Neuro-Symbolic Concepts for Robotic Manipulation

Jiayuan Mao

MIT CSAIL

jiayuanm@mit.edu

Towards Generalist Robots

Goal:

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

Towards Generalist Robots

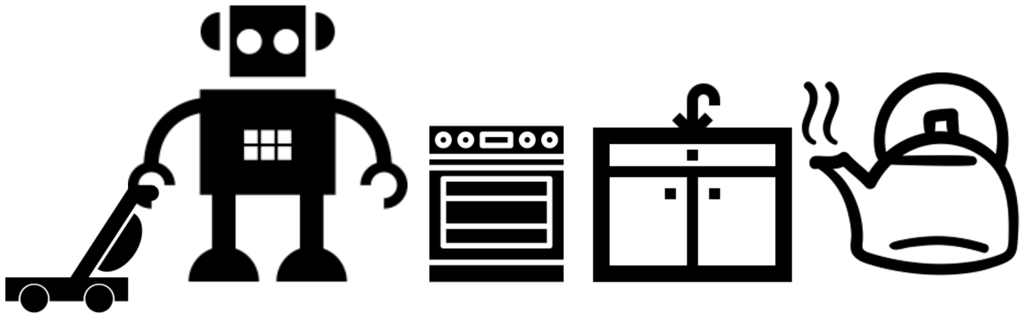
Goal:

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

Towards Generalist Robots

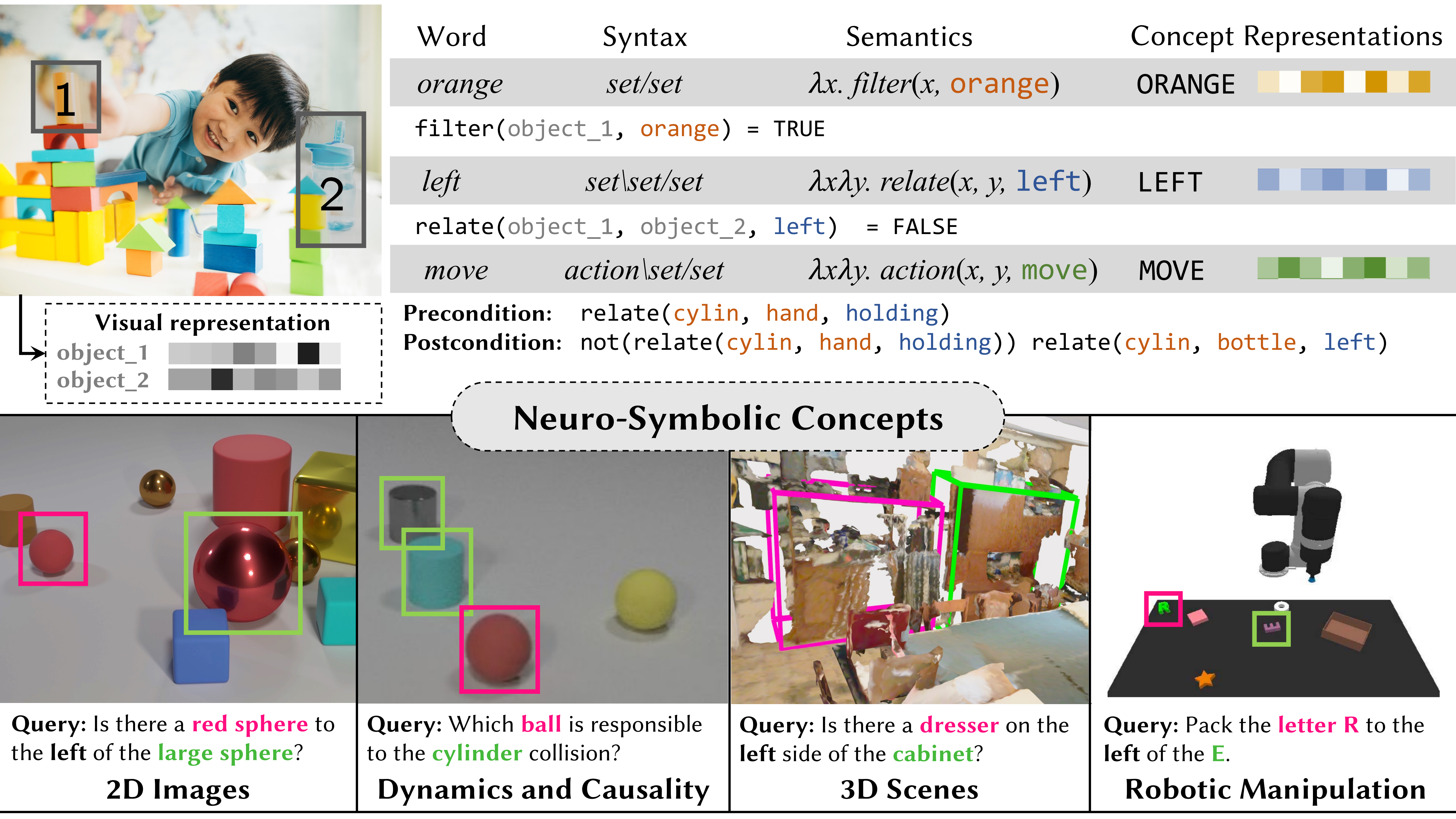
Goal:

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



Learning a single goal-conditioned policy for many tasks can be hard.

A promising direction is to combine model learning, reasoning, planning.



1
2
Word
Syntax
Semantics
Concept Representations
orange
set/set
λx. filter(x, orange)
ORANGE
filter(object_1, orange) = TRUE
left
set\set/set
λxλy. relate(x, y, left)
LEFT
relate(object_1, object_2, left) = FALSE
move
action\set/set
λxλy. action(x, y, move)
MOVE
Visual representation
object_1
object_2
Precondition: relate(cylin, hand, holding)
Postcondition: not(relate(cylin, hand, holding)) relate(cylin, bottle, left)
Neuro-Symbolic Concepts
Query: Is there a red sphere to the left of the large sphere?
2D Images
Query: Which ball is responsible to the cylinder collision?
Dynamics and Causality
Query: Is there a dresser on the left side of the cabinet?
3D Scenes
Query: Pack the letter R to the left of the E.
Robotic Manipulation

Word	Syntax	Semantics	Concept Representations
orange	*set/set*	$\lambda x.\ filter(x, \texttt{orange})$	ORANGE
`filter(object_1, orange) = TRUE`			
left	*set\set/set*	$\lambda x \lambda y.\ relate(x, y, \texttt{left})$	LEFT
`relate(object_1, object_2, left)  = FALSE`			
move	*action\set/set*	$\lambda x \lambda y.\ action(x, y, \texttt{move})$	MOVE

Visual representation
object_1
object_2

Precondition: `relate(cylin, hand, holding)`
Postcondition: `not(relate(cylin, hand, holding)) relate(cylin, bottle, left)`

Neuro-Symbolic Concepts

Key Aspects of Representations:

Compositional.

