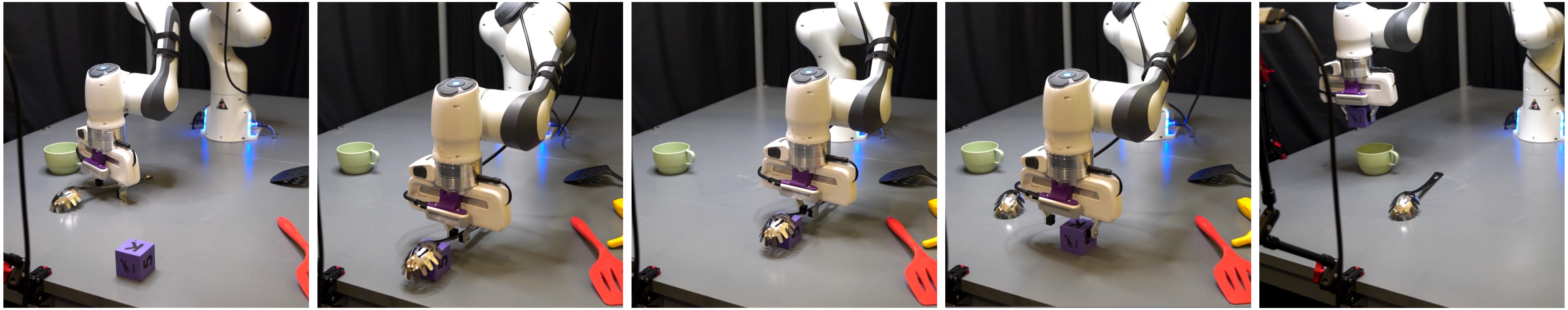


Building General-Purpose Robots with Compositional Action Abstractions

Jiayuan Mao

Massachusetts Institute of Technology

Towards General-Purpose Robots

Goal:

Having a robot that can do many tasks, across many environments

Towards General-Purpose Robots

Goal:

Having a robot that can do many tasks, across many environments

Towards General-Purpose Robots

Goal:

Having a robot that can do <u>many tasks</u>, across <u>many environments</u>

The robot should make long-horizon plans with rich contact with the environment, and generalize to unseen objects, states, and goals

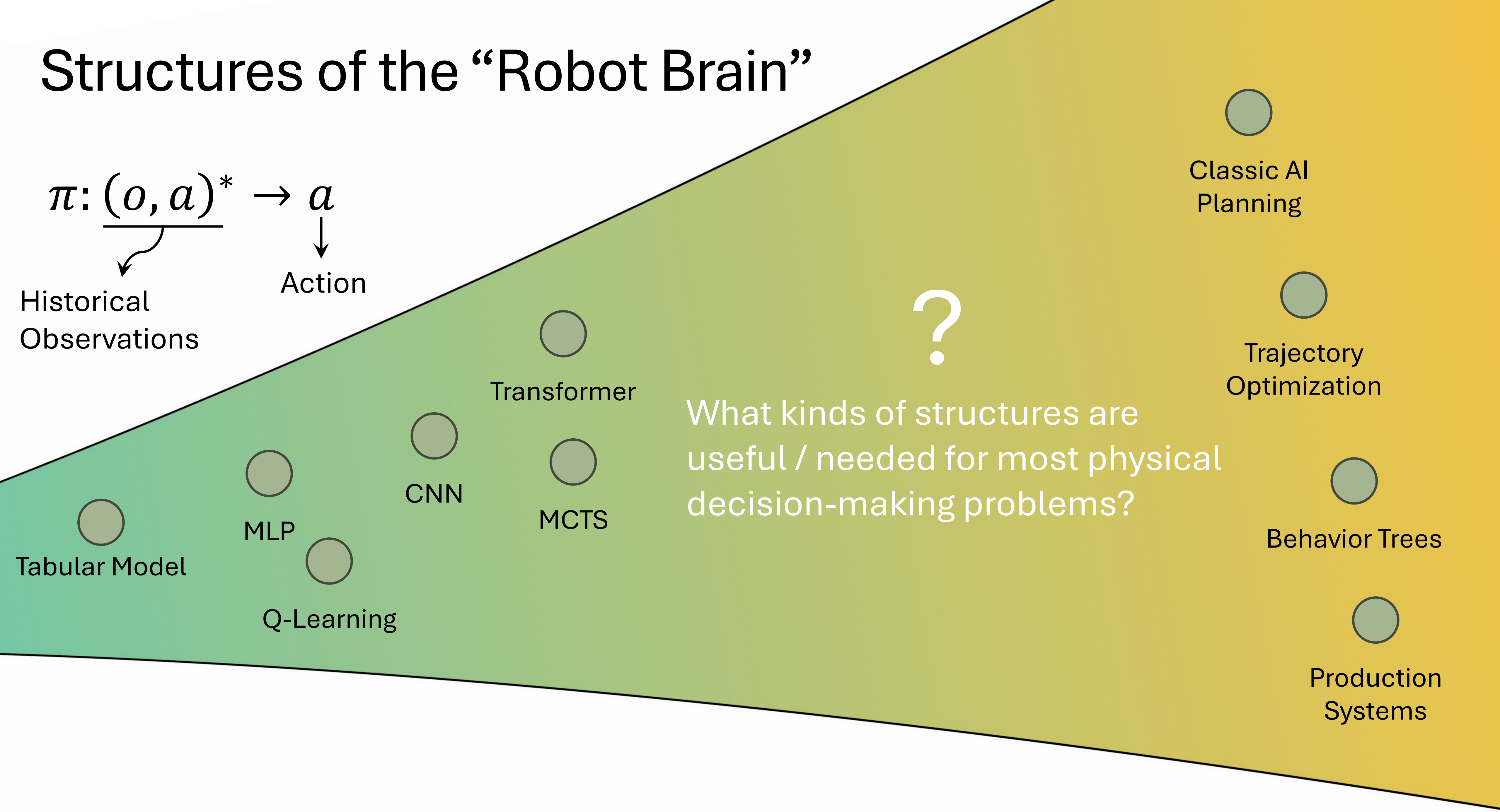
We want to achieve generalization from a feasible amount of data

Structures of the "Robot Brain"

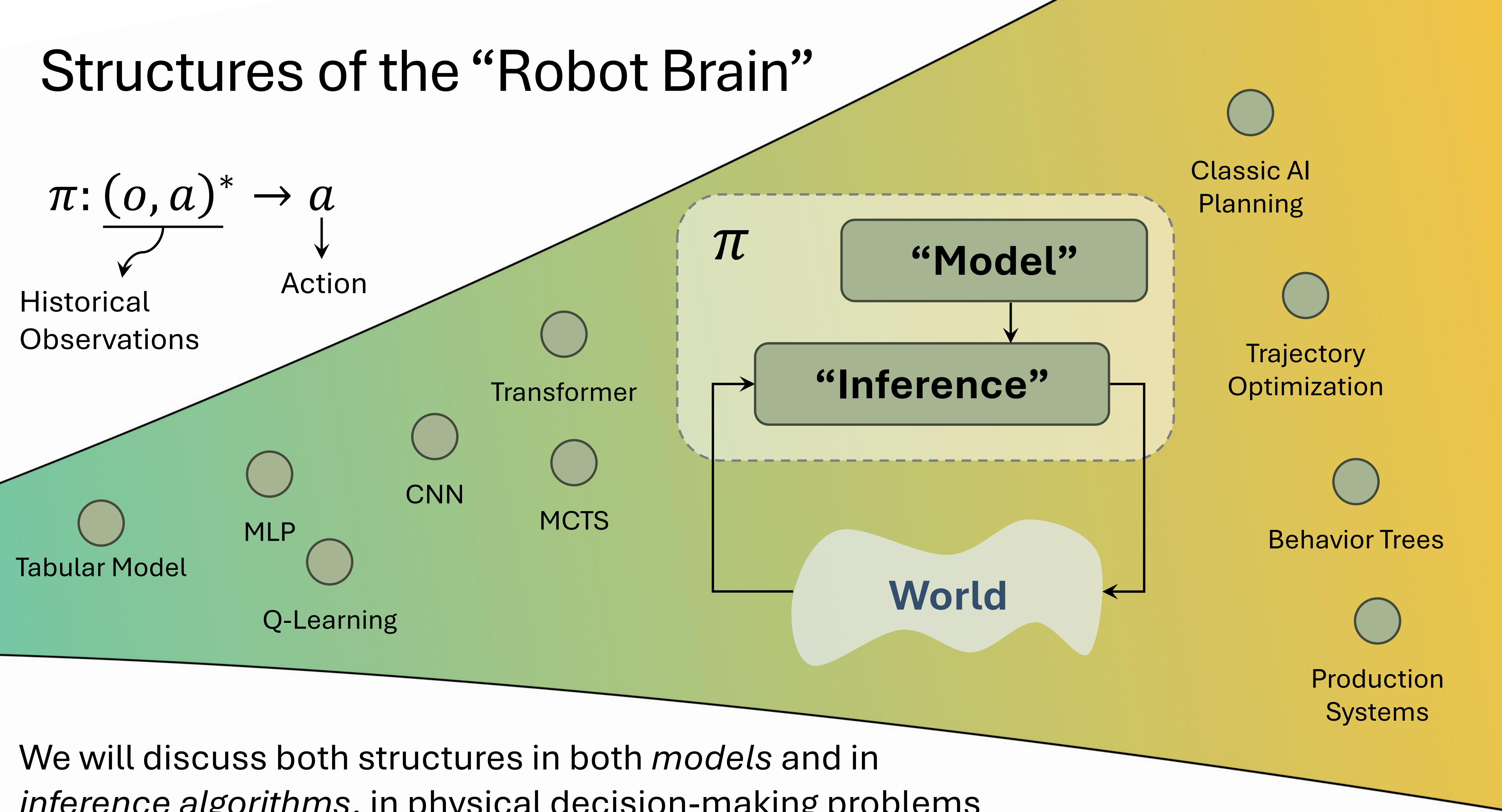
$$\pi: \underbrace{(o, a)^*}_{\text{Historical Observations}} \to \underbrace{a}_{\text{Action}}$$

Structures of the "Robot Brain"

$$\pi: \underbrace{(o, a)^*}_{\text{Historical Observations}} \rightarrow \underset{\text{Action}}{a}$$



Structures of the "Robot Brain"



We will discuss both structures in both *models* and in *inference algorithms*, in physical decision-making problems

Structures of the "Robot Brain"

State Representation

Monolithic — Compositional

Action Representation

As Feedforward Policies — As Causal Models

Model Acquisition

Machine Learned — Human Programmed

Structures of the “Robot Brain”

Tabular Policy Optimization

State Representation

Monolithic — Compositional

Action Representation

As Feedforward Policies — As Causal Models

Model Acquisition

Machine Learned — Human Programmed

Structures of the "Robot Brain"

Classical Planning

State Representation
Monolithic — Compositional

Action Representation
As Feedforward Policies — As Causal Models

Model Acquisition
Machine Learned — Human Programmed

Structures of the "Robot Brain"

Alpha Go

State Representation

Monolithic — Compositional

Action Representation

As Feedforward Policies — As Causal Models

Model Acquisition

Machine Learned — Human Programmed

Structures of the "Robot Brain"

Today

State Representation

Monolithic — Compositional

Action Representation

As Feedforward Policies — As Causal Models

Model Acquisition

Machine Learned — Human Programmed

Key Question: What’s an Action Anyway?

A Generic Action Description

```
action move-to-grasp(o: obj)
  body:
    find g: valid-grasp(g, o)
    find t: valid-trajectory(o, g, t)
    achieve robot-at == t[0]
    call robot-controller-move(t)
    call robot-controller-grasp()
  eff:
    robot-at = t[-1])
    holding[o] = g
```

Find parameters that has certain properties

Achieve state conditions

Call controllers to generate torque

Update the state

Connection to (Hierarchical) Policy

```
action move-to-grasp(o: obj)
  body:
    find g: valid-grasp(g, o)
    find t: valid-trajectory(o, g, t)
    achieve robot-at == t[0]
    call robot-controller-move(t)
    call robot-controller-grasp()
  eff:
    robot-at = t[-1])
    holding[o] = g
```

Option parameter prediction

Hierarchical decomposition

Call controllers to generate torque

Hierarchical forward models (for planning)

Connection to (Hierarchical) Task and Motion Planning

```
action move-to-grasp(o: obj)
  body:
    find g: valid-grasp(g, o)
    find t: valid-trajectory(o, g, t)
    achieve robot-at == t[0]
    call robot-controller-move(t)
    call robot-controller-grasp()
  eff:
    robot-at = t[-1])
    holding[o] = g
```

Constraints to satisfy

Preconditions

Call controllers to generate torque

Effects

Connecting Policies and Planning Descriptions

```
action move-to-grasp(o: obj)
  body:
    find g: valid-grasp(g, o)
    find t: valid-trajectory(o, g, t)
    achieve robot-at == t[0]
    call robot-controller-move(t)
    call robot-controller-grasp()
  eff:
    robot-at = t[-1])
    holding[o] = g
```

Insight: If you know the order and the way for achieving preconditions...

Actions as Causal Models   Actions as Feed-forward Policies

What planning problems can a relational neural network solve? *Mao*, Lozano-Perez, Tenenbaum, Kaelbling. NeurIPS 2023.

Connecting Policies and Planning Descriptions

```
action move-to-grasp(o: obj)
  body:
    find g: valid-grasp(g, o)
    find t: valid-trajectory(o, g, t)
    achieve robot-at == t[0]
    call robot-controller-move(t)
    call robot-controller-grasp()
  eff:
    robot-at = t[-1])
    holding[o] = g
```

Theorem:
When there exist decompositions where at most k subgoals interact, policy complexity is $N^{O(k)}$

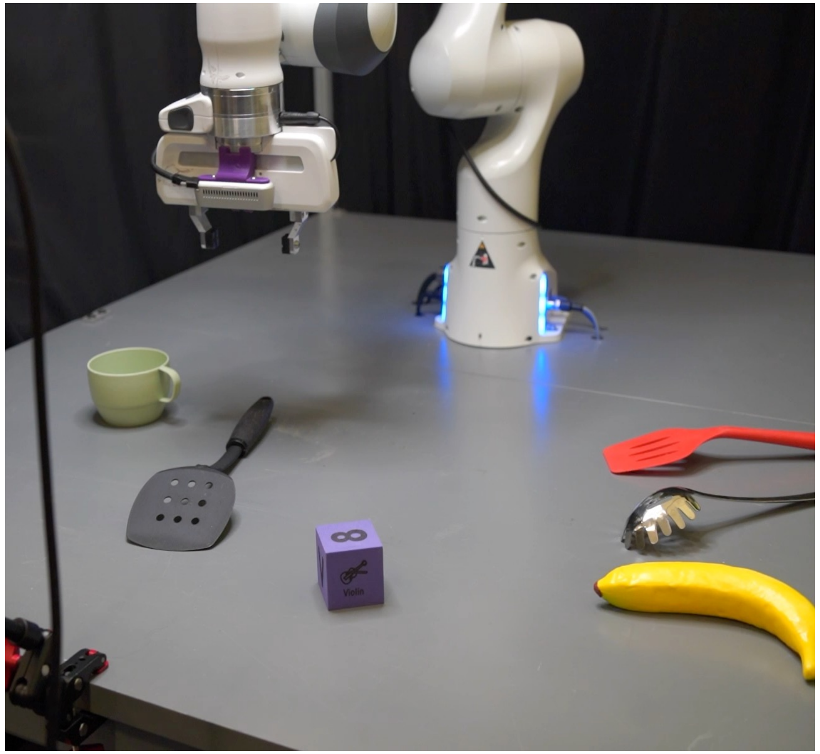
Planning enables constructing those policies with a compact model

Insight: If you know the order and the way for achieving preconditions...

Actions as Causal Models — Actions as Feed-forward Policies

What planning problems can a relational neural network solve? *Mao*, Lozano-Perez, Tenenbaum, Kaelbling. NeurIPS 2023.

Why Do We Need Compositional Abstractions?



States are described using (state abstraction) :

`holding(cube)`

`in(cube, cup)`

They can be composed: “all cubes in the cup”

Actions are described using (temporal abstraction):

`grasp(object)`

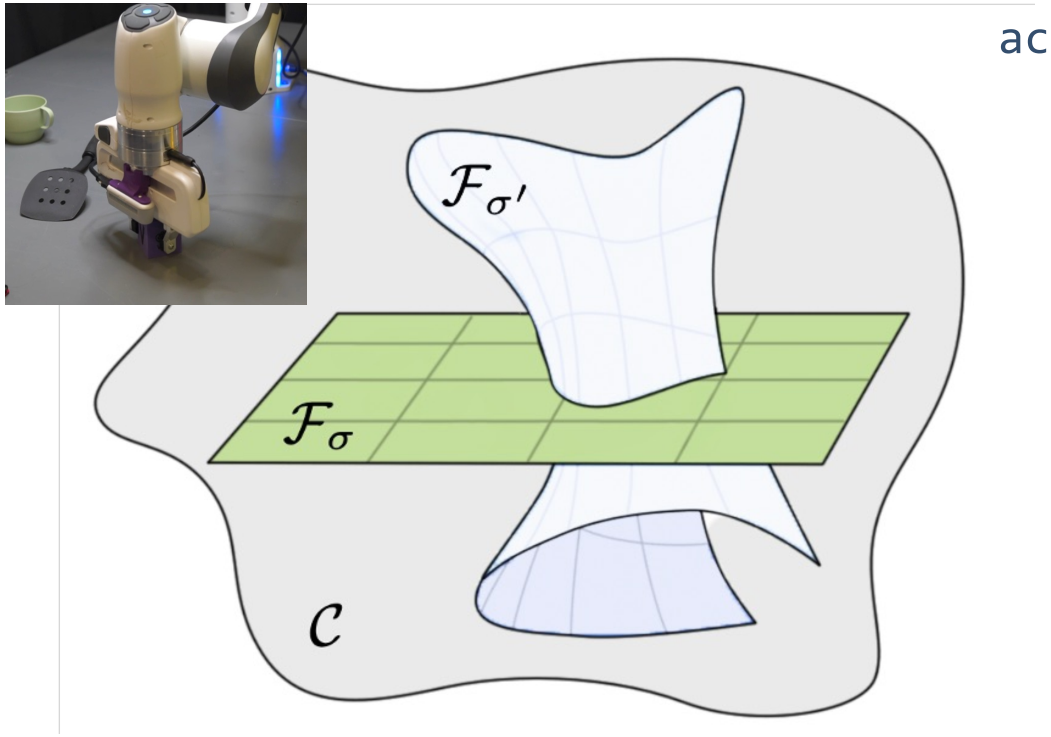
`place-in(object, container)`

They can be sequentially or hierarchically composed

Why Do We Need Compositional Abstractions?

Compositional abstraction brings **sparsity** and **temporal decomposition**

Models are sets of low-dimensional manifolds in the configuration space



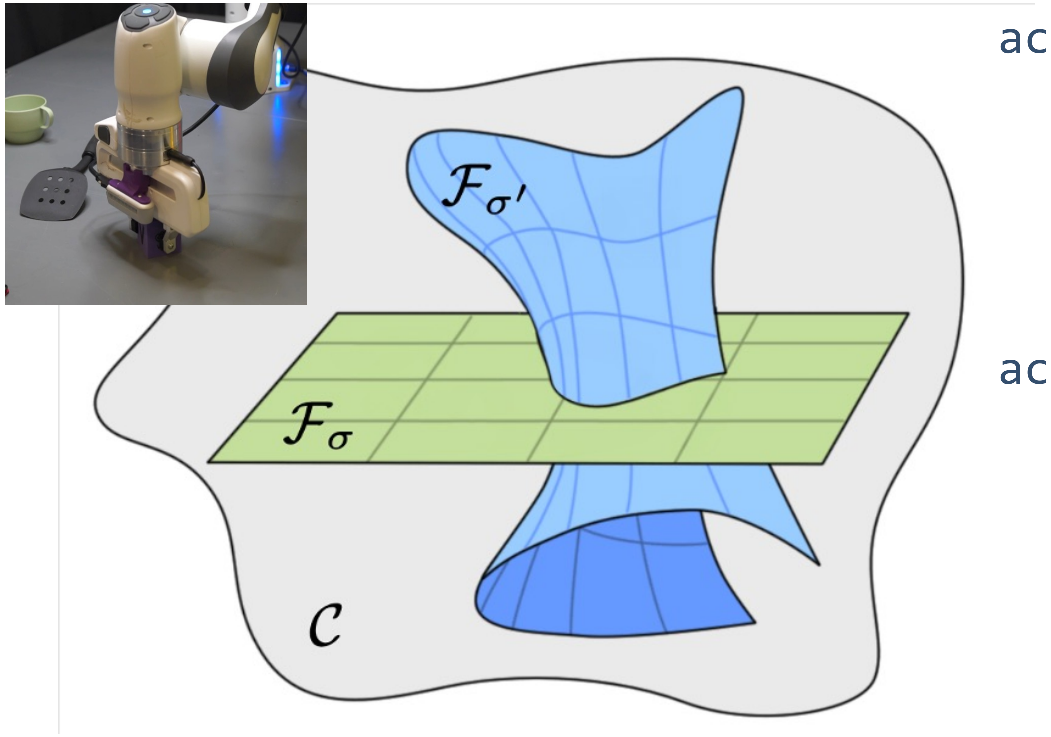

```
action move-to-grasp(o: obj)
  find valid-grasp(o, g) valid-traj(o, g, t);
  achieve robot-at(t[0]); call ...
  eff: robot-at(t[-1]), holding(o, g)
```

Figure: Hauser and Latombe. Multi-Modal Motion Planning in Non-Expansive Spaces.

Why Do We Need Compositional Abstractions?

Compositional abstraction brings **sparsity** and **temporal decomposition**

Models are sets of low-dimensional manifolds in the configuration space




```
action move-to-grasp(o: obj)
  find valid-grasp(o, g) valid-traj(o, g, t);
  achieve robot-at(t[0]); call ...
  eff: robot-at(t[-1]), holding(o, g)
action move-while-holding(o: obj, g: grasp)
  find valid-traj(o, g, t);
  achieve holding(o, g), robot-at(t[0]); call ...
  eff: robot-at(t[-1]), obj-at(...)
```

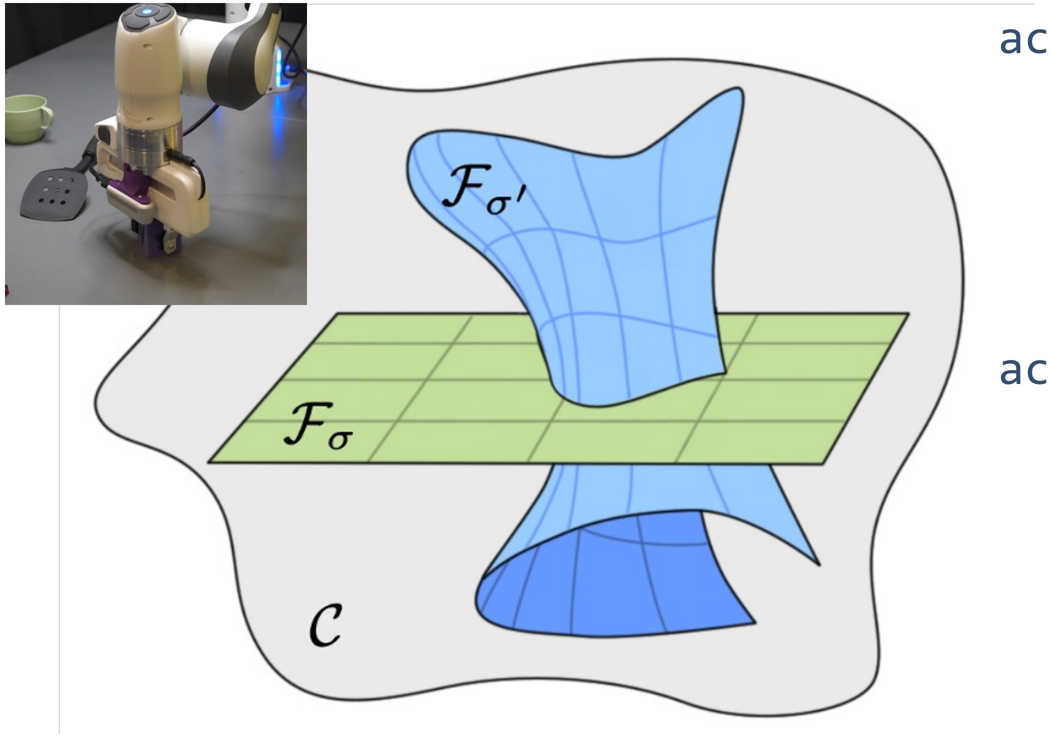
Figure: Hauser and Latombe. Multi-Modal Motion Planning in Non-Expansive Spaces.

Why Do We Need Compositional Abstractions?

Compositional abstraction brings **sparsity** and **temporal decomposition**

Models are sets of low-dimensional manifolds in the configuration space

They are connected at regions modeled by preconditions and effects




```
action move-to-grasp(o: obj)
  find valid-grasp(o, g) valid-traj(o, g, t);
  achieve robot-at(t[0]); call ...
  eff: robot-at(t[-1]), holding(o, g)
action move-while-holding(o: obj, g: grasp)
  find valid-traj(o, g, t);
  achieve holding(o, g), robot-at(t[0]); call ...
  eff: robot-at(t[-1]), obj-at(...)
```

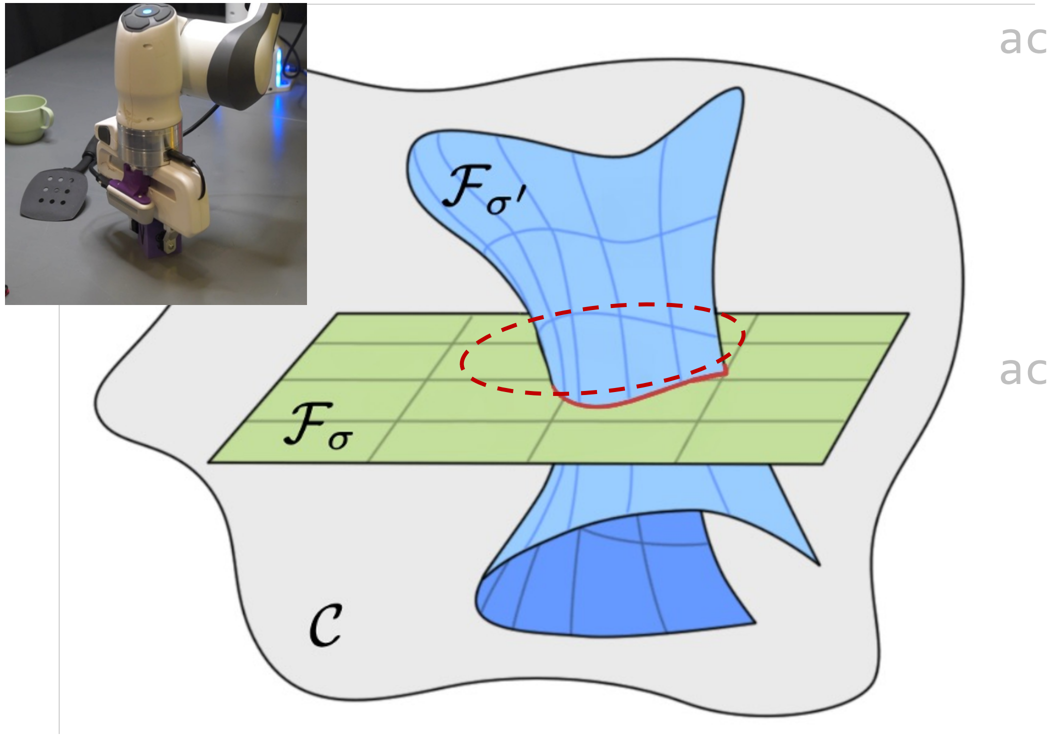
Figure: Hauser and Latombe. Multi-Modal Motion Planning in Non-Expansive Spaces.

Why Do We Need Compositional Abstractions?

Compositional abstraction brings **sparsity** and **temporal decomposition**

Models are sets of low-dimensional manifolds in the configuration space

They are connected at regions modeled by preconditions and effects




```
action move-to-grasp(o: obj)
  find valid-grasp(o, g) valid-traj(o, g, t);
  achieve robot-at(t[0]); call ...
  eff: robot-at(t[-1]), holding(o, g)
action move-while-holding(o: obj, g: grasp)
  find valid-traj(o, g, t);
  achieve holding(o, g), robot-at(t[0]); call ...
  eff: robot-at(t[-1]), obj-at(...)
```

Figure: Hauser and Latombe. Multi-Modal Motion Planning in Non-Expansive Spaces.

