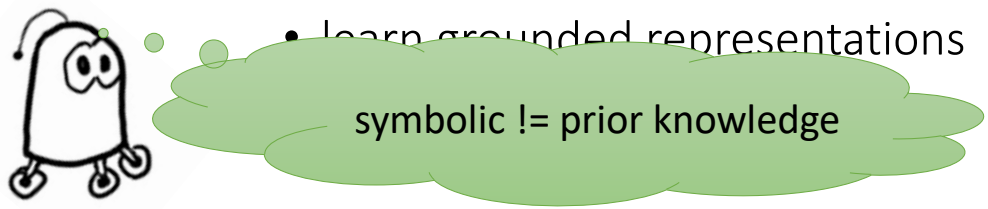


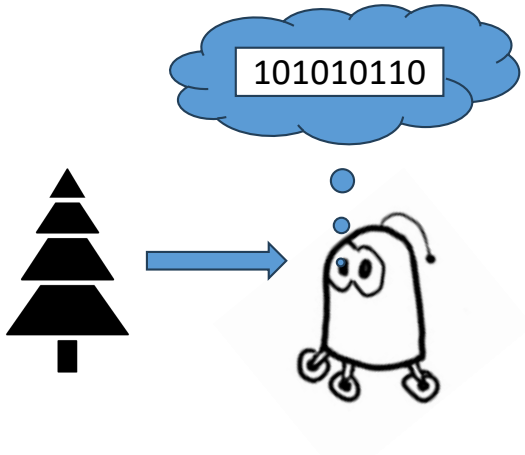
Symbol: a bit pattern that represents an aspect of the world

- Represents the same thing no matter where it is in memory
- For now, adopt a correlational theory of representation
- Not necessarily discrete: a float can represent the temperature outside

What are symbols good for?

- **Inference**: meaning-preserving manipulation of representations
  - correct **independent** of the actual meaning relationship
  - re-uses the same circuits for computations "about" completely different things
  - important for efficient learning:
    - build (networks for) inference algorithms into the agent
    - learn grounded representations



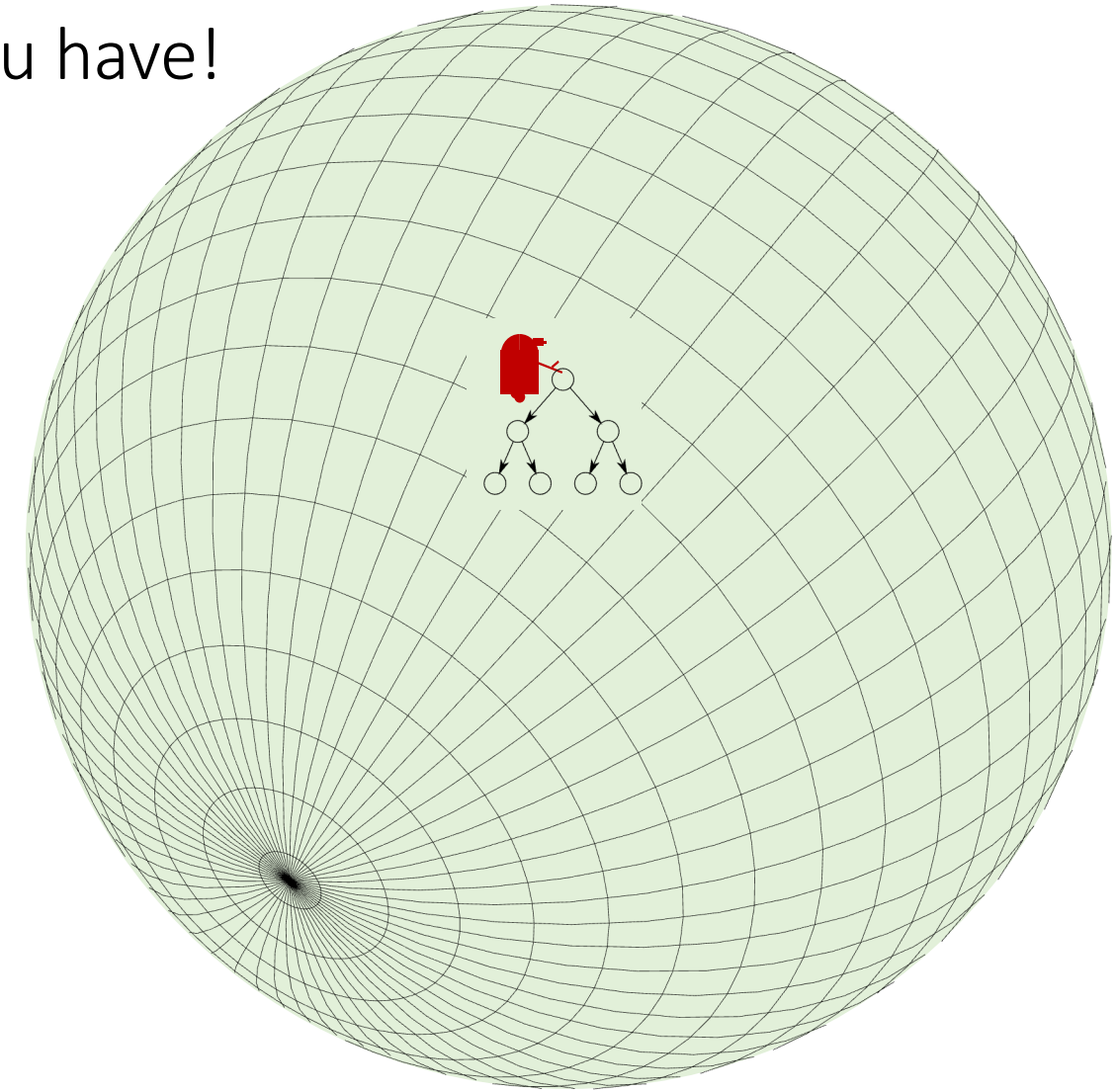


Solve the problem that you have!

The set of situations any individual is likely to encounter in its lifetime is orders of magnitude smaller than set of actually possible situations

Policies contain answers to all the problems: exponentially large circuit complexity for some domains

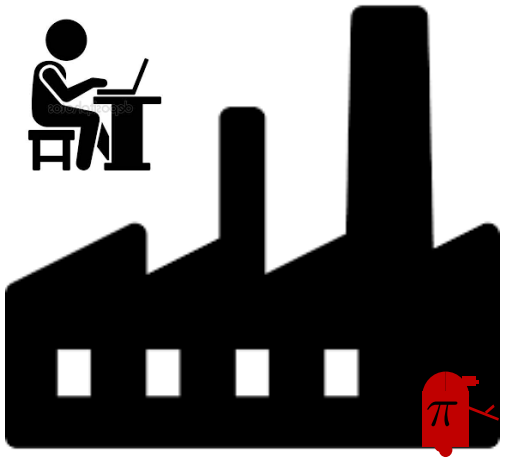
Planners find the answer to the problem at hand at some computational cost



(Rational robot) factory

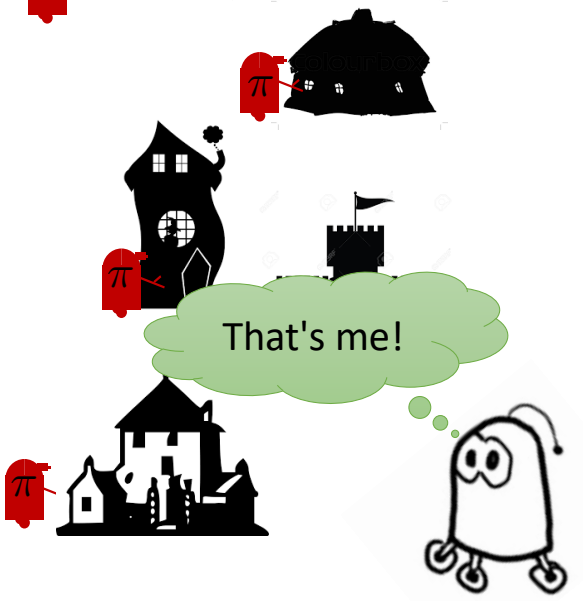
Find an action a that optimizes expected future reward

$$E_{s \sim \text{Belief}}\left[\sum_t R(s,a) \mid \text{continue optimally}\right]$$



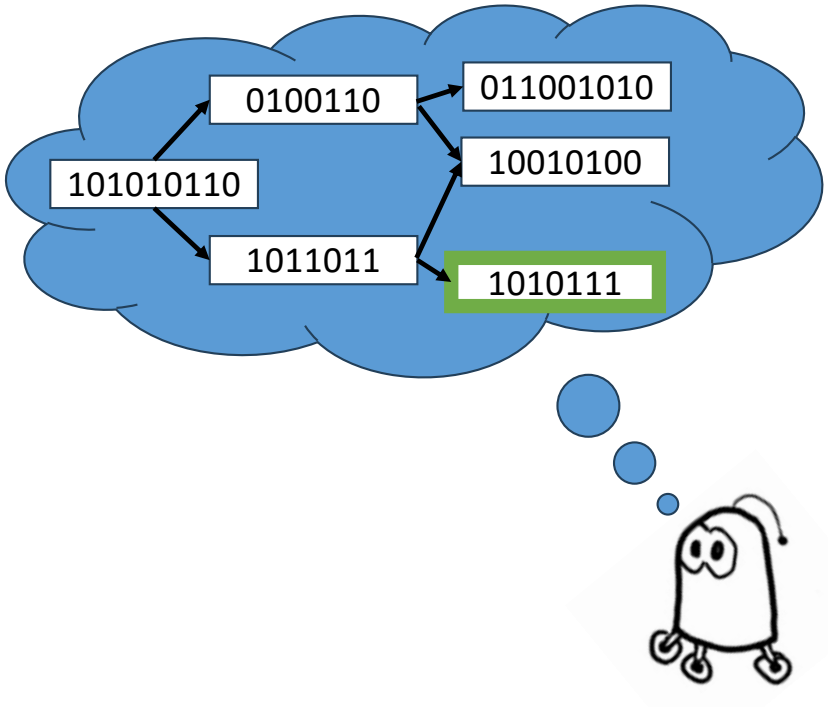
Robot has uncertainty over

- physical state
- social state (human desires)
- physical dynamics
- ...



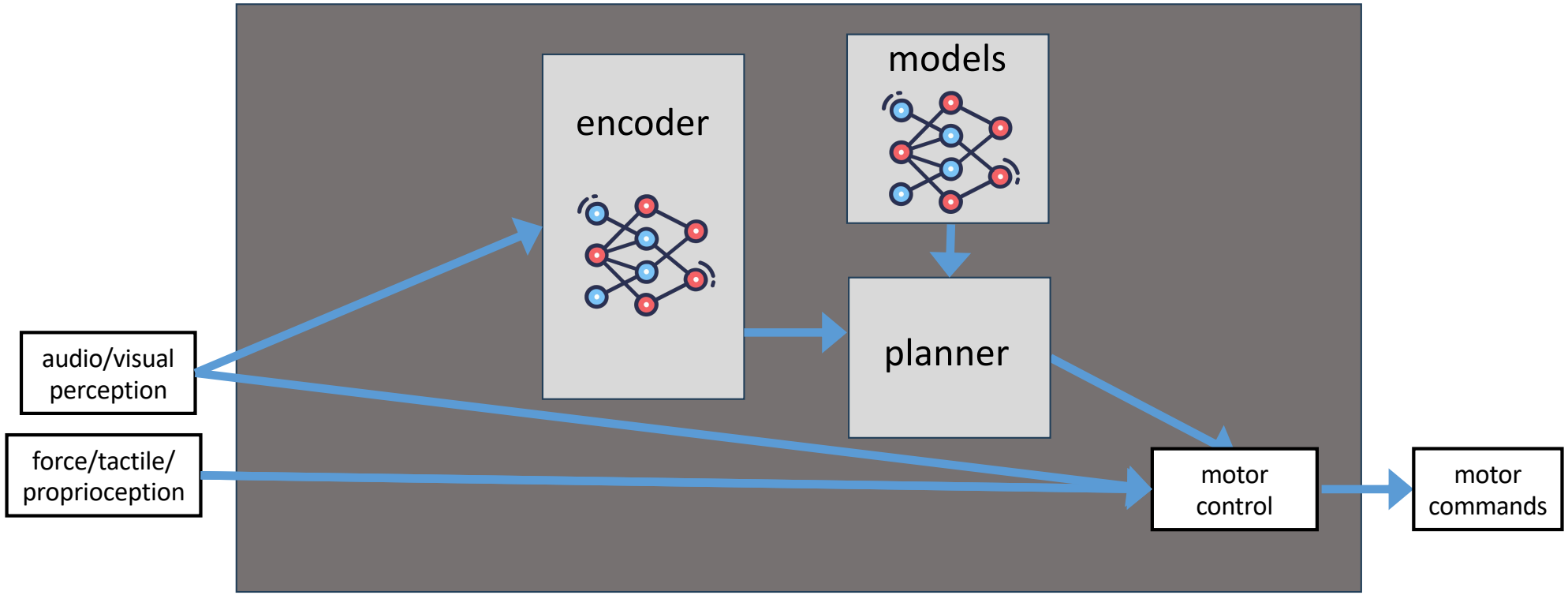
Planning: one type of symbolic inference

- Symbolic expressions represent states of the external world
  - Current world state
  - Transition model represents dynamics of world states
  - Goal / reward model represents utility of world states
- Planning algorithm
  - Semantics preserving, domain independent



Robot policy representation: prevailing view

Policy representation: with inference



Flexible, general world model representation : desiderata

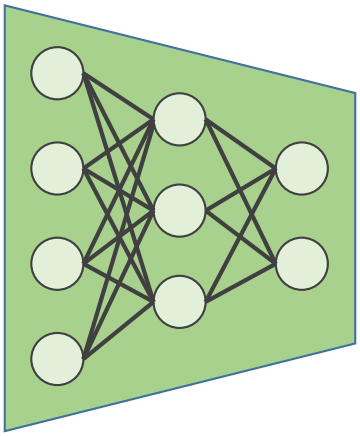
- can represent very complex state / goal spaces
  - including hybrid discrete-continuous
- learnable with low sample complexity / high generalization ability
  - key is (nearly) independent substructure : factoring, lifting
- efficient for planning
  - nearly deterministic outcomes
  - independent substructure

- can be guided by linguistic input / LLMs, but not required
- can incorporate learned search-control knowledge

- can model partial information

What are our symbols? Images?