Can be learned from little data.

Can support fast reasoning and planning.

Task and Motion Planning for Robotics

Instruction: Put all food items in the fridge.
Initial State: in(Cabbage, Pot),
on(Potato, Table), ...

Task Plan:

① *Open the left fridge door* ② *Remove the pot lid* ③ *Move the cabbage from pot to fridge* ④ *Move potato to fridge*

Task and Motion Planning for Robotics

Instruction: Put all food items in the fridge.

Initial State: in(Cabbage, Pot),
on(Potato, Table), ...

Photo Credit: Yang et al.
PIGINet: A Transformer-based Plan Feasibility Predictor for Robotic Rearrangement in Geometrically Complex Environments.
RSS 2023 Poster on Wed. #29

Refine
+
Feedback

Task Plan:

1 *Open the left fridge door* 2 *Remove the pot lid* 3 *Move the cabbage from pot to fridge* 4 *Move potato to fridge*

Motion Plan:

Basic Elements in Task and Motion Planning

- Basic predicates.

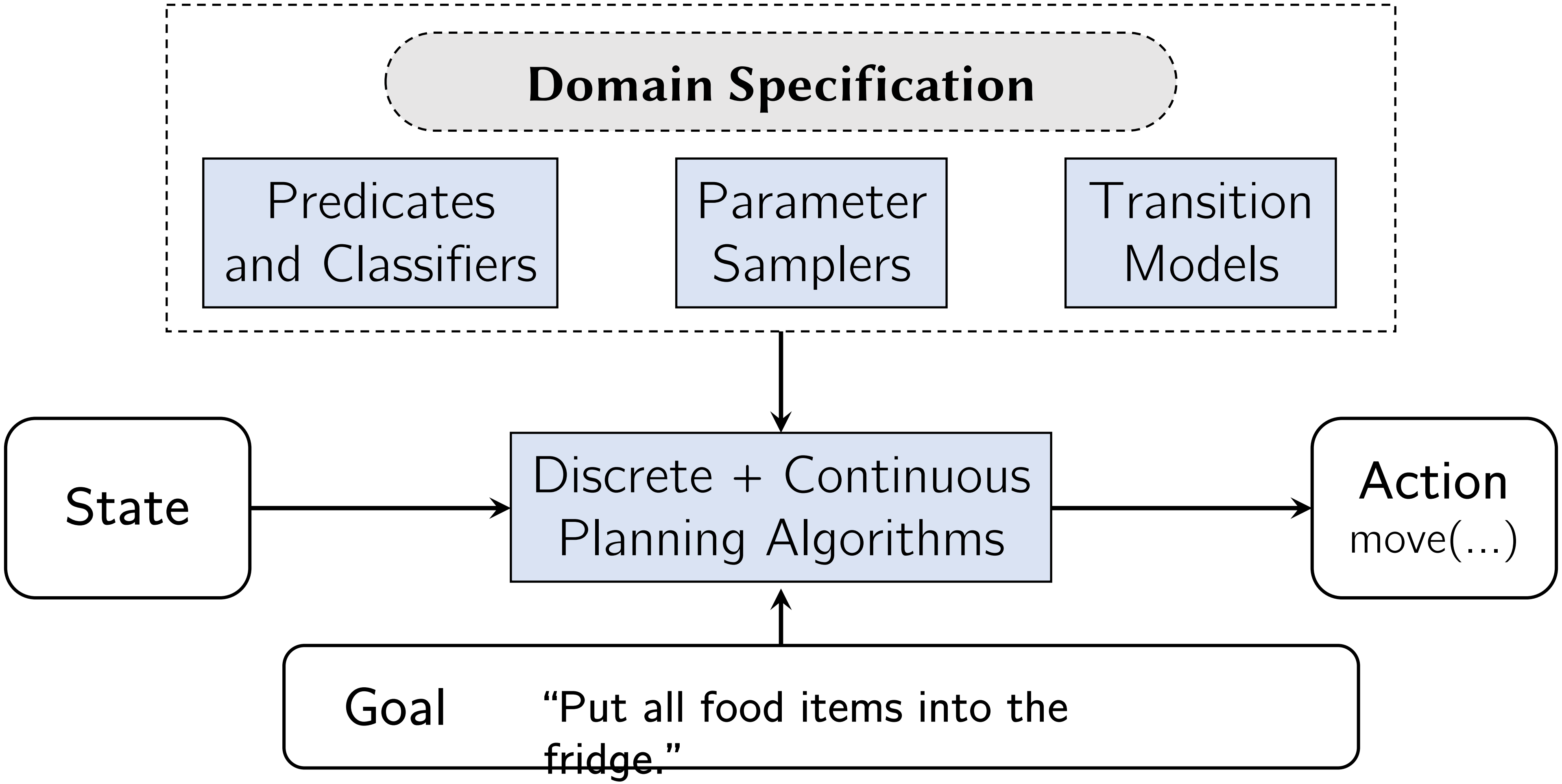
```
predicate is-food(o: object)
  classifier: ...
predicate in(o: object, r: receptacle)
  classifier: ...
```

- Basic operators: the transition models, samplers, controllers.

```
action pick-place(o: object, p1: pose, p2: pose, g: grasp, t: traj)
  pre: obj-at(p1), valid-trajectory(t, g, p1, p2)
  eff: obj-at(p2)
  samplers: grasp samplers, trajectory samplers (e.g., RRT)
  controller: ...
```

- These operators usually only models the object in contact.