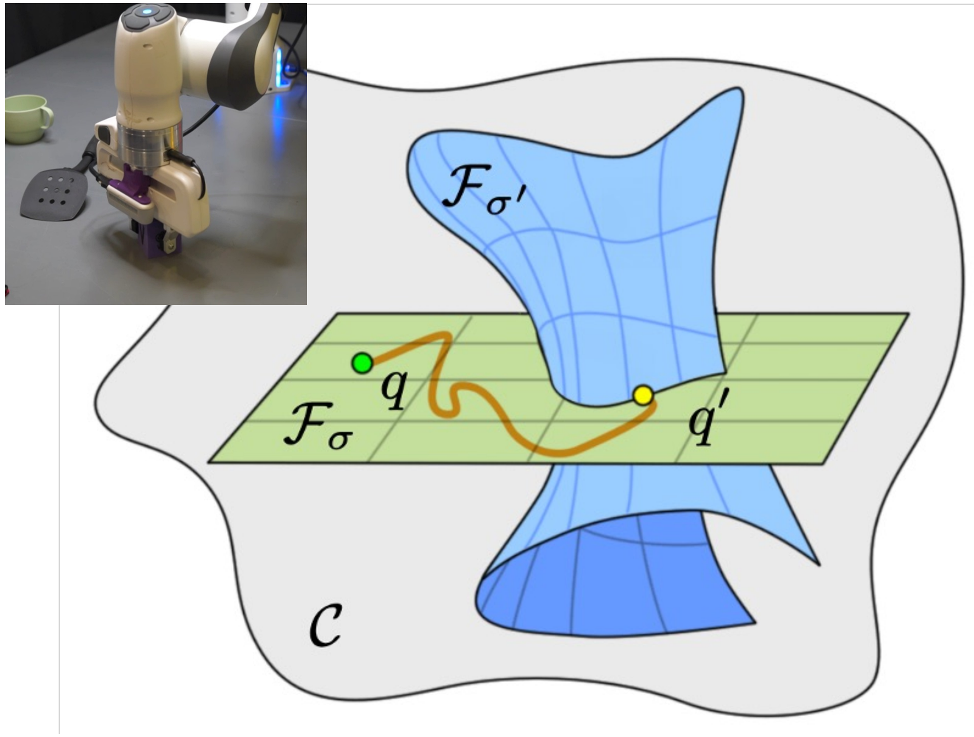
Why Do We Need Compositional Abstractions?

Compositional abstraction brings **sparsity** and **temporal decomposition**

Models are sets of low-dimensional manifolds in the configuration space

They are connected at regions modeled by preconditions and effects

They make planning easier by suggesting subgoals for planning




```
action move-to-grasp(o: obj)
  find valid-grasp(o, g) valid-traj(o, g, t);
  achieve robot-at(t[0]); call ...
  eff: robot-at(t[-1]), holding(o, g)

action move-while-holding(o: obj, g: grasp)
  find valid-traj(o, g, t);
  achieve holding(o, g), robot-at(t[0]); call ...
  eff: robot-at(t[-1]), obj-at(...)
```

Figure: Hauser and Latombe. Multi-Modal Motion Planning in Non-Expansive Spaces.

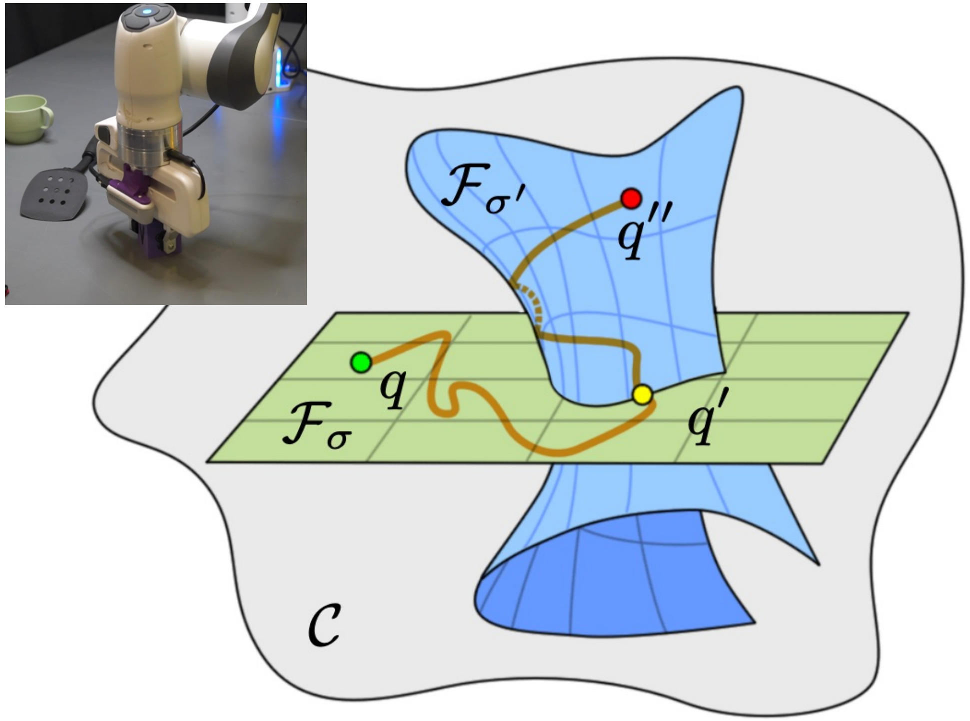
Why Do We Need Compositional Abstractions?

Compositional abstraction brings **sparsity** and **temporal decomposition**

Models are sets of low-dimensional manifolds in the configuration space

They are connected at regions modeled by preconditions and effects

They make planning easier by suggesting subgoals for planning




```
action move-to-grasp(o: obj)
  find valid-grasp(o, g) valid-traj(o, g, t);
  achieve robot-at(t[0]); call ...
  eff: robot-at(t[-1]), holding(o, g)

action move-while-holding(o: obj, g: grasp)
  find valid-traj(o, g, t);
  achieve holding(o, g), robot-at(t[0]); call ...
  eff: robot-at(t[-1]), obj-at(...)
```

Figure: Hauser and Latombe. Multi-Modal Motion Planning in Non-Expansive Spaces.

Structures of the "Robot Brain"

Compositional Action

State Representation
Monolithic — Compositional

Action Representation
As Feedforward Policies — As Causal Models

Model Acquisition
Machine Learned — Human Programmed

Structures of the “Robot Brain”

State Representation

Monolithic — Compositional

Compositional Action

Action Representation

As Feedforward Policies — As Causal Models

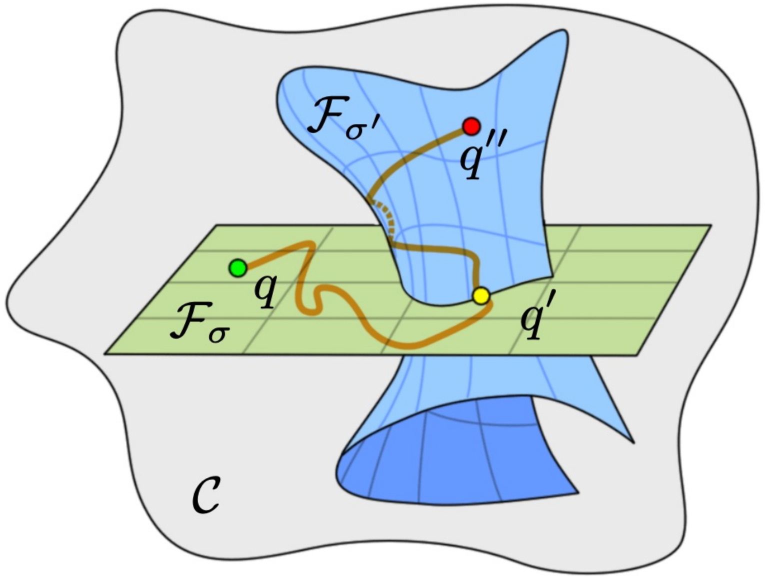
Model Acquisition

Machine Learned — Human Programmed

?

Two Parts of The Representation in Physical Domains

- **Structure:** abstract descriptions of mode families, object-invariant
  - Can be described by a factorized model over objects and their relations
  - Easy for humans to specify or get from LLMs. We call them “sketches”
- **Detail:** perception, geometry, physics
  - Hard for humans to write down. They are the **grounding** of the functions




```
action move-while-holding(o: obj, g: grasp)
  find valid-traj(o, g, t);
  achieve holding(o, g), robot-at(t[0]); call ...
  eff: robot-at(t[-1]), obj-at(...)
```

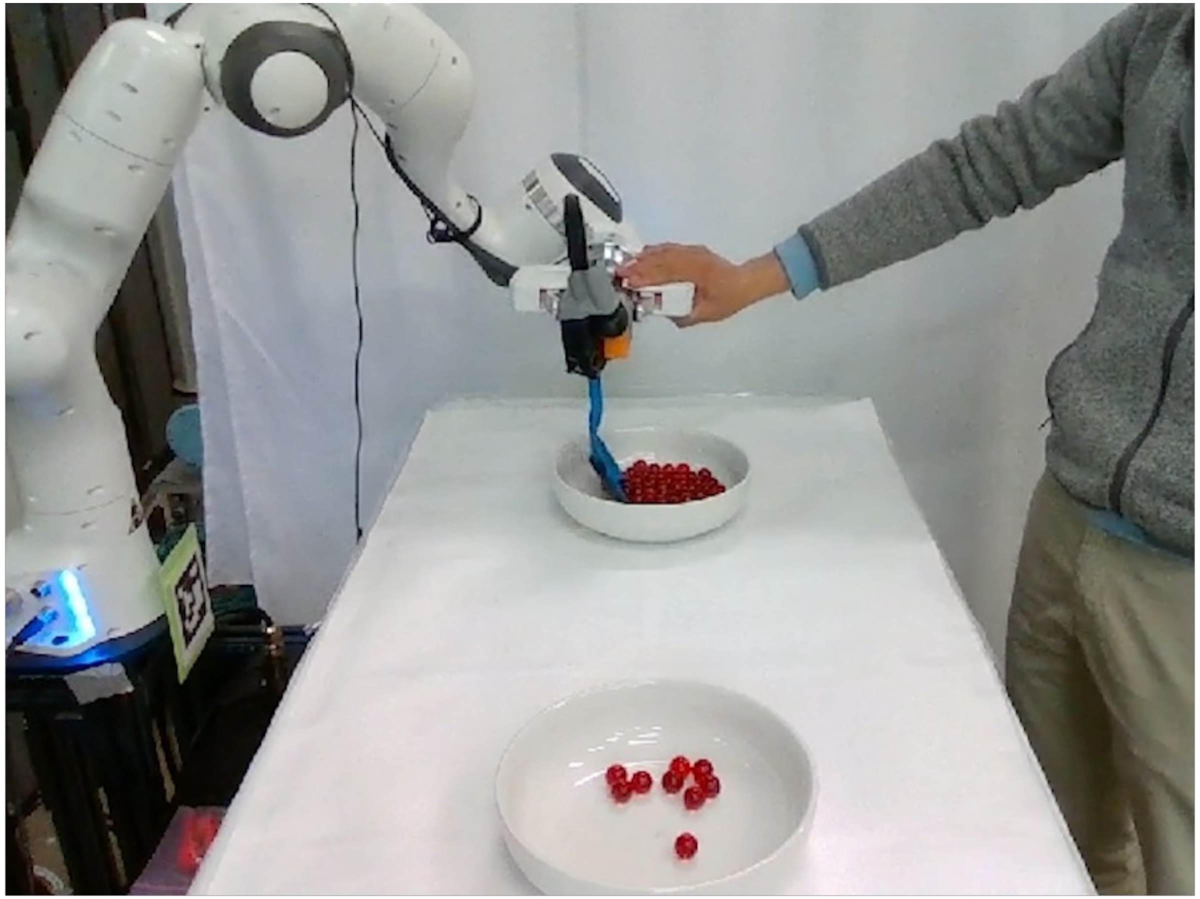
Structured Models With Factorization and Decomposition

- A skill (e.g., scooping) is a sequence of *intra-mode movements and inter-mode transitions, with parameters*

PDSketch: Integrated Domain Programming, Learning, and Planning. *Mao*, Lozano-Perez, Tenenbaum, Kaelbling. NeurIPS 2022.
Grounding Language Plans in Demonstrations through Counter-factual Perturbations. Wang, Wang, *Mao*, Hagenow, Shah. ICLR 2024.

Structured Models With Factorization and Decomposition

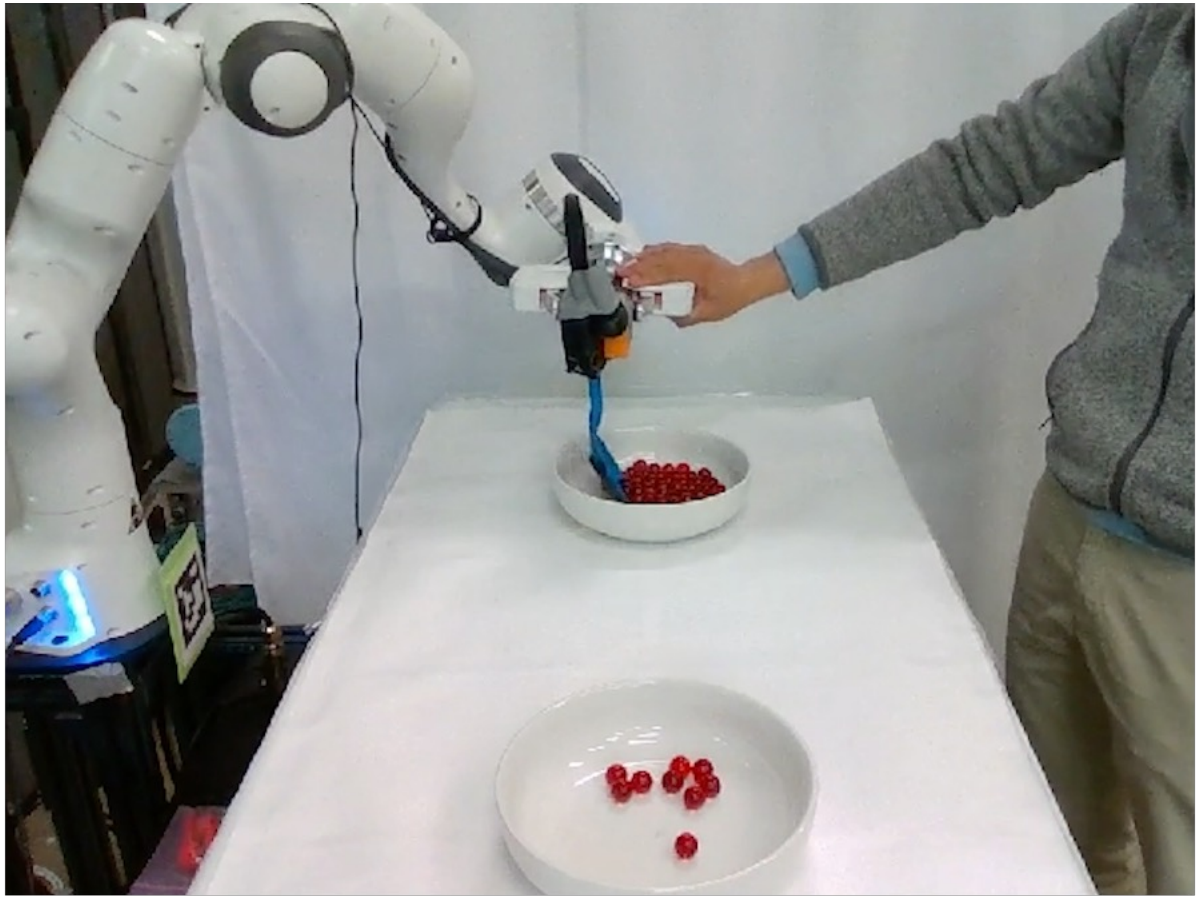
- A skill (e.g., scooping) is a sequence of *intra-mode movements and inter-mode transitions, with parameters*



```
# move to the bowl to scoop from
scoop-move-empty(tool, bowlA)

# scoop the piles
scoop-move-with-contact(tool, bowlA)

# move to the bowl to drop the piles
scoop-move-full(tool, bowlB)

# drop the piles
scoop-move-dump(tool)
```

Structured Models With Factorization and Decomposition

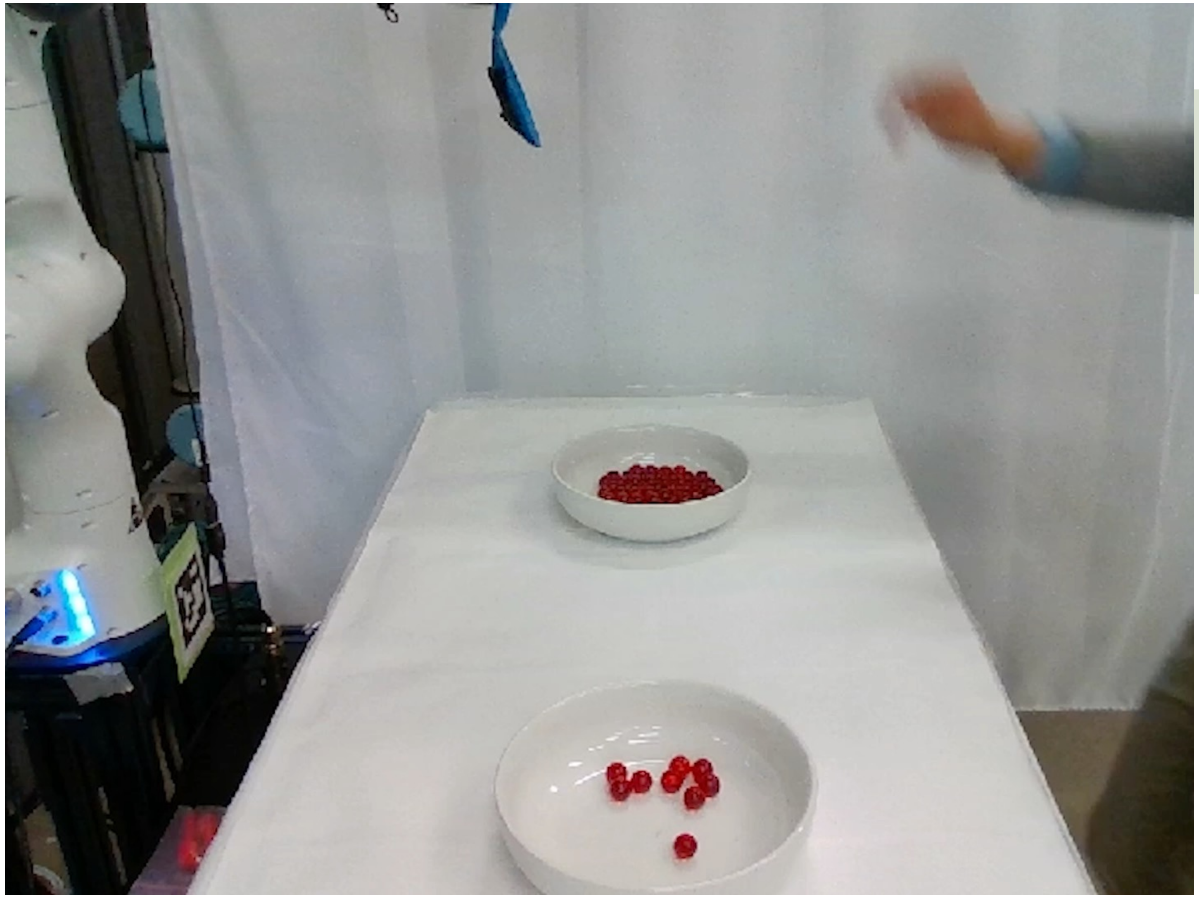
- A skill (e.g., scooping) is a sequence of *intra-mode movements and inter-mode transitions, with parameters*



```
# move to the bowl to scoop from
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
# scoop the piles
scoop-move-with-contact(tool, bowlA)
  achieve hold, empty, close(tool, bowlA)
  eff: marble-upd(tool), marble-upd(bowlA)
# move to the bowl to drop the piles
scoop-move-full(tool, bowlB)
  ...
# drop the piles
scoop-move-dump(tool)
```

Structured Models With Factorization and Decomposition

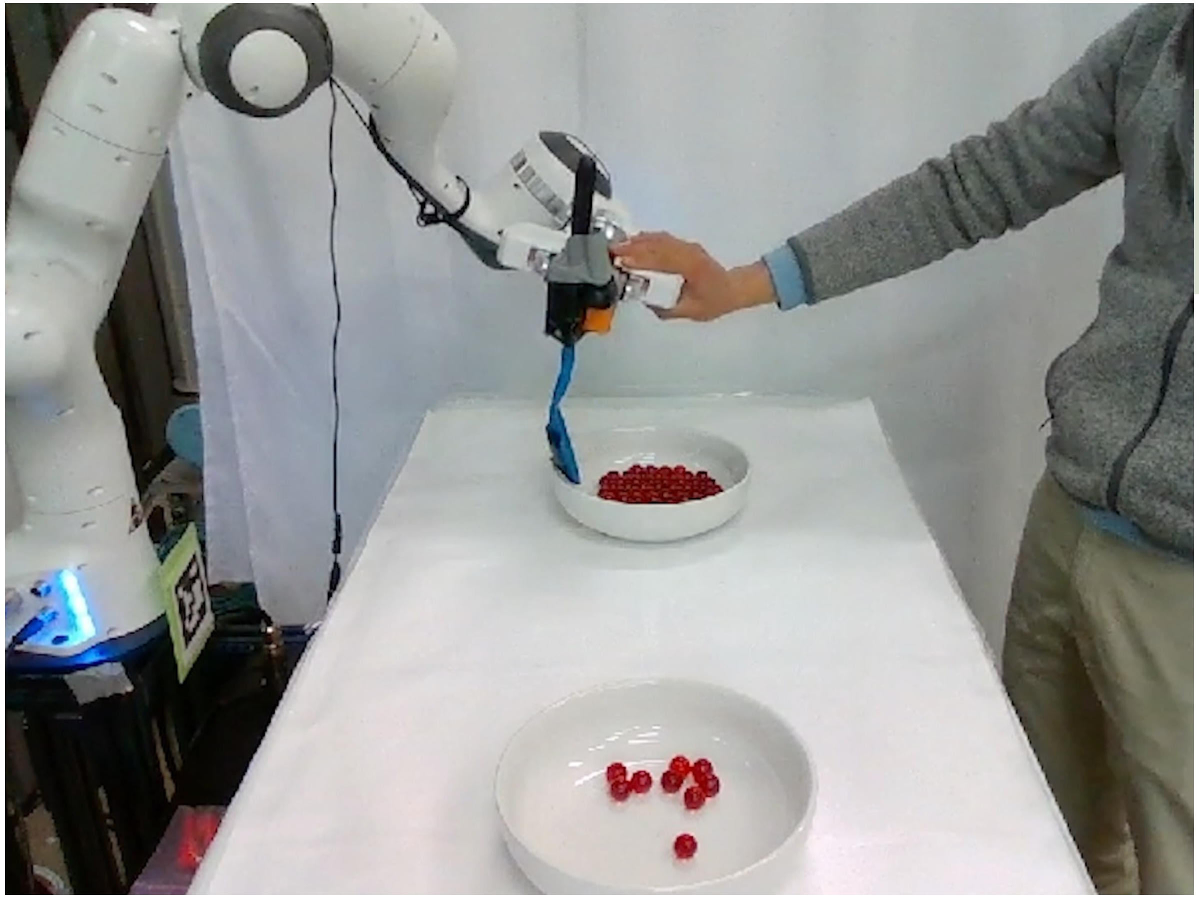
- A skill (e.g., scooping) is a sequence of *intra-mode movements and inter-mode transitions, with parameters*



```
# move to the bowl to scoop from
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
# scoop the piles
scoop-move-with-contact(tool, bowlA)
  achieve hold, empty, close(tool, bowlA)
  eff: marble-upd(tool), marble-upd(bowlA)
# move to the bowl to drop the piles
scoop-move-full(tool, bowlB)
  ...
# drop the piles
scoop-move-dump(tool)
```

Structured Models With Factorization and Decomposition

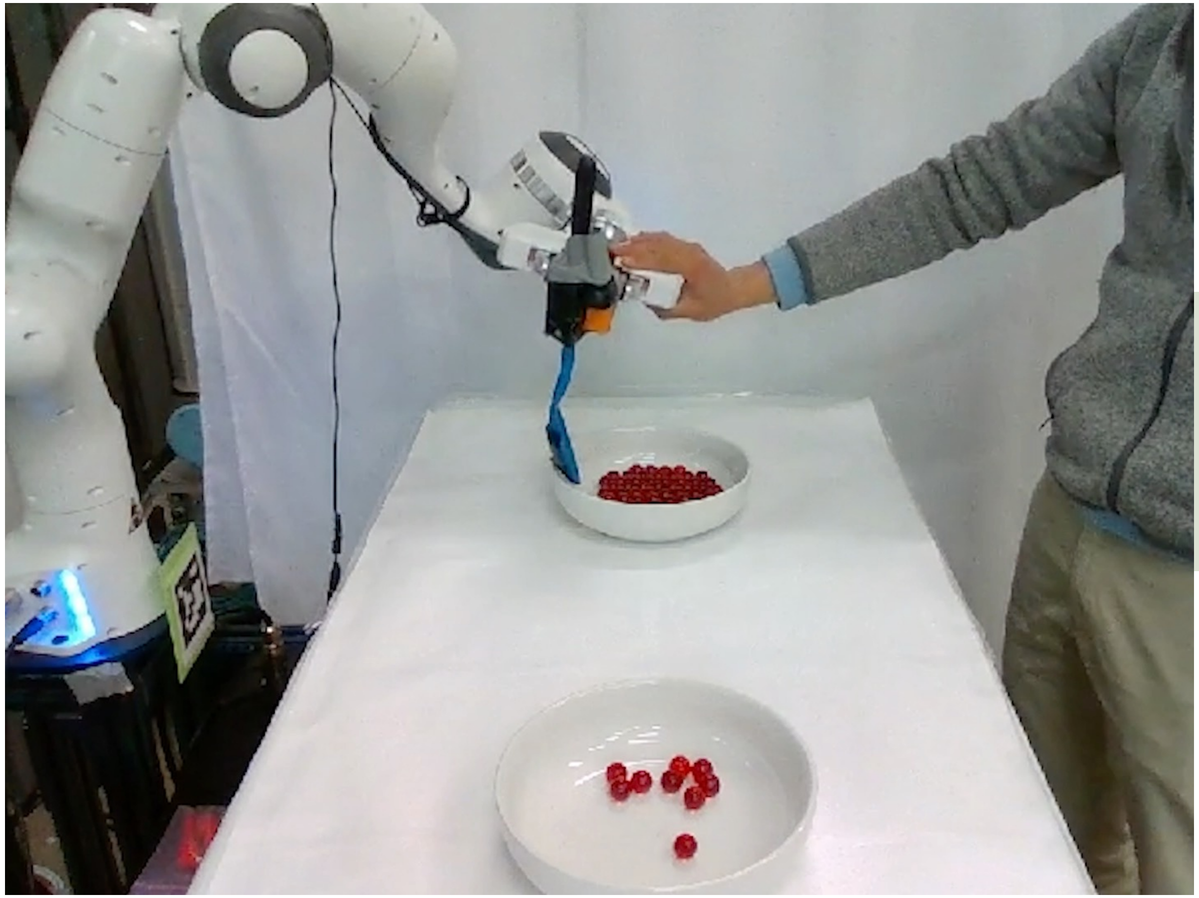
These function names are meaningful for humans, but completely ungrounded for the robot



```
# move to the bowl to scoop from
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
# scoop the piles
scoop-move-with-contact(tool, bowlA)
  achieve hold, empty, close(tool, bowlA)
  eff: marble-upd(tool), marble-upd(bowlA)
# move to the bowl to drop the piles
scoop-move-full(tool, bowlB)
  ...
# drop the piles
scoop-move-dump(tool)
```

Structured Models With Factorization and Decomposition

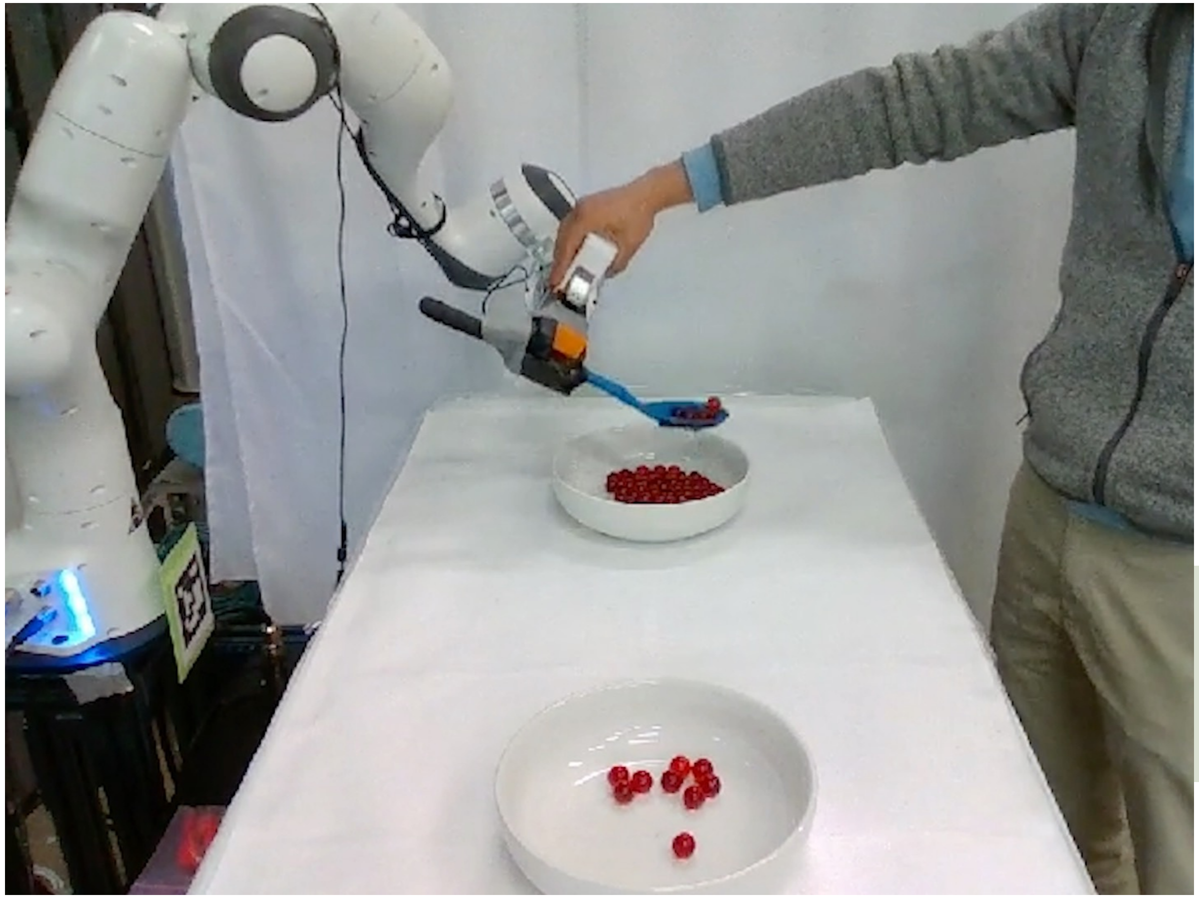
These function names are meaningful for humans, but completely ungrounded for the robot



```
# move to the bowl to scoop from
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
# scoop the piles
scoop-move-with-contact(tool, bowlA)
  achieve hold, empty, close(tool, bowlA)
  eff: marble-upd(tool), marble-upd(bowlA)
# move to the bowl to drop the piles
scoop-move-full(tool, bowlB)
  ...
# drop the piles
scoop-move-dump(tool)
```