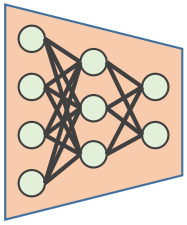
T



Latent, compressed encoding

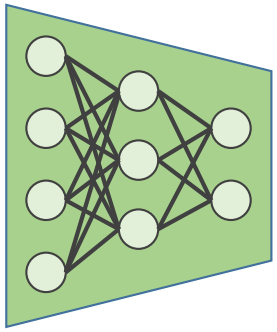
Map input state into lower-dimensional vector

- $\hat{s}$ is an abstraction
- $\hat{T}$ is smaller, easier to learn



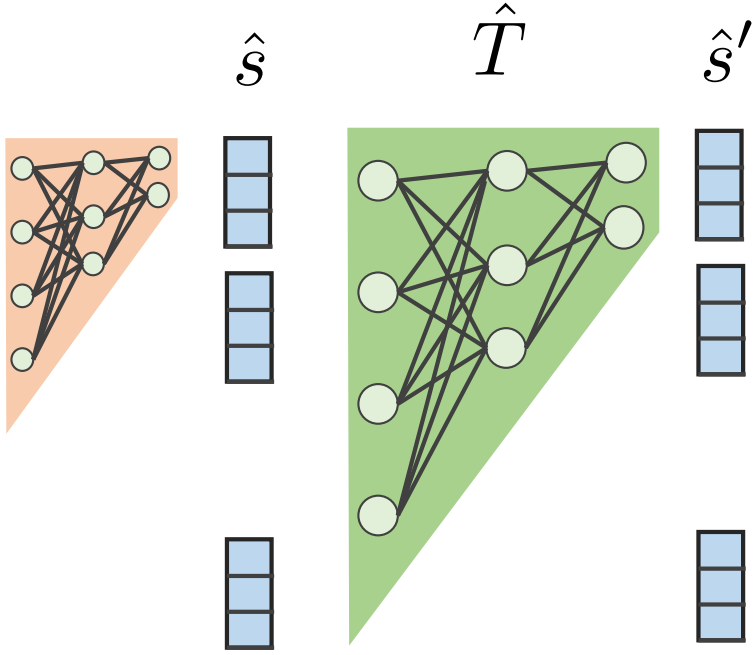
$\hat{s}$

$\hat{T}$



$\hat{s}'$

Factored encoding

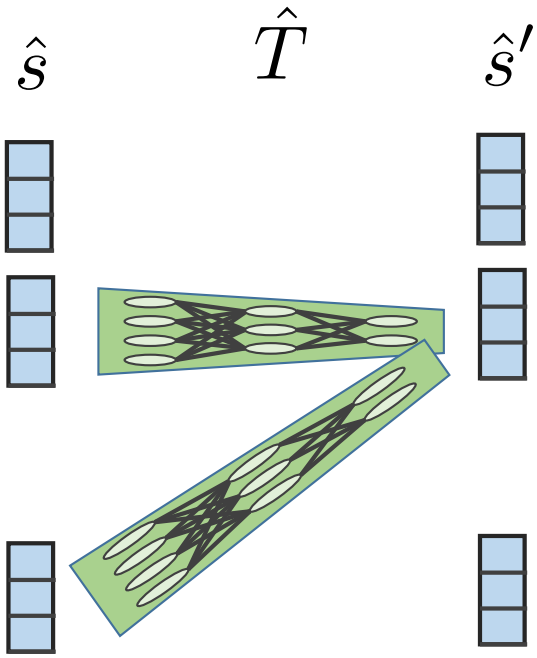


Divide latent vector into individually meaningful pieces called factors

- learn separate transition model for each factor
- not yet very advantageous (but does enable IW() style planning

F1: temperature
F2: heater-running
F3: window-open
F4: precipitation

Factored encoding: enables locality and sparsity



Locality: whether and how factors change depends on only a few other factors

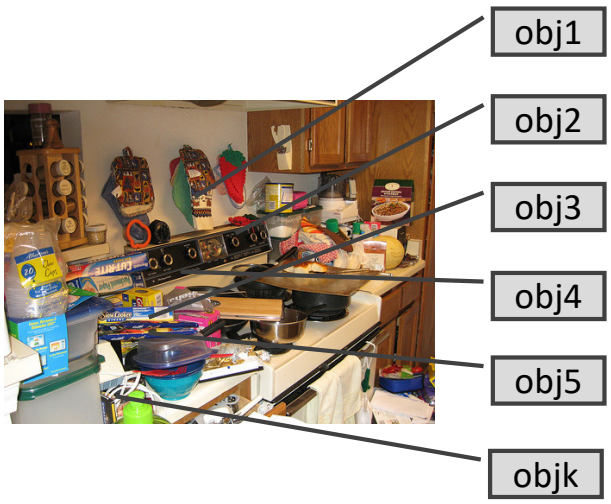
Sparsity: most factors don't change most of the time

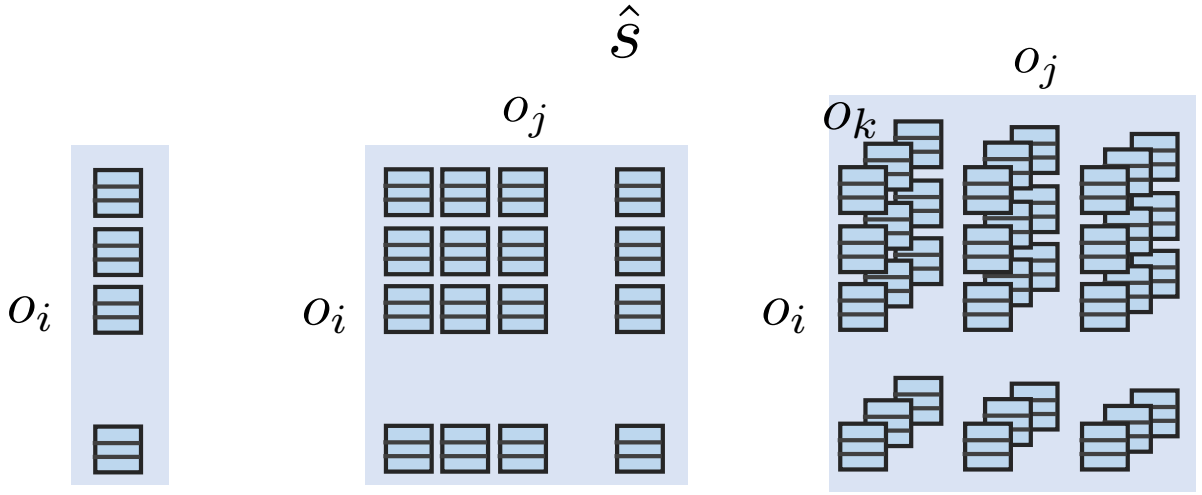
if heater-running:
temperature := f(temperature, window-opening)

Can result in **dramatic reduction in complexity** of the function to be learned

Enables **powerful general-purpose search "heuristics"** based on an assumption that actions don't "interfere" with each other

“Lifted” (object-based) encoding: objects index factors





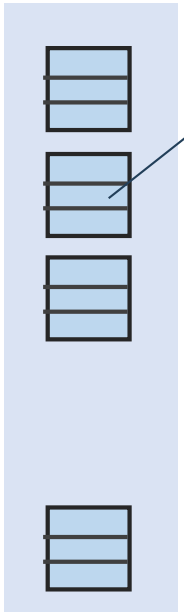
several factors
for each object:
properties

$temp(o_i)$
$open(o_i)$

several factors for each
tuple of objects:
relations

$relpose(o_i, o_j)$
$contains(o_i, o_j)$

Two kinds of symbols: object names, properties

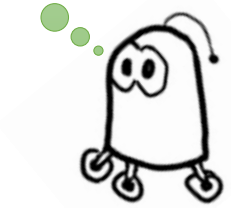


color(obj2) = #b1b

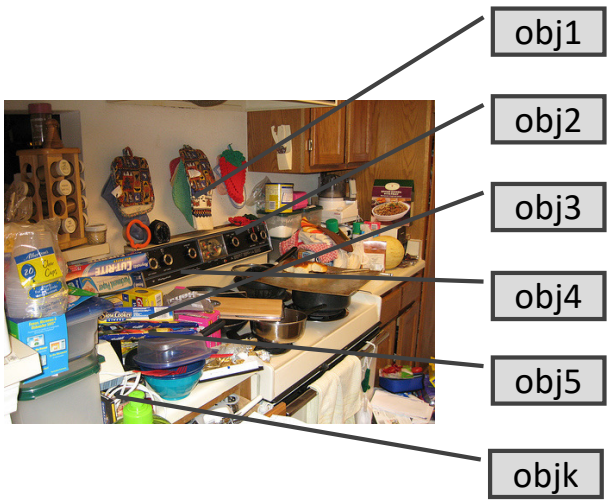
Properties grounded via perceptual detectors

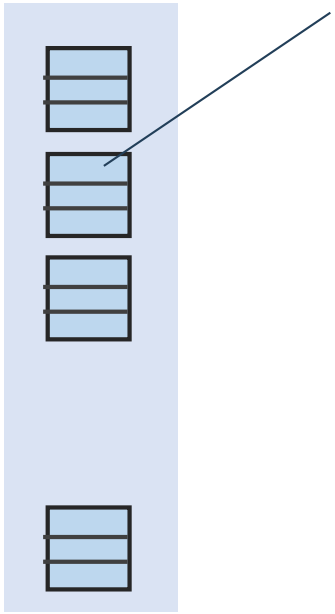
Object names grounded via segmentation and tracking; can be flexible, fluid

Very compact models



Properties can encode belief

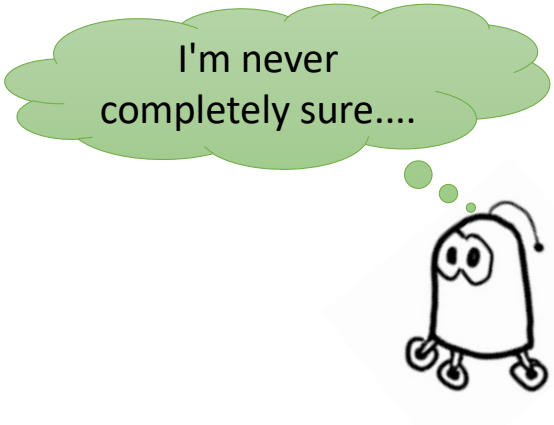




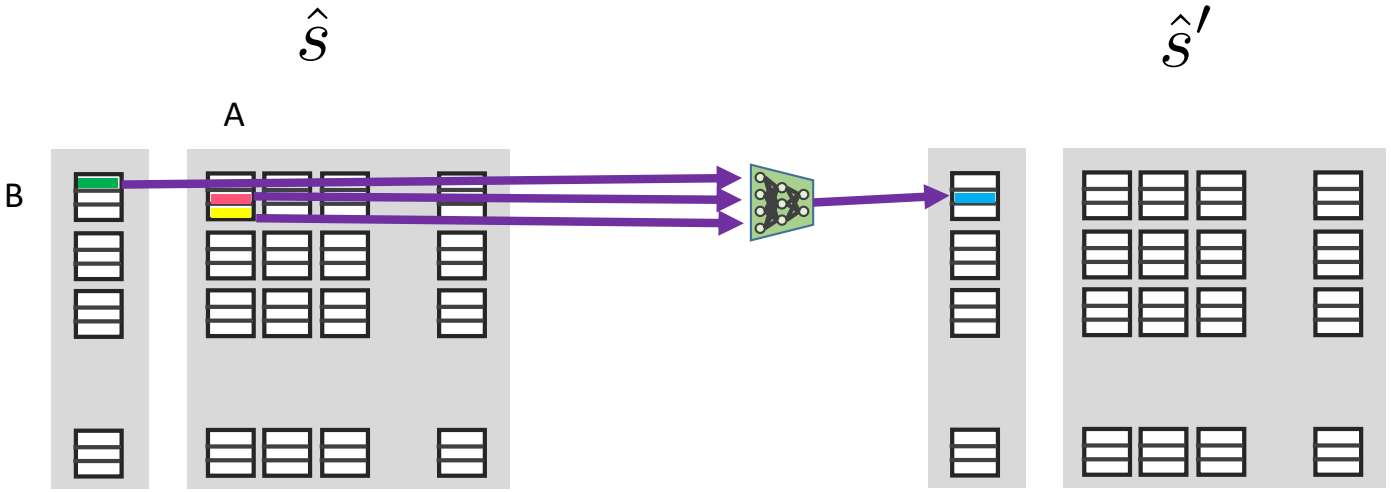
P(color(obj2) = #b1b) > 0.9

Belief space = space of distributions over world states

Planning in belief space is not technically different from planning in "state space"

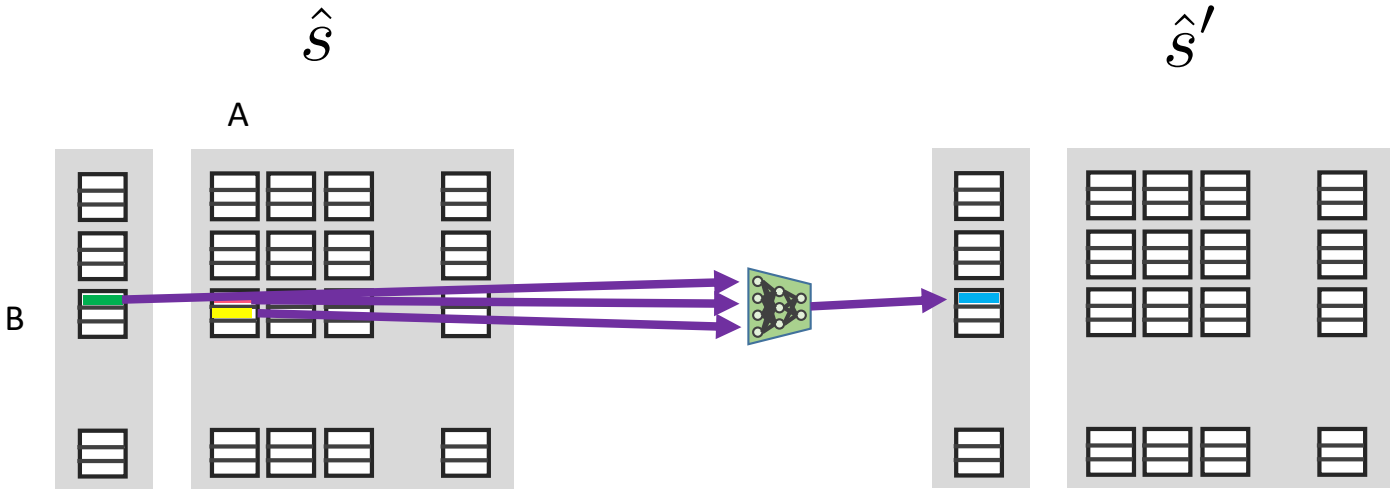


Sparse, factored, lifted transition rules



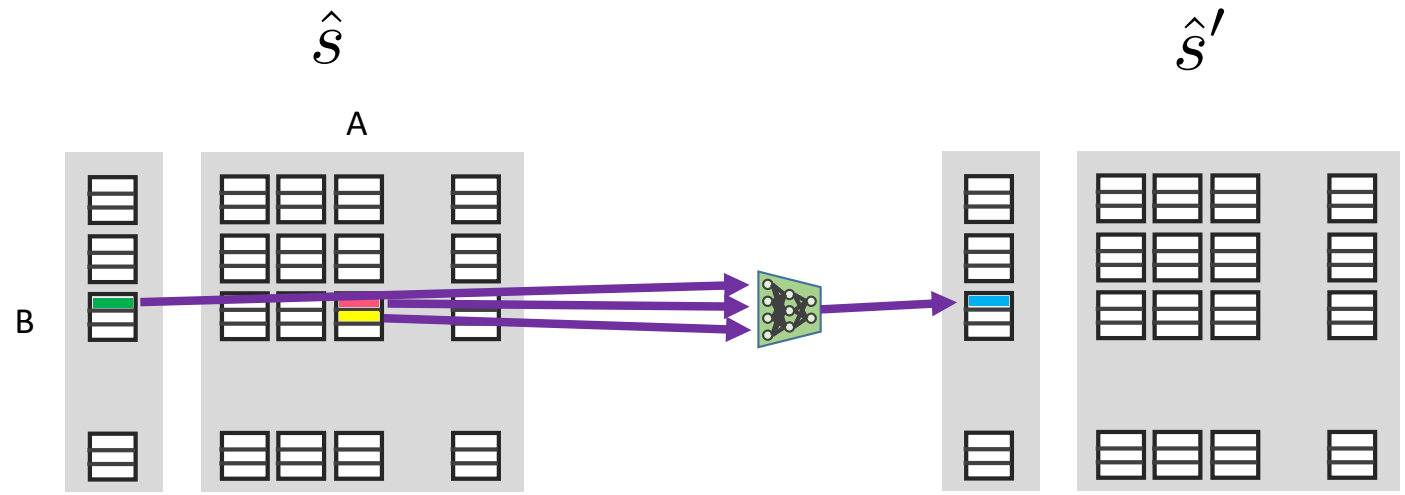

```
move(A, d):
if contains(A, B) and light(B):
   pose(B) := f(relpose(A, B), d)
```