Basic Elements in Task and Motion Planning



Many Difficult Choices Must Still Be Made

However, implementing such systems can be difficult:

① *Open the left fridge door*

- Where to grasp?
- How to move?
- How far?
- ...

② *Remove the pot lid*

- Where to grasp?
- Where to put?
- Any side-effects? (e.g., hot item?)
- ...

③ *Move the cabbage from pot to fridge*

- Where to grasp?
- Where to place to be stable?
- Enough space for later items?
- Enough space for robot hand?
- Maybe need non-prehensile manipulation?
- What will happen to the cabbage?
- ...

④ *Move potato to fridge*

- Where to grasp?
- Where to place to be ...
- How to organize the fridge?
- ...

Again, we want to solve these problems for many objects, for many tasks, across many environments.

Learning to Tackle These Challenges

- Task and motion planning is a general framework.
- Manually programming everything can be challenging, especially when dealing with perception and continuous parameters.
- We are interested in learning to tackle these challenges, in particular, learning neuro-symbolic representations for objects, relations, and actions.

How Should We Combine Learning and Planning

- We start from some understanding of a “base operators.”
- They are general, and hard to learn from little data.

```
action pick-place(o: object, p1: pose, p2: pose, t: traj): ...
```

- Then we can “specialize” these operators.

```
action paint(o: object, t: object, g: grasp, t: trajectory)
action use-hammer(o: object, r: object, g: grasp, t: trajectory)
action pouring(o: object, r: object, g: grasp, t: trajectory)
action open-cabinet-door(o: door, g: grasp, t: trajectory)
action sort-in-fridge(o: object, r: fridge, g: grasp, t: trajectory)
action place-in-pot(o: object, r: pot, g: grasp, t: trajectory)
......
```

The Objective of Learning

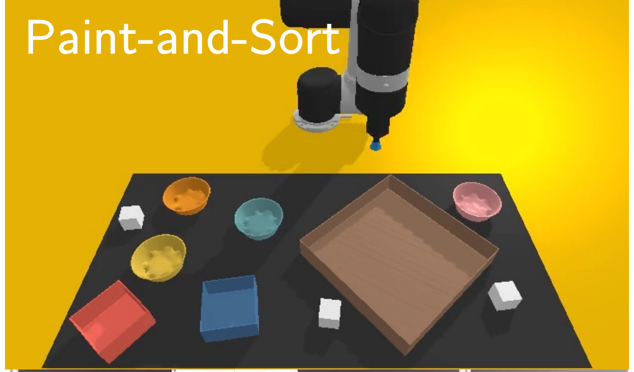
- Learning additional "transition models"

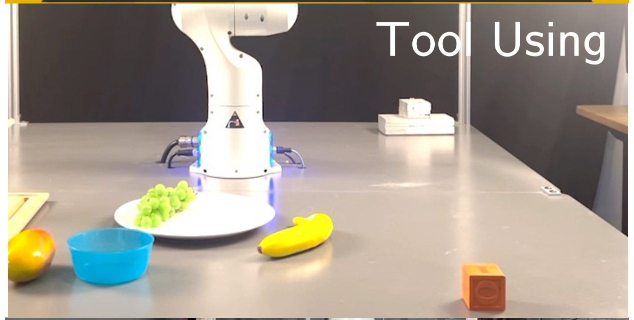
```
action paint(o: object, t: object, g: grasp, t: trajectory)
  eff: if is-green-painter(t) and is-clean(o) then is-green(o)

action use-hook(o: object, r: object, g: grasp, t: trajectory)
  eff: object-at(r, new-pose(...))

action pouring(o: object, r: object, g: grasp, t: trajectory)
  eff: if has-water(o) then has-water(r)
```

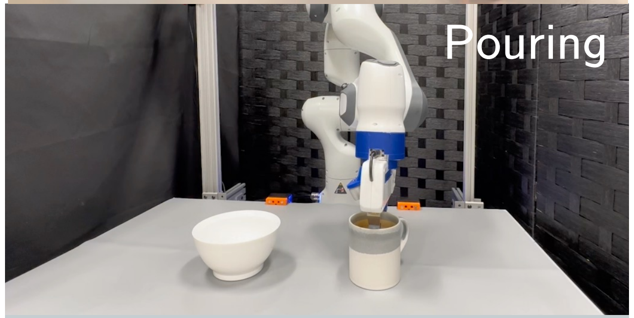
- Learning samplers

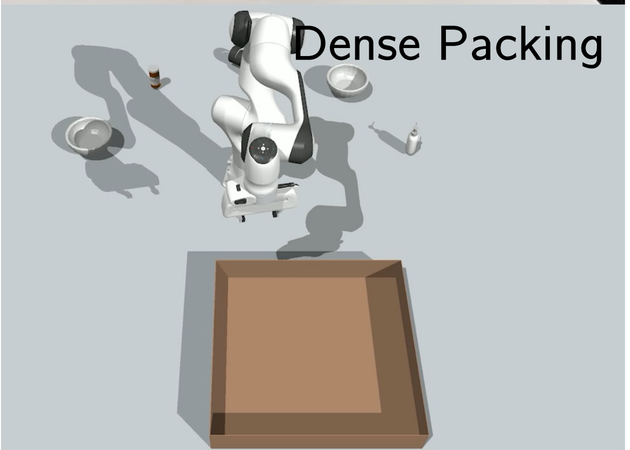
```
action use-hook(o: object, r: object, g: grasp, t: trajectory)
  sampler: t ~ trajectory-s(o, r, g)

action sort-in-fridge(o: object, r: fridge, g: grasp, t: trajectory)
  sampler: pose_of_o ~ placement-s(o, r)
```

- In many cases you don't need to learn new controllers.









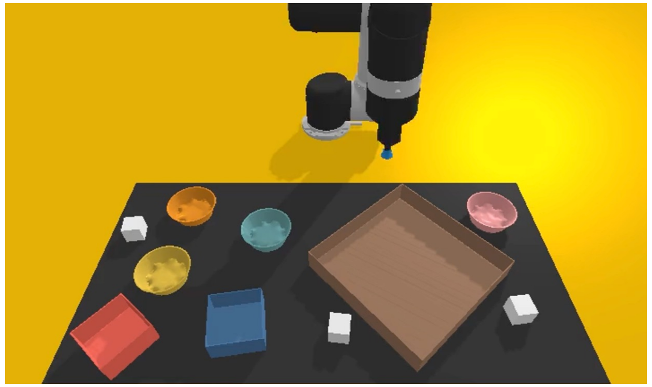
PDSketch

Integrated Domain Programming, Learning, and Planning

Jiayuan Mao, Tomas Lozano-Perez, Joshua B. Tenenbaum, Leslie Pack Kaelbling. NeurIPS 2022.