Structured Models With Factorization and Decomposition

These function names are meaningful for humans, but completely ungrounded for the robot



```
# move to the bowl to scoop from
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
# scoop the piles
scoop-move-with-contact(tool, bowlA)
  achieve hold, empty, close(tool, bowlA)
  eff: marble-upd(tool), marble-upd(bowlA)
# move to the bowl to drop the piles
scoop-move-full(tool, bowlB)
  ...
# drop the piles
scoop-move-dump(tool)
```

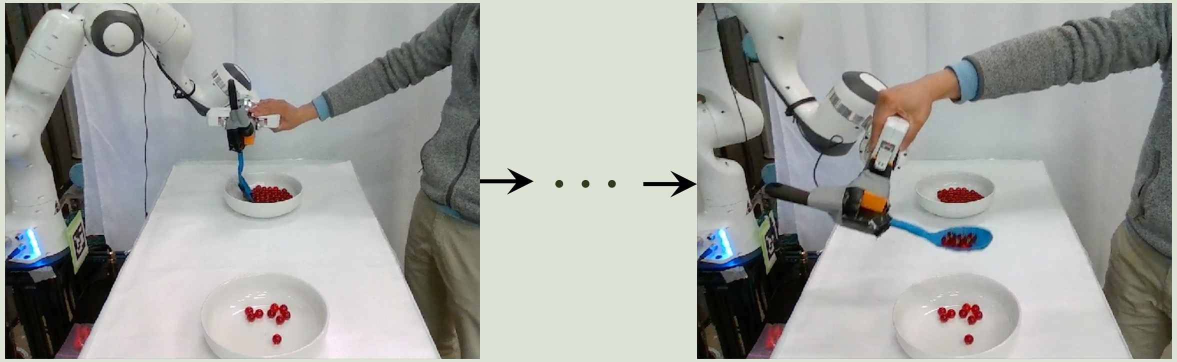
Structured Models With Factorization and Decomposition

These function names are meaningful for humans, but completely ungrounded for the robot

```
# move to the bowl to scoop from
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
# scoop the piles
scoop-move-with-contact(tool, bowlA)
  achieve hold, empty, close(tool, bowlA)
  eff: marble-upd(tool), marble-upd(bowlA)
# move to the bowl to drop the piles
scoop-move-full(tool, bowlB)
  ...
# drop the piles
scoop-move-dump(tool)
```

PDSketch

Integrated Domain Programming, Learning, and Planning



Training Data: Trajectories (e.g., demonstrations)

```
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
......
```

Abstract Sketch (from Humans or LLMs)

Learning Algo. → Structured Model Representations

Assumptions:

Access to object segmentations

Access to trajectory segmentations*

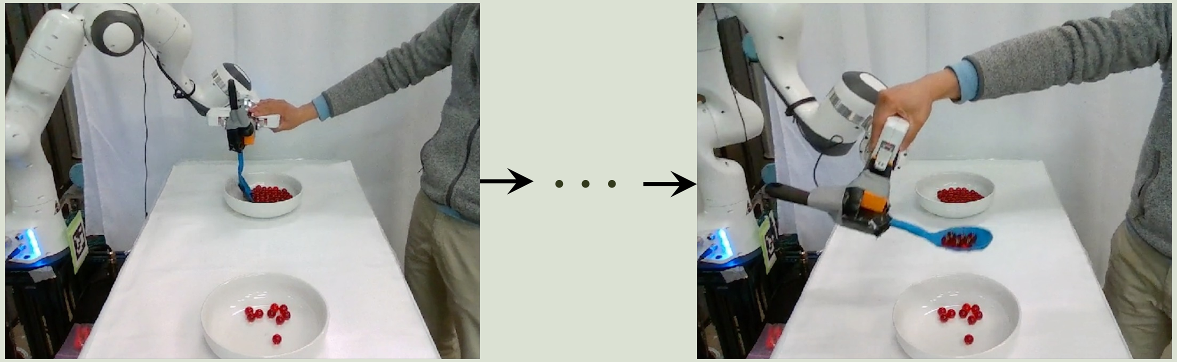
Learning:

Functions for classifiers, transition models, and controllers

PDSketch: Integrated Domain Programming, Learning, and Planning. *Mao*, Lozano-Perez, Tenenbaum, Kaelbling. NeurIPS 2022.
Grounding Language Plans in Demonstrations through Counter-factual Perturbations. Wang, Wang, *Mao*, Hagenow, Shah. ICLR 2024.

PDSketch

Integrated Domain Programming, Learning, and Planning



Training Data: Trajectories (e.g., demonstrations)

```
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
......
```

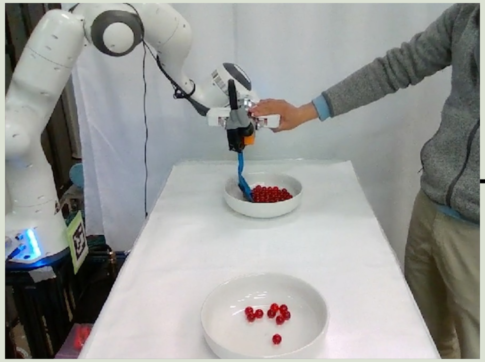
Abstract Sketch (from Humans or LLMs)

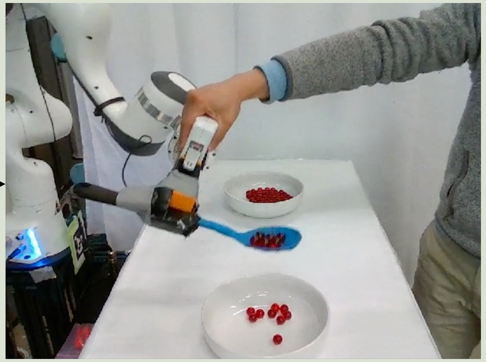
Learning Algo. → Structured Model Representations → **Planning Algo.** ← **New State** **New Goal**

Planning Algo. → **Actions**

PDSketch: Integrated Domain Programming, Learning, and Planning. *Mao*, Lozano-Perez, Tenenbaum, Kaelbling. NeurIPS 2022.
Grounding Language Plans in Demonstrations through Counter-factual Perturbations. Wang, Wang, *Mao*, Hagenow, Shah. ICLR 2024.

PDSketch
Integrated Domain Programming, Learning, and Planning





Training Data: Trajectories (e.g., demonstrations)

```
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
```

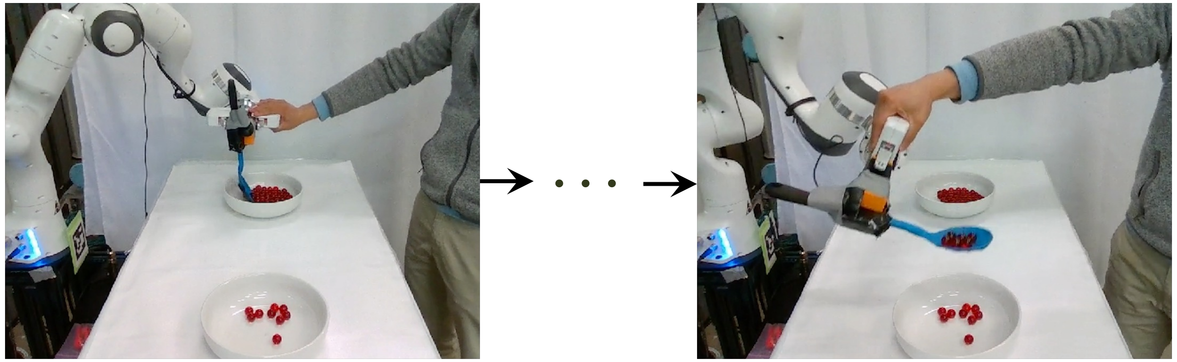
We will first assume they are given
Later talk about how to learn them

Learning Algo. → Structured Model Representations → **Planning Algo.** ← **New State** **New Goal**

Planning Algo. → **Actions**

PDSketch: Integrated Domain Programming, Learning, and Planning. *Mao*, Lozano-Perez, Tenenbaum, Kaelbling. NeurIPS 2022.
Grounding Language Plans in Demonstrations through Counter-factual Perturbations. Wang, Wang, *Mao*, Hagenow, Shah. ICLR 2024.

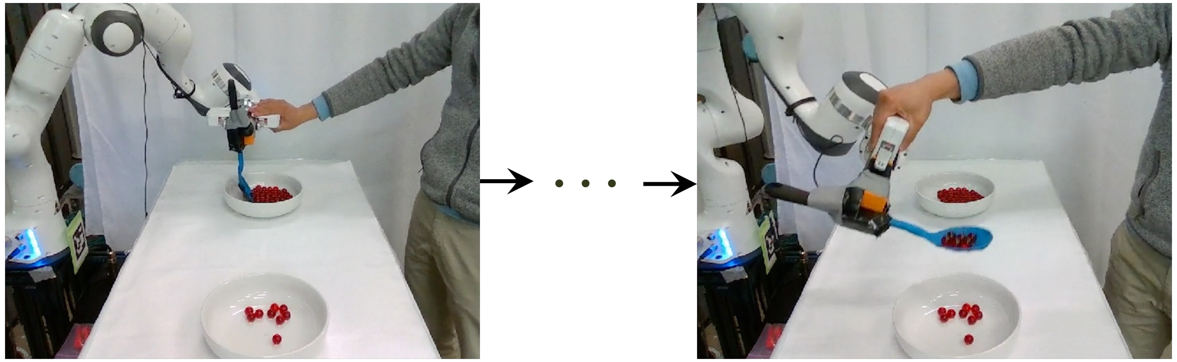
The Objective of Learning



Training Data: Trajectories (e.g., demonstrations)

```
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
scoop-move-with-contact(tool, from)
  achieve hold, empty, close(tool, bowlA)
  eff: marble-upd(tool), marble-upd(bowlA)
scoop-move-full(tool, to)
  ...
scoop-move-dump(tool)
  ...
```

The Objective of Learning



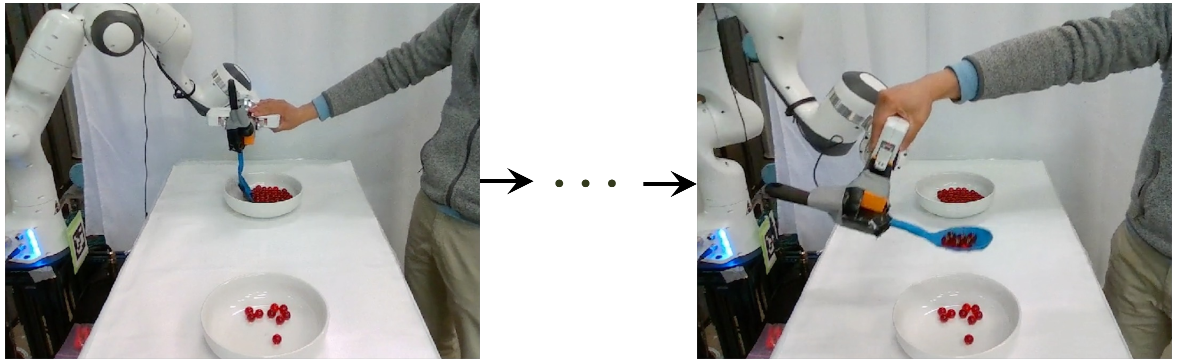
Training Data: Trajectories (e.g., demonstrations)

```
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
scoop-move-with-contact(tool, from)
  achieve hold, empty, close(tool, bowlA)
  eff: marble-upd(tool), marble-upd(bowlA)
scoop-move-full(tool, to)
  ...
scoop-move-dump(tool)
  ...
```

Target 1: Classifiers for predicates

Learning to classify objects and relations

The Objective of Learning



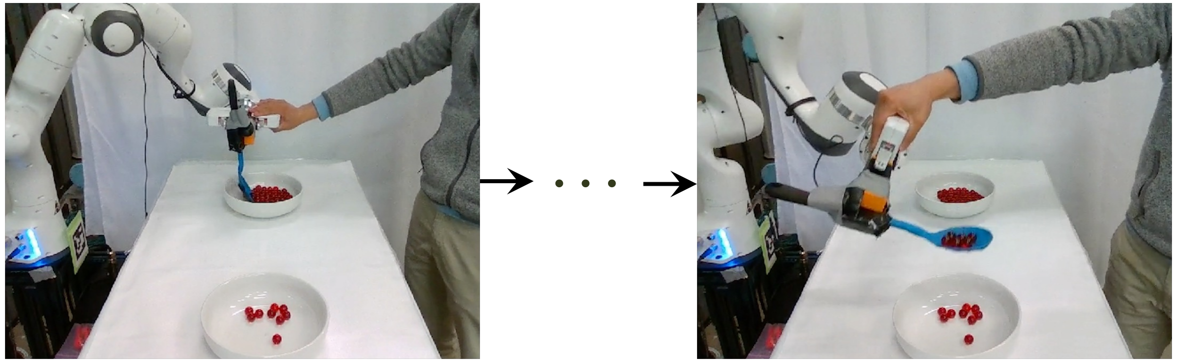
Training Data: Trajectories (e.g., demonstrations)

```
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
scoop-move-with-contact(tool, from)
  achieve hold, empty, close(tool, bowlA)
  eff: marble-upd(tool), marble-upd(bowlA)
scoop-move-full(tool, to)
  ...
scoop-move-dump(tool)
  ...
```

Target 1: Classifiers for predicates.
Learning to classify objects and relations

Target 2: Controllers for sub-actions

The Objective of Learning



Training Data: Trajectories (e.g., demonstrations)

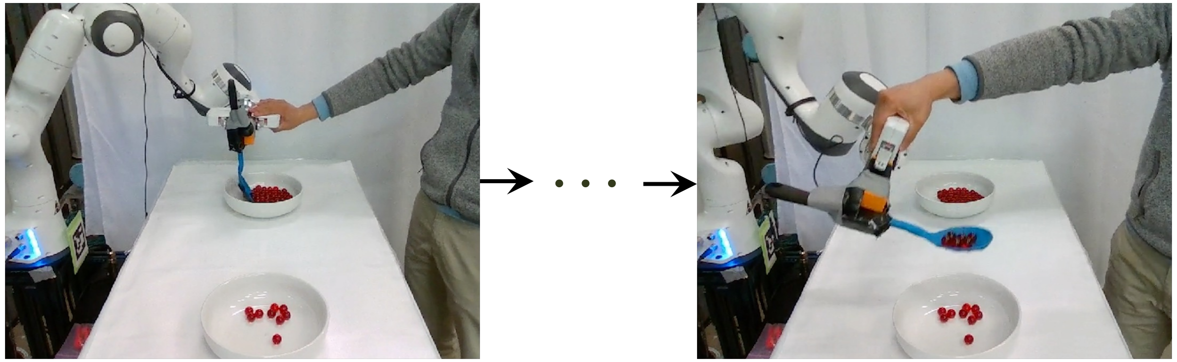
```
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
scoop-move-with-contact(tool, from)
  achieve hold, empty, close(tool, bowlA)
  eff: marble-upd(tool), marble-upd(bowlA)
scoop-move-full(tool, to)
  ...
scoop-move-dump(tool)
  ...
```

Target 1: Classifiers for predicates
Learning to classify objects and relations

Target 2: Controllers for sub-actions

Target 3: Transition models

The Objective of Learning



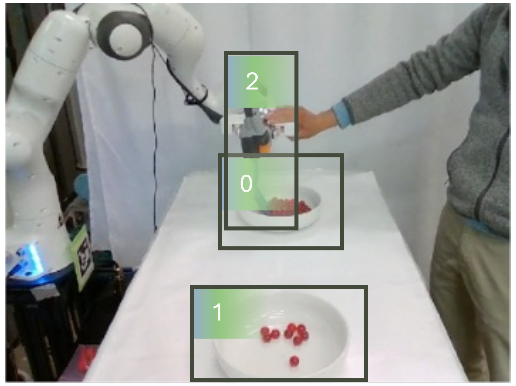
Training Data: Trajectories (e.g., demonstrations)

```
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
scoop-move-with-contact(tool, from)
  achieve hold, empty, close(tool, bowlA)
  eff: marble-upd(tool), marble-upd(bowlA)
scoop-move-full(tool, to)
  ...
scoop-move-dump(tool)
  ...
```

Target 1: Classifiers for predicates

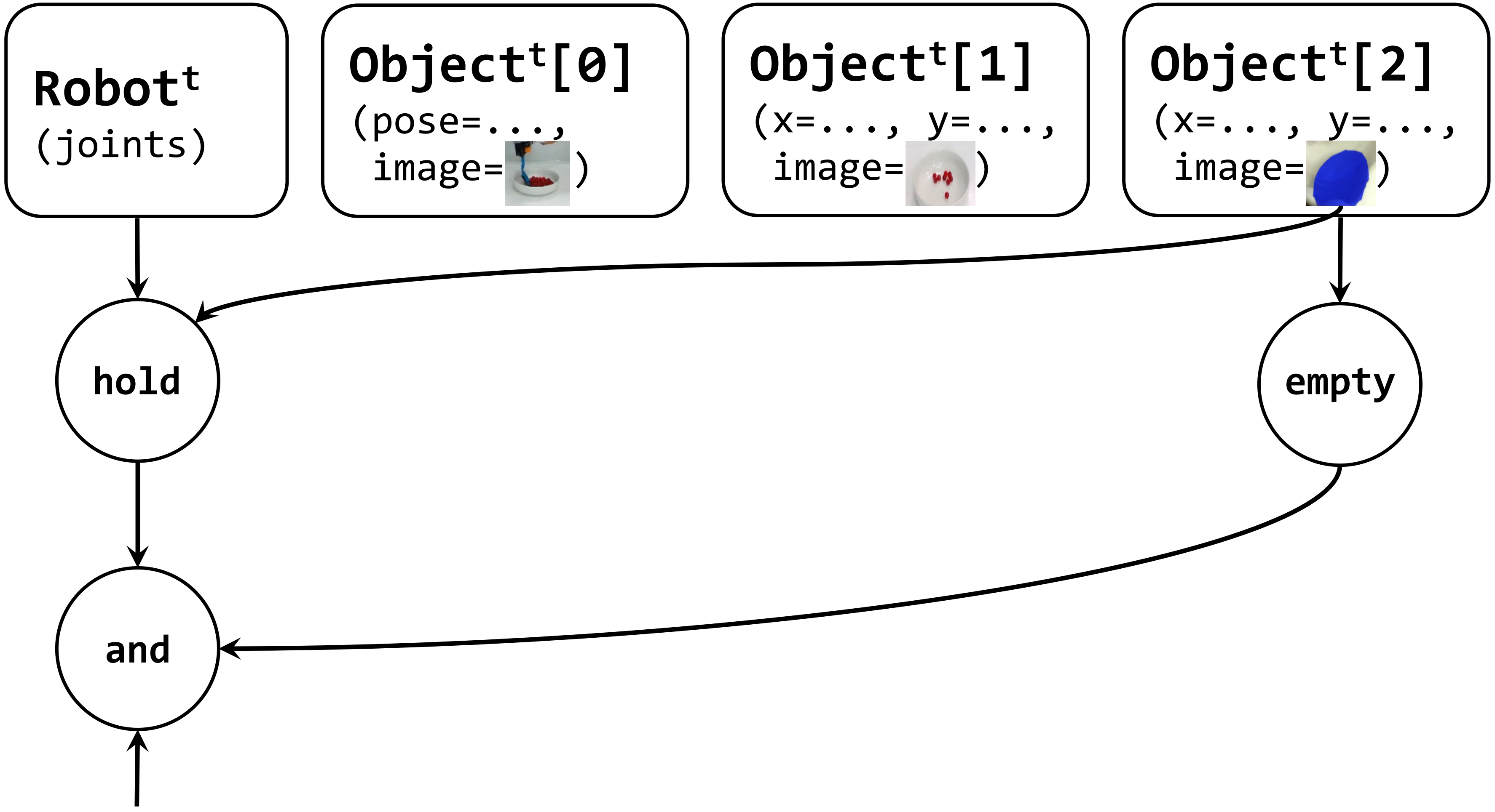
Learning to classify objects and relations

Learning Through the Computation Graph of Preconditions



(Before)

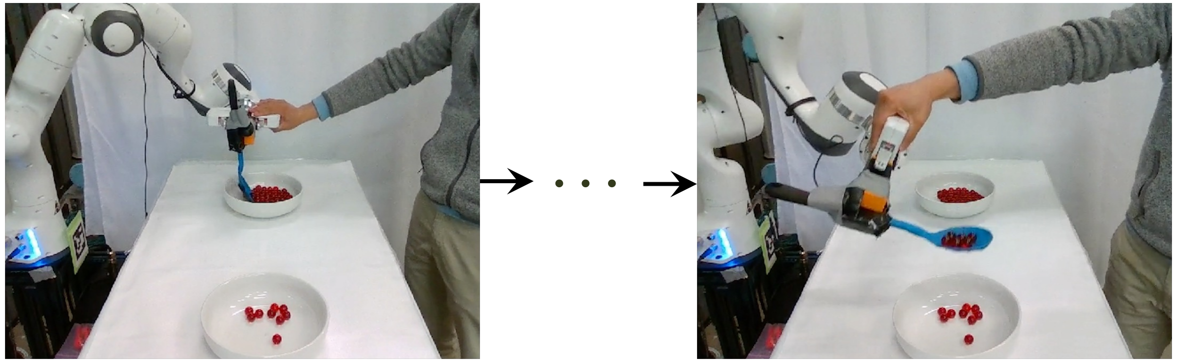
```
precondition:
 holding(tool),
 empty(tool)
```



Label: 1 In demonstration, precondition has been achieved before executing the action.

Back Prop

The Objective of Learning

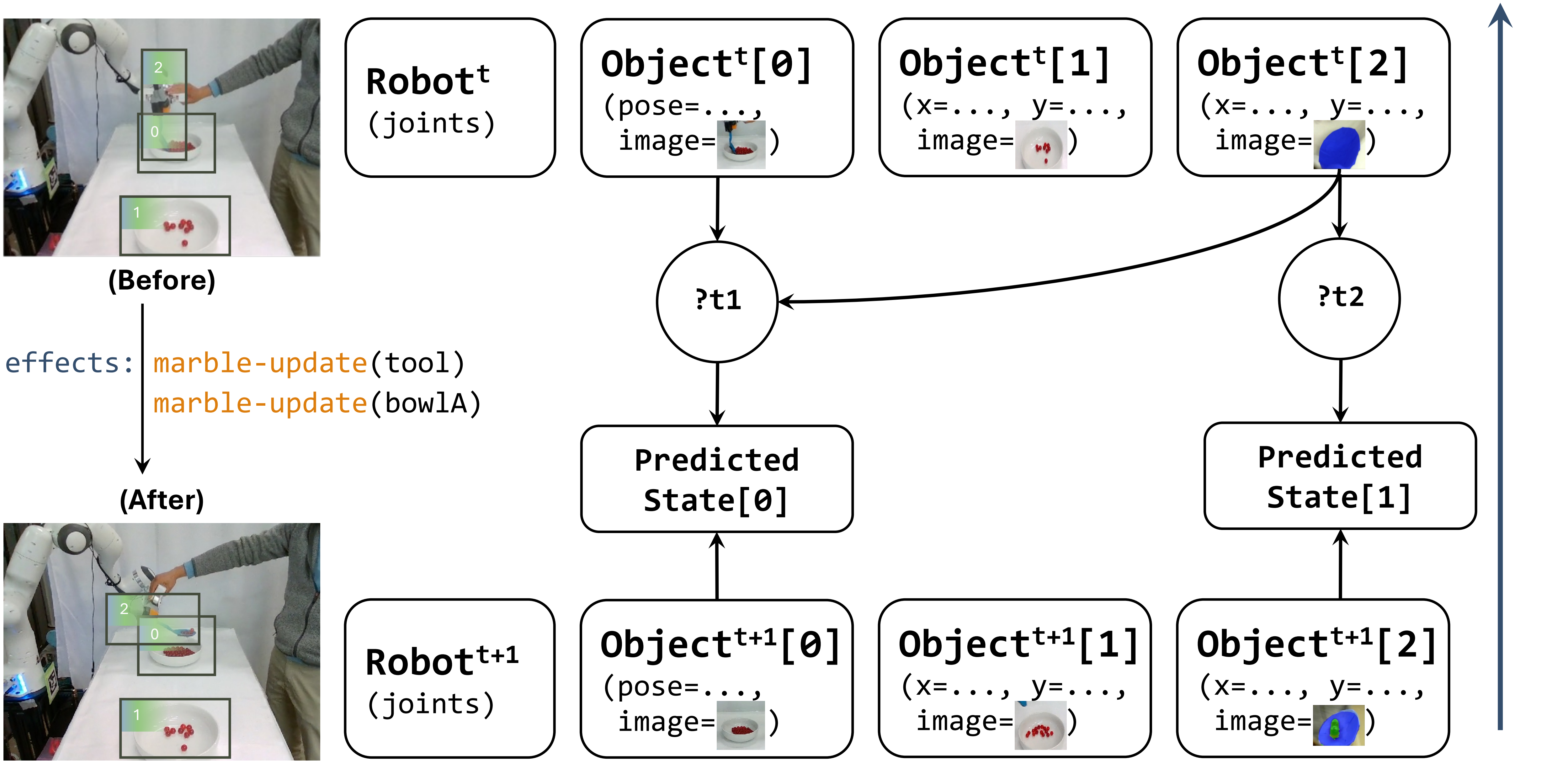


Training Data: Trajectories (e.g., demonstrations)

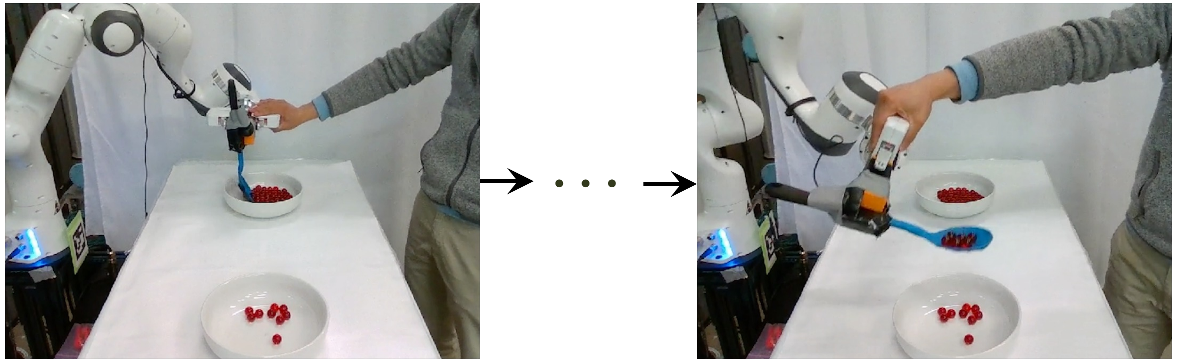
```
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
scoop-move-with-contact(tool, from)
  achieve hold, empty, close(tool, bowlA)
  eff: marble-upd(tool), marble-upd(bowlA)
scoop-move-full(tool, to)
  ...
scoop-move-dump(tool)
  ...
```

Target 3: Transition models

Learning Transitions from Self-Supervision



The Objective of Learning

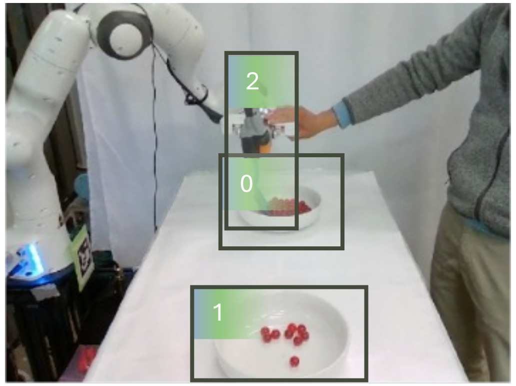


Training Data: Trajectories (e.g., demonstrations)

```
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
scoop-move-with-contact(tool, from)
  achieve hold, empty, close(tool, bowlA)
  eff: marble-upd(tool), marble-upd(bowlA)
scoop-move-full(tool, to)
  ...
scoop-move-dump(tool)
  ...
```

Target 2: Controllers for sub-actions

Learning Continuous Parameters or Controllers



Robot^t
(joints)

Object^t[0]
(pose=...,
image=)

Object^t[1]
(x=..., y=...,
image=)

Object^t[2]
(x=..., y=...,
image=)

`scoop-move-with-contact(tool, from)`

π1

A simple implementation can be done with segmented trajectories, but we can also jointly learn to segment them

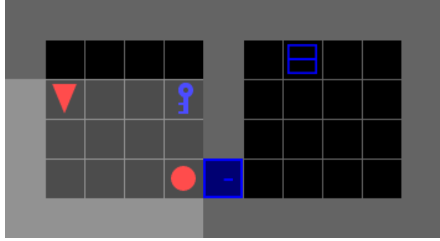
Option 1: Directly output a joint command

+: Most general. Does not rely on any prior knowledge
-: Poor generalization for unseen configurations and obstacles.