Sparse, factored, lifted transition rules




```
move(A, d):
if contains(A, B) and light(B):
   pose(B) := f(relpose(A, B), d)
```

Sparse, factored, lifted transition rules




```
move(A, d):
if contains(A, B) and light(B):
   pose(B) := f(relpose(A, B), d)
```

Causal Action Model: lifted, factored, transition model

```
behavior move_to(target: loc) :
  body:
    bind d: {[1, 0], [0, 1], [-1, 0], [0, -1]}
    achieve not wall_at(target)
    achieve robot_loc == target - d
    move(d)
  effect:
    robot_loc == target
```

object var: target
parameter: d
predicate: wall_at, robot_loc
controller: move

Range from completely deterministic program to highly non-deterministic search space

Jiayuan Mao, Crow Description Language, Hybrid Representations for Decision-Making, WAFR24

Learning the motion rule using Popper Inductive Logic Programming algorithm

Data

```
pos(new_at(1, 1, 2)).
pos(new_at(1, -1, 0)).
pos(new_at(2, 1, 2)).
pos(new_at(4, -1, 4)).
pos(new_at(-3, 1, -3)).

neg(new_at(1, 1, 1)).
neg(new_at(2, 1, 3)).
neg(new_at(-3, 1, -2)).
neg(new_at(4, -1, 3)).
```

Facts

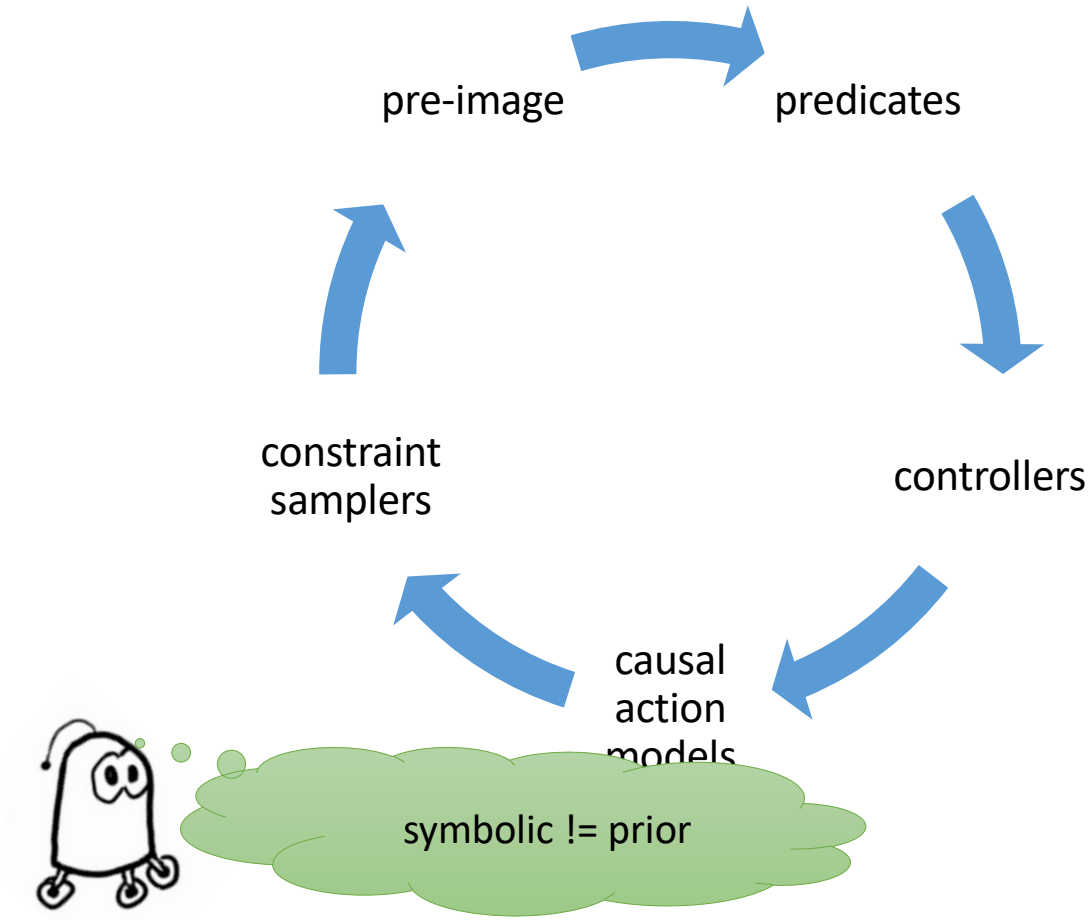
```
wall_at(3).
wall_at(-2).

mysum(X, Y, Z) :- Z #= X + Y.

free(X) :- \+ wall_at(X).
```

RESULT

```
new_at(V0,V1,V2):- mysum(V0,V1,V2),free(V2).
new_at(V0,V1,V2):- wall_at(V3),mysum(V1,V0,V3),mysum(V1,V2,V3).
```

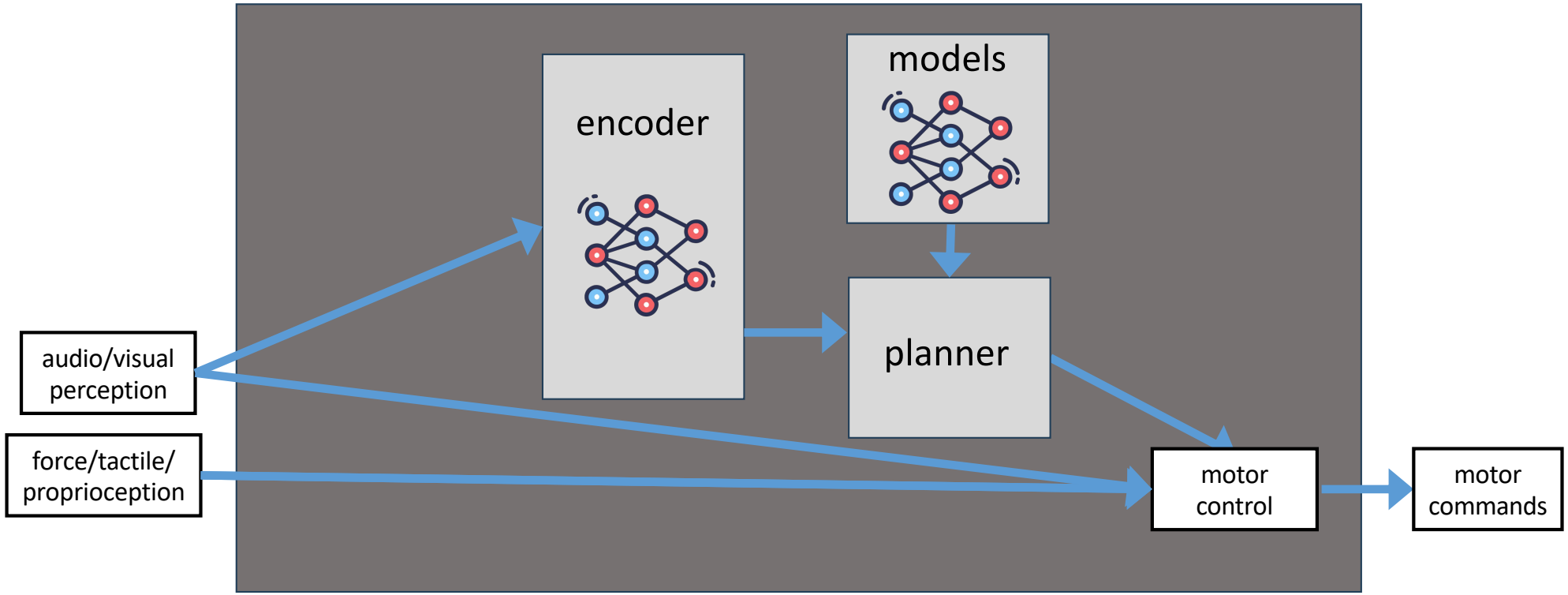
Cropper & Morel , Learning Programs by Learning From Failures , MLJ21

Continual Learning for Causal Action Models

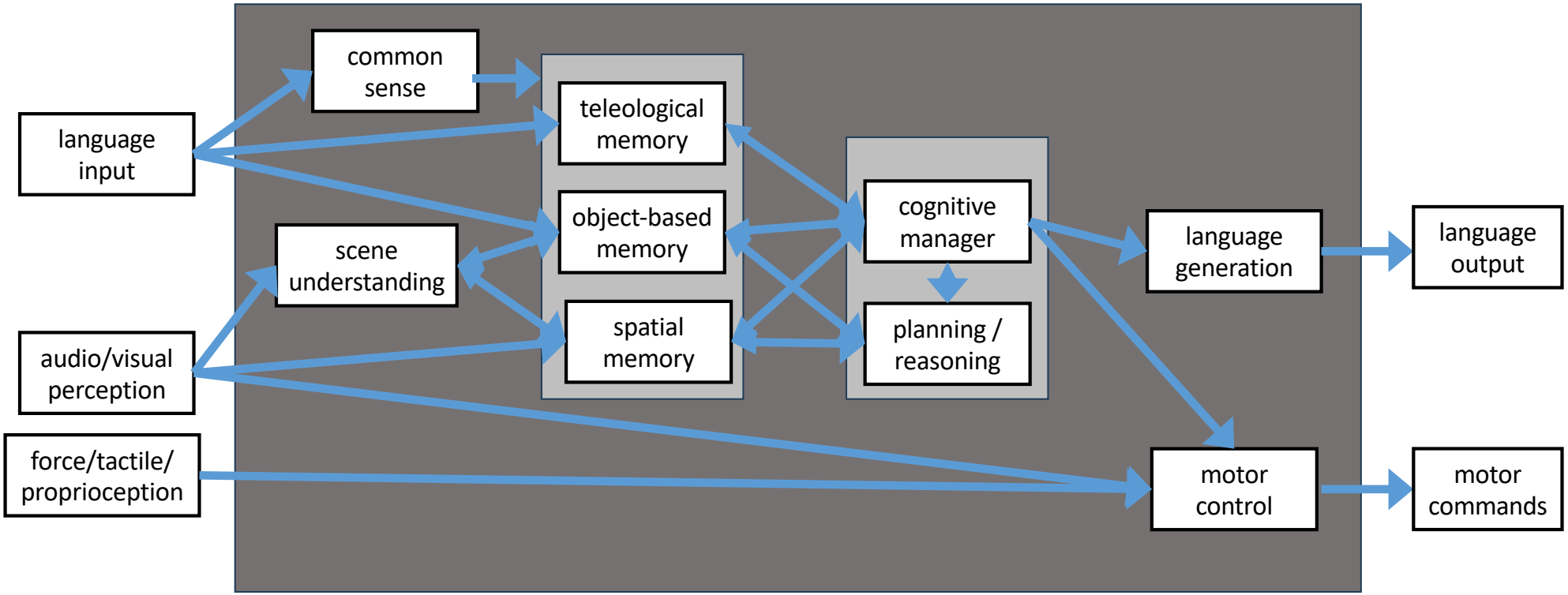


- Start with some **built-in predicates** (innate or via LVM or teaching)
- **Learn controllers** to achieve them (RL on top of sensible control-theoretic primitives)
- **Instantiate CAMs** with the predicate as effect and controller
- Learn, by experimentation, **generative models for sampling** continuous parameters
- **Describe the preimage** of the effect under the controller (sort of like but not the same as an initiation set) using existing predicates, if possible, but if not, instantiate **new predicates**
- ....