- Manually programming everything can be challenging.
- However, humans are good at describing qualitative structural and causal aspects of a domain.
- ML methods are good at learning detailed parametric models.
- We get great generalization and from very little training.

The Objective of Learning



State Space:

```
s.agent = (x, y, yaw)
s.objects[i] = (x, y, image)
```

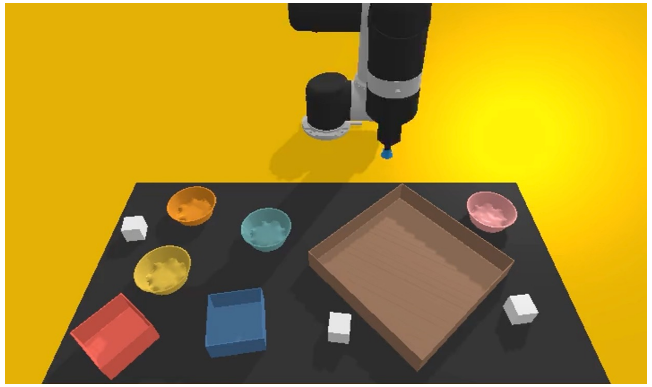
Predicates

```
next_to(agent, object)
is_box(object)
......
```

Transition Model

```
def pick_place(s, o): ...
```

The Objective of Learning



State Space:

```
s.agent = (x, y, yaw)
s.objects[i] = (x, y, image)
```

Predicates

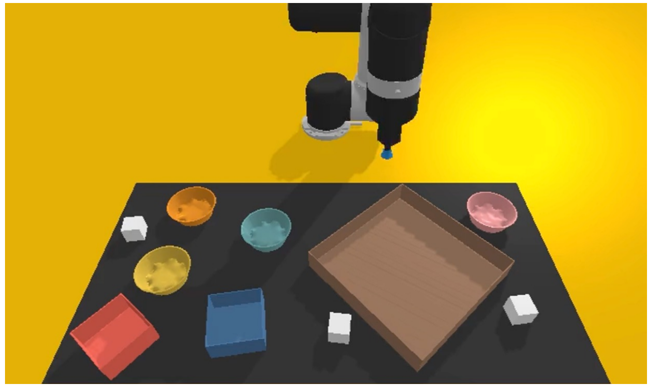
```
next_to(agent, object)
is_box(object)
......
```

Transition Model

```
def pick_place(s, o): ...
```

Target 1: Classifiers for predicates
Learning to classify objects and relations.
Samplers for certain relations (e.g., “in”)

The Objective of Learning



State Space:

```
s.agent = (x, y, yaw)
s.objects[i] = (x, y, image)
```

Predicates

```
next_to(agent, object)
is_box(object)
......
```

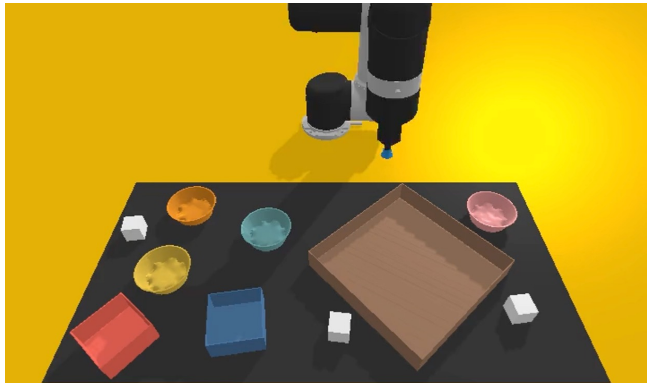
Transition Model

```
def pick_place(s, o): ...
```

Target 1: Classifiers for predicates
Learning to classify objects and relations.
Samplers for certain relations (e.g., "in")

Target 2: Details in the transition model.
"How objects will be painted?"

Combining Human “Sketch” and Learning



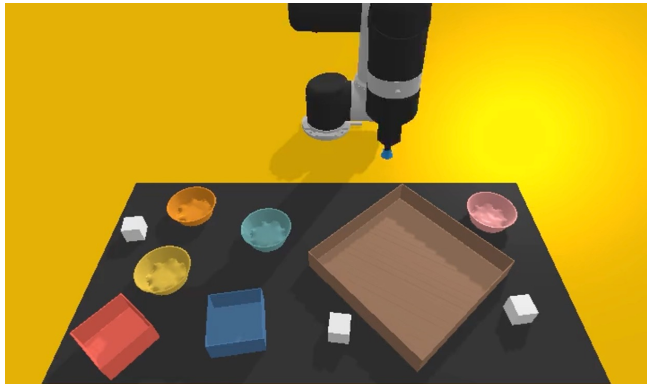
Goal: Paint all blocks red and put them into the box.

```
action pick-place(o: object, p1: pose, p2: pose, t: traj)
  pre: obj-at(p1), valid-trajectory(t, p1, p2)
  eff: obj-at(p2)
```

Specializes

```
action paint(o: object, p1: pose, p2: pose, t: traj)
  pre: obj-at(p1), valid-trajectory(t, p1, p2)
  eff: obj-at(p2)
       forall b:
         if in(o, b) and ?f(o, b):
           o.color = ?g(o, b)
```

Combining Human "Sketch" and Learning



Agent^t
(x, y, r)

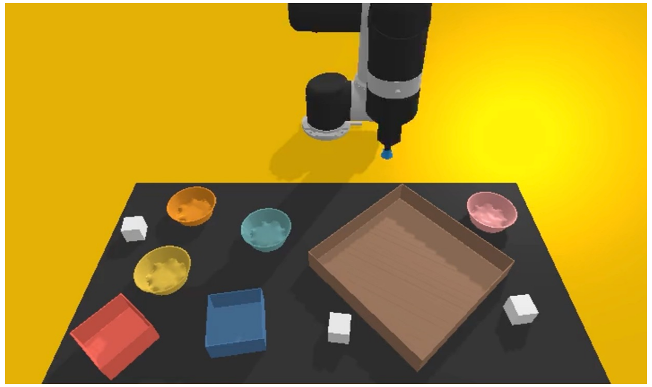
Objects^t[0]
(x=..., y=...,
image=)