Option 2: Output a target relative pose, and then call a motion planner

-: Need additional knowledge
+: Better generalization for unseen configurations and obstacles

Learning and Planning Efficiency



PDS-Rob

Full robot movement models
Learn to interpret goals

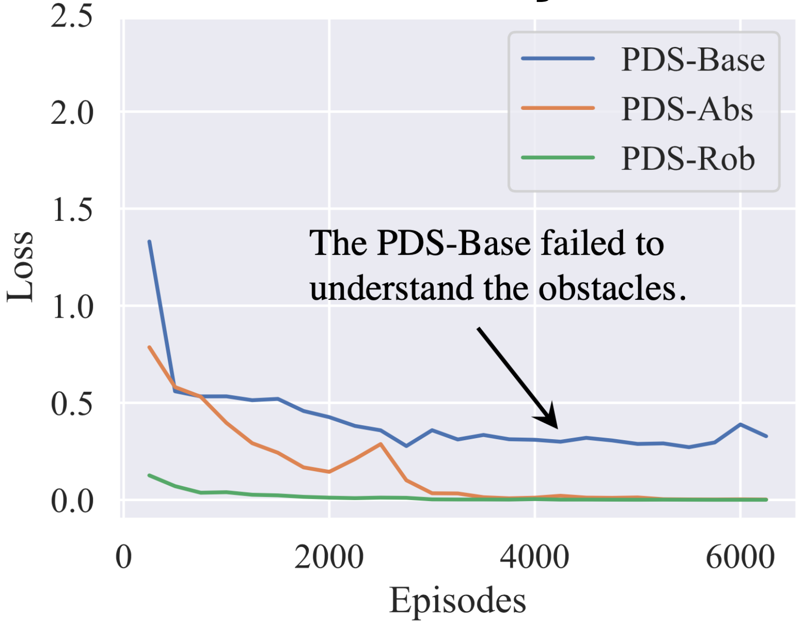
PDS-Abs

Abstract robot models
(With uninterpreted symbols)

PDS-Base

GNNs
(Weakest prior)

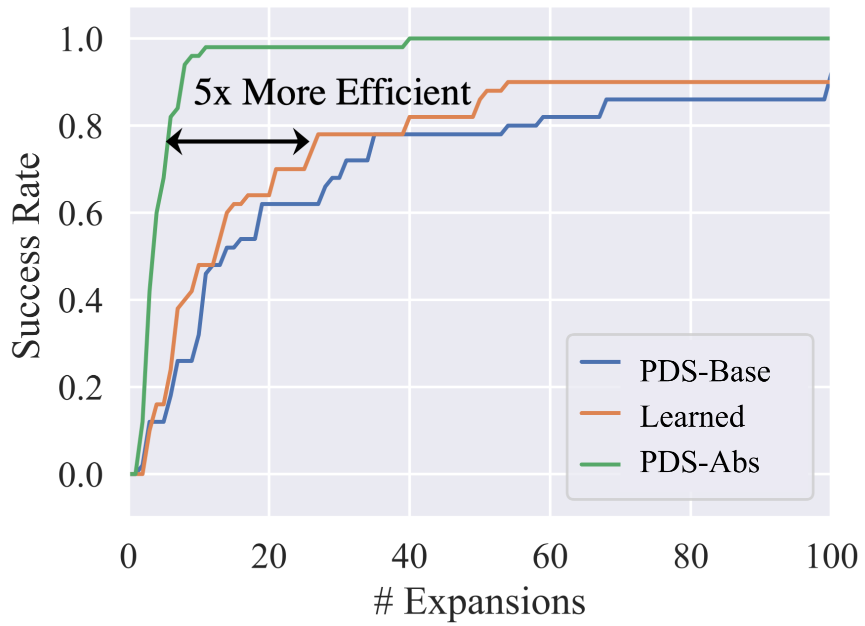
Data Efficiency



Success Rate

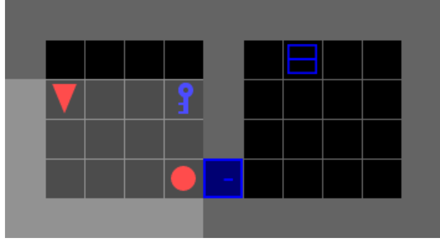
Behavior Cloning	0.79
Decision Xformer	0.82
DreamerV2	0.79
PDS-Base	0.62
PDS-Abs	0.98
PDS-Rob	1.00

Planning Efficiency



Environment from: Chevalier-Boisvert et al. 2019.

Learning and Planning Efficiency



PDS-Rob

Full robot movement models
Learn to interpret goals

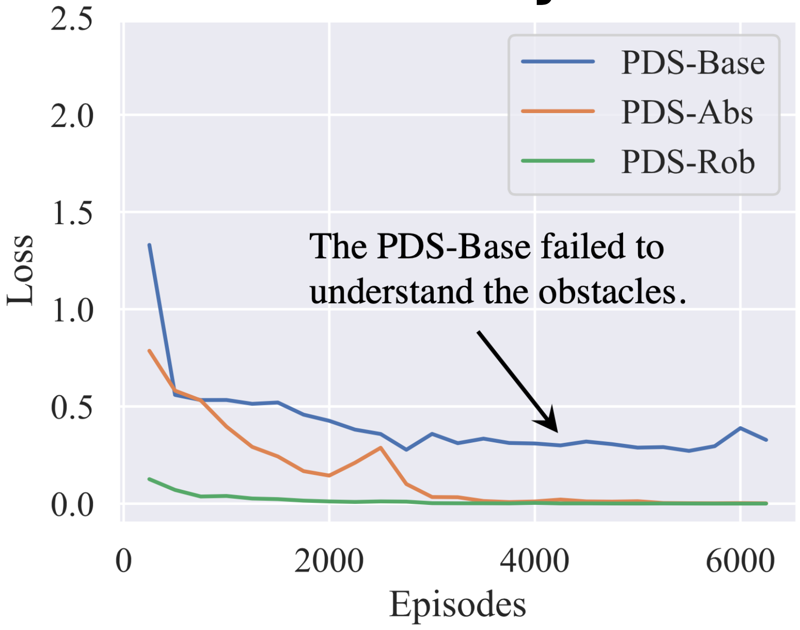
PDS-Abs

Abstract robot models
(With uninterpreted symbols)

PDS-Base

GNNs
(Weakest prior)

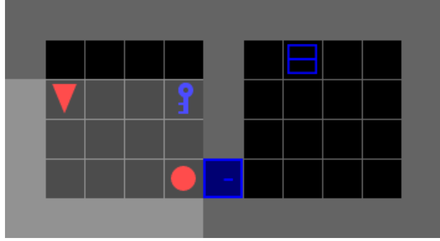
Data Efficiency



Very small amount of prior knowledge significantly improves the *data efficiency*

Environment from: Chevalier-Boisvert et al. 2019.

Learning and Planning Efficiency



PDS-Rob

Full robot movement models
Learn to interpret goals

PDS-Abs

Abstract robot models
(With uninterpreted symbols)

PDS-Base

GNNs
(Weakest prior)

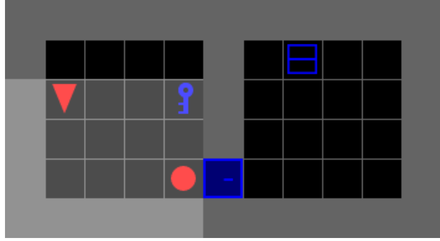
Success Rate

Behavior Cloning	0.79
Decision Xformer	0.82
DreamerV2	0.79
PDS-Base	0.62
PDS-Abs	0.98
PDS-Rob	1.00

The performance in model learning also translates to *better performance*

Environment from: Chevalier-Boisvert et al. 2019.

Learning and Planning Efficiency



PDS-Rob

Full robot movement models
Learn to interpret goals

PDS-Abs

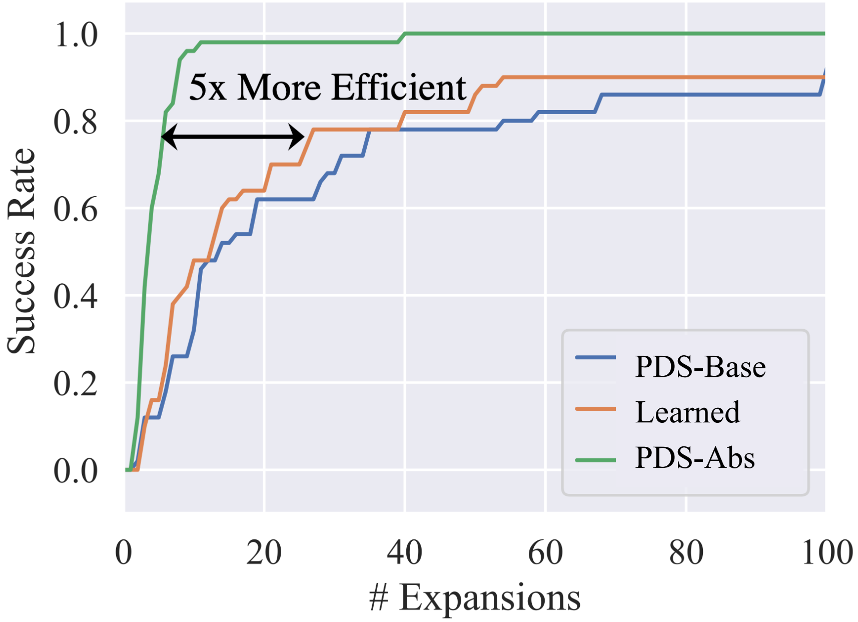
Abstract robot models
(With uninterpreted symbols)

PDS-Base

GNNs
(Weakest prior)

The factored representation yields domain-independent heuristics which improves *planning efficiency*

Planning Efficiency



Environment from: Chevalier-Boisvert et al. 2019.

Planning Efficiency via Domain-Independent Heuristics

- Suppose an action has two preconditions
- Solve two planning problems separately, and “add” the costs together

`goal: filled(bowlB) and on(bowlB, tableB)`

need to change
marble-state(bowlB)

`drop(spoon, bowlB)`

need to change
marble-state(tool)

`scoop(spoon, bowlA)`

...

need to change
pose(tool)

`move(spoon)`

...

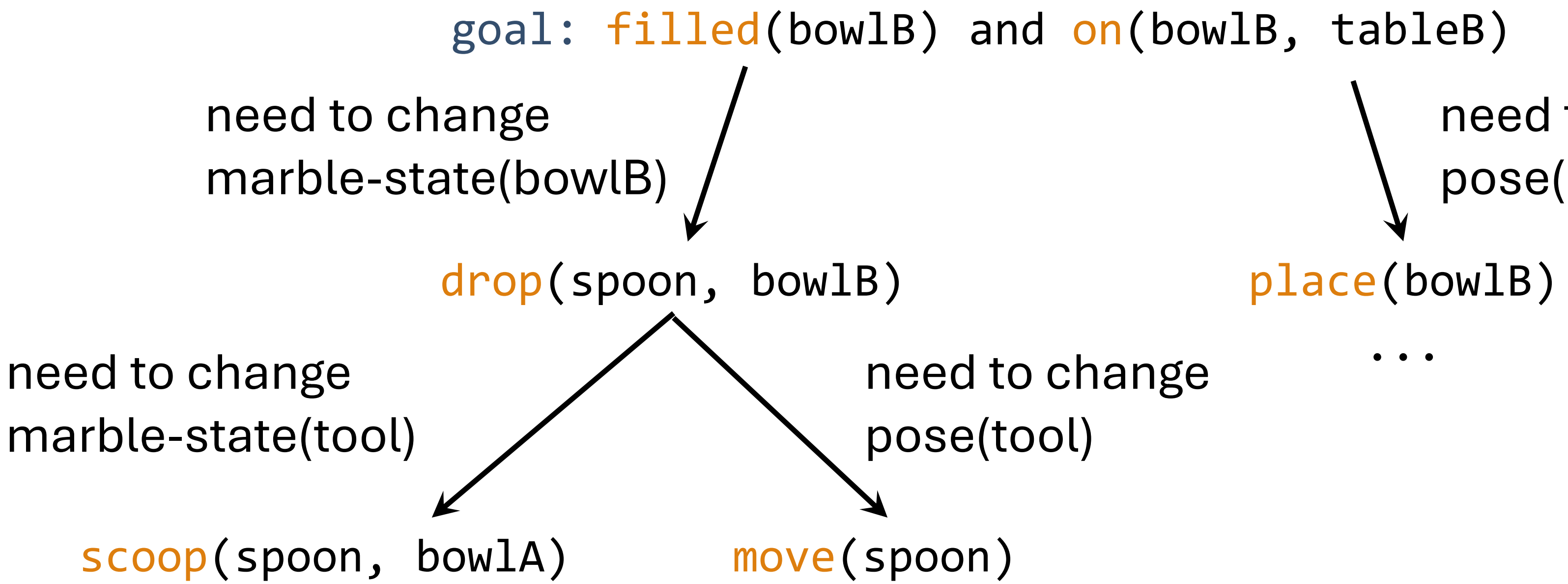
need to change
pose(bowlB)

`place(bowlB)`

...

Planning Efficiency via Domain-Independent Heuristics

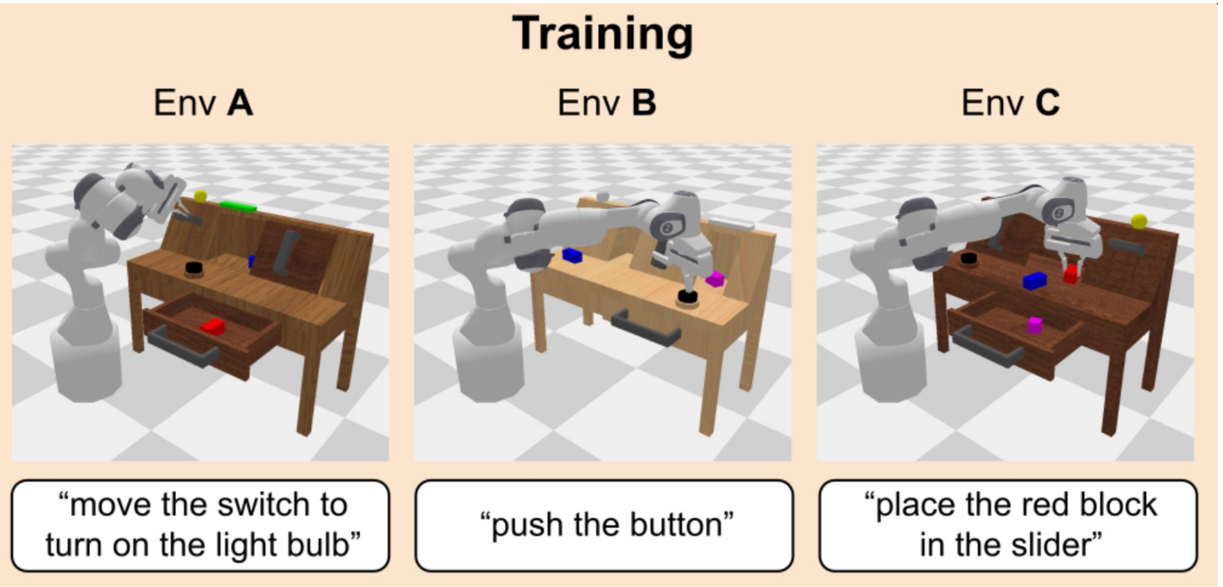
- Suppose an action has two preconditions
- Solve two planning problems separately, and "add" the costs together



- This gives a good estimate of the cost-to-go and it's efficient to compute
- PDSketch generalizes this to the (neural) computation graphs of preconditions and transitions

FF: The Fast-Forward Planning System. Hoffmann. AAAI 2001.
PDSketch: Integrated Domain Programming, Learning, and Planning. *Mao*, Lozano-Perez, Tenenbaum, Kaelbling. NeurIPS 2022.

Generalization to Unseen Goals



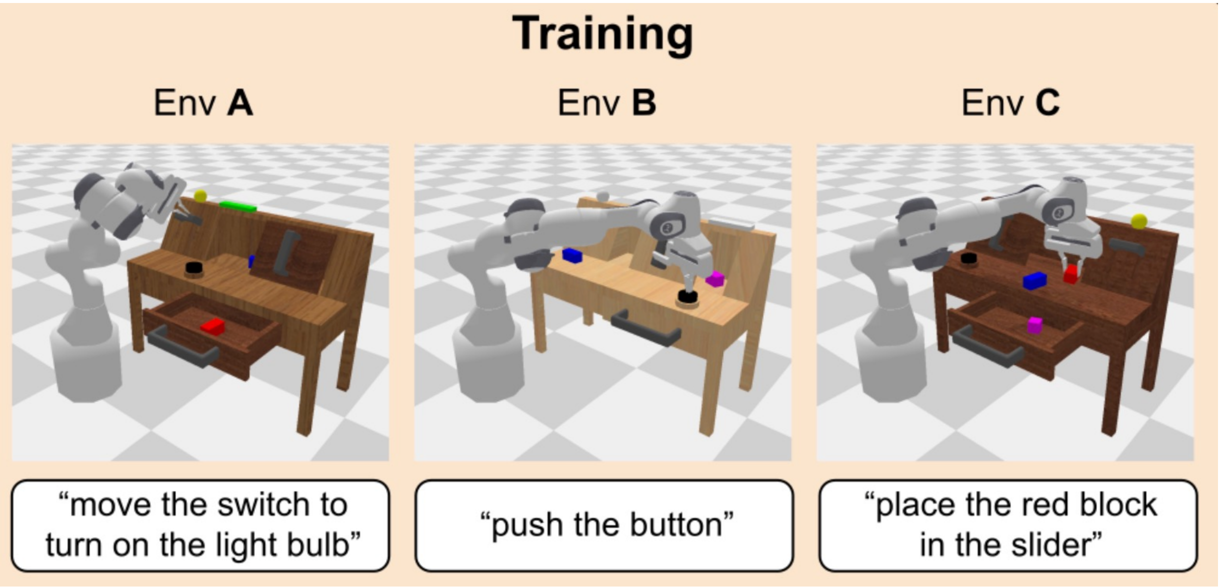
Data: language-annotated demonstrations

What it learns:

- Classifiers for relations (e.g., light-on)
- (Diffusion) policies for a set of primitive actions, based on a motion planner

Liu*, *Mao**, Nie*, et al. In Preparation. Environment from Mees et al. CALVIN. RA-L. 2022.

Generalization to Unseen Goals



Novel goal: all blocks in the drawer

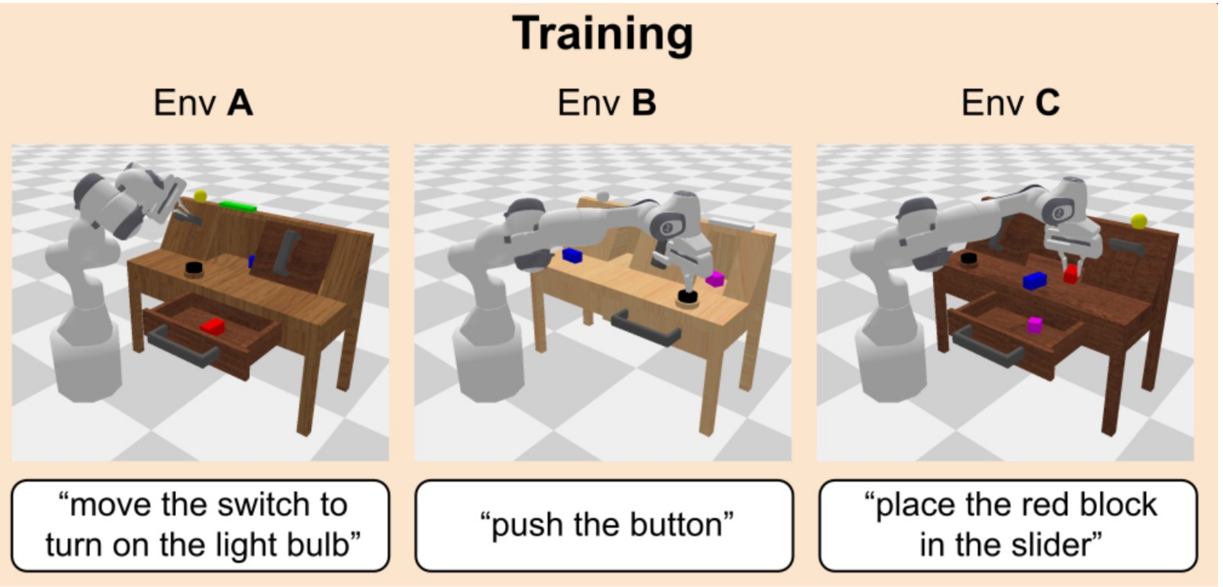
Data: language-annotated demonstrations

What it learns:

- Classifiers for relations (e.g., light-on)
- (Diffusion) policies for a set of primitive actions, based on a motion planner

Liu*, *Mao**, Nie*, et al. In Preparation. Environment from Mees et al. CALVIN. RA-L. 2022.

Generalization to Unseen Goals



Data: language-annotated demonstrations

What it learns:

- Classifiers for relations (e.g., light-on)
- (Diffusion) policies for a set of primitive actions, based on a motion planner

Novel goal: all lights turned off

Liu*, *Mao**, Nie*, et al. In Preparation. Environment from Mees et al. CALVIN. RA-L. 2022.

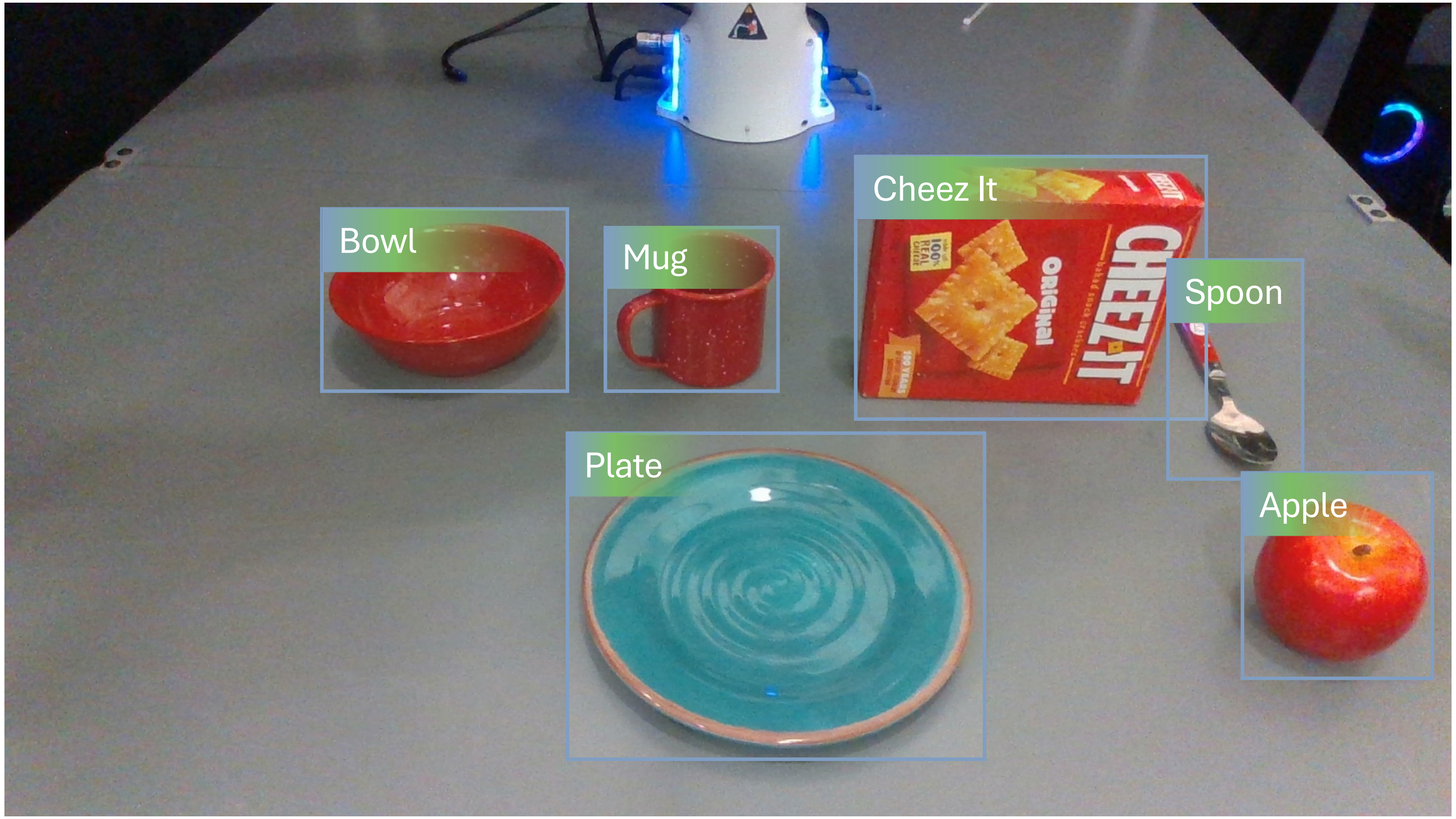
Generalization to Underspecified Goals

Instruction: Set up a table for my breakfast, please. I have set the plate for you

Compositional Diffusion-Based Continuous Constraint Solvers. Yang, *Mao*, Du, Wu, Tenenbaum, Lozano-Perez, Kaelbling. CoRL 2023.
Functional Object Arrangement with Compositional Generative Models. Xu, *Mao*, Du, Hsu, Kaelbling. In submission.

Generalization to Underspecified Goals

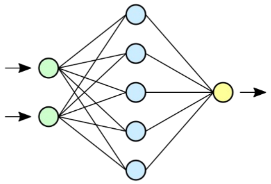
Instruction: Set up a table for my breakfast, please. I have set the plate for you



LLM

Relational Graph

```
on(bowl, plate),
left_of(apple, plate),
right_of(spoon, plate),
aligned_horizontally(
  apple, plate, spoon, mug
),
...
```

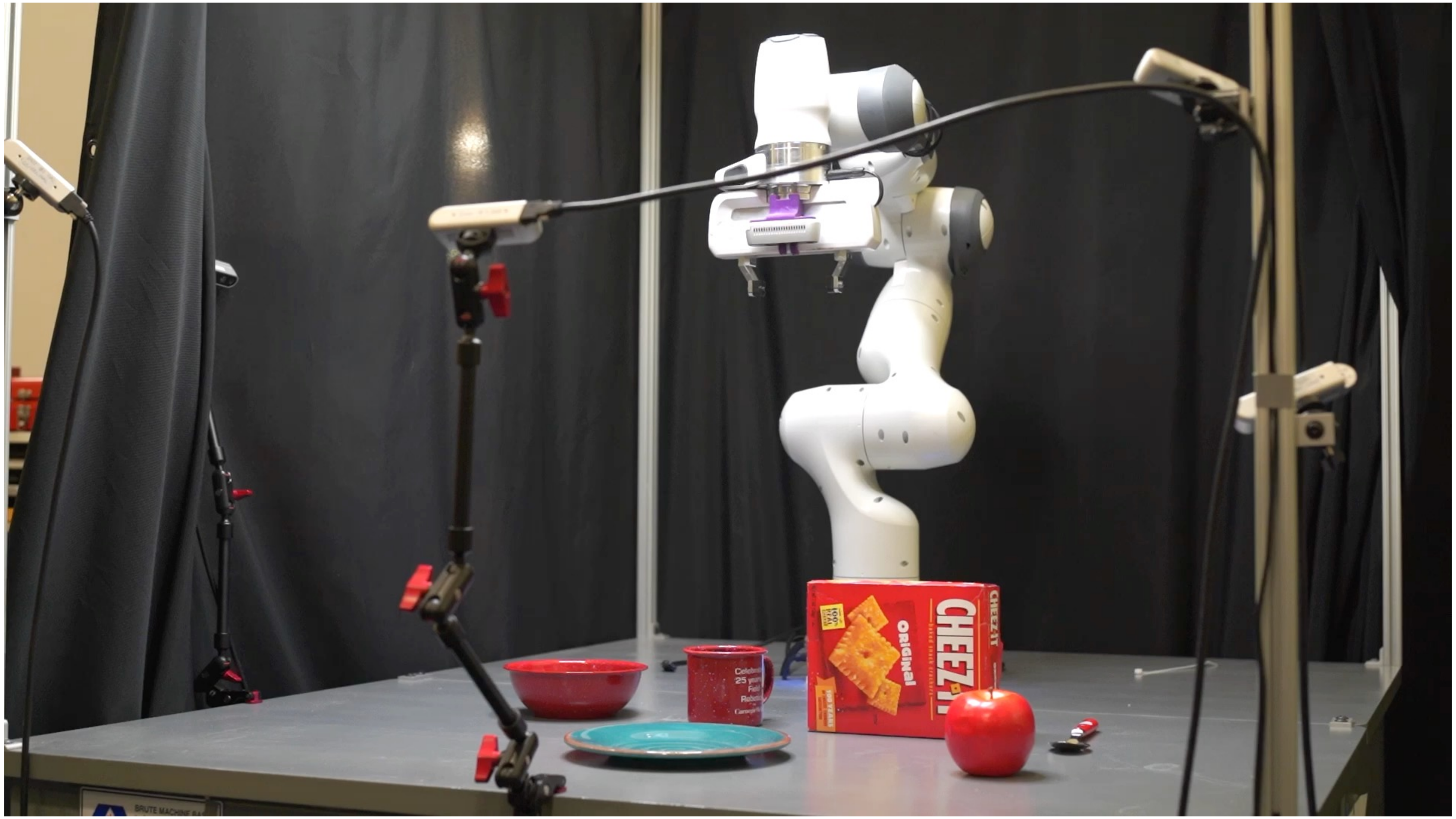


Compositional Diffusion-Based Constraint Solver

Compositional Diffusion-Based Continuous Constraint Solvers. Yang, *Mao*, Du, Wu, Tenenbaum, Lozano-Perez, Kaelbling. CoRL 2023.
Functional Object Arrangement with Compositional Generative Models. Xu, *Mao*, Du, Hsu, Kaelbling. In submission.