Policy representation: with inference

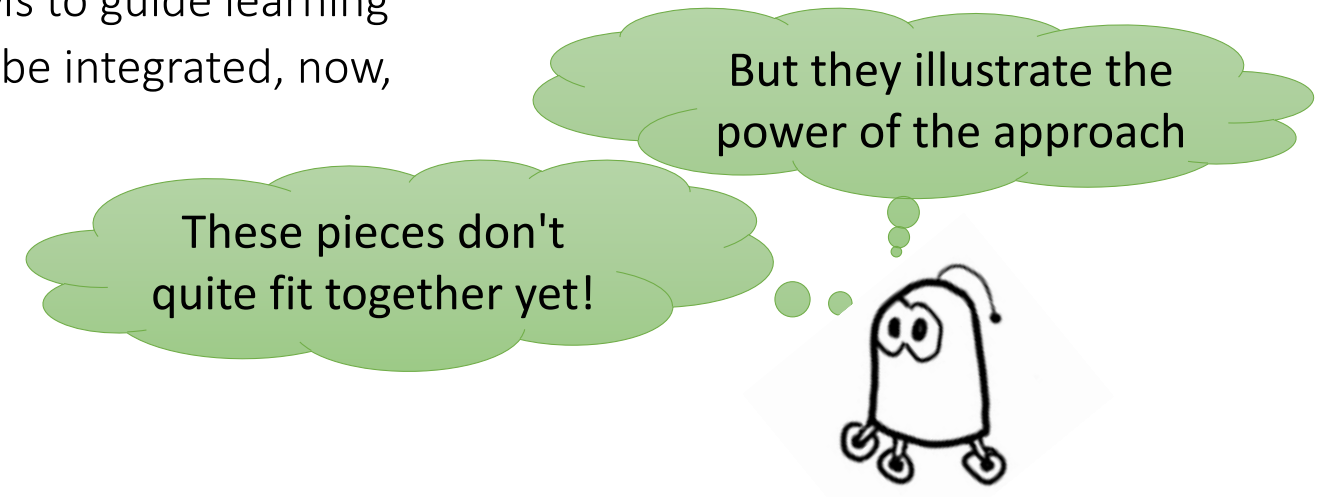


Policy representation: our target architecture



We currently have a patchwork of the needed parts

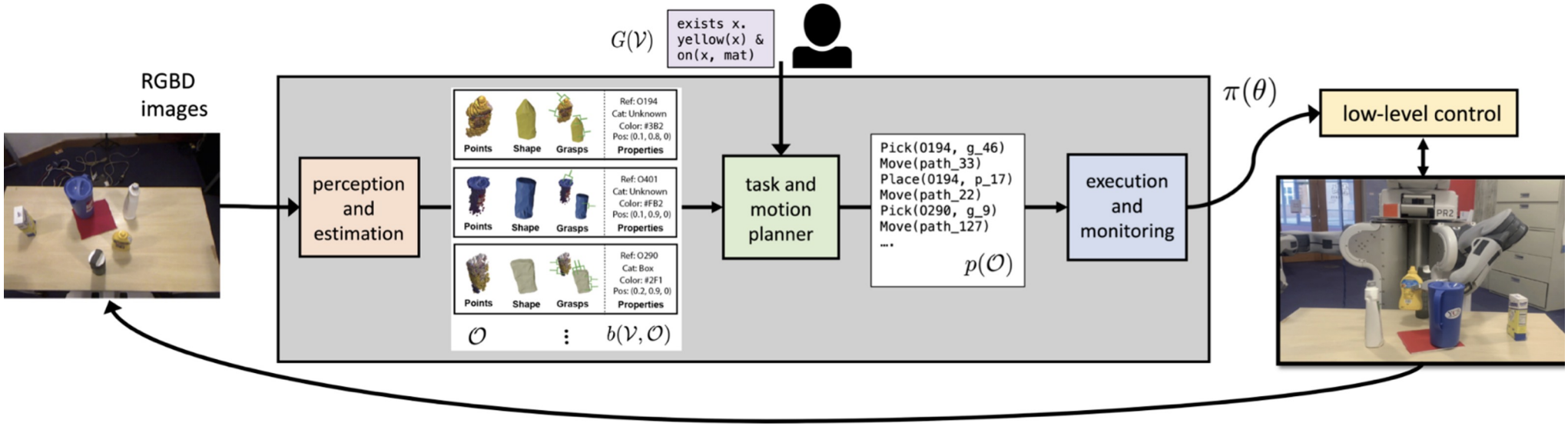
1. Learned perception plus planning yields strong generalization
2. We can learn to ground predicate symbols from unlabeled demonstrations
3. We can learn generative models for continuous parameters by experimentation
4. We can learn causal action model structures
5. We can learn "mechanisms" from a single example
6. Generative models can be compositionally structured and learned
7. We can use LLM and LVMs to guide learning
8. Many of these ideas can be integrated, now, into a real robot

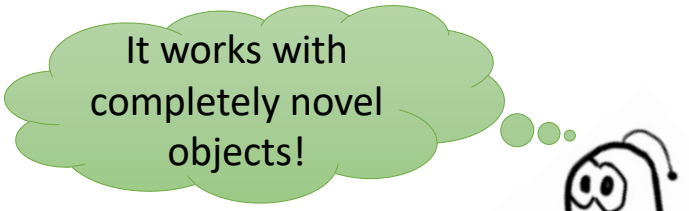


We currently have a patchwork of the needed parts

1. Learned perception plus planning yields strong generalization
2. We can learn to ground predicate symbols from unlabeled demonstrations
3. We can learn generative models for continuous parameters by experimentation
4. We can learn causal action model structures
5. We can learn "mechanisms" from a single example
6. Generative models can be compositionally structured and learned
7. We can use LLM and LVMs to guide learning
8. Many of these ideas can be integrated, now,
   into a real robot

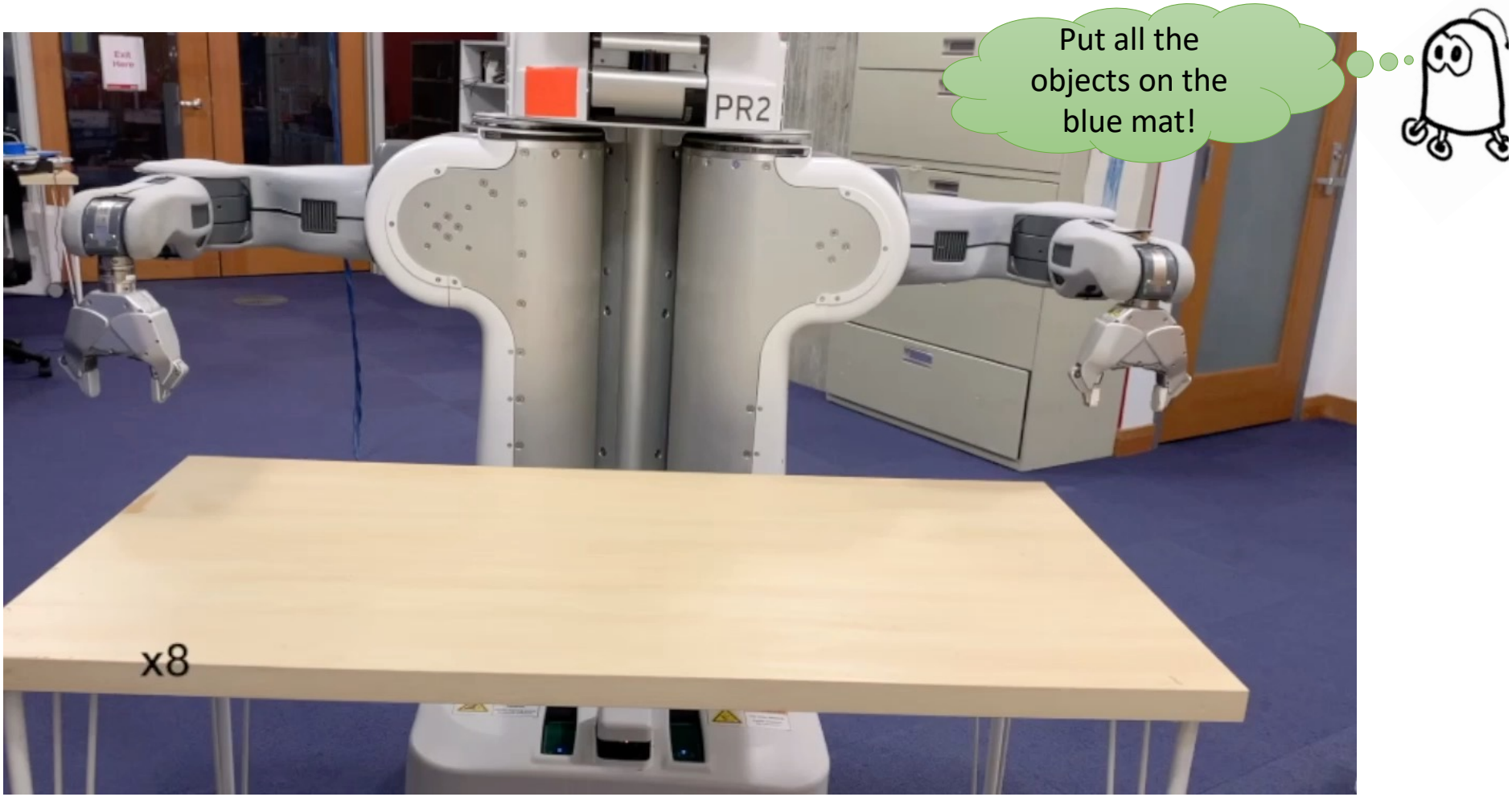
Learned perception plus planning yields generalized policy



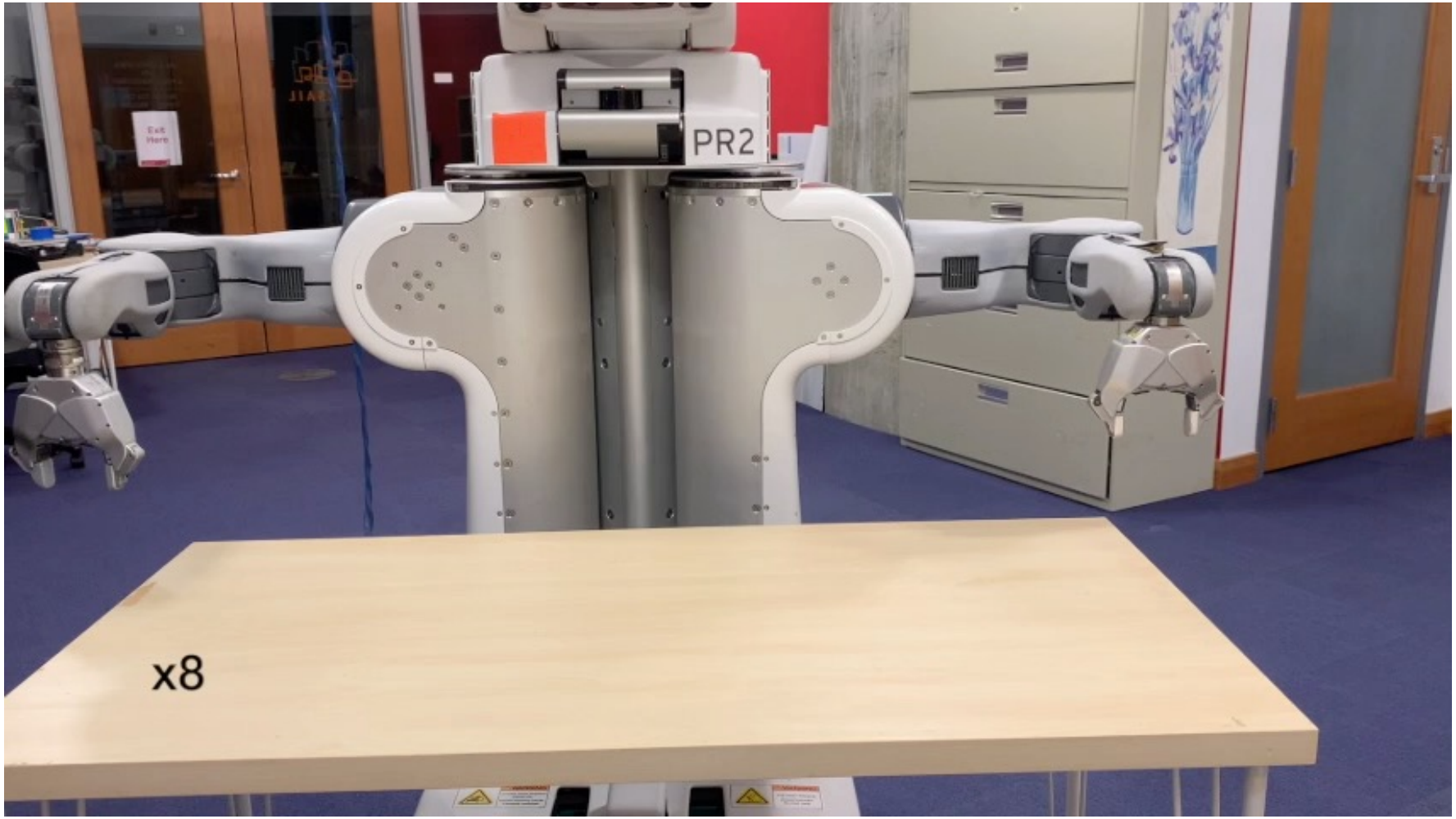


Aidan Curtis, Xiaolin Fang, ..., Caelan Garrett, ICRA 2022

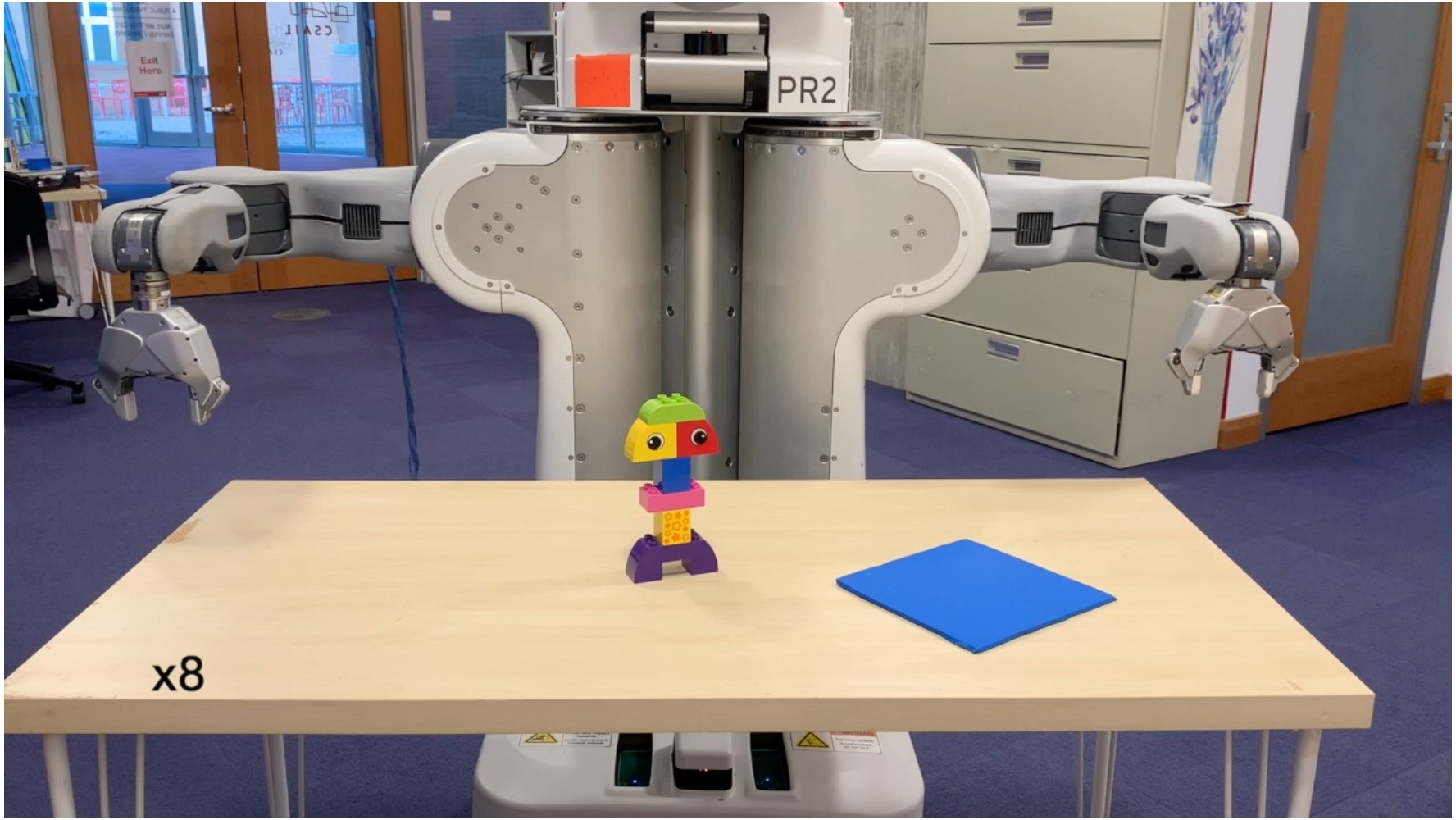
Immediate generalization to novel objects, goals, arrangements