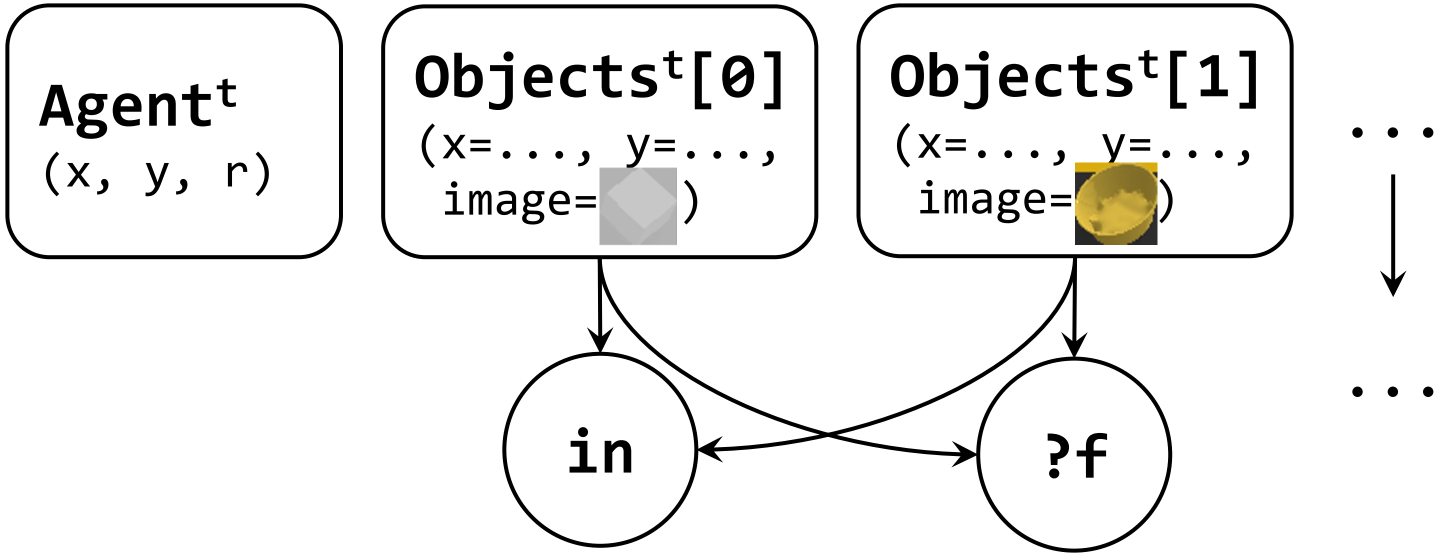
Objects^t[1]
(x=..., y=...,
image=)

...

```
action paint(o, p1, p2, t)
 forall b:
   if in(o, b) and ?f(o, b):
     o.color = ?g(o, b)
```

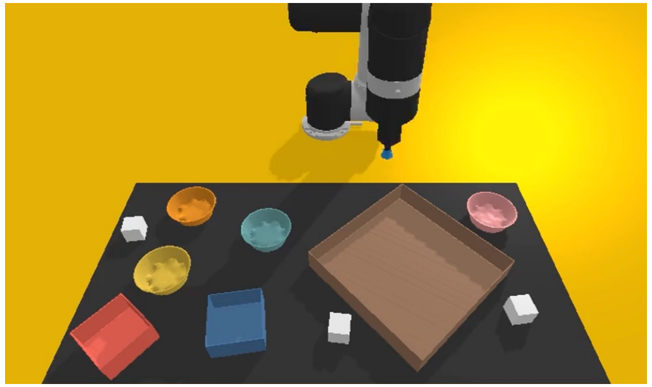
Combining Human "Sketch" and Learning

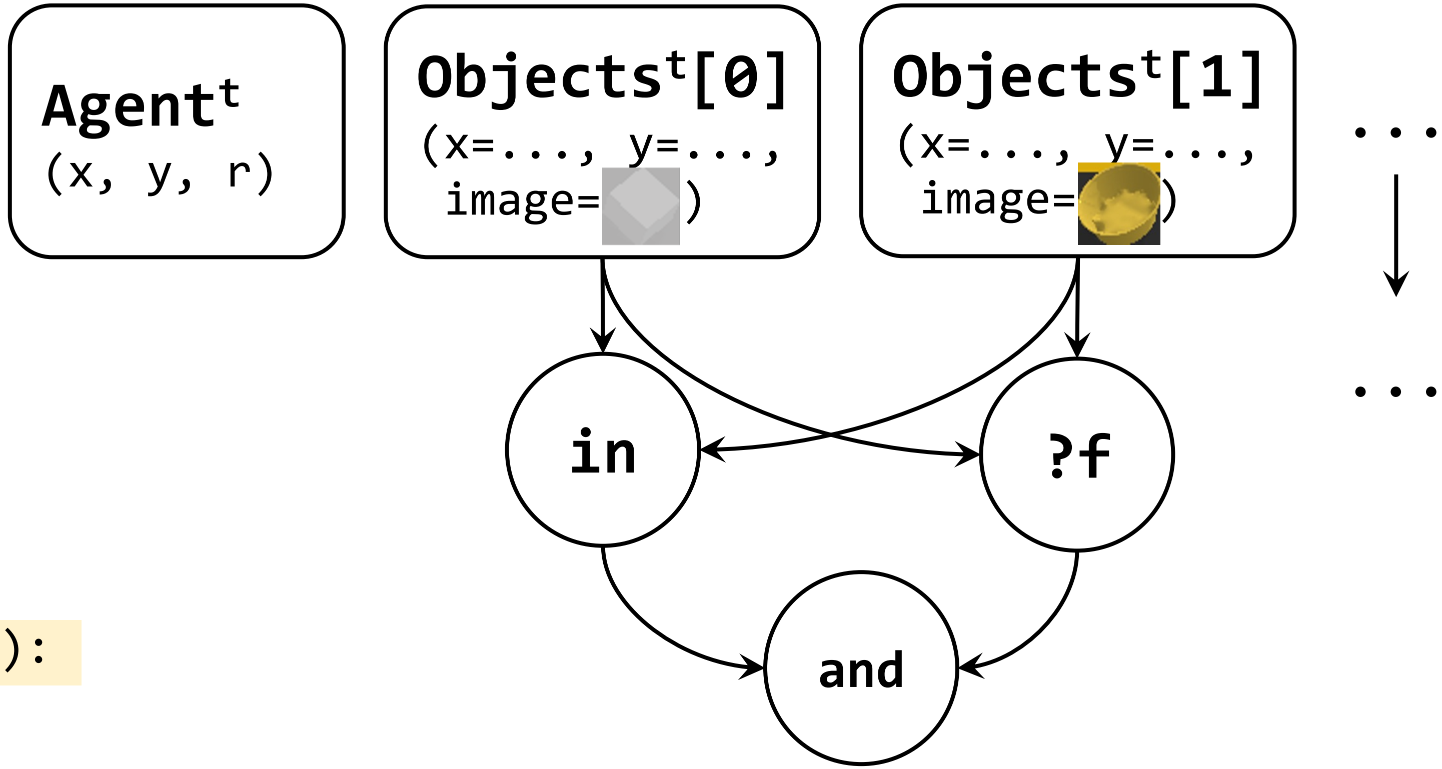





```
action paint(o, p1, p2, t)
 forall b:
   if in(o, b) and ?f(o, b):
     o.color = ?g(o, b)
```