Generalization to Underspecified Goals

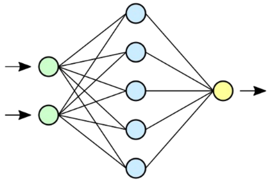
Instruction: Set up a table for my breakfast, please. I have set the plate for you



LLM

Relational Graph

```
on(bowl, plate),
left_of(apple, plate),
right_of(spoon, plate),
aligned_horizontally(
  apple, plate, spoon, mug
),
...
```



Compositional Diffusion-Based Constraint Solver

Compositional Diffusion-Based Continuous Constraint Solvers. Yang, *Mao*, Du, Wu, Tenenbaum, Lozano-Perez, Kaelbling. CoRL 2023.
Functional Object Arrangement with Compositional Generative Models. Xu, *Mao*, Du, Hsu, Kaelbling. In submission.

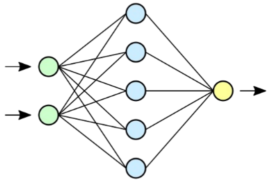
Generalization to Underspecified Goals

Instruction: Set up a table for my breakfast, please. I have set the plate for you

LLM

Relational Graph

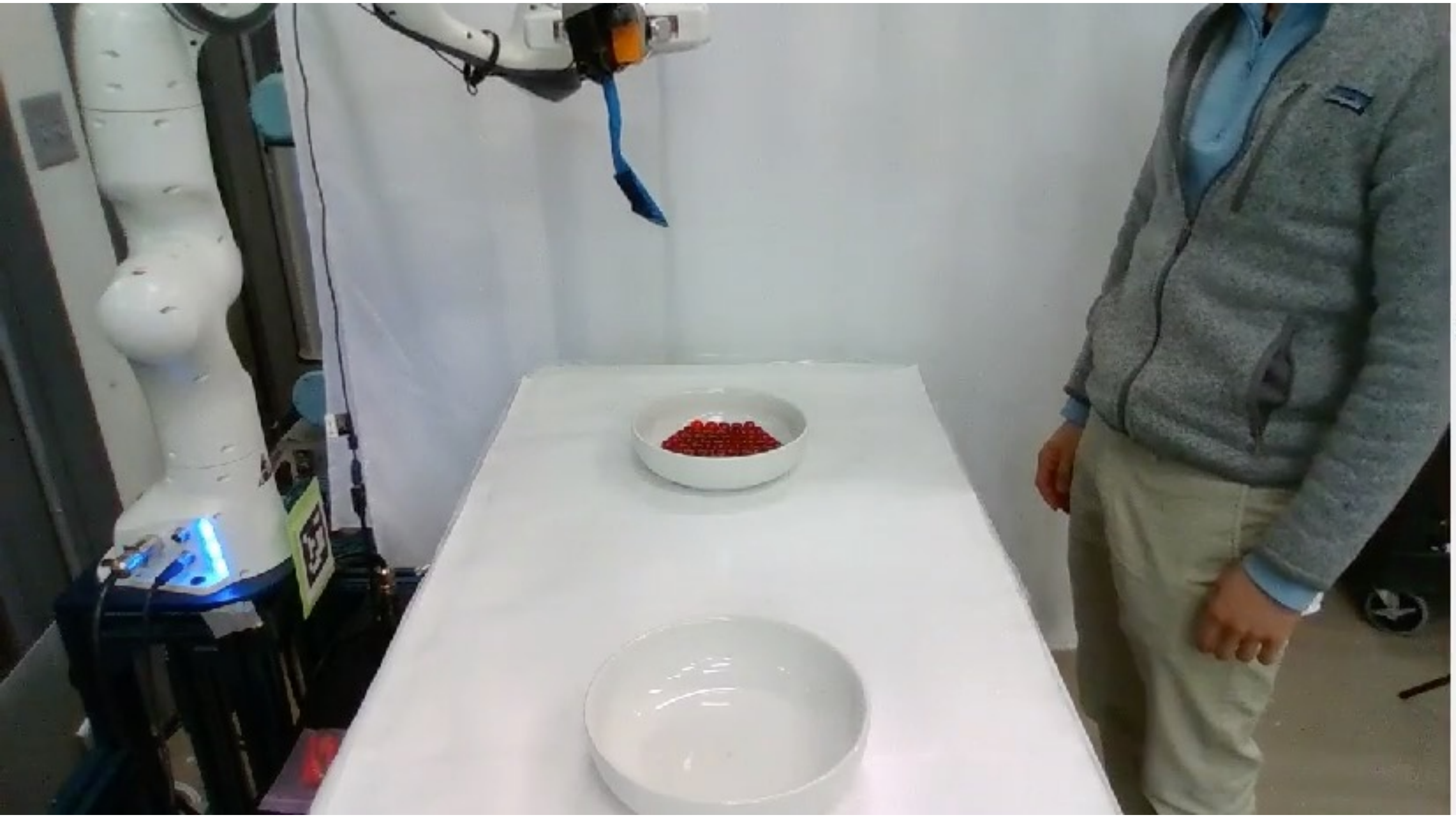
```
on(bowl, plate),
left_of(apple, plate),
right_of(spoon, plate),
aligned_horizontally(
  apple, plate, spoon, mug
),
...
```

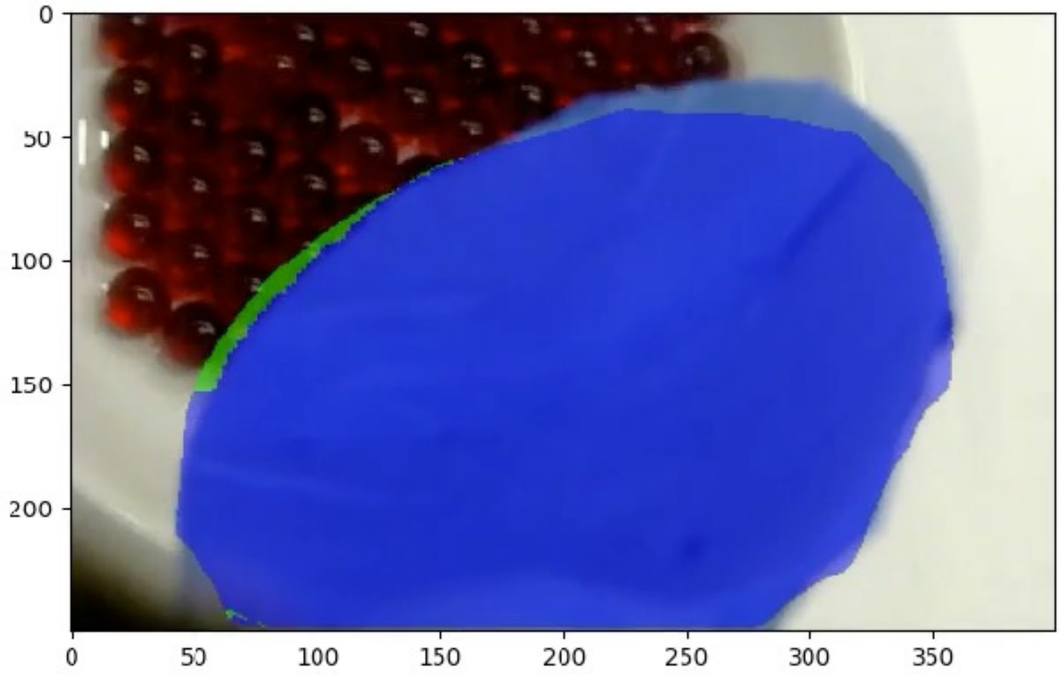


Compositional Diffusion-Based Constraint Solver

Compositional Diffusion-Based Continuous Constraint Solvers. Yang, *Mao*, Du, Wu, Tenenbaum, Lozano-Perez, Kaelbling. CoRL 2023.
Functional Object Arrangement with Compositional Generative Models. Xu, *Mao*, Du, Hsu, Kaelbling. In submission.

Robust under Local and Global Perturbation



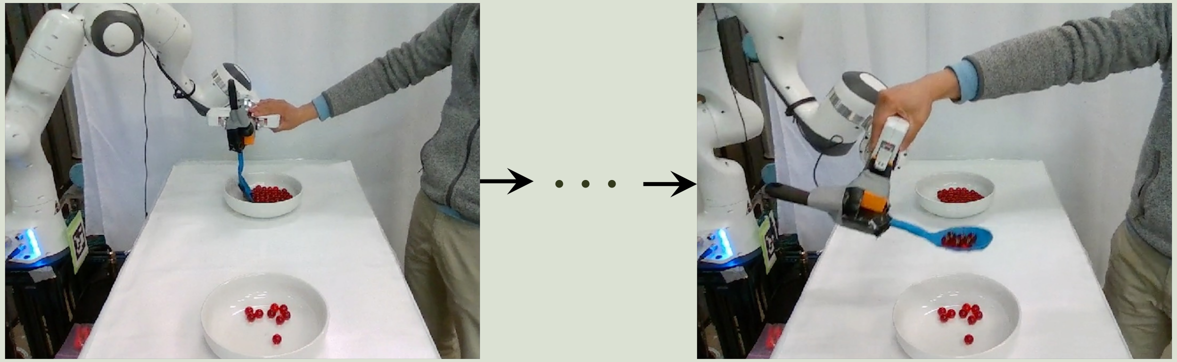


- Explicitly learned mode classifiers and transition rules enable online re-planning
- Using motion planners enables generalization in “getting back to pre-scoop poses”

* Trained with 17 human-collected demonstrations, and ~200 counterfactual replays.
Grounding Language Plans in Demonstrations through Counter-factual Perturbations. Wang, Wang, *Mao*, Hagenow, Shah. ICLR 2024.

PDSketch

Integrated Domain Programming, Learning, and Planning



Training Data: Trajectories (e.g., demonstrations)

```
scoop-move-empty(tool, bowlA)
  achieve hold(tool), empty(tool)
  eff: close(tool, bowlA)
```

Now let's talk about how we can get this *automatically* from language

Learning Algo.

Structured Model Representations

Planning Algo.

New State
New Goal

Actions

PDSketch: Integrated Domain Programming, Learning, and Planning. *Mao*, Lozano-Perez, Tenenbaum, Kaelbling. NeurIPS 2022.
Grounding Language Plans in Demonstrations through Counter-factual Perturbations. Wang, Wang, *Mao*, Hagenow, Shah. ICLR 2024.

Learning Abstractions from Language

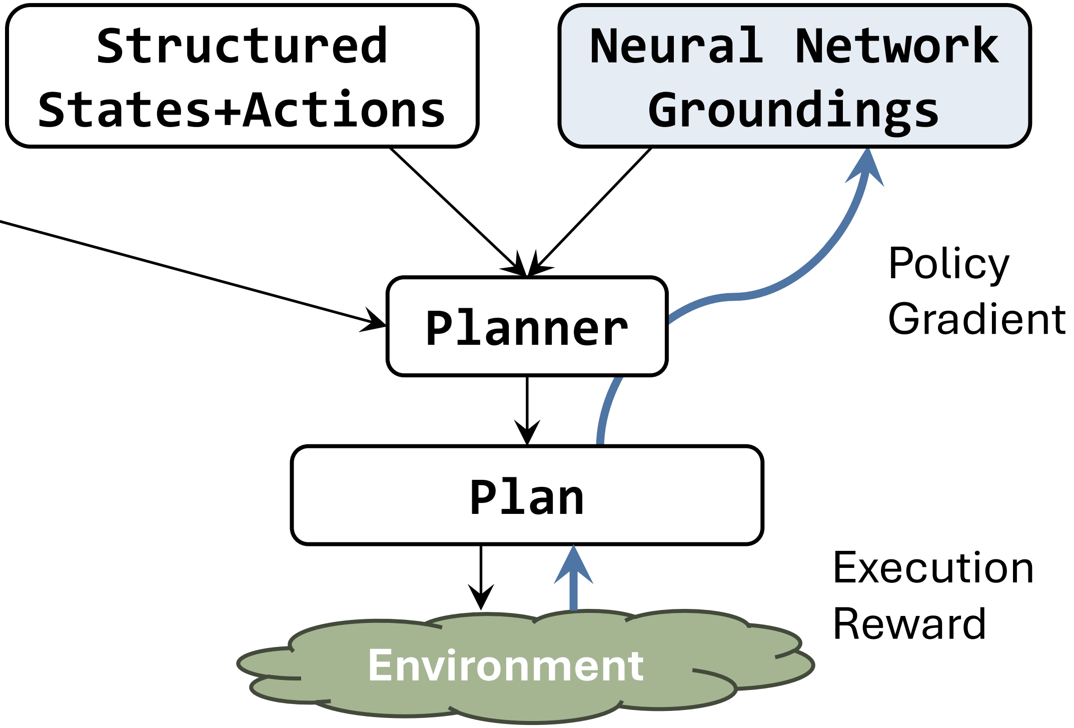
- We start with a distribution of tasks, including the environments and possible goals
- We want to **automatically** build a compositional abstraction for states and actions

Put chilled wine in the cabinet.

Warm a plate and place it on the table.

Wash the dirty bowl before putting the bowl on the counter.

Place a cold potato slice in the oven.



Learning Adaptive Planning Representations with Natural Language Guidance. Wong*, *Mao**, Sharma*, et al. ICLR 2024.

Learning Abstractions from Language

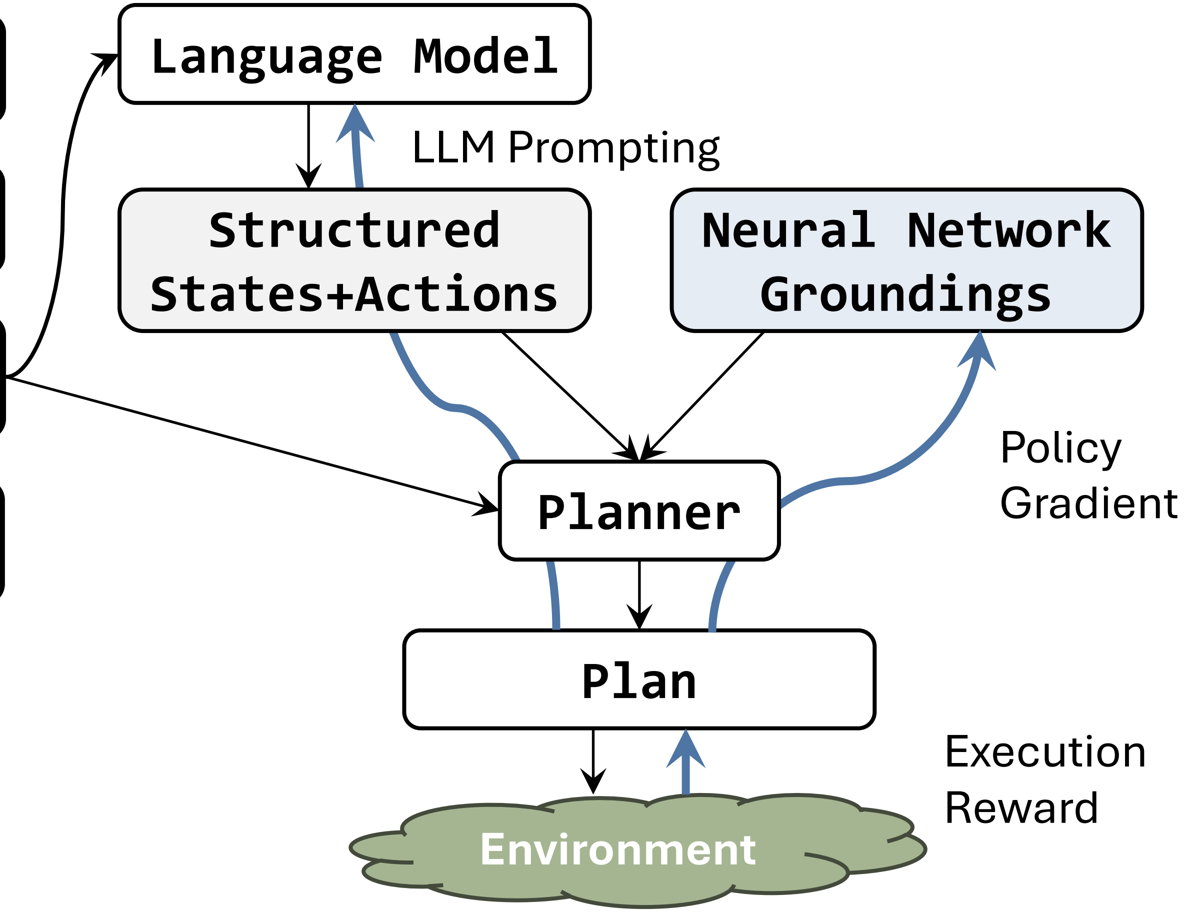
- We start with a distribution of tasks, including the environments and possible goals
- We want to **automatically** build a compositional abstraction for states and actions

Put chilled wine in the cabinet.

Warm a plate and place it on the table.

Wash the dirty bowl before putting the bowl on the counter.

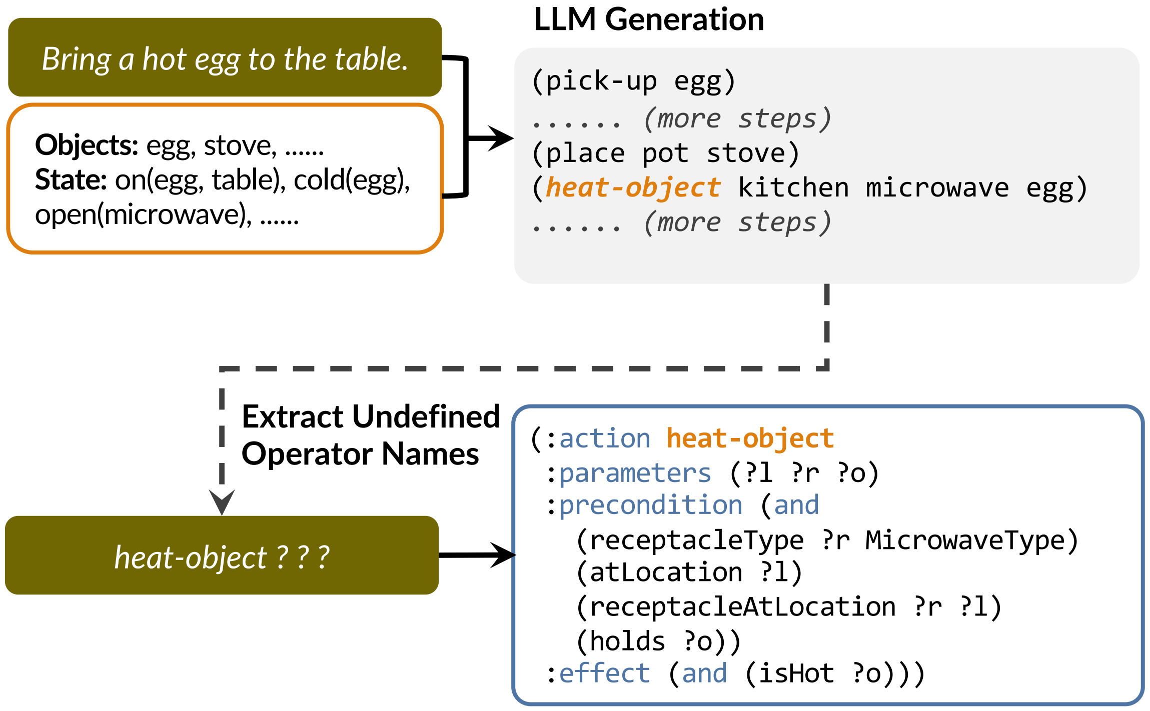
Place a cold potato slice in the oven.

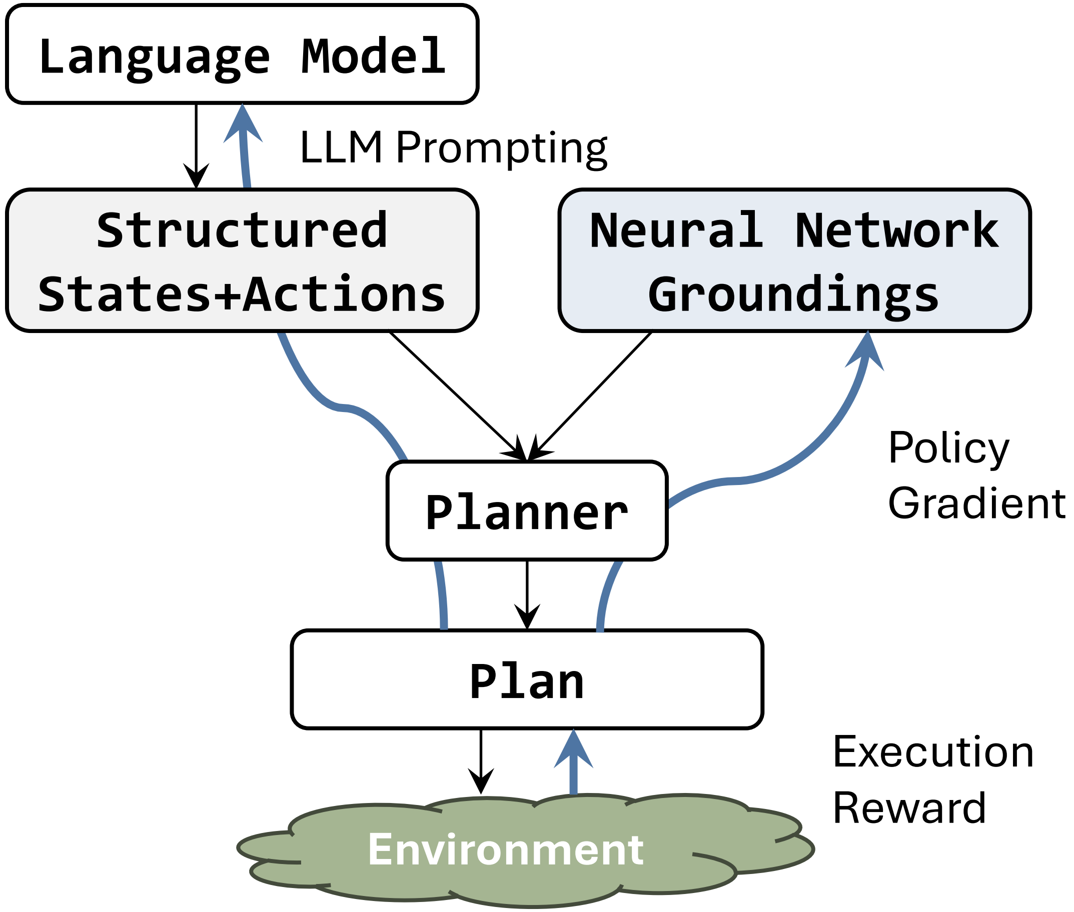


Learning Adaptive Planning Representations with Natural Language Guidance. Wong*, *Mao**, Sharma*, et al. ICLR 2024.

Learning Abstractions from Language

- We start with a distribution of tasks, including the environments and possible goals
- We want to **automatically** build a compositional abstraction for states and actions





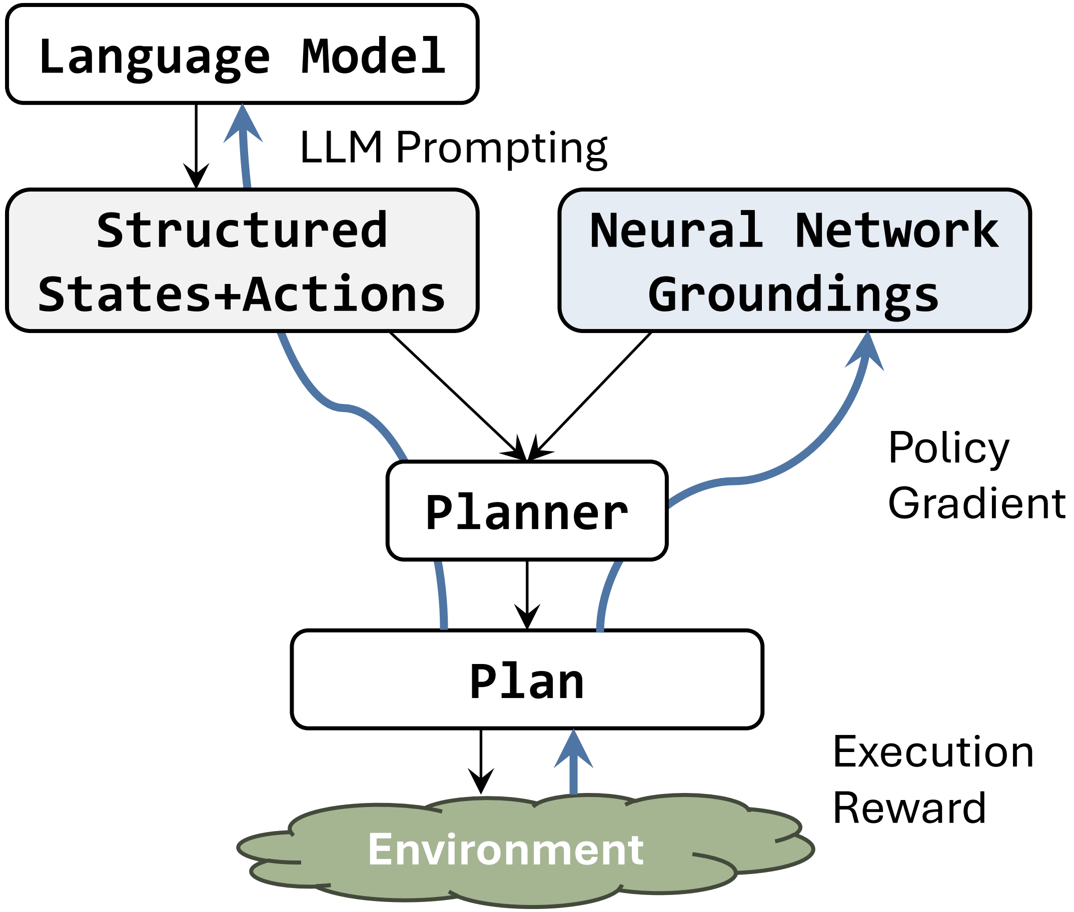
Learning Adaptive Planning Representations with Natural Language Guidance. Wong*, *Mao**, Sharma*, et al. ICLR 2024.

Learning Abstractions from Language

- We start with a distribution of tasks, including the environments and possible goals
- We want to **automatically** build a compositional abstraction for states and actions

```
(:action SliceObject
 :parameters (
   ?toolobject - object ?a – agent ?l - location ?o - object)
 :precondition (and
   (objectType ?toolobject KnifeType)
   (atLocation ?a ?l)
   (objectAtLocation ?o ?l)
   (sliceable ?o)
   (holds ?a ?toolobject))
 :effect (and
   (isSliced ?o)))
```

```
(:action CoolObject
 :parameters (
   ?toolreceptacle - receptacle ?a – agent ?l - location ?o - object)
 :precondition (and
   (receptacleType ?toolreceptacle FridgeType)
   (atLocation ?a ?l)
   (holds ?a ?o)
   (receptacleAtLocation ?toolreceptacle ?l))
 :effect (and
   (isCool ?o)))
```



Learning Adaptive Planning Representations with Natural Language Guidance. Wong*, *Mao**, Sharma*, et al. ICLR 2024.

Learning Abstractions from Language

Put chilled wine in the cabinet.

Warm a plate and place it on the table.

Wash the dirty bowl before putting the bowl on the counter.

Learning Adaptive Planning Representations with Natural Language Guidance. Wong*, *Mao**, Sharma*, et al. ICLR 2024.