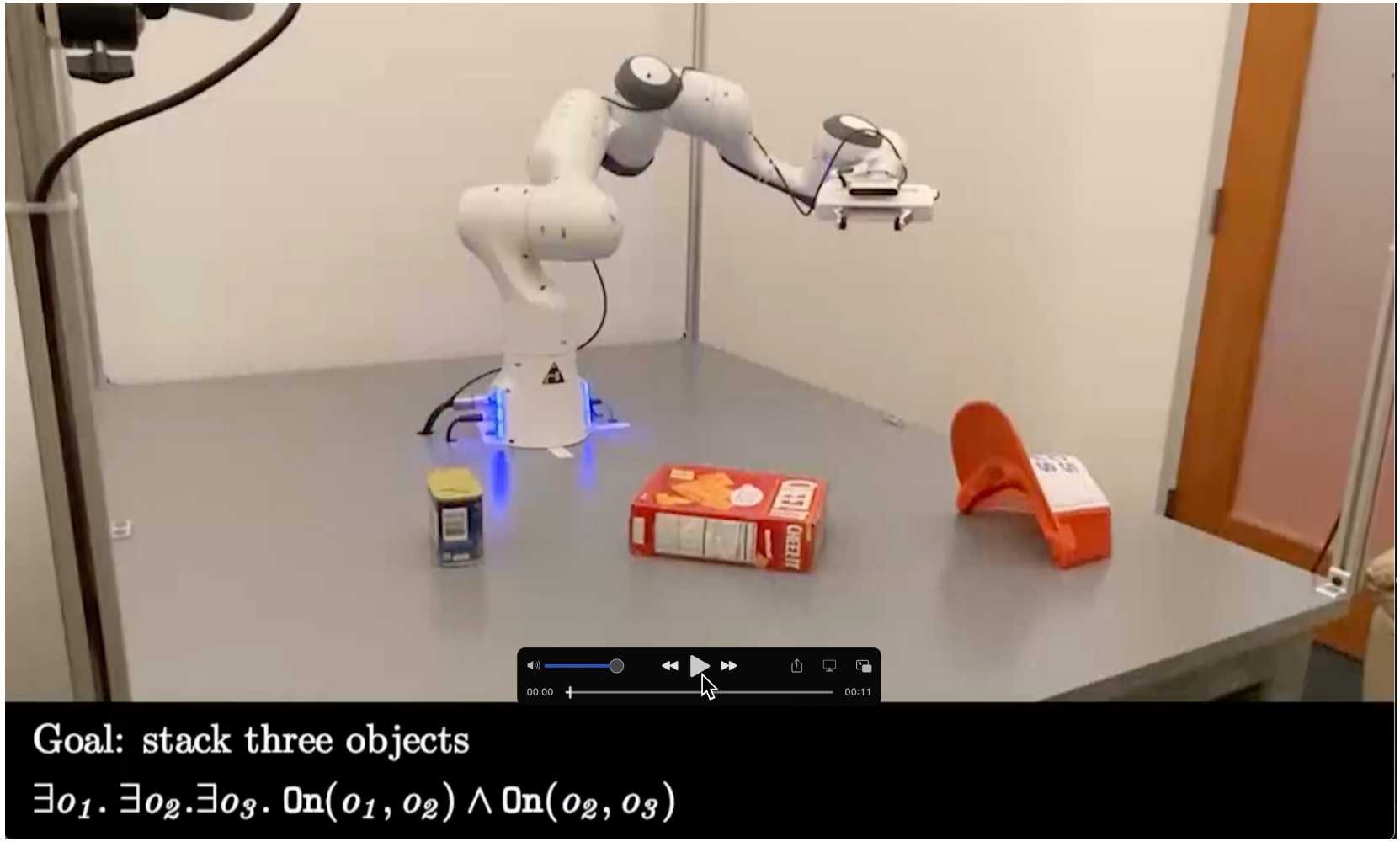
Put each object in the bowl of the closest color



Put all the objects on the blue mat



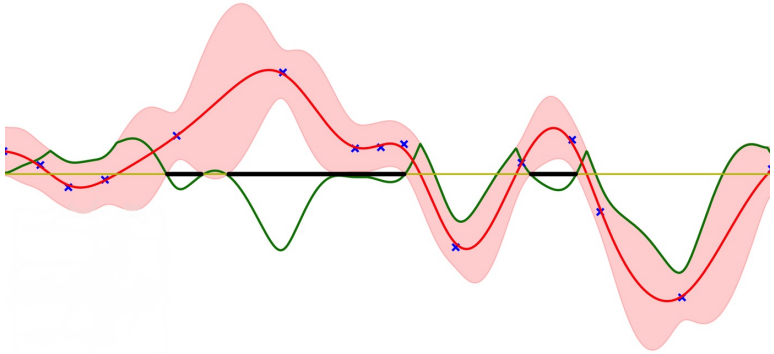
Cross-embodiment generalization



We currently have a patchwork of the needed parts

1. Learned perception plus planning yields strong generalization
2. We can learn to ground predicate symbols from unlabeled demonstrations
3. We can learn generative models for continuous parameters by experimentation
4. We can learn causal action model structures
5. We can learn "mechanisms" from a single example
6. Generative models can be compositionally structured and learned
7. We can use LLM and LVMs to guide learning
8. Many of these ideas can be integrated, now,
   into a real robot

Active learning: training on real robot is expensive!



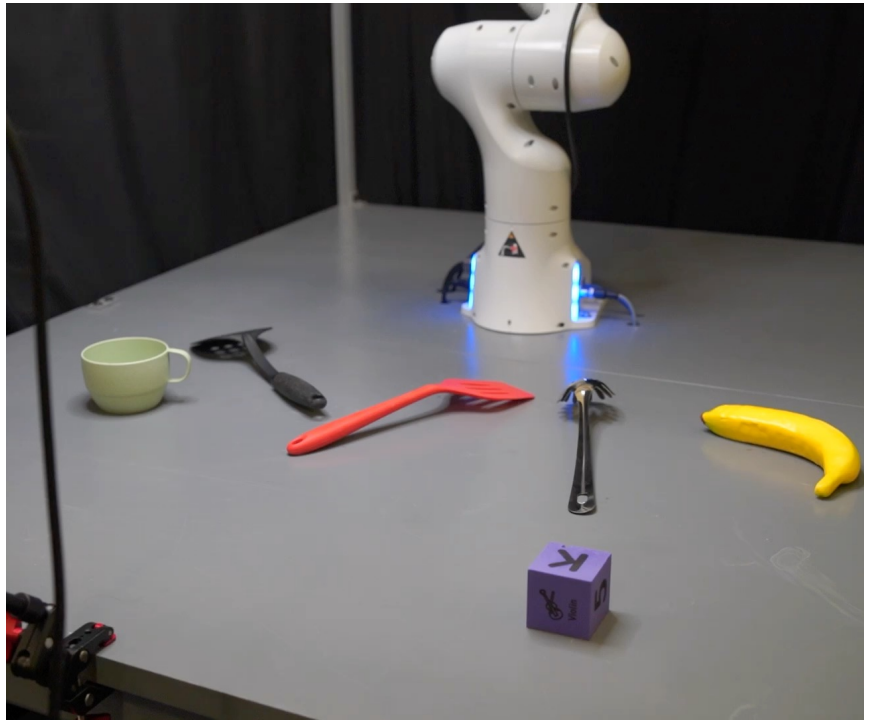
Zi Wang, Caelan Garrett, ..., IROS 2018

Note that we do grasp and motion planning during data acquisition!

We currently have a patchwork of the needed parts

1. Learned perception plus planning yields strong generalization
2. We can learn to ground predicate symbols from unlabeled demonstrations
3. We can learn generative models for continuous parameters by experimentation
4. We can learn causal action model structures
5. We can learn "mechanisms" from a single example
6. Generative models can be compositionally structured and learned
7. We can use LLM and LVMs to guide learning
8. Many of these ideas can be integrated, now,
   into a real robot

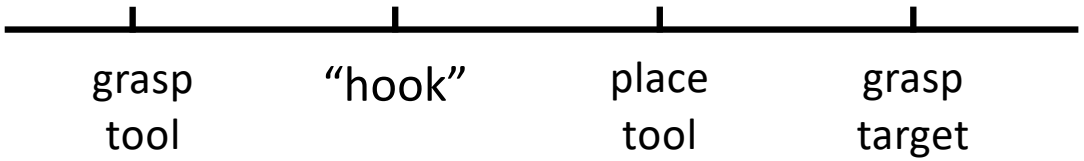
What can We Learn from a Single Demonstration



⟨ name, type, **??** ⟩

hook (r: robot, tool: object, target: object)
→ action_sequence

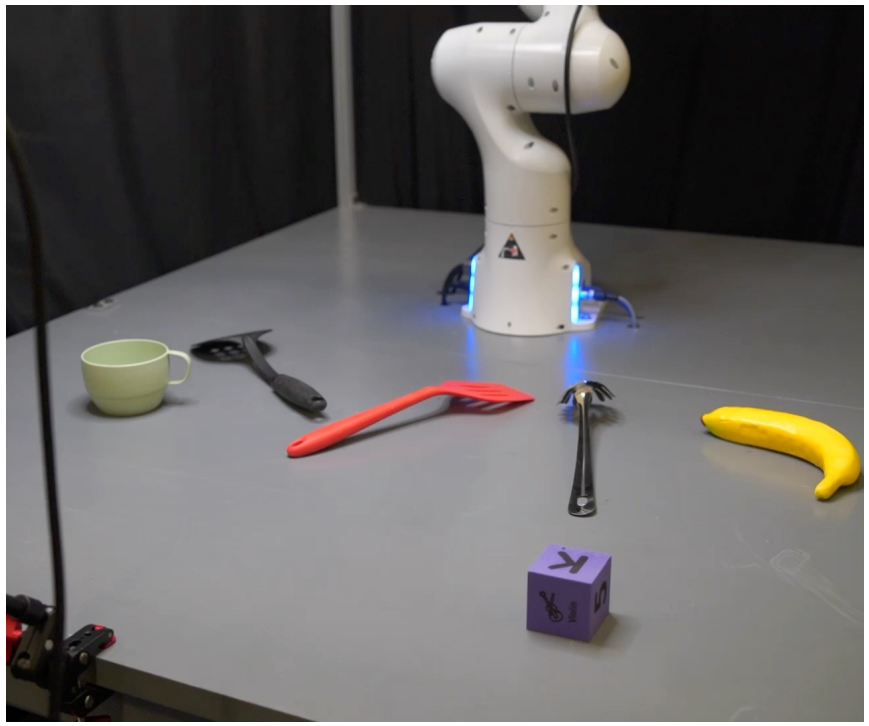
High-order concepts as a "program"!
Modeled as a sequence of contacts



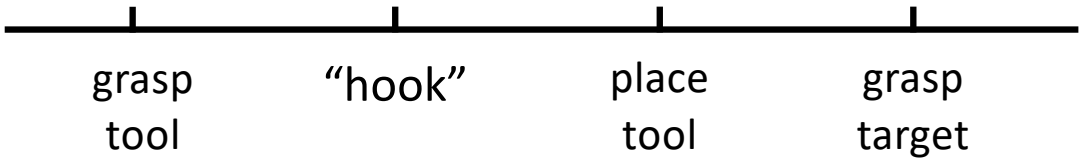
Mao, Lozano-Perez, Tenenbaum, Kaelbling. CoRL 2023. Learning Reusable Manipulation Strategies.

The Contact Mode Subgoals in Hook-Using

Key idea: model manipulation "strategies" as a program of a sequence of subgoals about contacts among objects



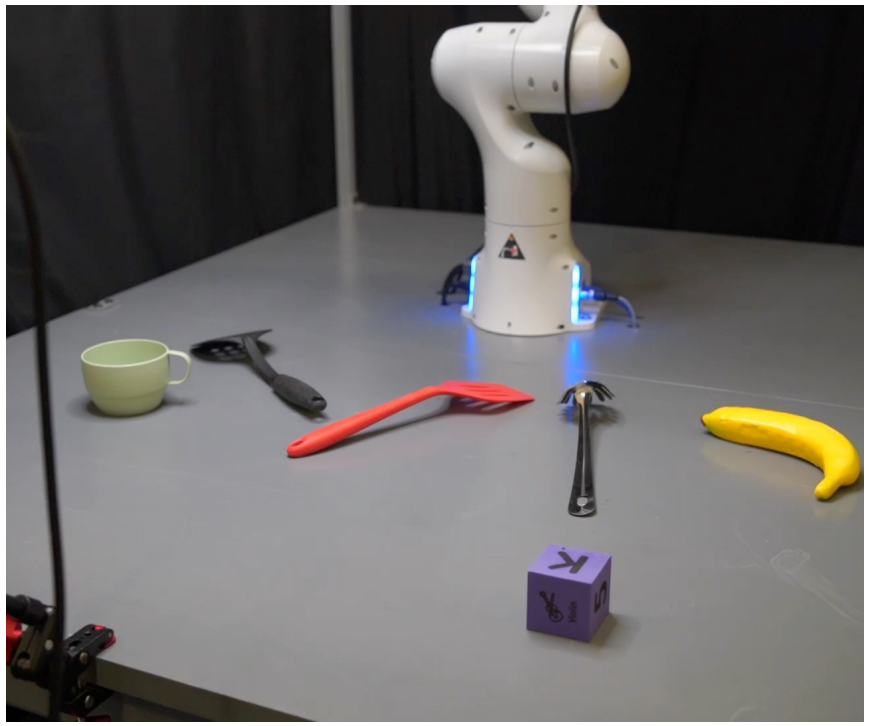
```
behavior hook(target, tool):
  body:
    achieve holding(tool, ?grasp1)
    move-with-contact(tool, target, ?traj)
    place_down(tool)
    grasp(target, ?grasp2)
```



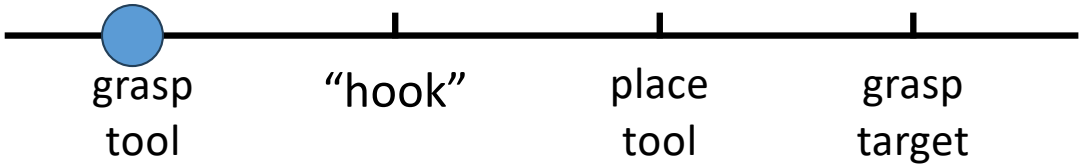
Mao, Lozano-Perez, Tenenbaum, Kaelbling. CoRL 2023. Learning Reusable Manipulation Strategies.