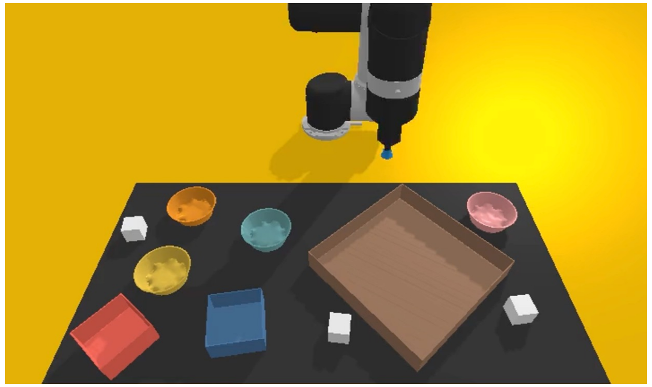
Combining Human "Sketch" and Learning





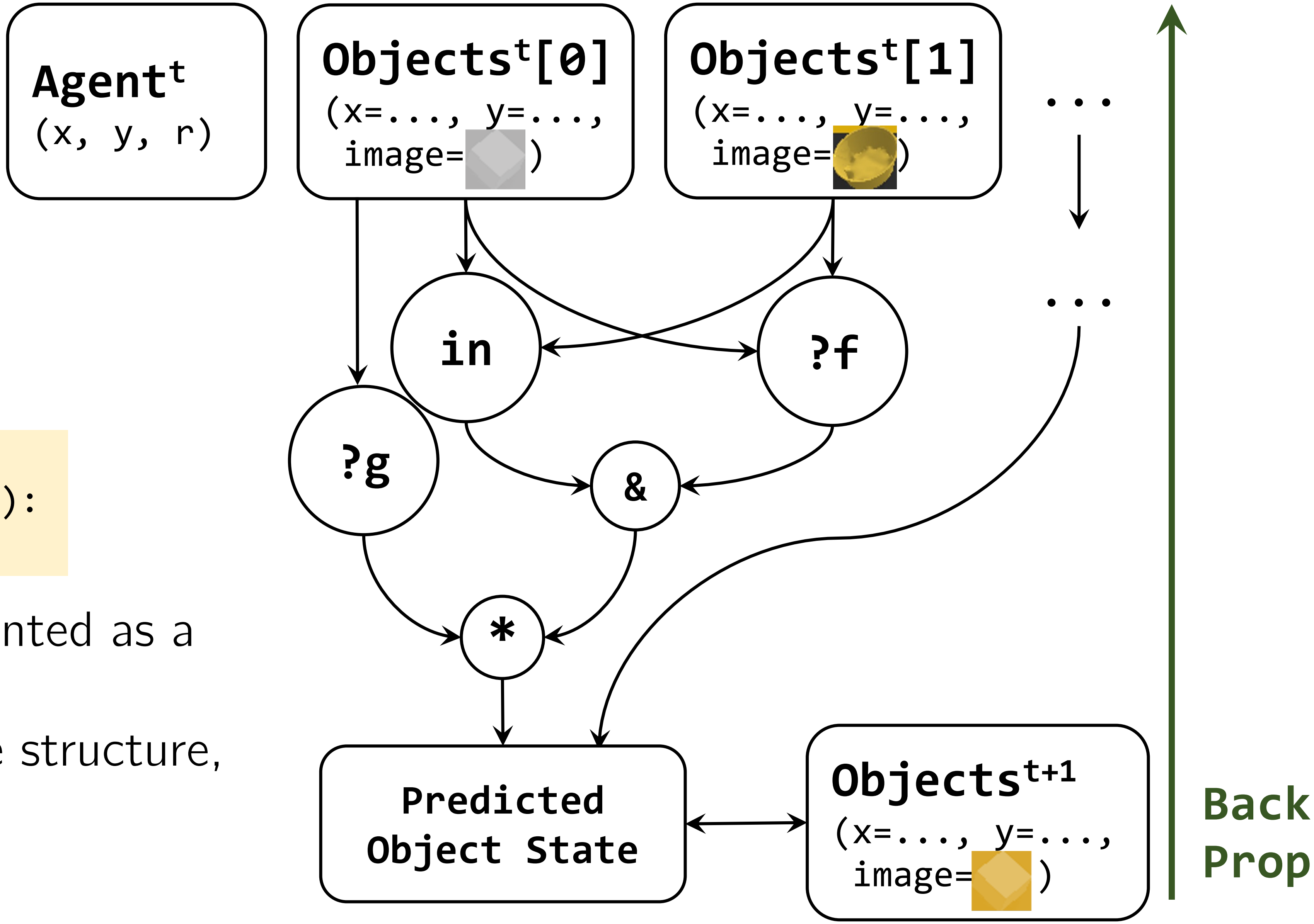

```
action paint(o, p1, p2, t)
 forall b:
   if in(o, b) and ?f(o, b):
     o.color = ?g(o, b)
```