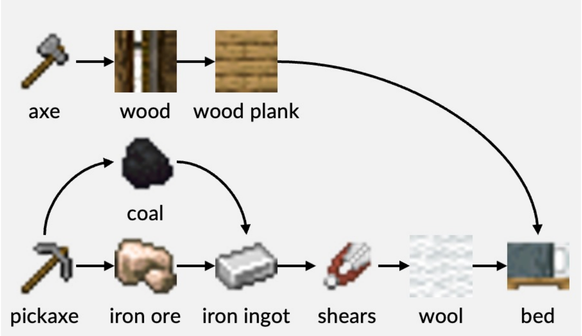
Learning Abstractions from Language

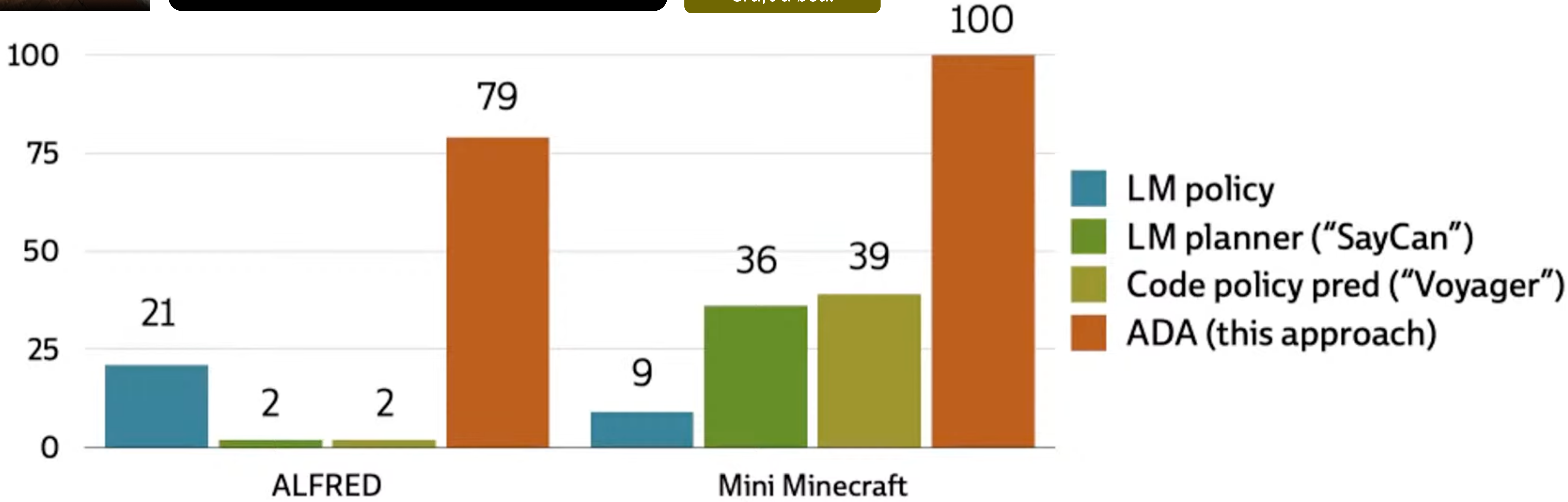
Put chilled wine in the cabinet.

Warm a plate and place it on the table.

Wash the dirty bowl before putting the bowl on the counter.

Craft a bed.





Learning Adaptive Planning Representations with Natural Language Guidance. Wong*, *Mao**, Sharma*, et al. ICLR 2024.

Structures of the "Robot Brain"

Compositional Action

State Representation
Monolithic — Compositional

Action Representation
As Feedforward Policies — As Causal Models

Model Acquisition
Machine Learned — Human Programmed

Factorization representations improve learning and planning efficiency
Temporal structures support generalization to unseen goals and states

Structures of the “Robot Brain”

Compositional Action

State Representation
Monolithic
Compositional

Action Representation
As Feedforward Policies
As Causal Models

?

Model Acquisition
Machine Learned
Human Programmed

So far we have been exploring learning only causal models for “primitives”
Next: going beyond causal models and beyond language

What Can We Learn from One Demonstration?

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

What Can We Learn from One Demonstration?

A “**strategy**” for picking up the cylinder

- Push to rotate
- Exert force on one end so that it tilts
- Move the bucket

You might not be able to execute it robustly now, but you have some “**ideas**”

We aim to learn such “strategies” from a single demonstration and apply them compositionally

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

Problem Formulation

We have a basic model for object manipulation & ***one demonstration***

What can we learn from the demonstration?

Single Demo

Learning Algo.

"Knowledge" of Manipulation Strategy

Basic Domain Model

(5 Actions: Transit, Grasp, Place, Push, Move)

Planning Algo.

New State
New Goal

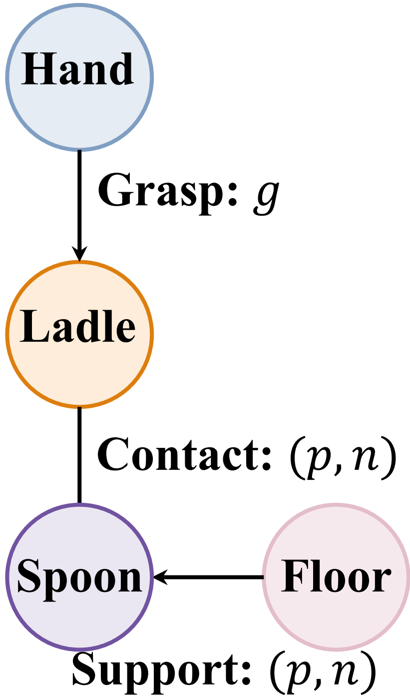
$u_1, u_2, \ldots$

Actions

What Can We Learn from One Demonstration?

Key idea: some manipulation “strategies” can be modeled by a sequence of subgoals about contacts among objects

Let’s talk about a familiar example: hook-using

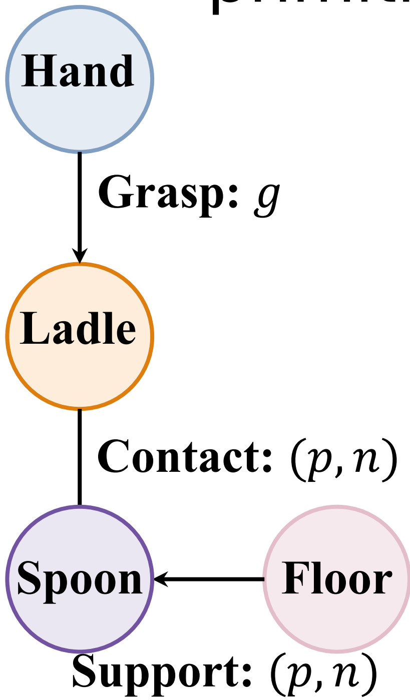


What Can We Learn from One Demonstration?

Key idea: some manipulation "strategies" can be modeled by a sequence of subgoals about contacts among objects

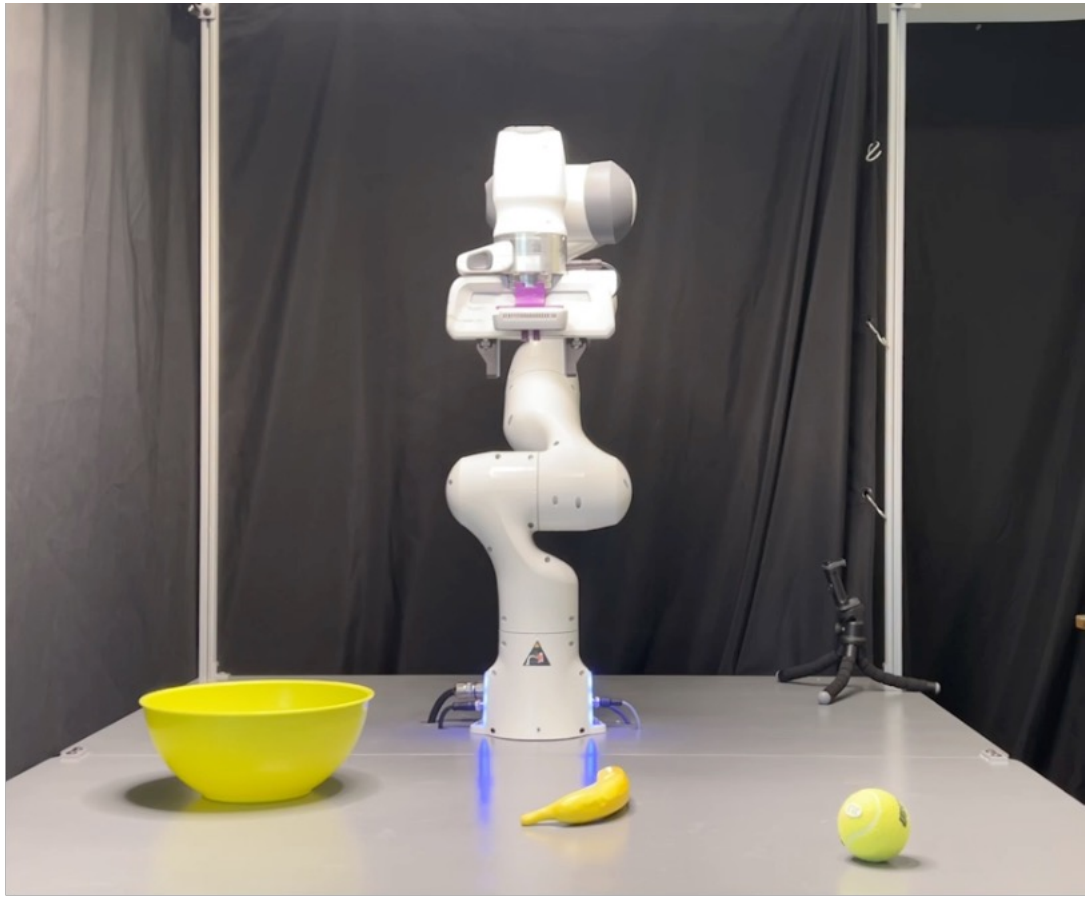
Let's talk about a familiar example: hook-using

Those are some most "primitive" mode families!

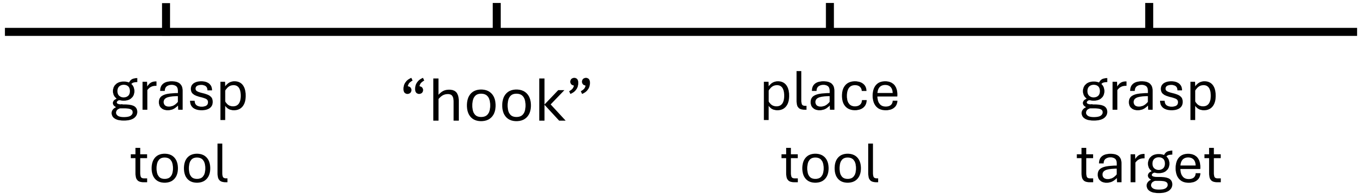


The Contact Mode Subgoals in Hook-Using

Key idea: some manipulation "strategies" can be modeled by
a sequence of subgoals about contacts among objects

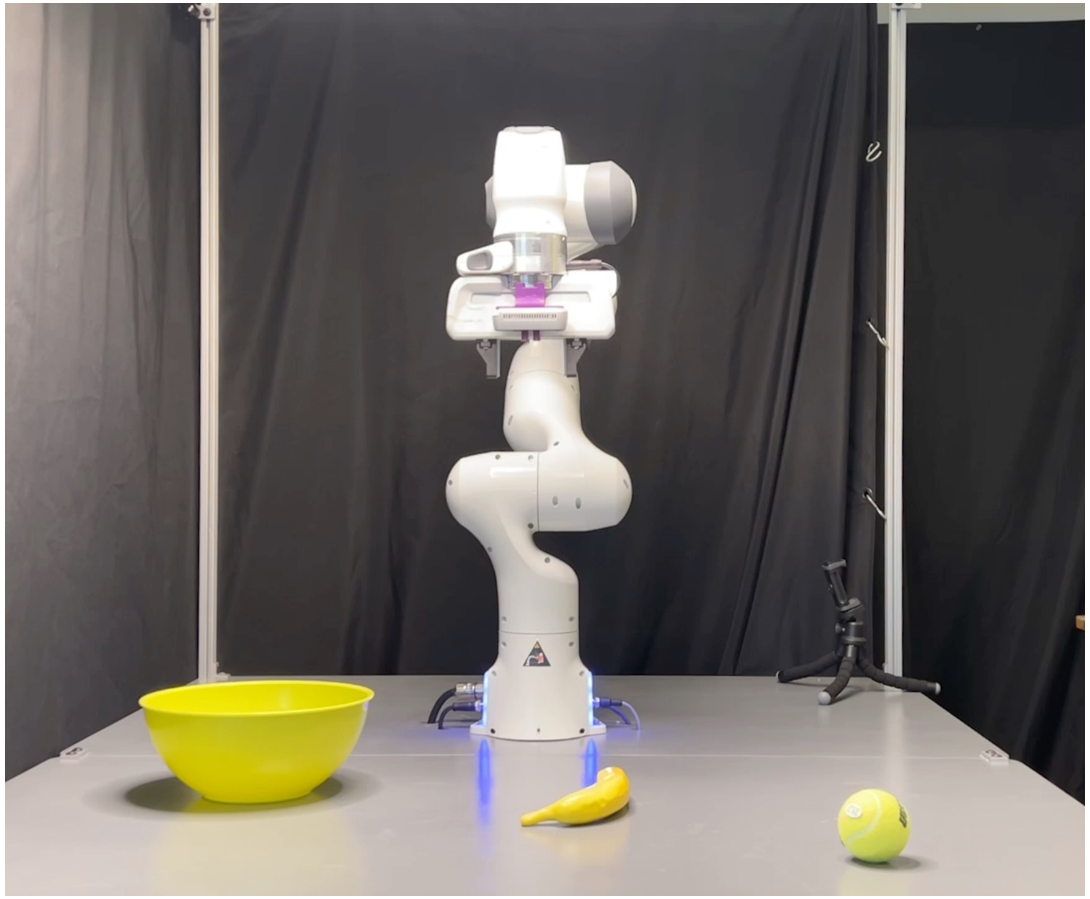


```
action hook(target, tool, support):
  body:
    achieve holding(tool, ?grasp1)
    move-with-contact(tool, target, ?traj)
    achieve holding-nothing
    grasp(target, ?grasp2)
  eff:
    holding(target, ?grasp2)
```

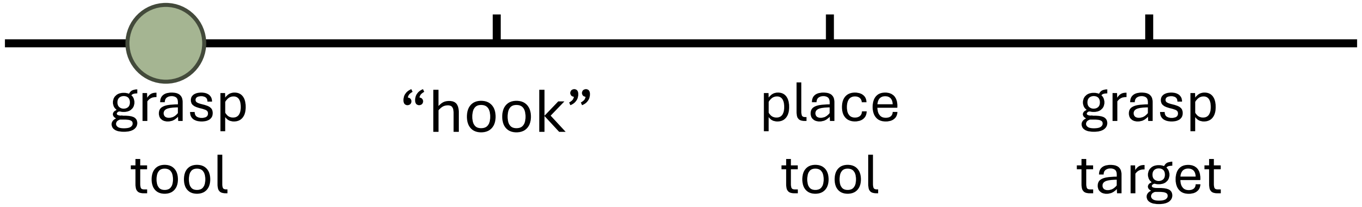


The Contact Mode Subgoals in Hook-Using

Key idea: some manipulation “strategies” can be modeled by a sequence of subgoals about contacts among objects

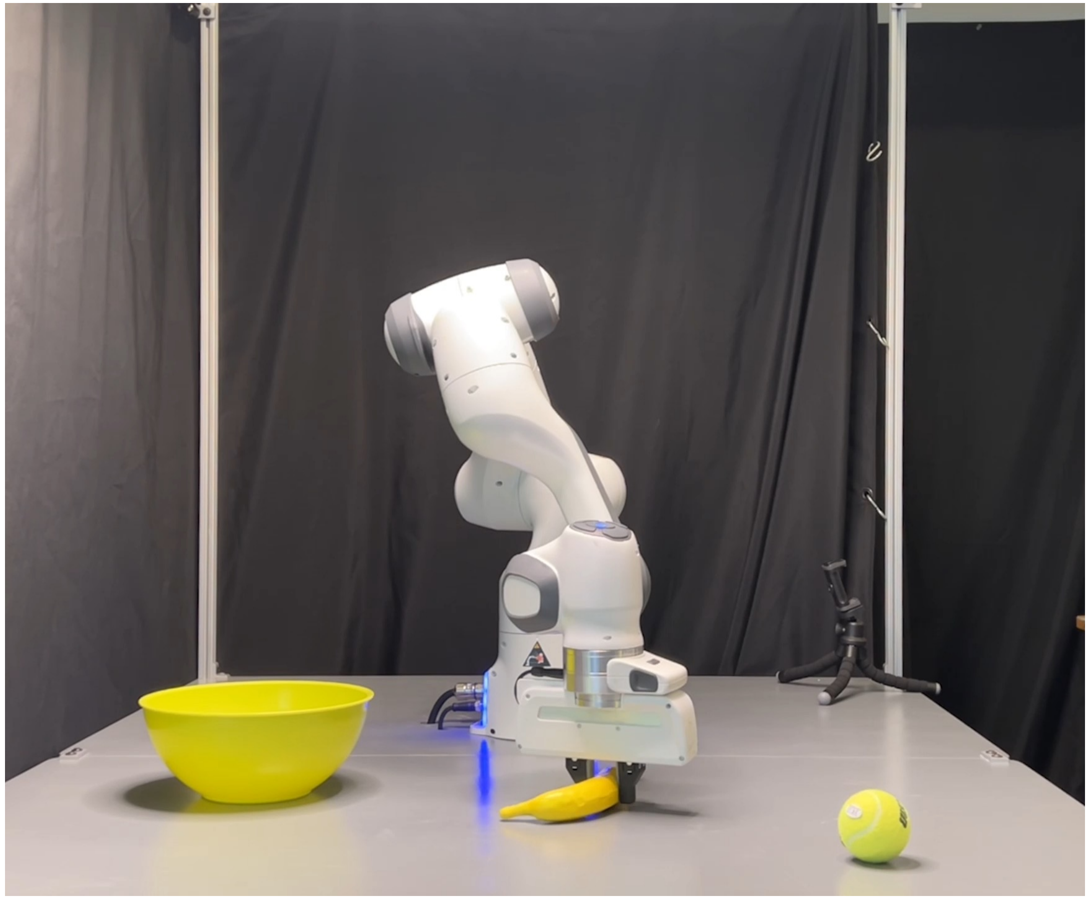


```
action hook(target, tool, support):
  body:
    achieve holding(tool, ?grasp1)
    move-with-contact(tool, target, ?traj)
    achieve holding-nothing
    grasp(target, ?grasp2)
  eff:
    holding(target, ?grasp2)
```

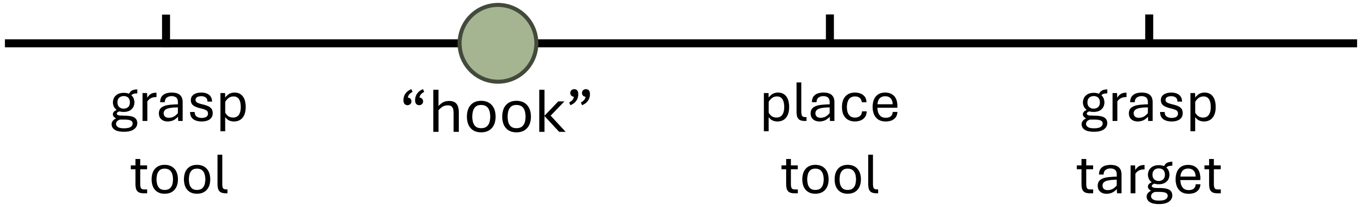


The Contact Mode Subgoals in Hook-Using

Key idea: some manipulation "strategies" can be modeled by a sequence of subgoals about contacts among objects

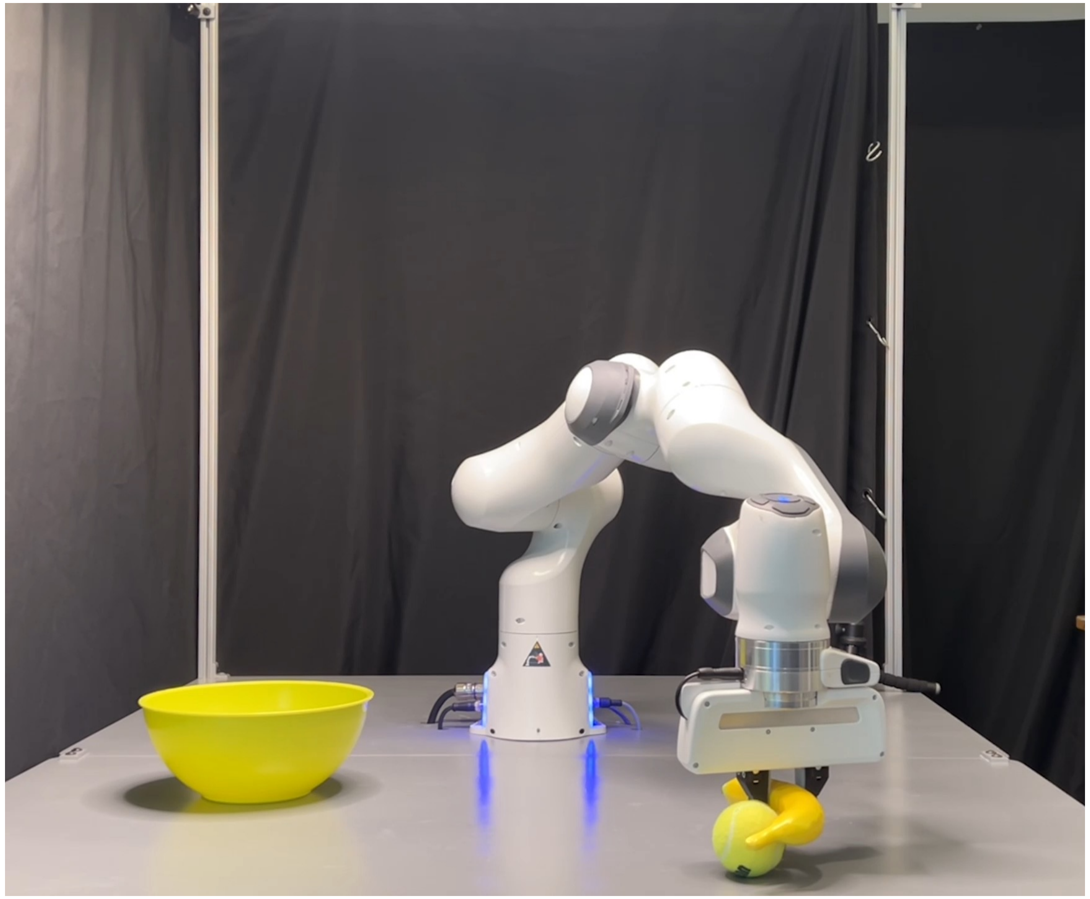


```
action hook(target, tool, support):
  body:
    achieve holding(tool, ?grasp1)
    move-with-contact(tool, target, ?traj)
    achieve holding-nothing
    grasp(target, ?grasp2)
  eff:
    holding(target, ?grasp2)
```

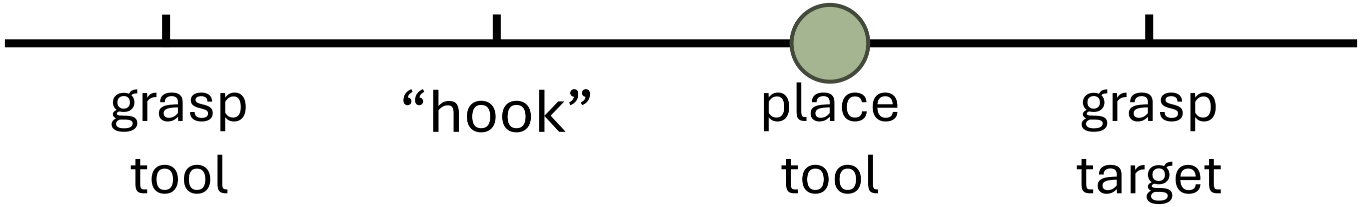


The Contact Mode Subgoals in Hook-Using

Key idea: some manipulation "strategies" can be modeled by a sequence of subgoals about contacts among objects

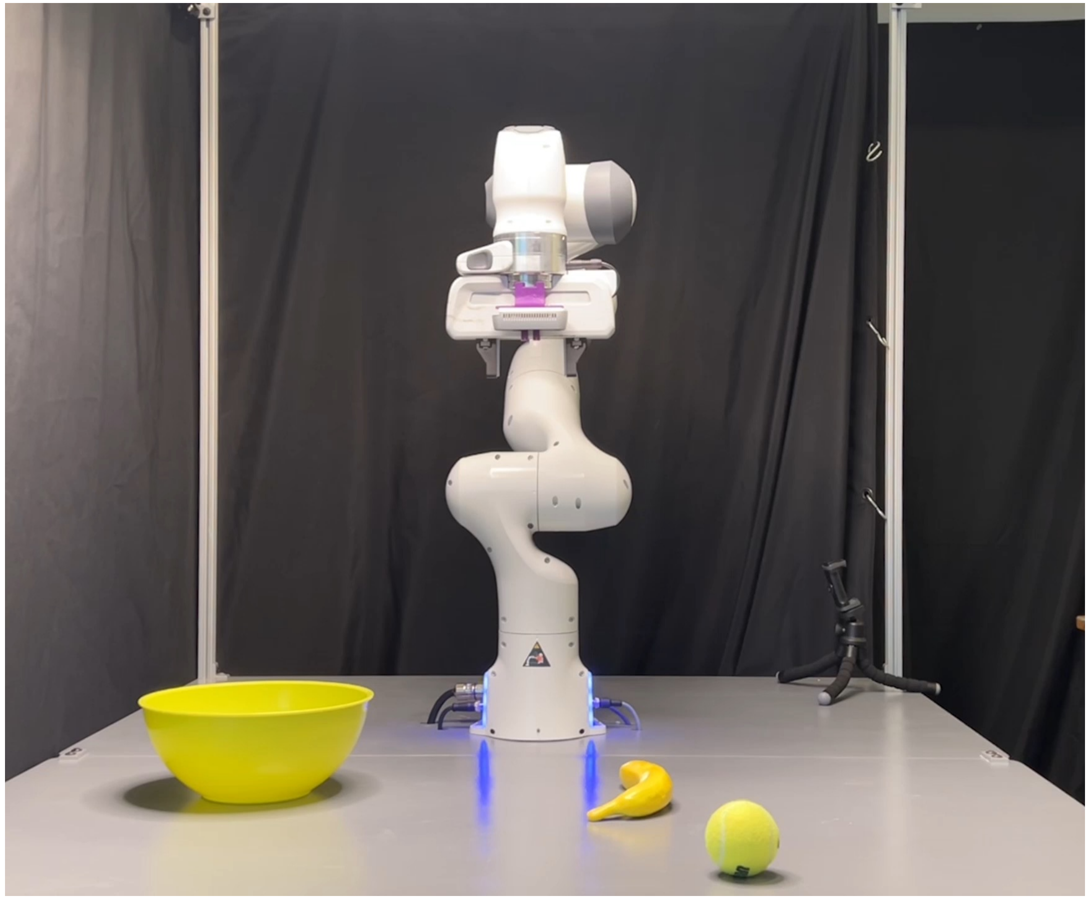


```
action hook(target, tool, support):
  body:
    achieve holding(tool, ?grasp1)
    move-with-contact(tool, target, ?traj)
    achieve holding-nothing
    grasp(target, ?grasp2)
  eff:
    holding(target, ?grasp2)
```

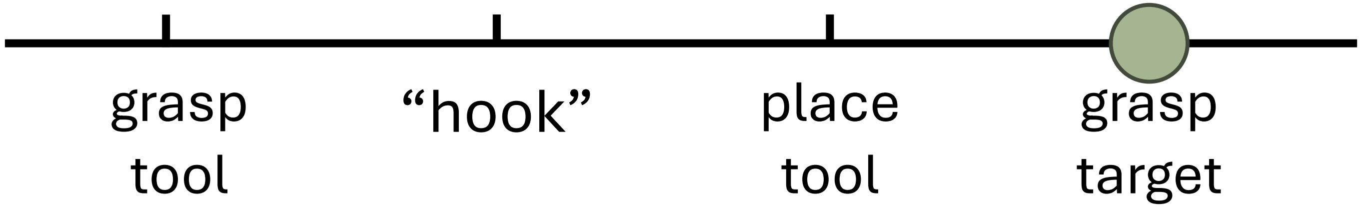


The Contact Mode Subgoals in Hook-Using

Key idea: some manipulation "strategies" can be modeled by a sequence of subgoals about contacts among objects

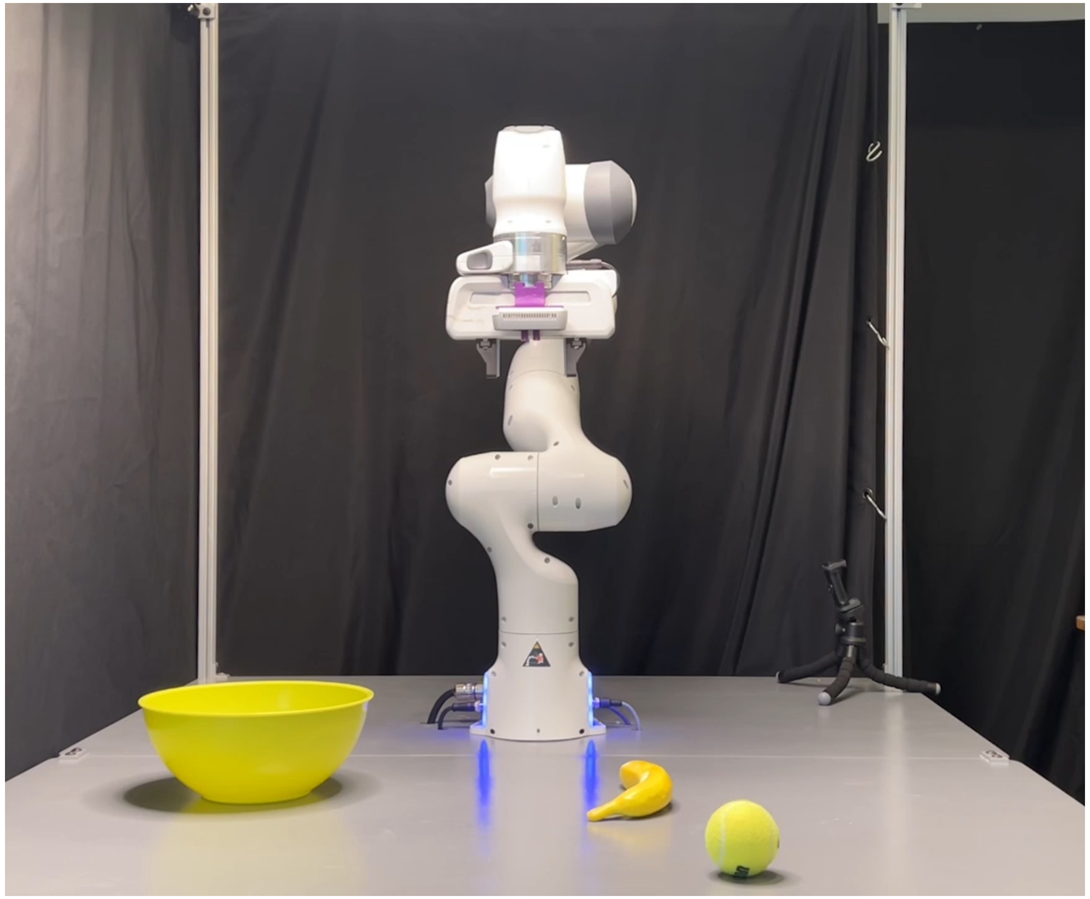


```
action hook(target, tool, support):
  body:
    achieve holding(tool, ?grasp1)
    move-with-contact(tool, target, ?traj)
    achieve holding-nothing
    grasp(target, ?grasp2)
  eff:
    holding(target, ?grasp2)
```