The Contact Mode Subgoals in Hook-Using

Key idea: model manipulation "strategies" as a program of a sequence of subgoals about contacts among objects



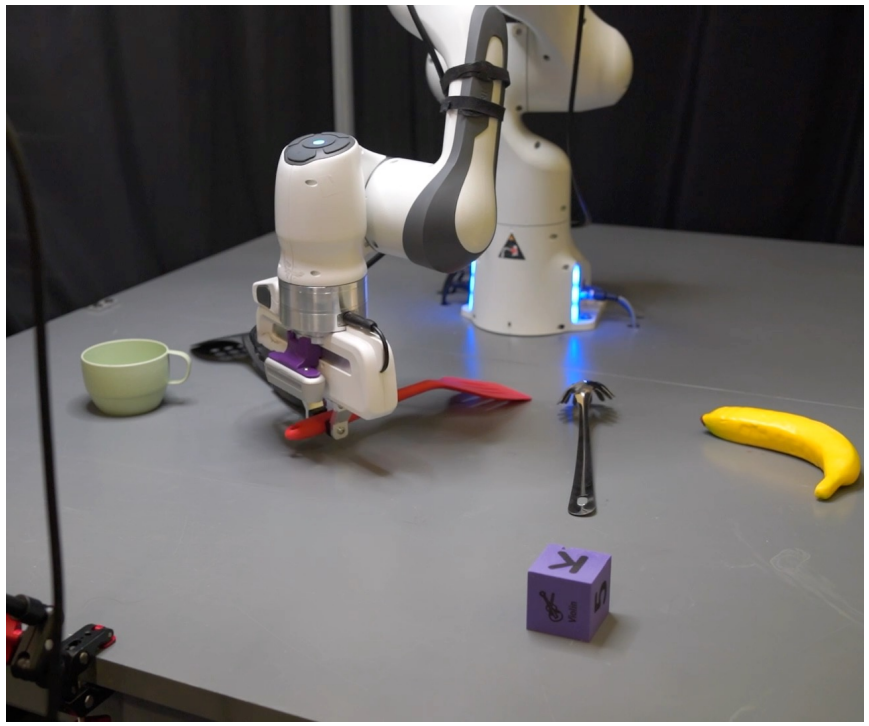
```
behavior hook(target, tool):
  body:
    achieve holding(tool, ?grasp1)
    move-with-contact(tool, target, ?traj)
    place_down(tool)
    grasp(target, ?grasp2)
```



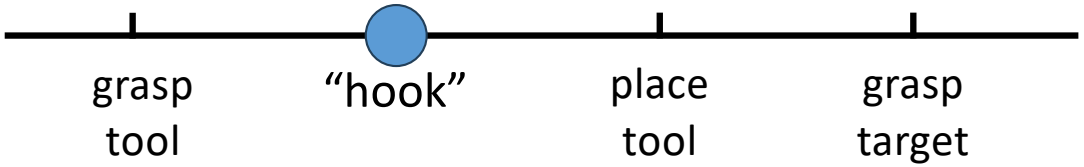
Mao, Lozano-Perez, Tenenbaum, Kaelbling. CoRL 2023. Learning Reusable Manipulation Strategies.

The Contact Mode Subgoals in Hook-Using

Key idea: model manipulation "strategies" as a program of a sequence of subgoals about contacts among objects



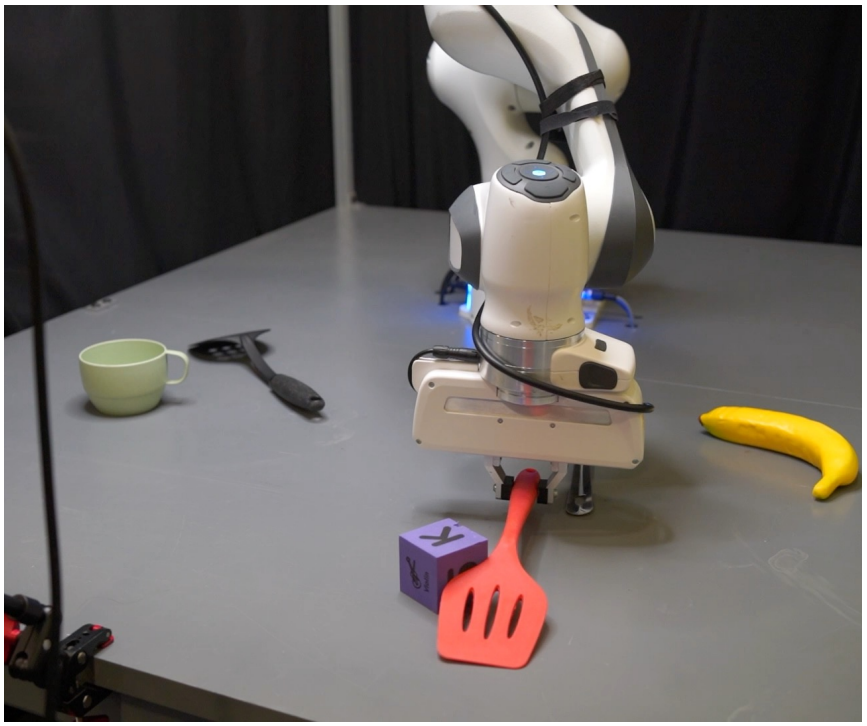
```
behavior hook(target, tool):
  body:
    achieve holding(tool, ?grasp1)
    move-with-contact(tool, target, ?traj)
    place_down(tool)
    grasp(target, ?grasp2)
```



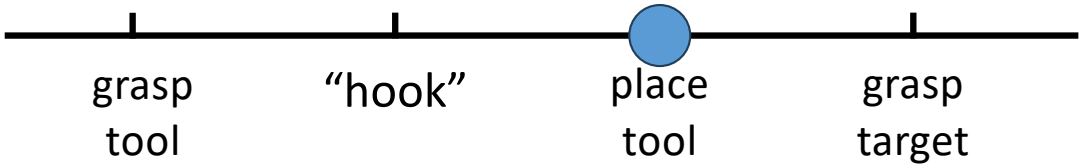
Mao, Lozano-Perez, Tenenbaum, Kaelbling. CoRL 2023. Learning Reusable Manipulation Strategies.

The Contact Mode Subgoals in Hook-Using

Key idea: model manipulation "strategies" as a program of a sequence of subgoals about contacts among objects



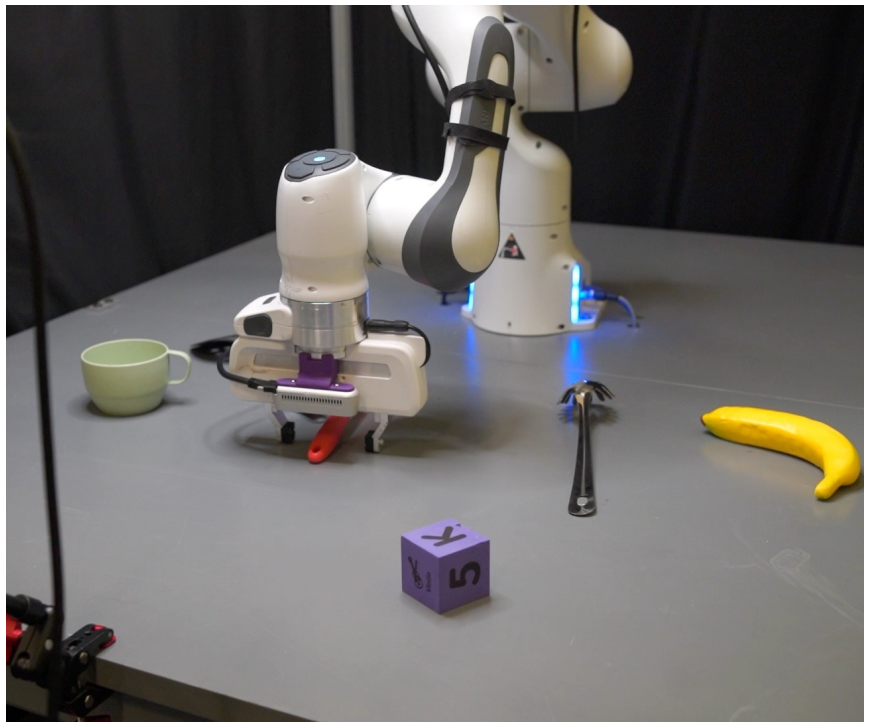
```
behavior hook(target, tool):
  body:
    achieve holding(tool, ?grasp1)
    move-with-contact(tool, target, ?traj)
    place_down(tool)
    grasp(target, ?grasp2)
```



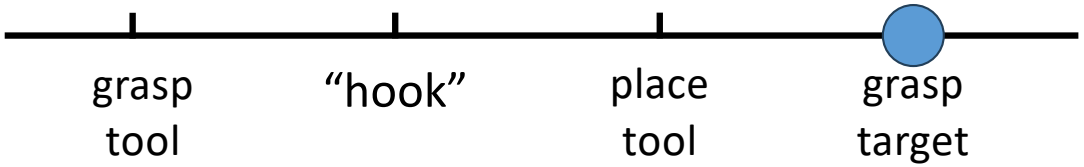
Mao, Lozano-Perez, Tenenbaum, Kaelbling. CoRL 2023. Learning Reusable Manipulation Strategies.

The Contact Mode Subgoals in Hook-Using

Key idea: model manipulation "strategies" as a program of a sequence of subgoals about contacts among objects



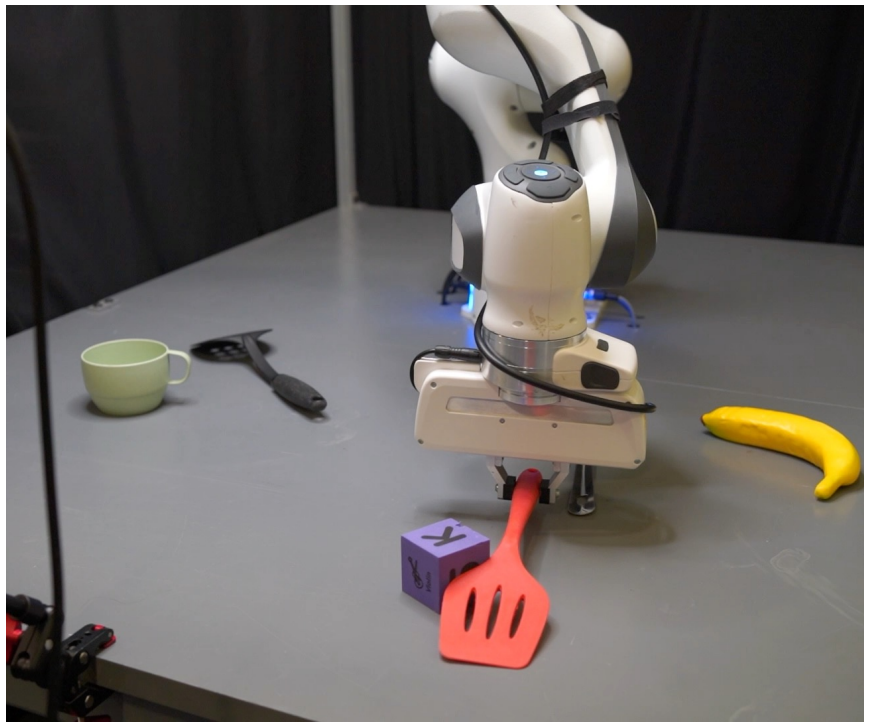
```
behavior hook(target, tool):
  body:
    achieve holding(tool, ?grasp1)
    move-with-contact(tool, target, ?traj)
    place_down(tool)
    grasp(target, ?grasp2)
```



Mao, Lozano-Perez, Tenenbaum, Kaelbling. CoRL 2023. Learning Reusable Manipulation Strategies.

The Contact Mode Subgoals in Hook-Using

Key idea: model manipulation "strategies" as a program of a sequence of subgoals about contacts among objects



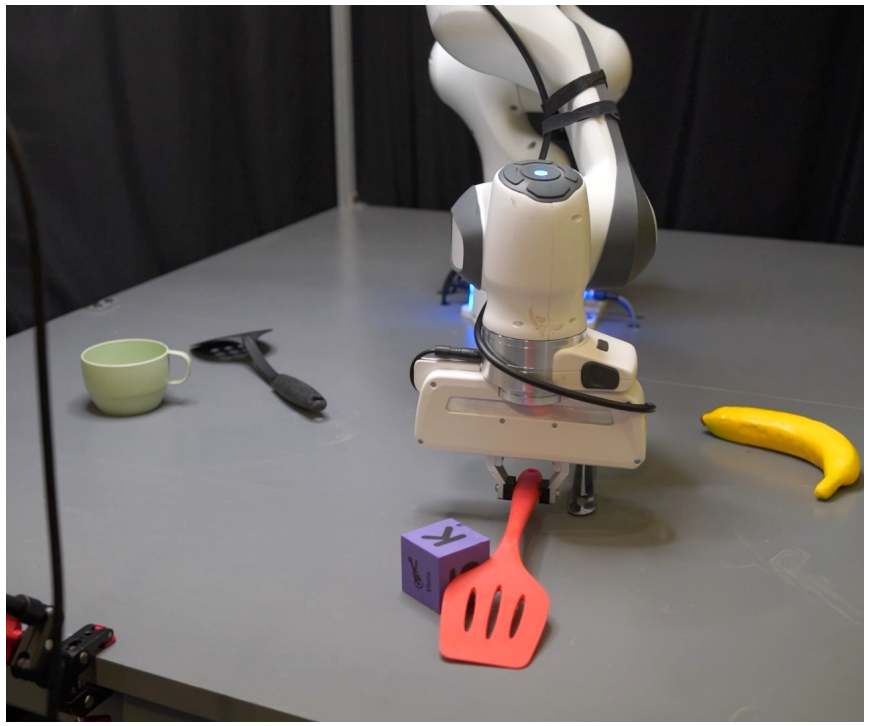
```
behavior hook(target, tool):
  body:
    achieve holding(tool, ?grasp1)
    move-with-contact(tool, target, ?traj)
    place_down(tool)
    grasp(target, ?grasp2)
```

The sequence of subgoals can be learned from a single demonstration!

Mao, Lozano-Perez, Tenenbaum, Kaelbling. CoRL 2023. Learning Reusable Manipulation Strategies.