Combining Human “Sketch” and Learning

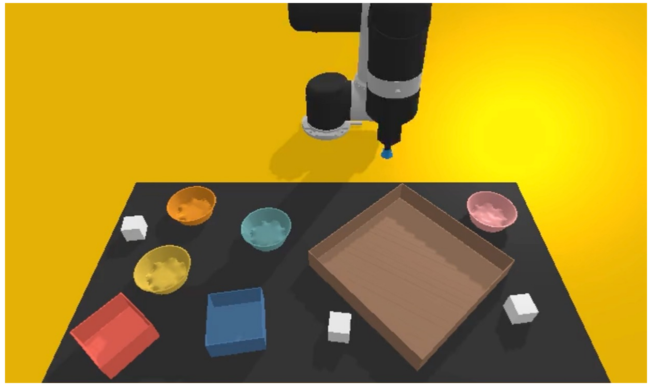


```
action paint(o, p1, p2, t)
  forall b:
    if in(o, b) and ?f(o, b):
      o.color = ?g(o, b)
```

Each **??** can be implemented as a neural network module.
Humans “sketch” out the structure, and ML fills in the gaps.



Learning Continuous Parameters



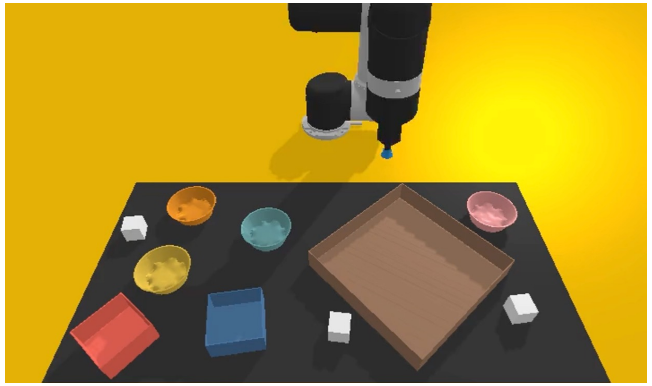
Goal: Paint all blocks red and put them into the box.

```
action pick-place(o: object, p1: pose, p2: pose, t: traj)
  pre: obj-at(p1), valid-trajectory(t, p1, p2)
  eff: obj-at(p2)
```

Specializes

```
action place-in(o: object, r: receptacle)
  p1 = s.pose[o]
  sample p2 ~ sample_in(o, r)
  sample t  ~ sample_trajectory(o, p1, p2)

  pick-place(o, p1, p2, t)
```

Learning Continuous Parameters



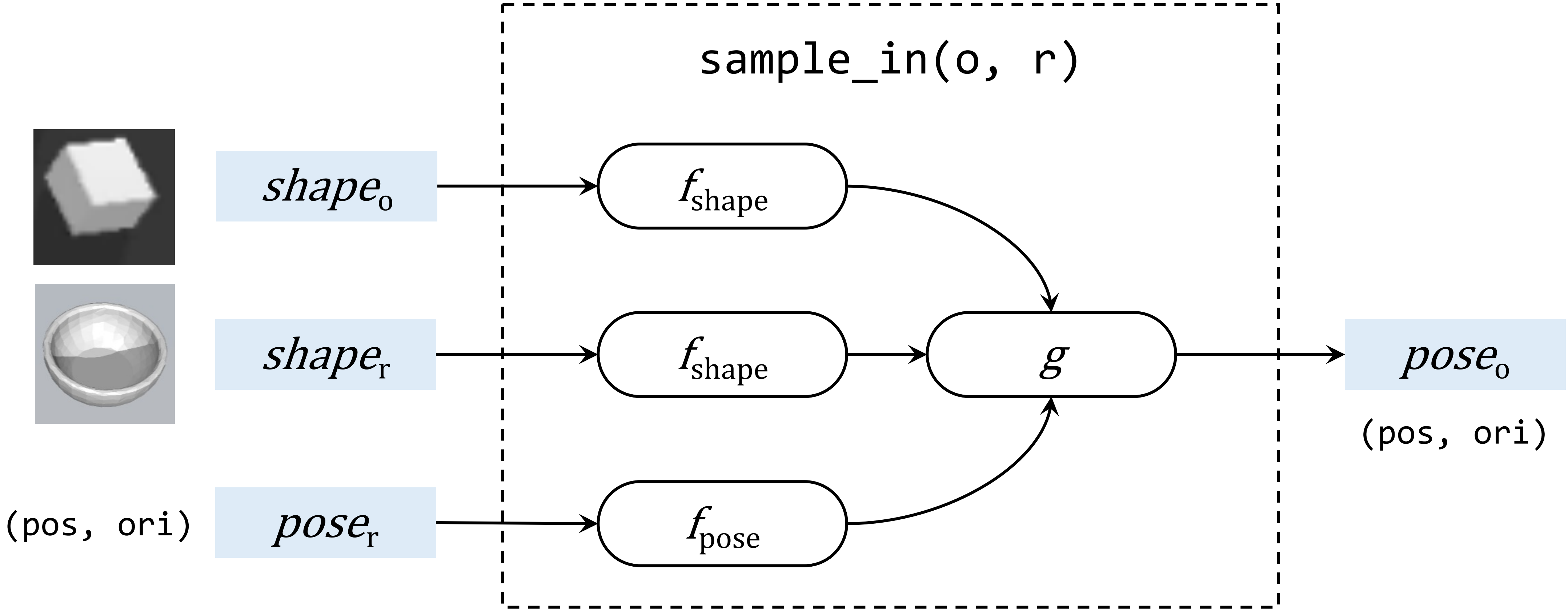
Goal: Paint all blocks red and put them into the box.

```
action pick-place(o: object, p1: pose, p2: pose, t: traj)
  pre: obj-at(p1), valid-trajectory(t, p1, p2)
  eff: obj-at(p2)
```

Specializes

```
action place-in(o: object, r: receptacle)
  p1 = s.pose[o]
  sample p2 ~ sample_in(o, r)
  sample t  ~ sample_trajectory(o, p1, p2) # RRT

  pick-place(o, p1, p2, t)
```

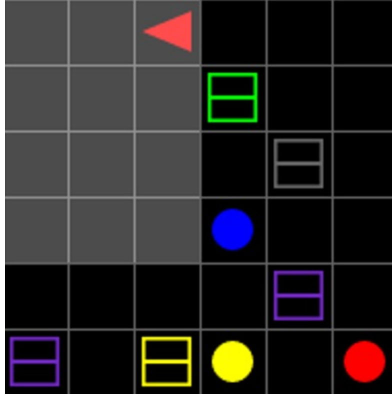
Learning Continuous Parameters



Can be implemented by any specific generative models (e.g., Diffusion).