The Contact Mode Subgoals in Hook-Using

Key idea: some manipulation "strategies" can be modeled by a sequence of subgoals about contacts among objects



```
action hook(target, tool, support):
  body:
    achieve holding(tool, ?grasp1)
    move-with-contact(tool, target, ?traj)
    achieve holding-nothing
    grasp(target, ?grasp2)
  eff:
    holding(target, ?grasp2)
```

Previously we were learning causal models of actions and planning with them. Now we can memorize "partial solutions" as shortcuts

Many Strategies Can Be Represented This Way

We call these manipulation strategies "***mechanisms***"

Mechanisms as sequence of contact mode families *generalizes*

We learn these mechanisms, and we *compose* them

Overview of the Framework

There are two **learning problems**:

1. Learning of the contact mode sequence

2. Learning samplers for parameters of the contact modes: where to grasp, how to move, *etc.*

Overview of the Framework

There are two **learning problems**:

1. Learning of the contact mode sequence

   We will recover it from the single demonstration

2. Learning samplers for parameters of the contact modes: where to grasp, how to move, *etc.*

Single Demo

```
action hook(
 tool, target,
 support
):
 body:
   achieve ...
   ...
 eff:
  (holding ?target)
```

Contact Modes and Goals

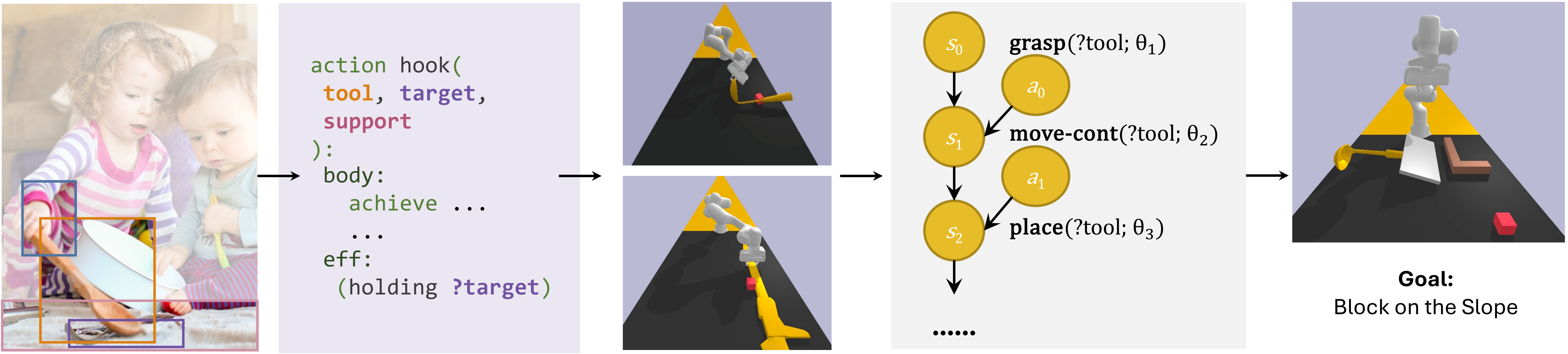
Overview of the Framework

There are two **learning problems**:

1. Learning of the contact mode sequence

   We will recover it from the single demonstration

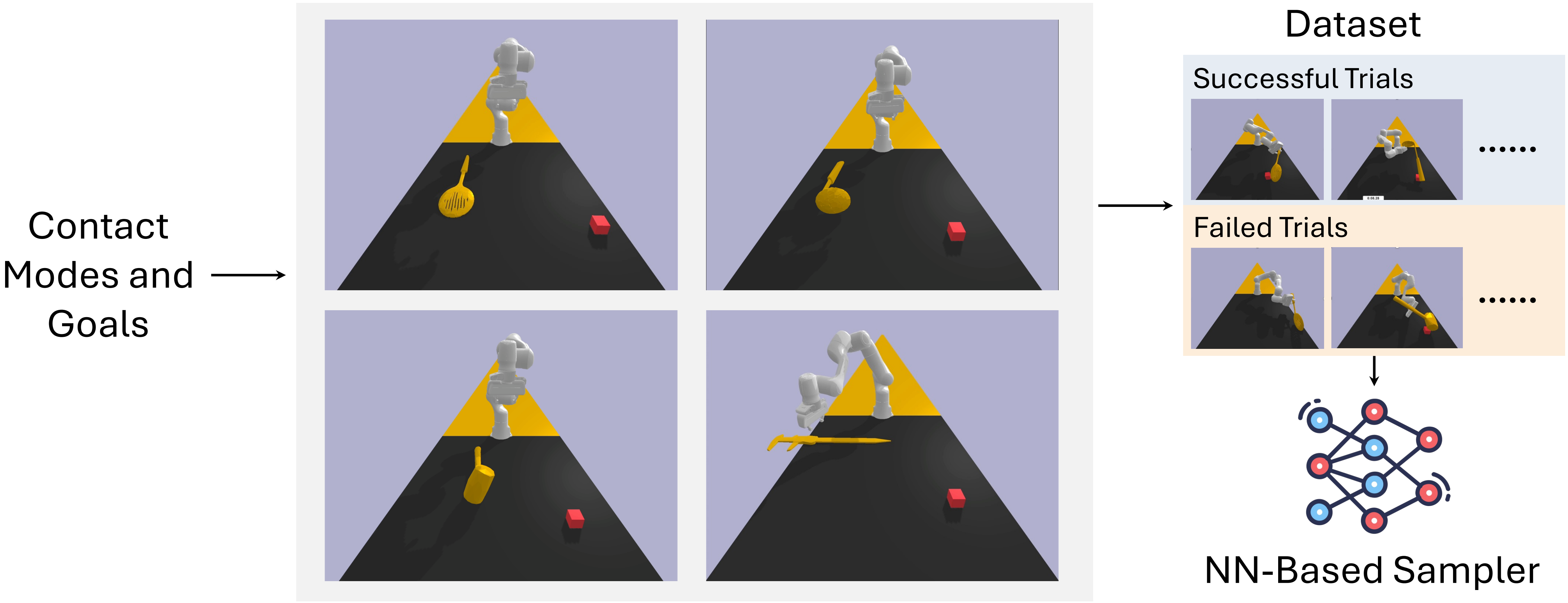
2. Learning samplers for parameters of the contact modes: where to grasp, how to move, *etc.*



Single Demo **Contact Modes and Goals** **Self-Play** **Learned Contact Distributions** **Compositional Planning**

Step 2: Learn Mechanism-Specific Samplers

We will learn those samplers (parameter generators) from self-play

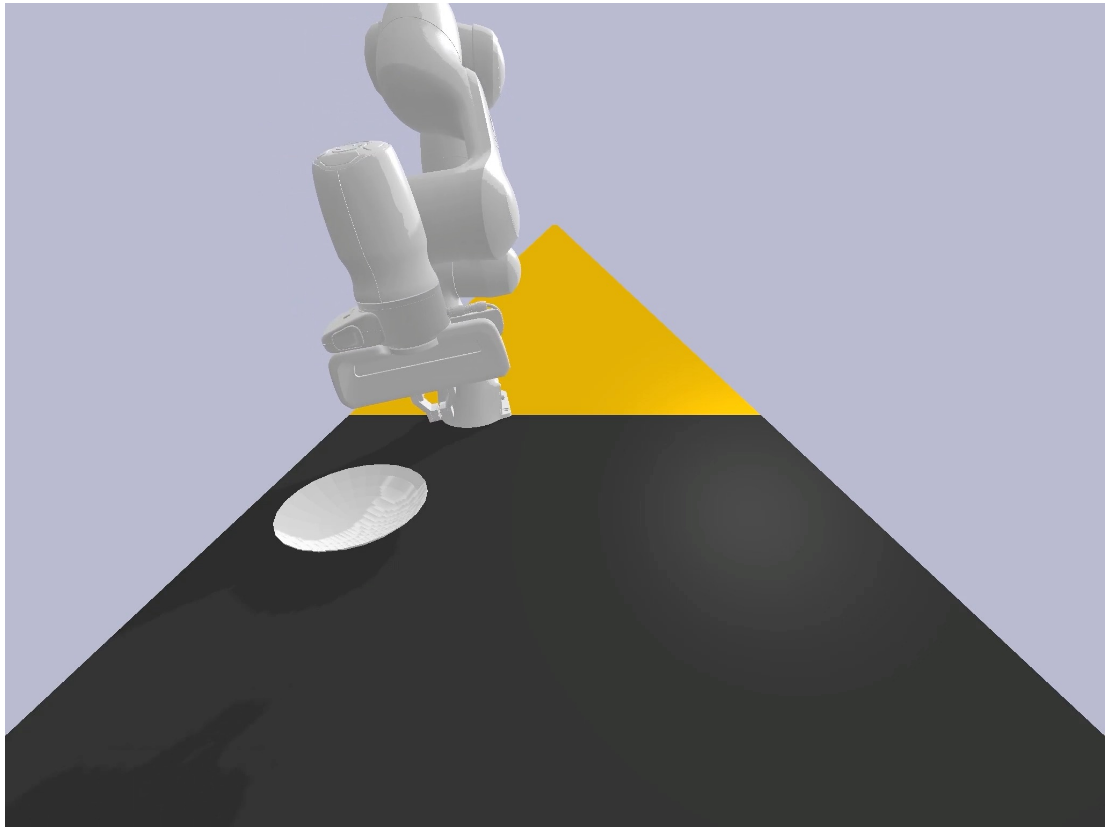


Self-Play with Randomly Sampled Objects and Poses

Learning Mechanisms Improves Planning Efficiency

Method
Basis Ops Only
Ours (Macro+Sampler)

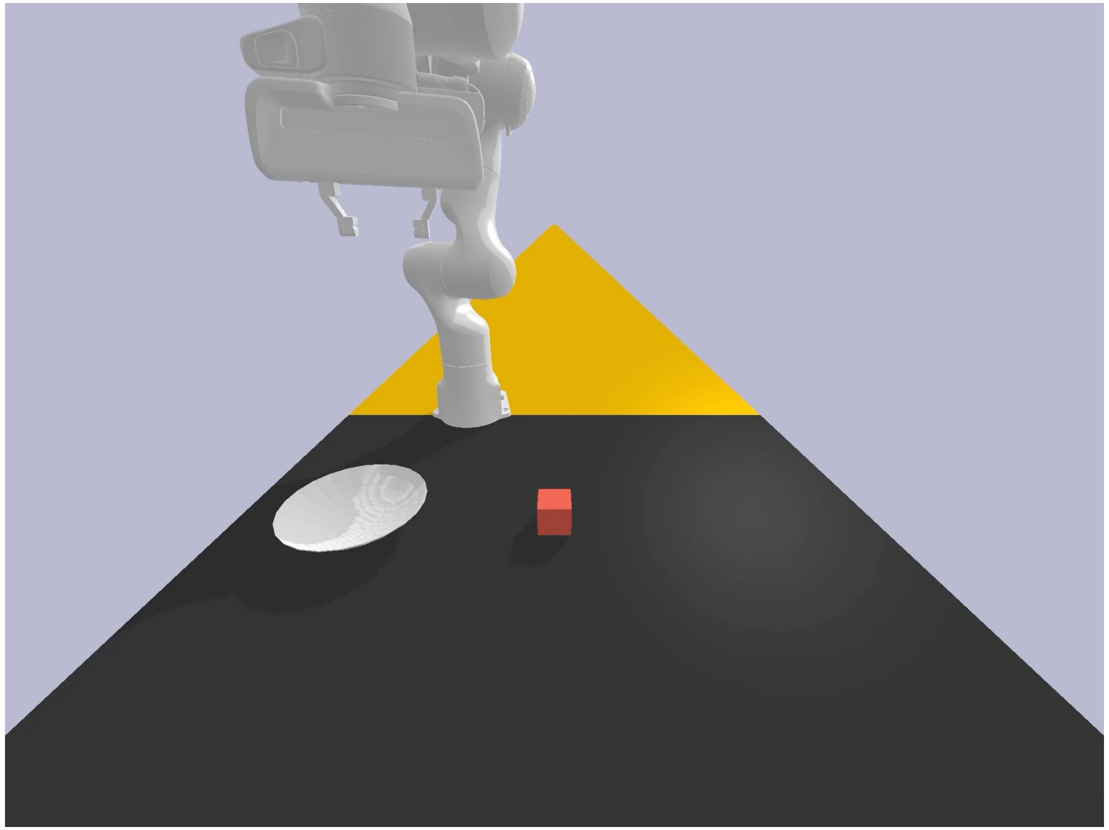
Learning Mechanisms Improves Planning Efficiency



Goal:
`holding(plate)`

Method	Edge
Basis Ops Only	89.45±5.53
Ours (Macro+Sampler)	**0.57**±0.05

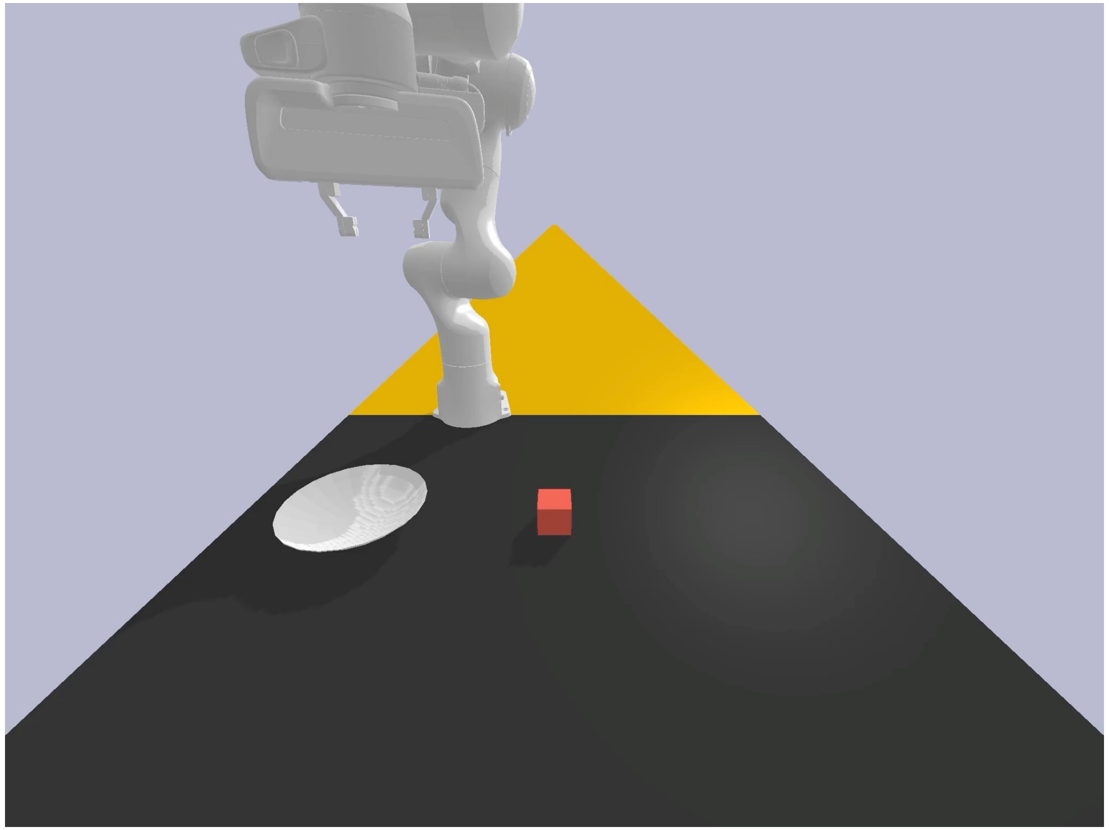
Learning Mechanisms Improves Planning Efficiency



Goal:
`holding(plate)`

Method	Edge	Hook	Lever
Basis Ops Only	89.45±5.53	>600	523.18±9.22
Ours (Macro+Sampler)	**0.57**±0.05	**3.84**±1.56	1.55±0.29

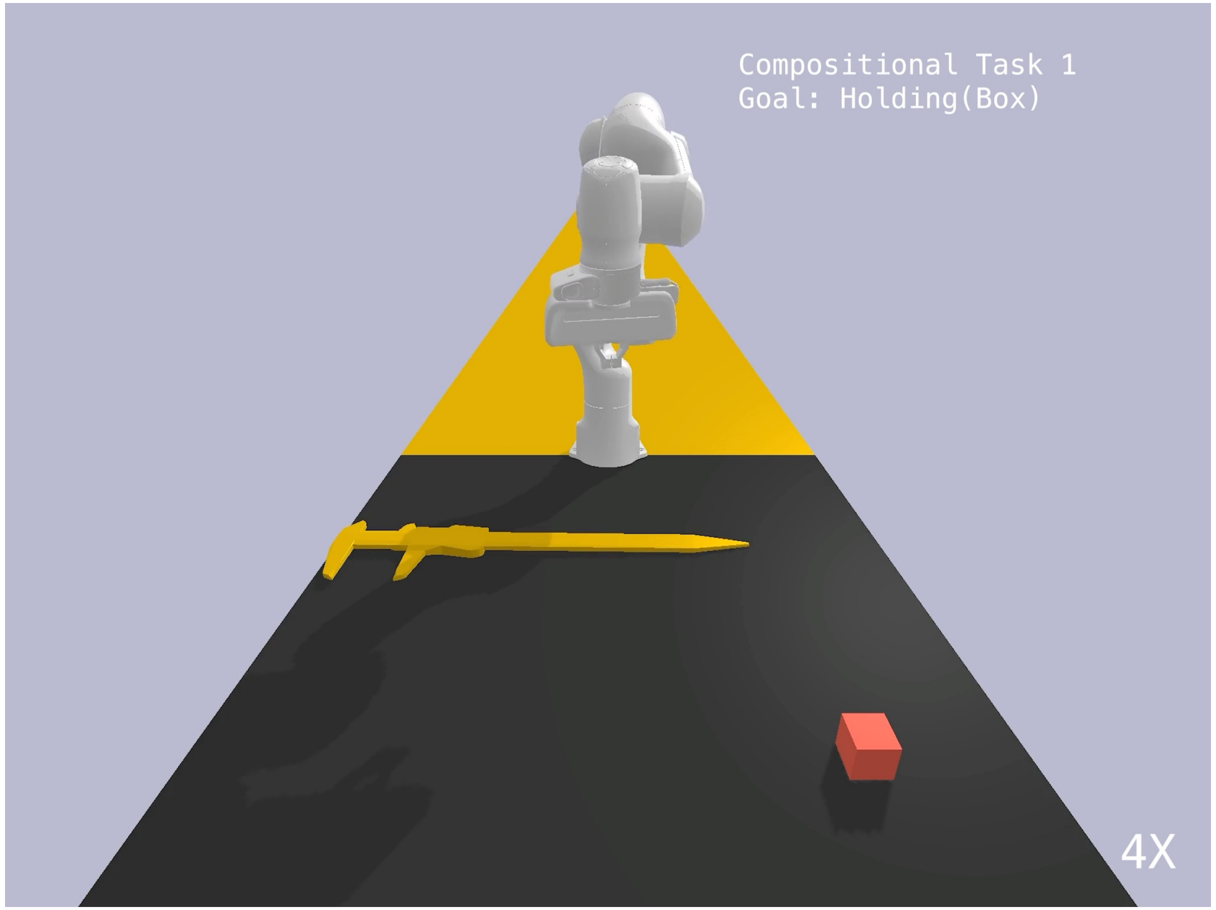
Learning Mechanisms Improves Planning Efficiency



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

Composing Mechanisms Automatically by Planning

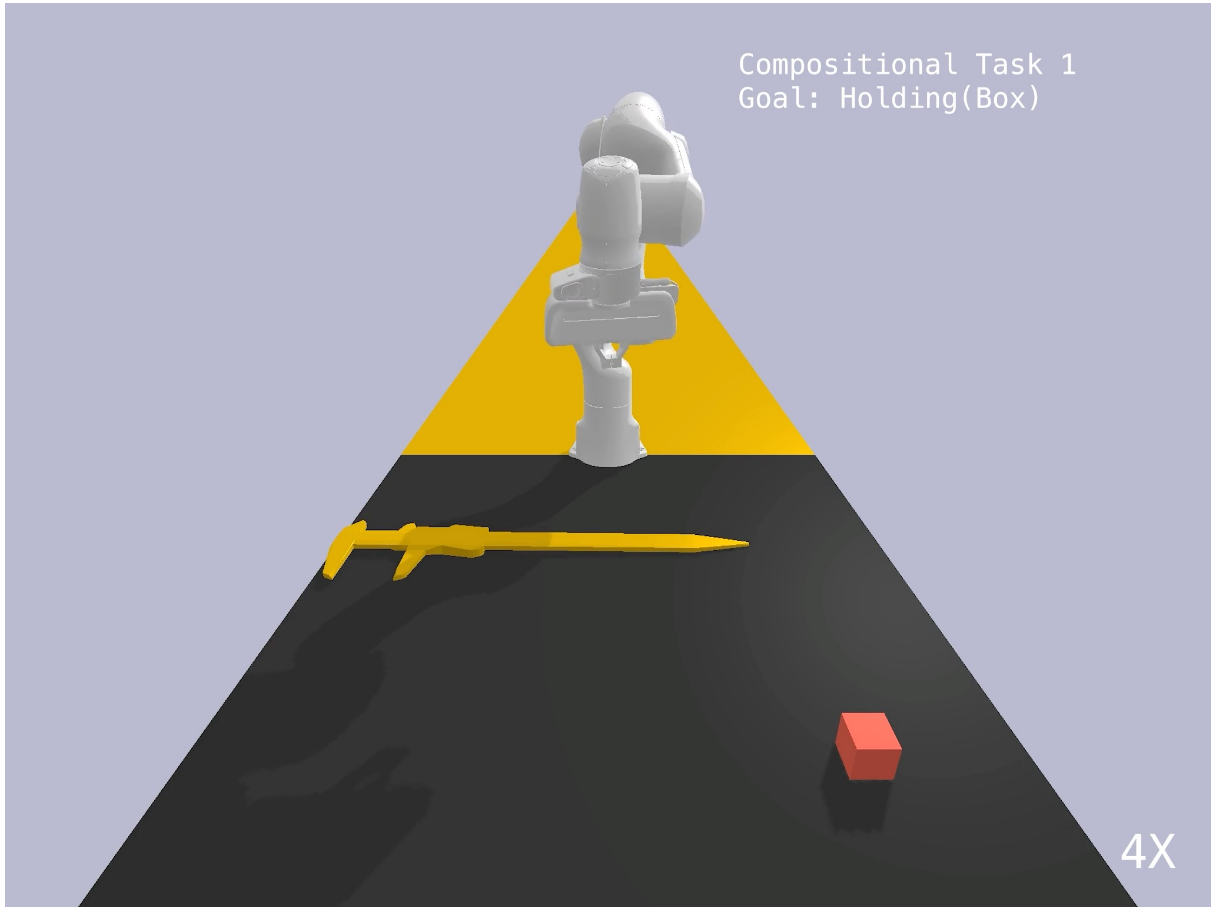



```
action hook(target, tool, support):
  body:
    achieve holding(tool, ?grasp1)
    move-with-contact(tool, target, ?t)
    ......
  eff:
    holding(target, ?grasp2)


action grasp-from-edge(target, support):
  body:
    push(target, support, ?t)
    grasp(target, ?grasp)
eff:
    holding(target, ?grasp)
```

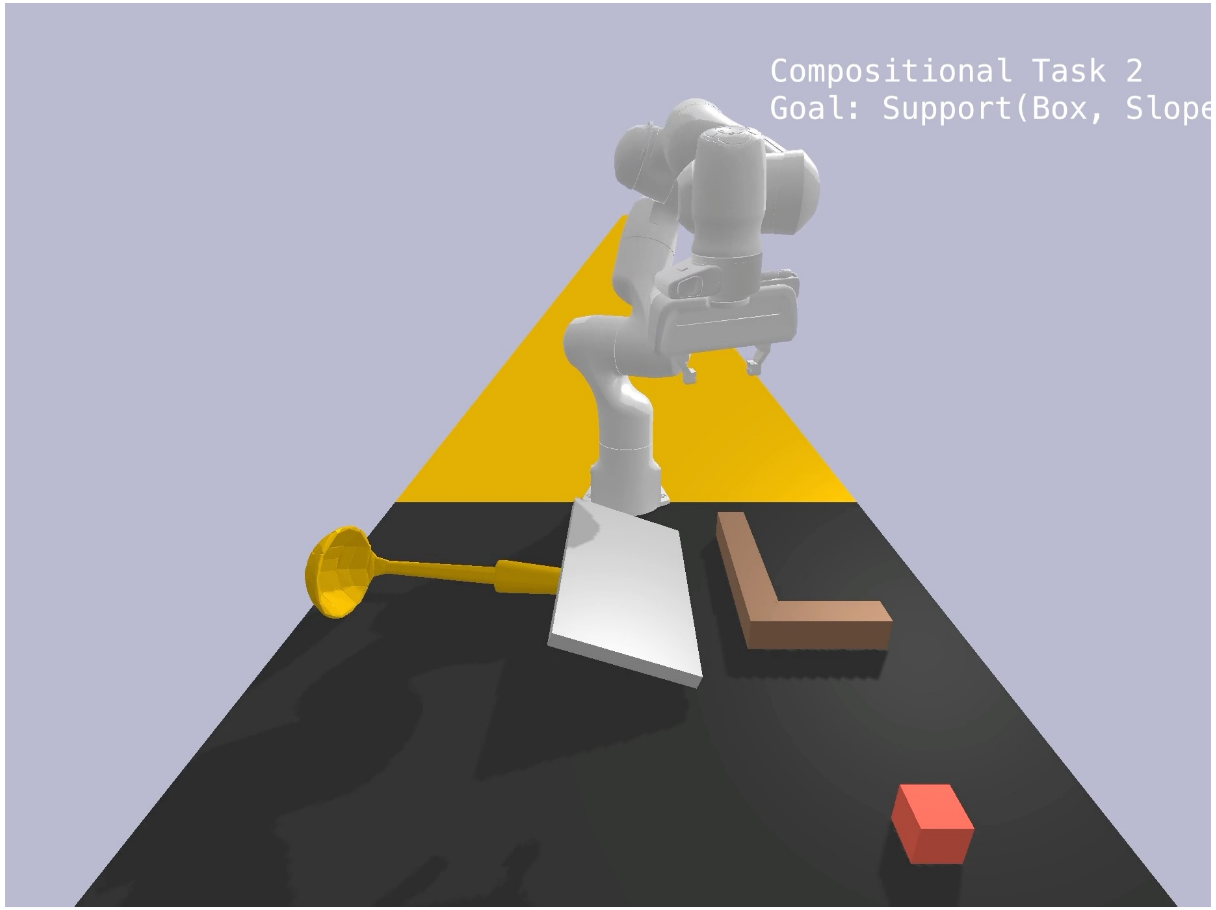
Goal: holding(box)
The caliper is too flat to be grasped

Automatically composed
by matching preconditions and effects

Composing Mechanisms Automatically by Planning



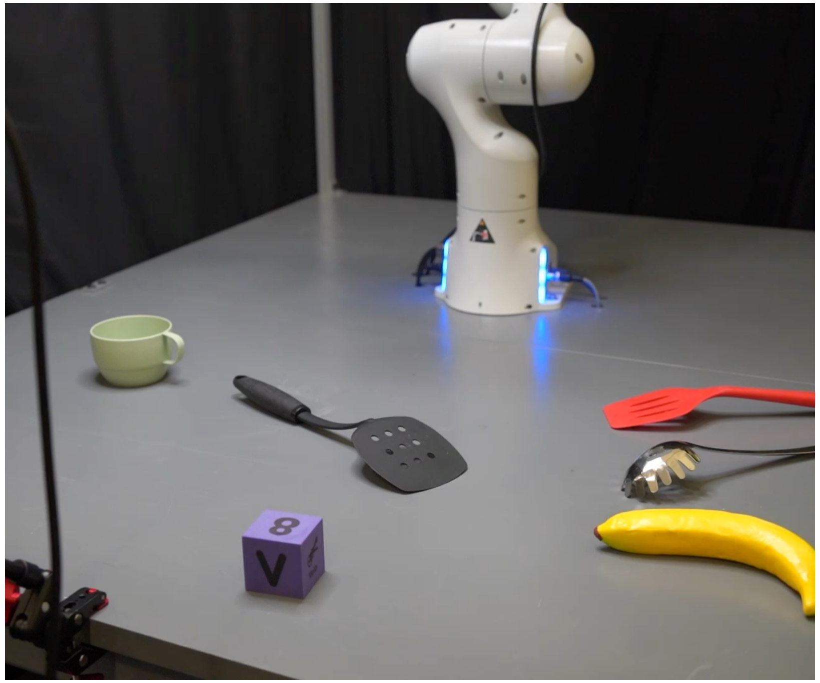
Goal: holding(box)
The caliper is too flat to be grasped