The Contact Mode Subgoals in Hook-Using

Key idea: model manipulation "strategies" as a program of a sequence of subgoals about contacts among objects



```
behavior hook(target, tool):
  body:
    achieve holding(tool, ?grasp1)
    move-with-contact(tool, target, ?traj)
    place_down(tool)
    grasp(target, ?grasp2)
```

The sequence of subgoals can be learned from a single demonstration!

Concrete values for grasping and trajectories vary across instances

Mao, Lozano-Perez, Tenenbaum, Kaelbling. CoRL 2023. Learning Reusable Manipulation Strategies.

Learning Contact Constraints and Values

Single Demo:

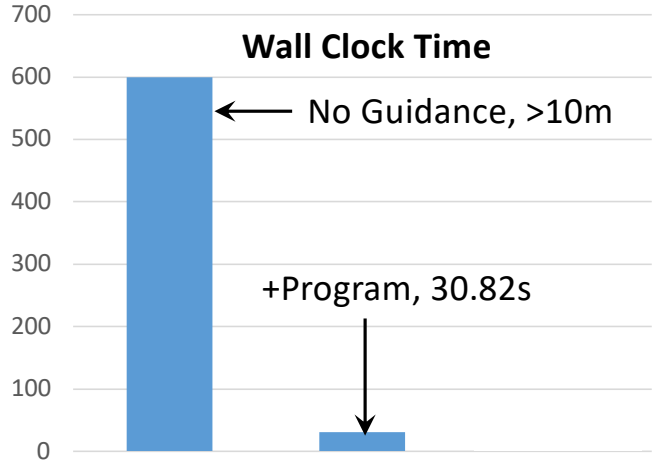
Using a "hook"

The high-level concept as a constraint set

```
behavior hook(target, tool):
  body:
    achieve holding(tool, ?grasp1)
    move-with-contact(tool, target, ?traj)
    place_down(tool)
    grasp(target, ?grasp2)
```

New Instance

Sampling-Based Opt. following the Program



*: The demonstration is given in simulation so we can extract contact points directly

Learning Contact Constraints and Values

Single Demo:

Using a "hook"

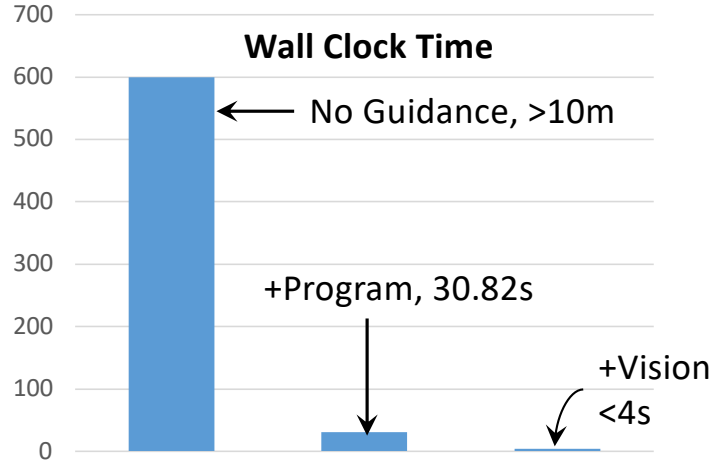
The high-level concept as a constraint set

```
behavior hook(target, tool):
  body:
    achieve holding(tool, ?grasp1)
    move-with-contact(tool, target, ?traj)
    place_down(tool)
    grasp(target, ?grasp2)
```

New Instance

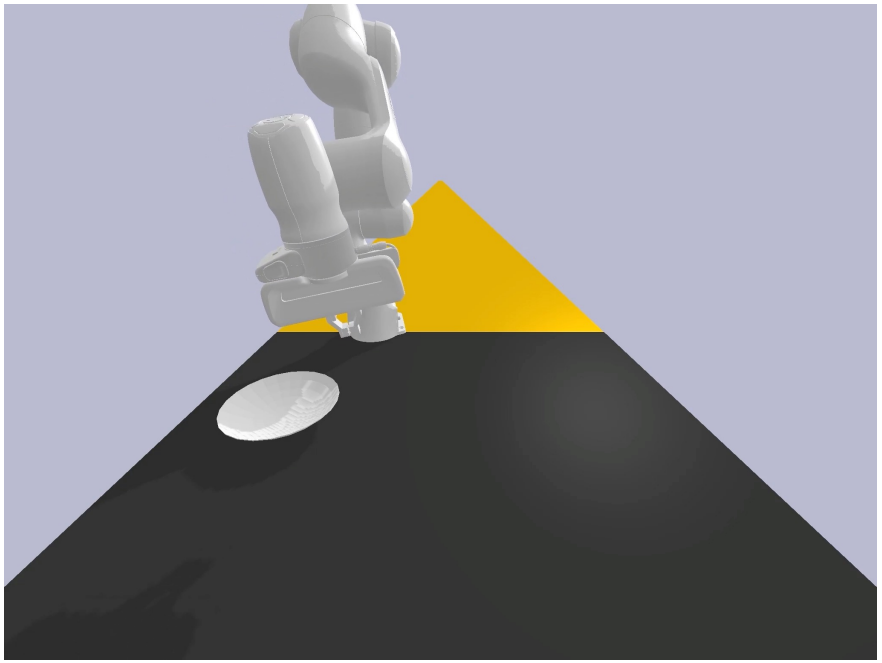
Sampling-Based Opt. following the Program

Leveraging Additional Visual Cues



*: The demonstration is given in simulation so we can extract contact points directly

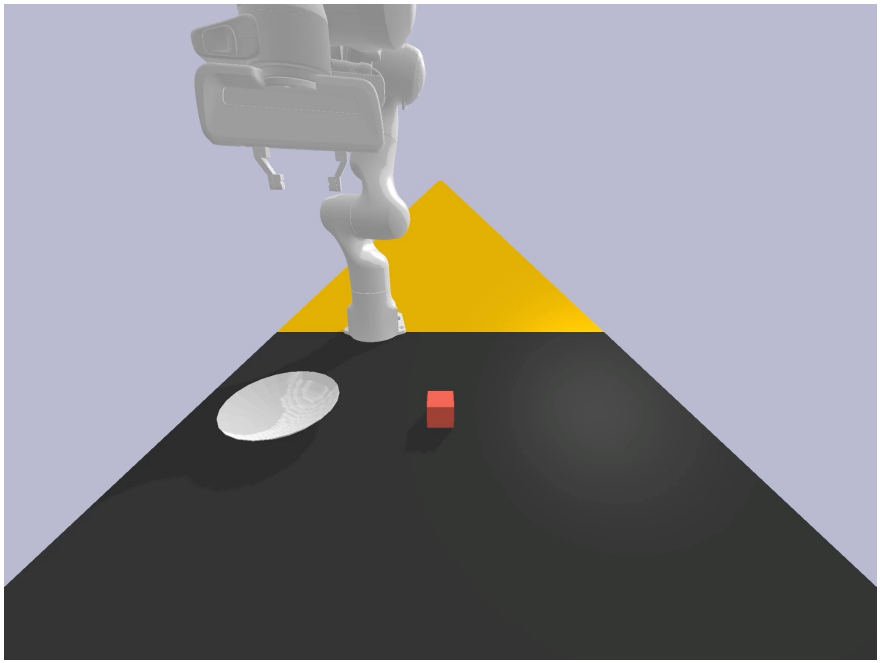
Learning Mechanisms Improves Planning Efficiency



Goal:
`holding(plate)`

Method	Edge	Hook	Lever	Poking	CoM	Slope&Blocker
Basis Ops Only	89.45±5.53	>600	523.18±9.22	>600	19.30±2.82	>600
Ours (Macro+Sampler)	**0.57**±0.05	**3.84**±1.56	1.55±0.29	**97.76**±10.67	**0.97**±0.09	**4.11**±0.94

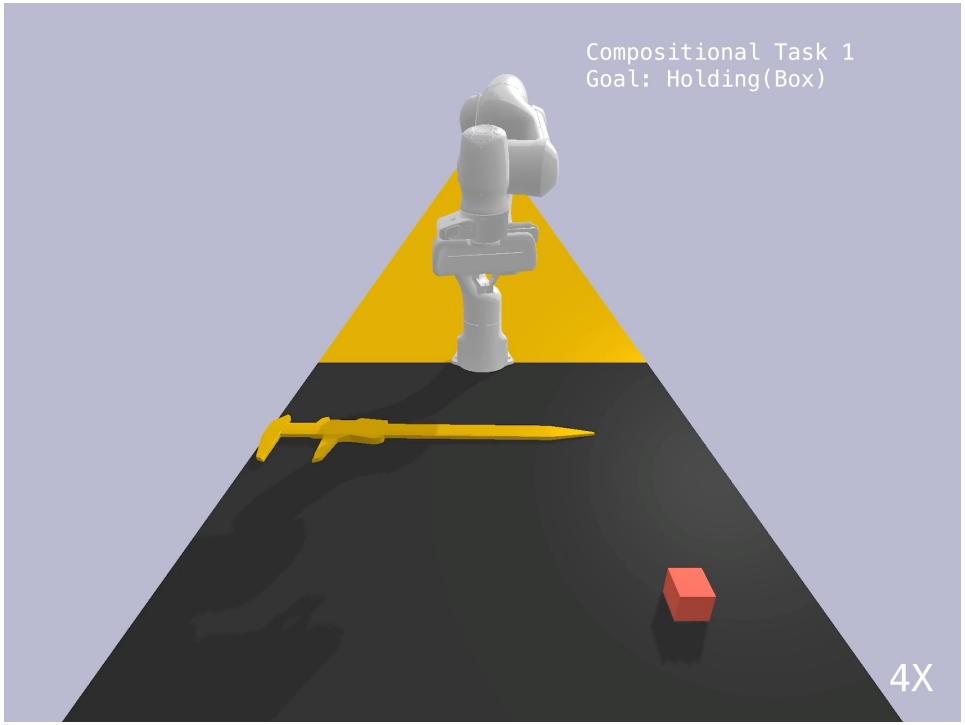
Learning Mechanisms Improves Planning Efficiency



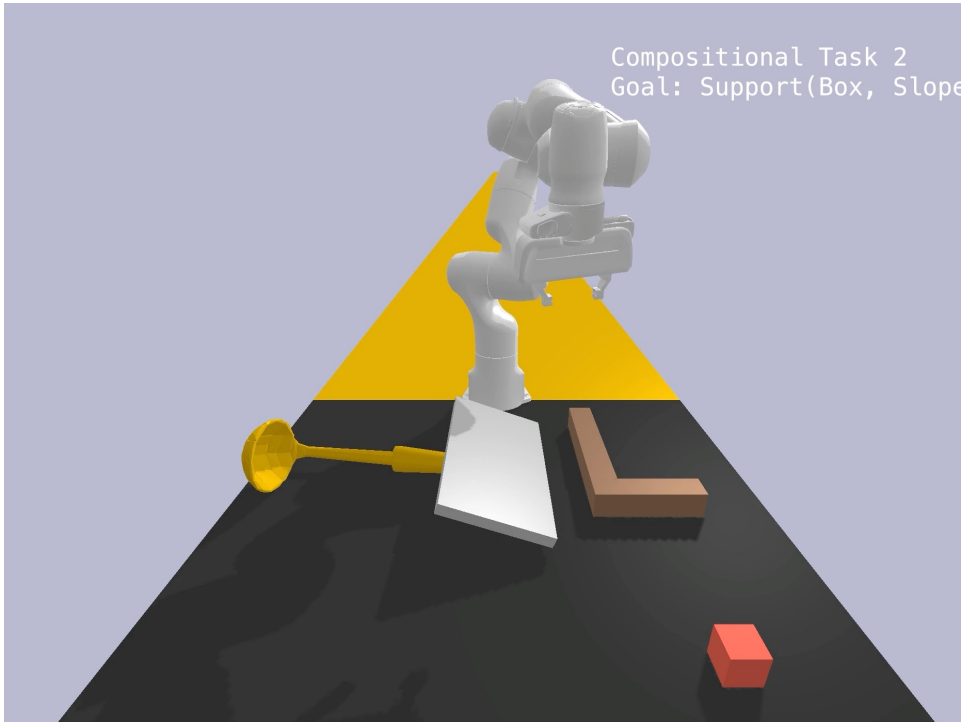
Goal:
`holding(plate)`

Method	Edge	Hook	Lever	Poking	CoM	Slope&Blocker
Basis Ops Only	89.45±5.53	>600	523.18±9.22	>600	19.30±2.82	>600
Ours (Macro+Sampler)	**0.57**±0.05	**3.84**±1.56	1.55±0.29	**97.76**±10.67	**0.97**±0.09	**4.11**±0.94

Further compose learned mechanisms via planning



Goal: holding(box)
The caliper is too flat to be grasped.



Goal: on(box, ramp)
Box may slide down the ramp.