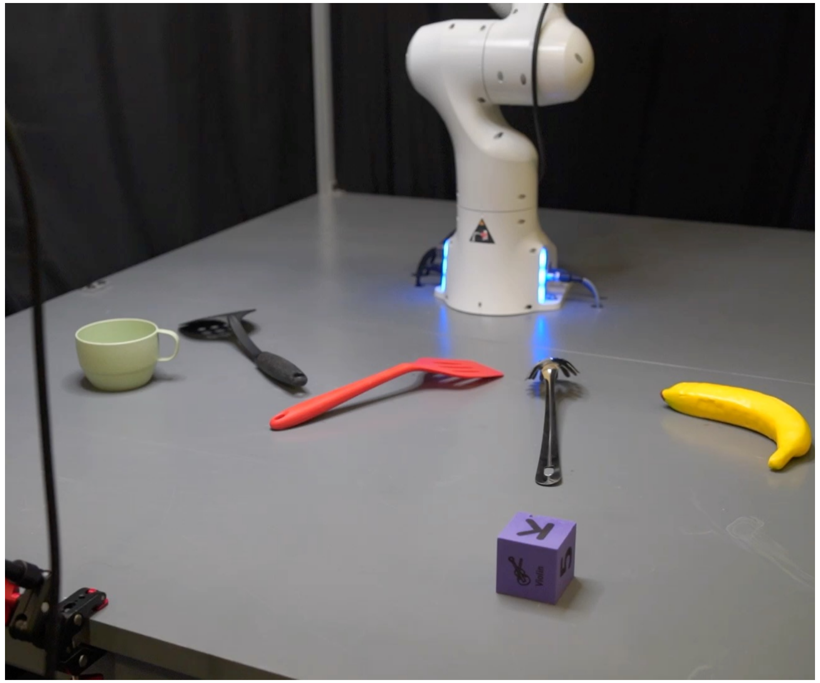


Goal: on(box, ramp)
Box may slide down the ramp

Real Robot Execution of the Learned Strategies

Goal: in(cube, cup)



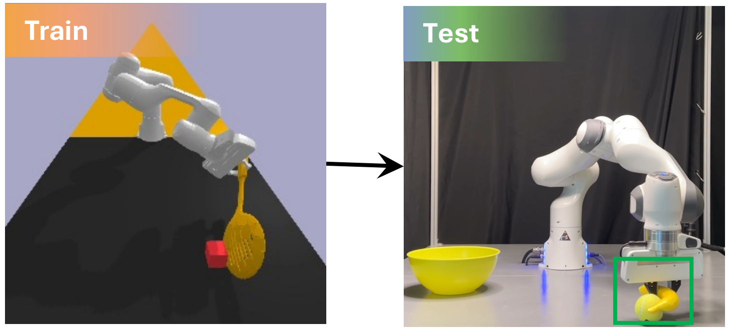


Generalizes to new tools, with no 3D model required

We apply our structured model and planner based on point cloud inputs

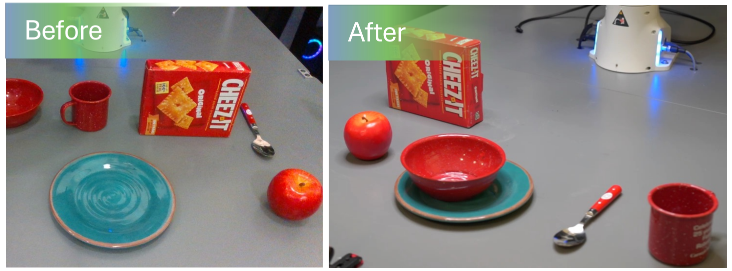
Compositional Abstractions Enable Generalization

Generalization to Novel Objects

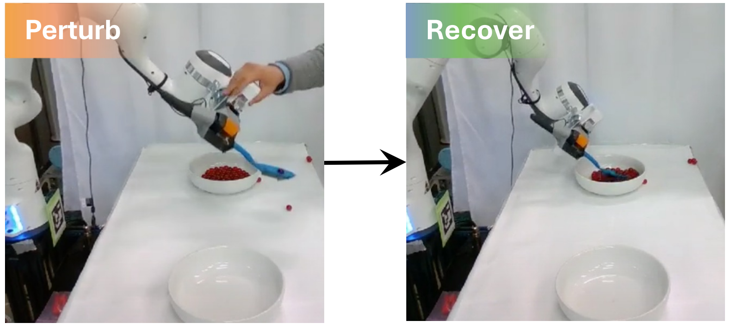


Generalization to Novel Goals

Set up a table for my breakfast.

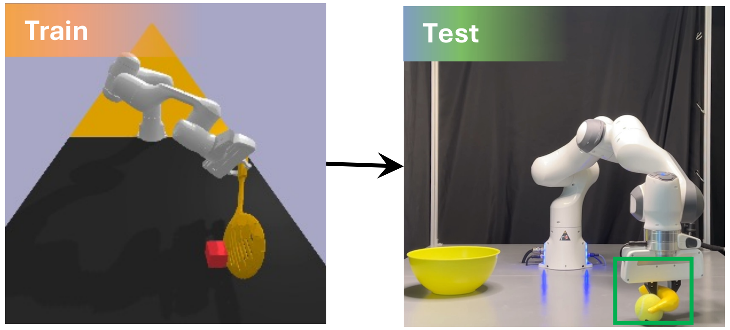


Generalization to Novel States



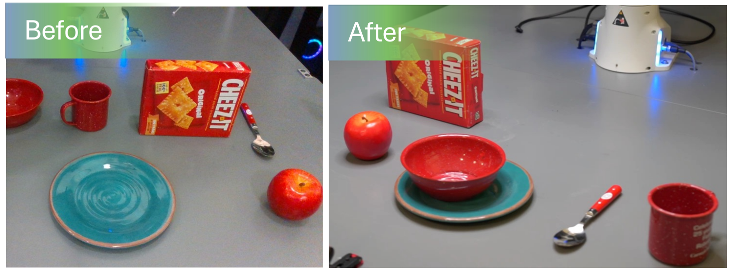
Compositional Abstractions Enable Generalization

Generalization to Novel Objects

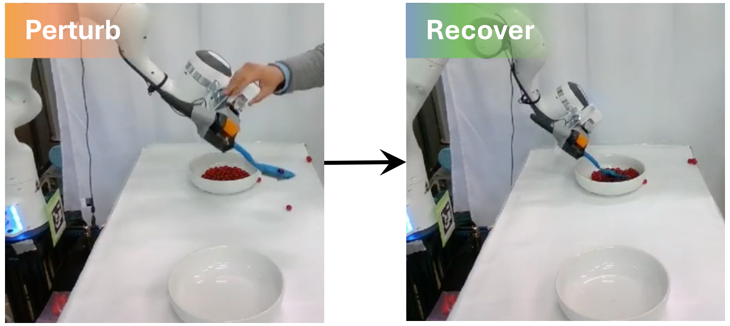


Generalization to Novel Goals

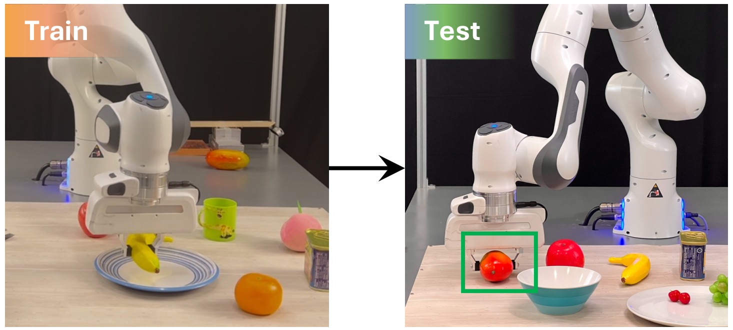
Set up a table for my breakfast.



Generalization to Novel States



Generalization to Novel Words

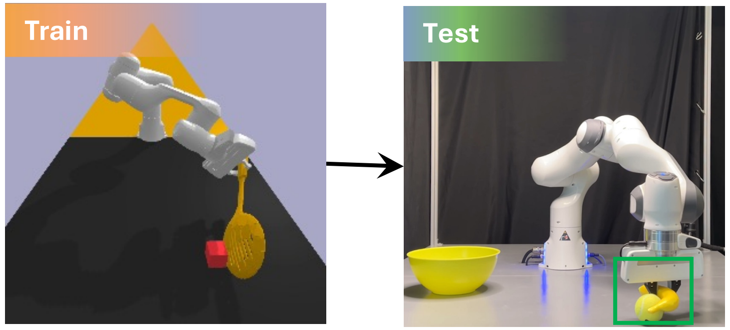


By composing learned action controllers and visual recognition models (e.g., CLIP), we can zero-shot generalize to instructions with previously unseen words

Programmatically Grounded, Compositionally Generalization Robotic Manipulation. Wang*, *Mao**, Hsu, Zhao, Gao, Wu. ICLR 2023.

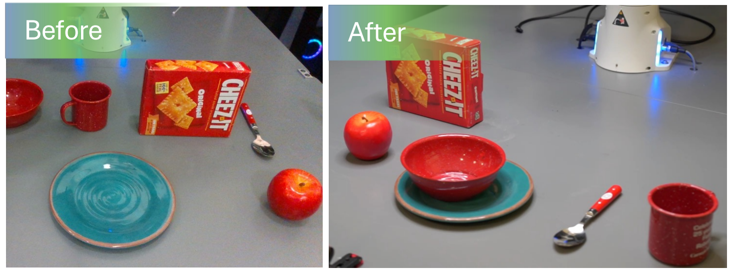
Compositional Abstractions Enable Generalization

Generalization to Novel Objects

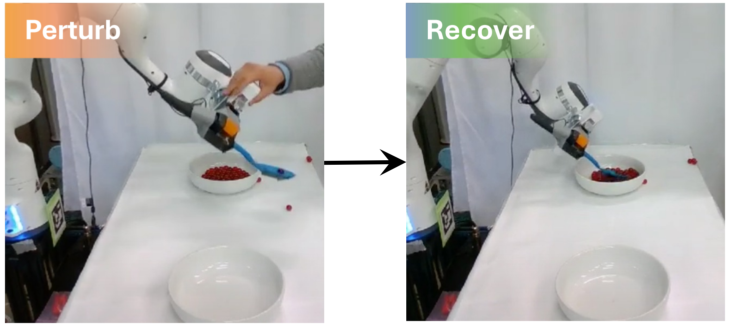


Generalization to Novel Goals

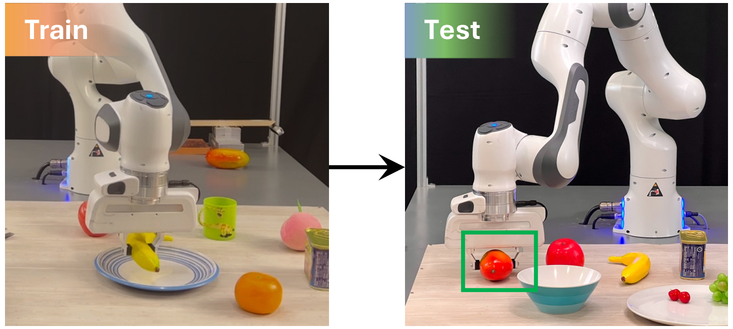
Set up a table for my breakfast.



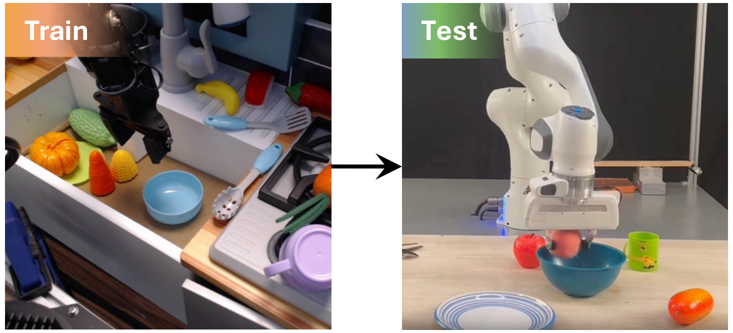
Generalization to Novel States



Generalization to Novel Words



Generalization to Novel Embodiments

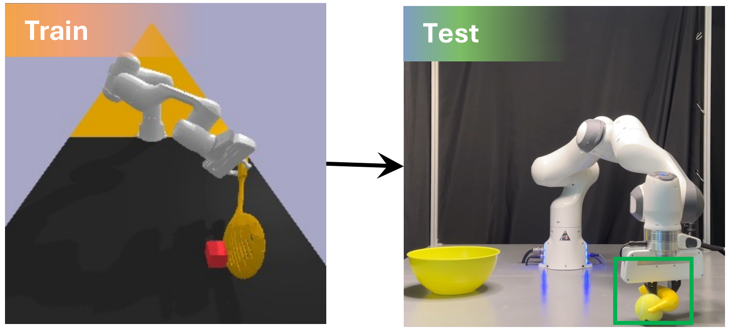


By composing the robot controller and the generation of object trajectories, we can train policies on videos of other robots and even humans, and deploy on a different robot

Learning to Act from Actionless Video through Dense Correspondences. Ko, *Mao*, Du, Sun, Tenenbaum. ICLR 2024.

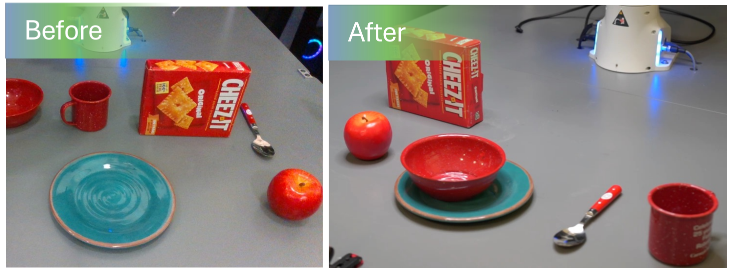
Compositional Abstractions Enable Generalization

Generalization to Novel Objects

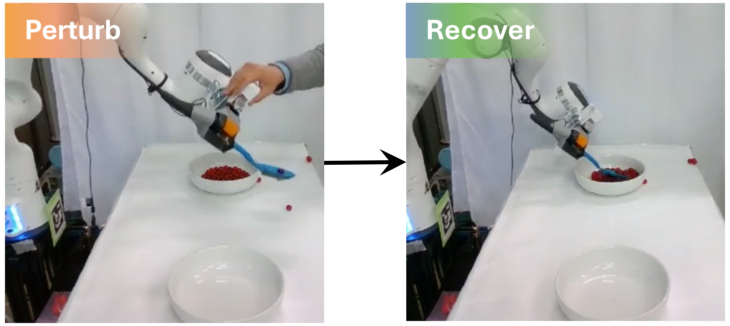


Generalization to Novel Goals

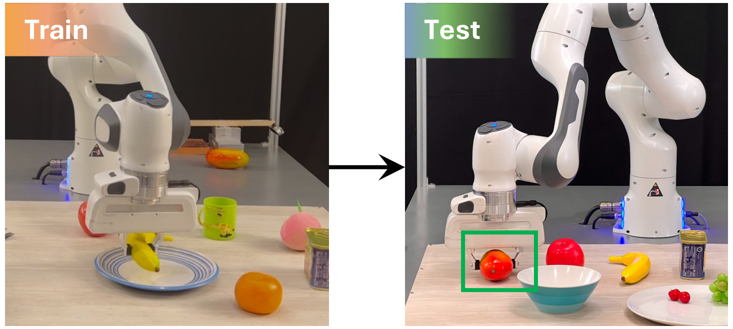
Set up a table for my breakfast.



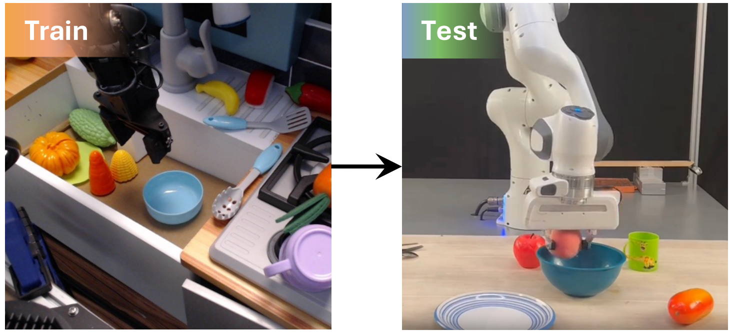
Generalization to Novel States



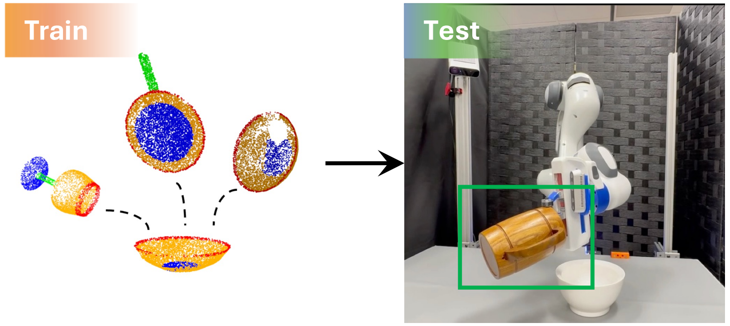
Generalization to Novel Words



Generalization to Novel Embodiments



Generalization to Novel Categories



By composing part-part interactions,
we build systems that can generalize to unseen object categories

Composable Part-Based Manipulation. Liu, *Mao*, Hsu, Hermans, Garg, Wu. CoRL 2023.

Compositional Abstractions Enable Generalization

Principle: Compositional abstractions for

- *states* (objects, relations, and sparse transition models), and
- *actions and plans* (hierarchical compositions and decompositions)

enable data-efficient learning, faster planning, and better generalization

We showed how to build in search algorithms and representational structures for learning

What planning problems can a relational neural network solve? *Mao*, Lozano-Perez, Tenenbaum, Kaelbling. NeurIPS 2023.

Compositional Abstractions Enable Generalization

Principle: Compositional abstractions for

- *states* (objects, relations, and sparse transition models), and
- *actions and plans* (hierarchical compositions and decompositions)

enable data-efficient learning, faster planning, and better generalization

We showed how to build in search algorithms and representational structures for learning

Of course, we can further relax the amount of built-in structures

Today, we give ideas about and constraints on the kinds of network models that could possible be used to learn the computations we need

Neural Logic Machines. Dong*, *Mao**, Lin, Wang, Li, Zhou. ICLR 2019.
Sparse and Local Hypergraph Reasoning Networks. Xiao, Kaelbling, Wu, *Mao*. LOG 2023.
What planning problems can a relational neural network solve? *Mao*, Lozano-Perez, Tenenbaum, Kaelbling. NeurIPS 2023.

Connections to Human Cognition

Broader principle : Concepts as the building block of compositional thoughts, formed based on representational structures over objects, space, physics, numbers, and agents

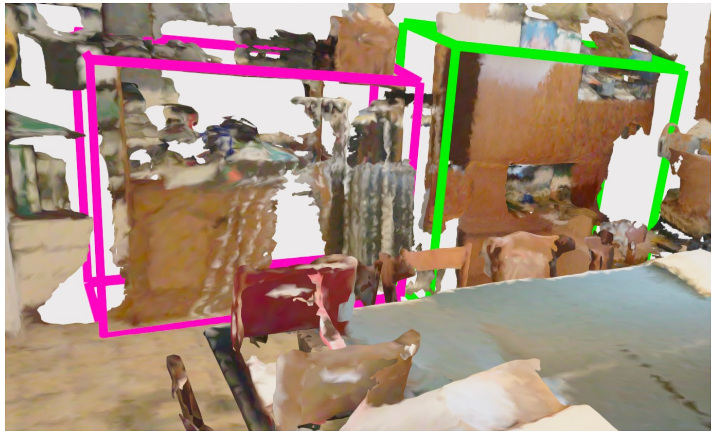
"Core Knowledge" in developmental psychology

A small set of the concepts are built-in (e.g., *contact*); the rest are learned language

Concept	Symbolic Programs		Neural Networks
orange	$\lambda x.\ filter(x, \texttt{orange})$		ORANGE
right	$\lambda x \lambda y.\ relate(x, y, \texttt{right})$		RIGHT
place	$\lambda x \lambda y.$ *precondition:*	$holding(x)$	PLACE
	postcondition:	$on(x, y)$	
	controller:	$action(x, y, \texttt{place})$	

Broader principle : Concepts as the building block of compositional thoughts, formed based on representational structures over objects, space, physics, numbers, and agents

Reasoning about Objects



Q: Is the **dresser** left of the **cabinet**?

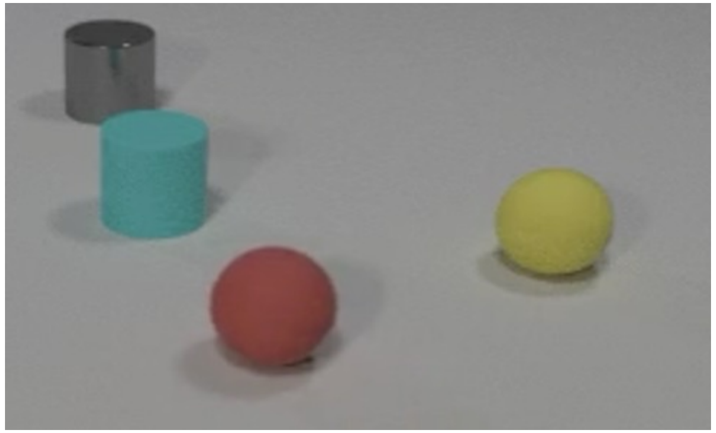
Mao et al. 2019. Hsu*, *Mao** et al. 2023.

Reasoning about Abstractions

Q: Who is wining this **tic-tac-toe** game?

Hsu et al. 2024.

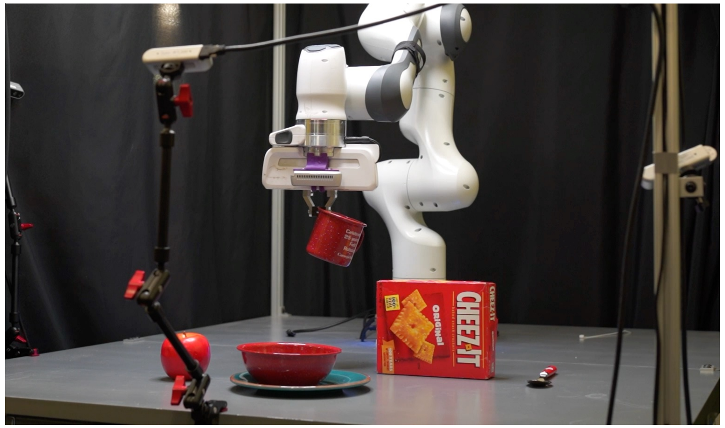
Causality in Humans



Q: Which **ball** caused the collision?

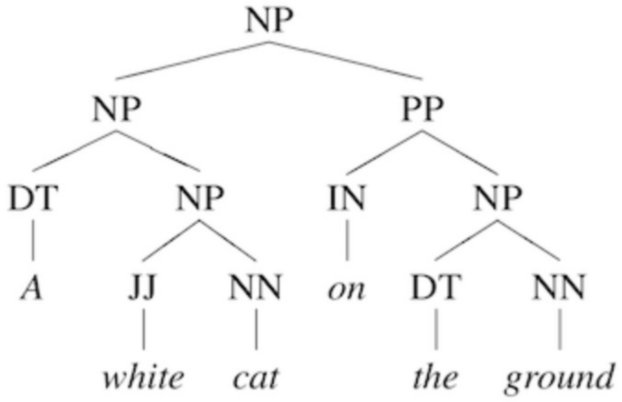
*Mao**, Yang* et al 2023.

Robotic Manipulation



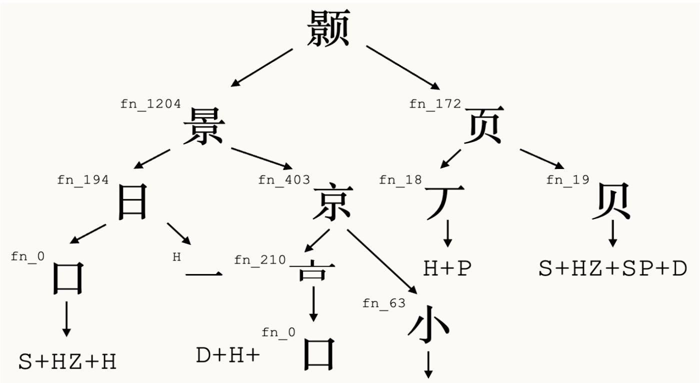
Q: Put the **mug** next to the **Plate**.

Grounded Syntax Learning



Shi*, *Mao** et al. 2019. *Mao* et al 2021.

Compositionality in Human Writing Systems



Jiang et al 2024.

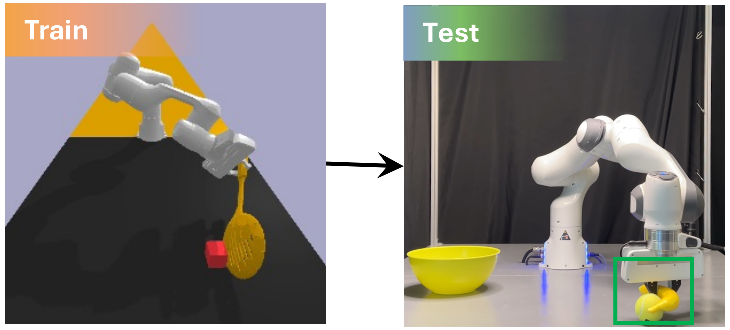
Compositional Abstractions Enable Generalization

Principles: Compositional abstractions for

- *states* (objects, relations, and sparse transition models), and
- *actions and plans* (hierarchical compositions and decompositions)

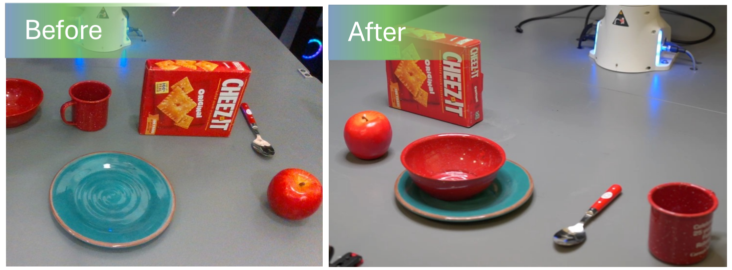
enable data-efficient learning, faster planning, and better generalization

Generalization to Novel Objects

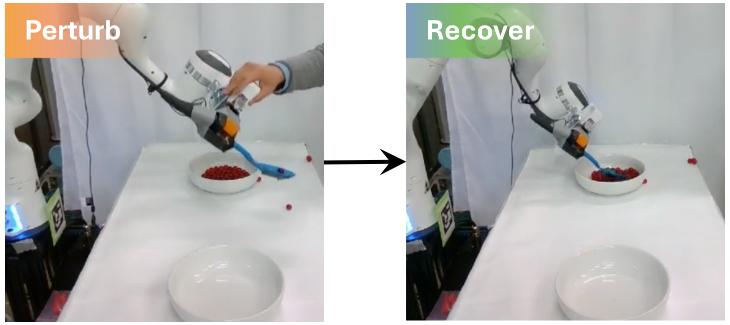


Generalization to Novel Goals

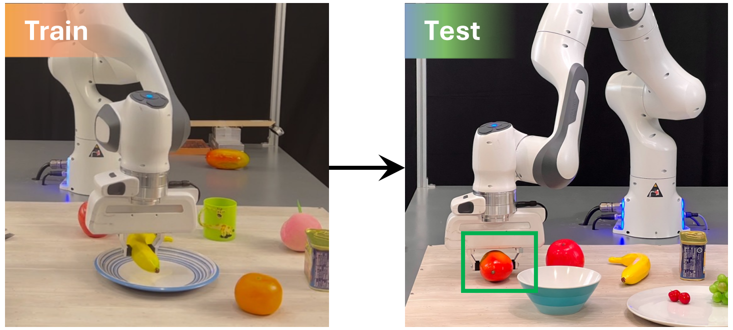
Set up a table for my breakfast.



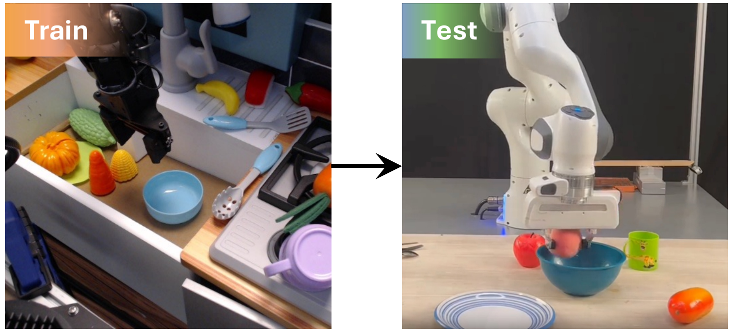
Generalization to Novel States



Generalization to Novel Words



Generalization to Novel Embodiments



Generalization to Novel Categories