We currently have a patchwork of the needed parts

1. Learned perception plus planning yields strong generalization
2. We can learn to ground predicate symbols from unlabeled demonstrations
3. We can learn generative models for continuous parameters by experimentation
4. We can learn causal action model structures
5. We can learn "mechanisms" from a single example
6. Generative models can be compositionally structured and learned
7. We can use LLM and LVMs to guide learning
8. Many of these ideas can be integrated, now,
   into a real robot

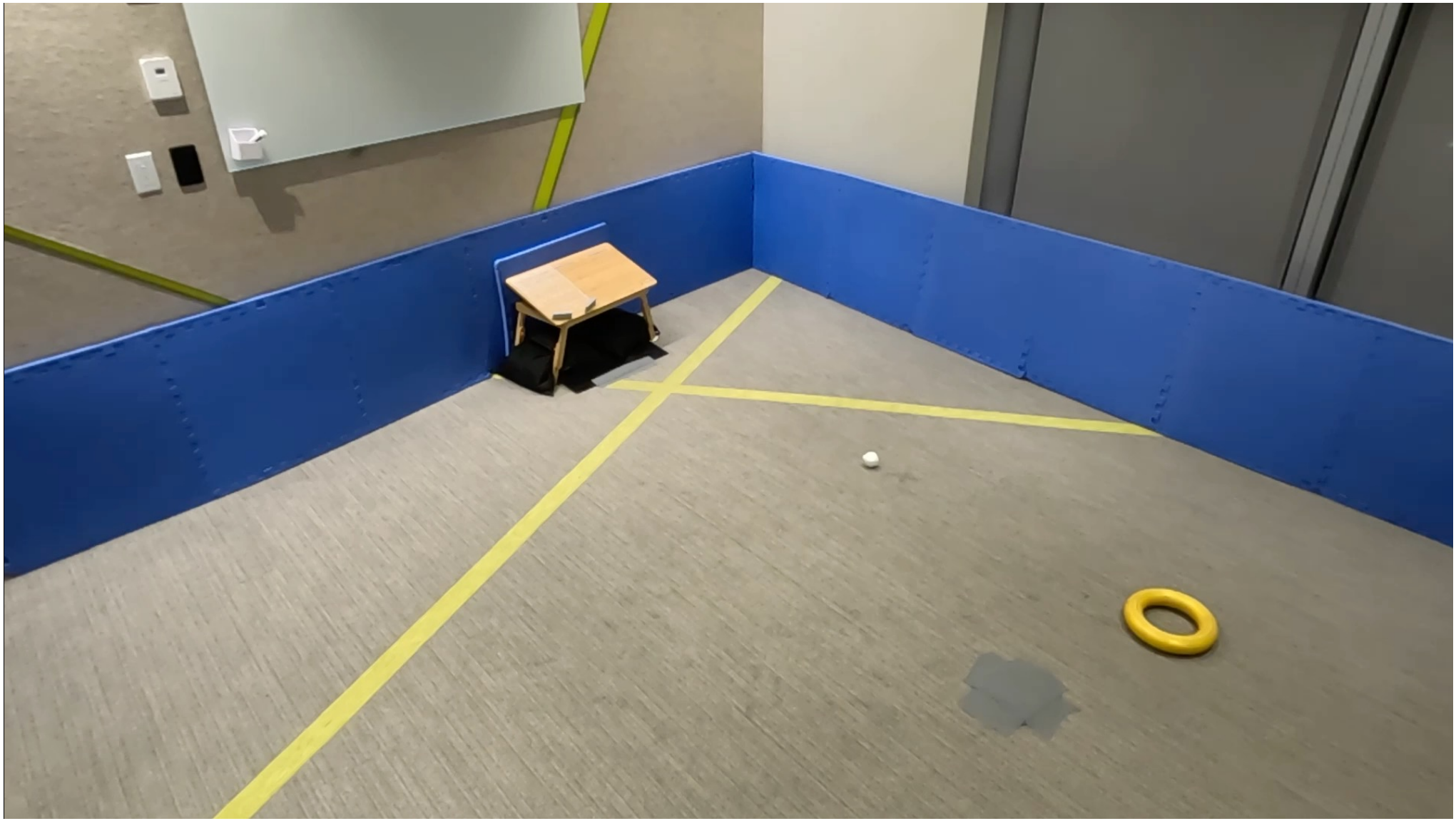
Planning to practice:
Self-motivated, fully independent, task driven learning

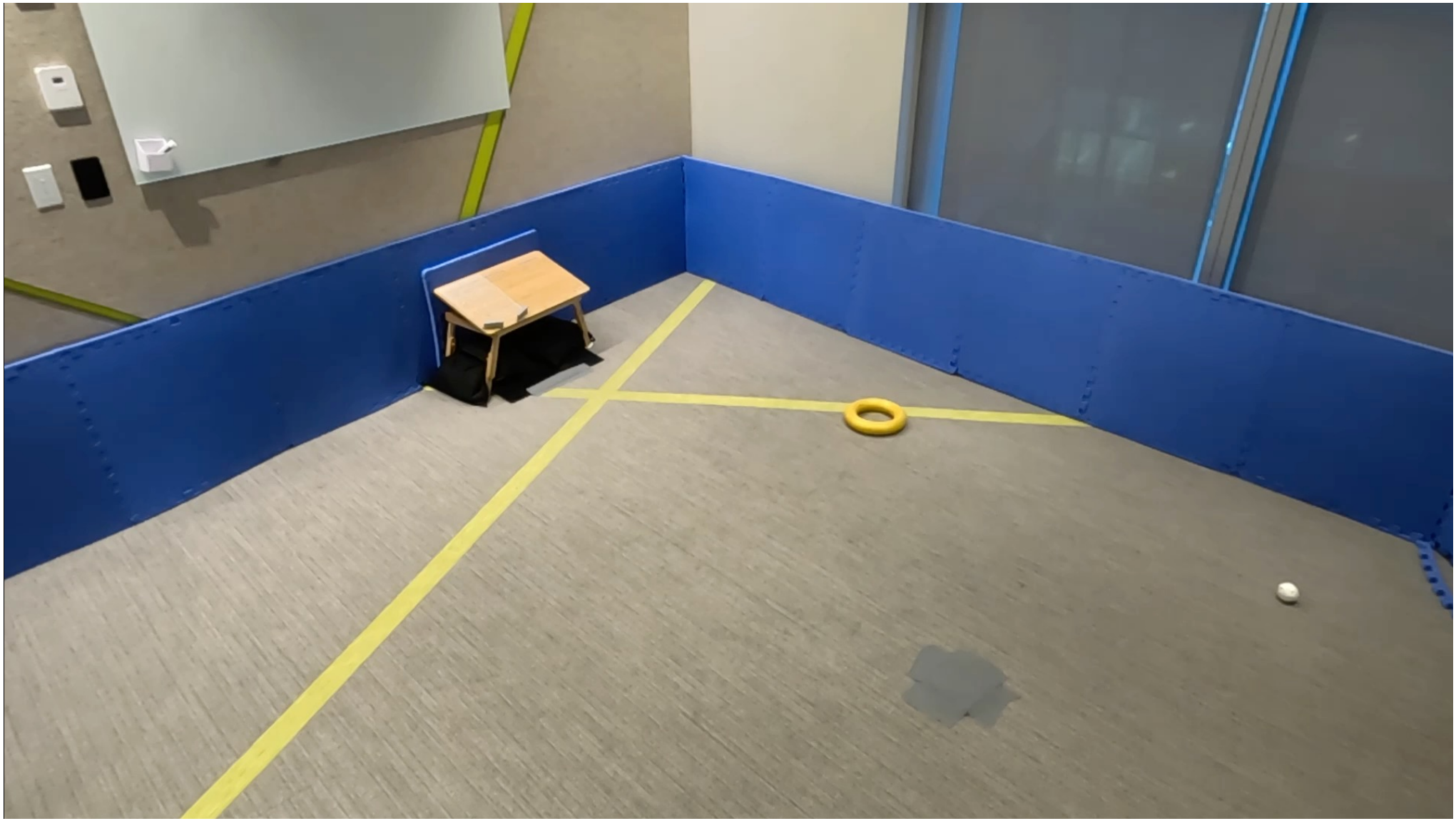
Robot

- Has a set of parameterized skills
- Knows a task distribution
- Is not very good at deploying those skills
- Has some free time on its hands
- Decides to practice:
  - Choose to practice skills that are important for its tasks and that it's bad at
  - Plans, using its current skills:
    - to decide which skills are important for its tasks
    - to set up the environment so that it can practice the skill
  - Plan to **practice important** skills you think you **can improve at** at!
  - Plan to **use** skills you think you are **good** at!

Boston Dynamics AI INSTITUTE

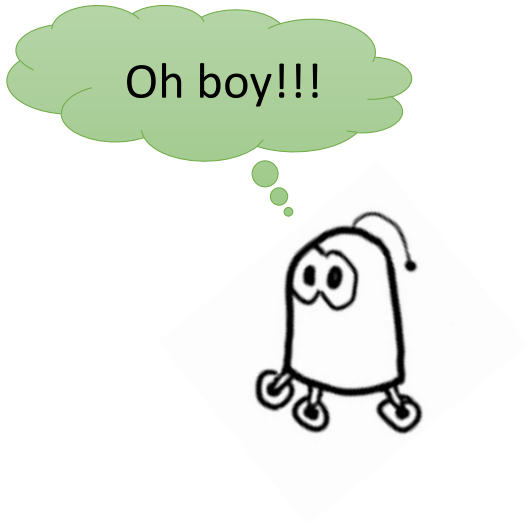
Nishanth Kumar, Tom Silver, Willie McClinton, Linfeng Xiao, BDAII colleagues, arxiv2024





Steps toward an integrated real robot

- Exploits factorial structure for planning
- Exploits world knowledge about geometry and kinematics
- Uses generative models for perception
- Learns generative model for actions
- Learns compositionally and incrementally
- Knows what it doesn't know how to do
- Uses these foundations to generalize sensibly and robustly



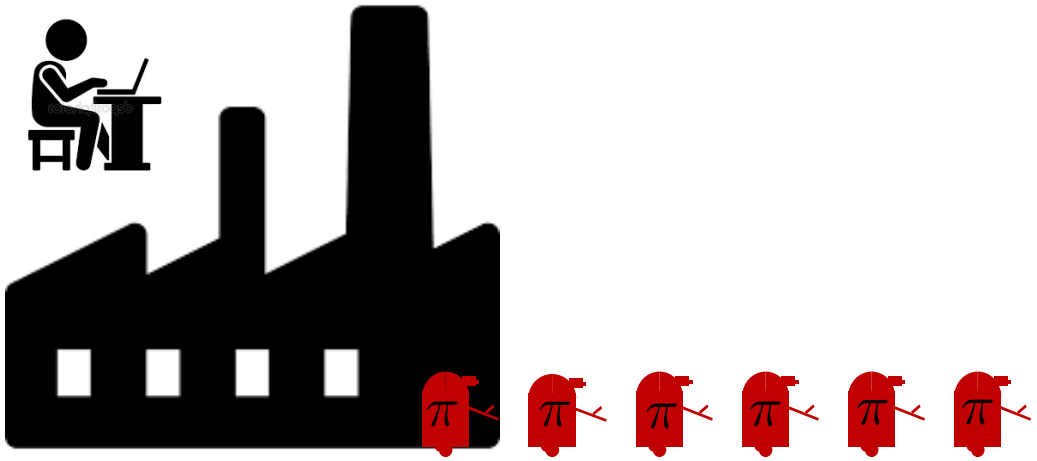

Cleanup Playroom Environment

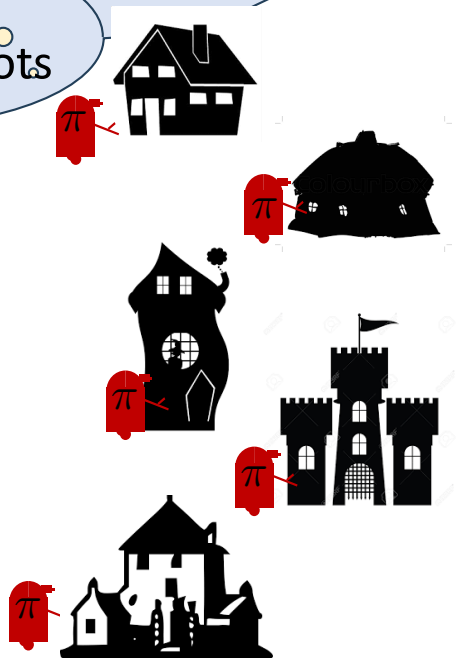
Practical (**rational robot**) factory

Compose
- fundamental algorithms
- physical constraints
- world knowledge in pretrained models

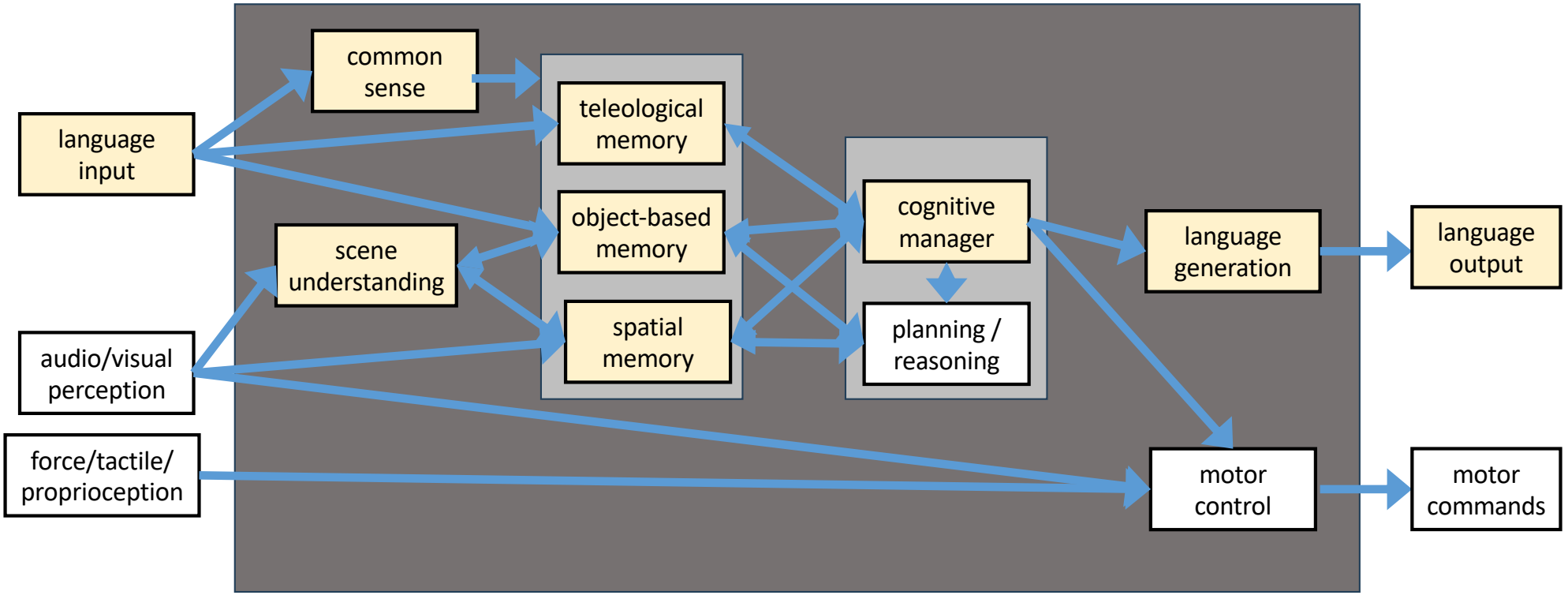
to provide rational designs for runtime rational robots

$$E_{s\sim\text{Belief}}\left[\sum_t R(s,a) \mid \text{continue optimally}\right]$$





Things we didn't talk much about!



Many other important directions!

- Meta-reasoning:
  - Hierarchical planning, intention management
  - Posing tractable sub-problems
- How can we design flexible memory systems for space and objects that play well with other modules?
- How can we address all of these problems when there is substantial, long-term partial observability?
  - See lpk rant : The last four frames is not all you need! https://youtu.be/AtNGJeXTXu4

Huge thanks to students and colleagues!

- Rohan Chitnis
- Aidan Curtis
- Yilun Du
- Xiaolin Fang
- Caelan Garrett
- Nishanth Kumar
- Tomas Lozano-Perez
- Jiayuan Mao
- Willie McClinton
- Jorge Mendez-Mendez
- Tom Silver
- Josh Tenenbaum
- Xi Wang

We gratefully acknowledge support from

- NSF grant 2214177
- AFOSR grant FA9550-22-1-0249
- ONR MURI grant N00014-22-1-2740
- ARO grant W911NF-23-1-0034
- the MIT-IBM Watson Lab
- the Boston Dynamics Artificial Intelligence Institute
- the MIT Quest for Intelligence