In PDSketch, models are learned from expert demonstrations.

Learning and Planning Efficiency



PDS-Rob

Full robot movement models.
Need to learn object classifiers.

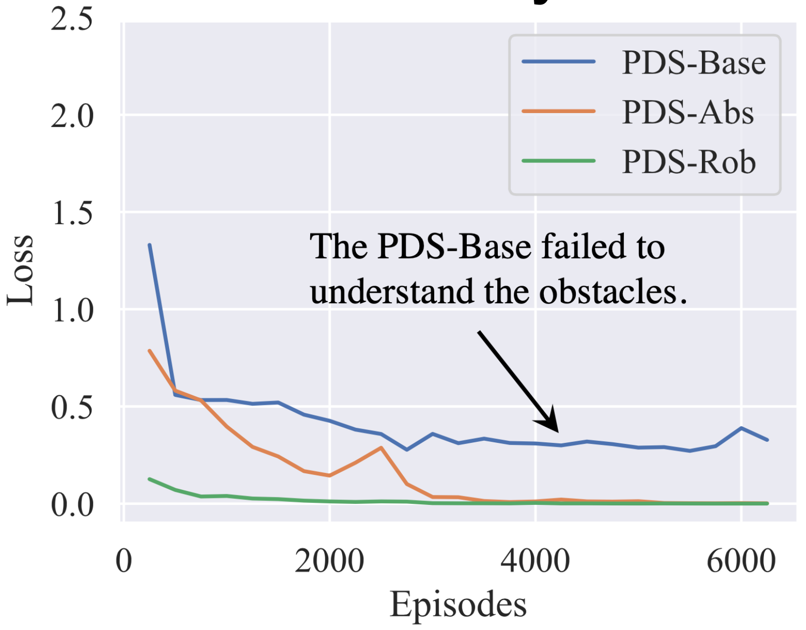
PDS-Abs

Abstract robot models.
(With ??)

PDS-Base

GNNs.
(Weakest prior)

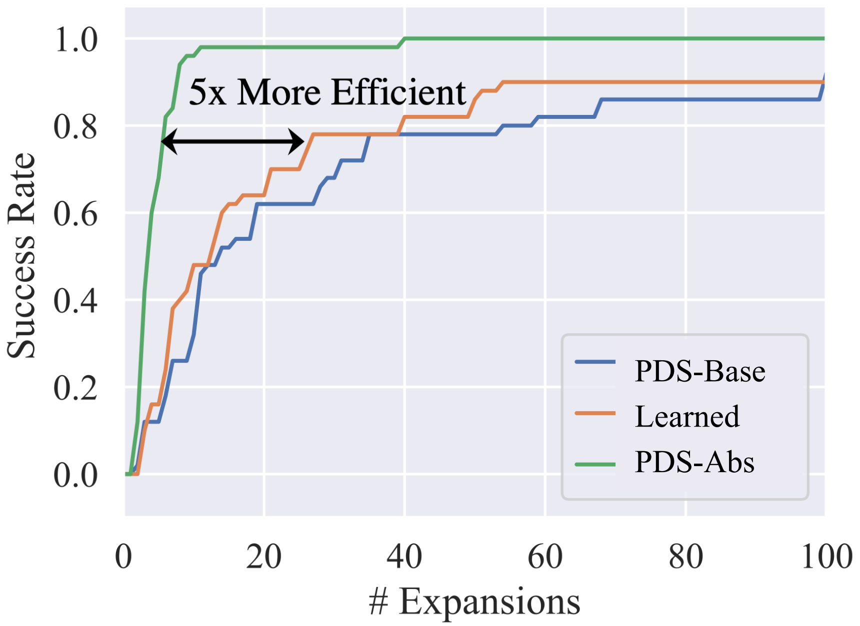
Data Efficiency



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

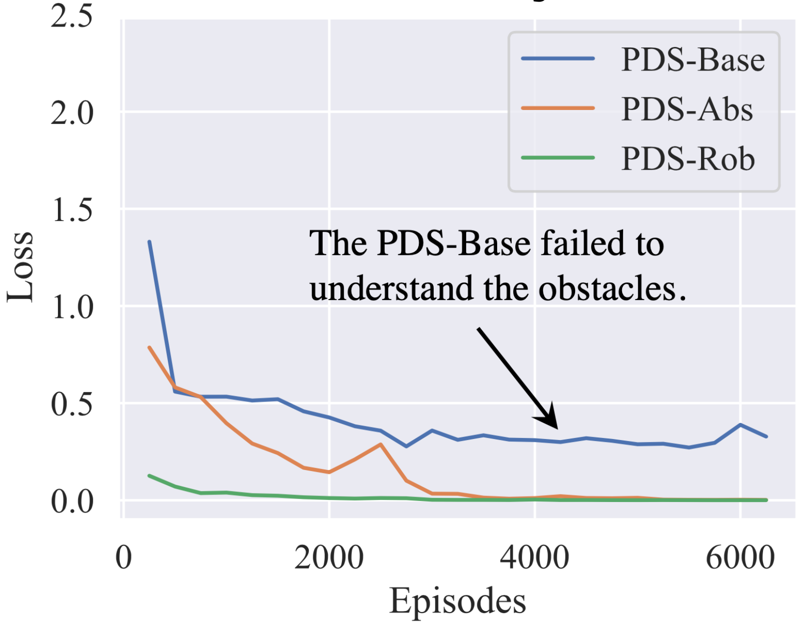
Planning Efficiency



Environment from: Chevalier-Boisvert et al. 2019.

Learning and Planning Efficiency

Data Efficiency



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

Learning and Planning 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

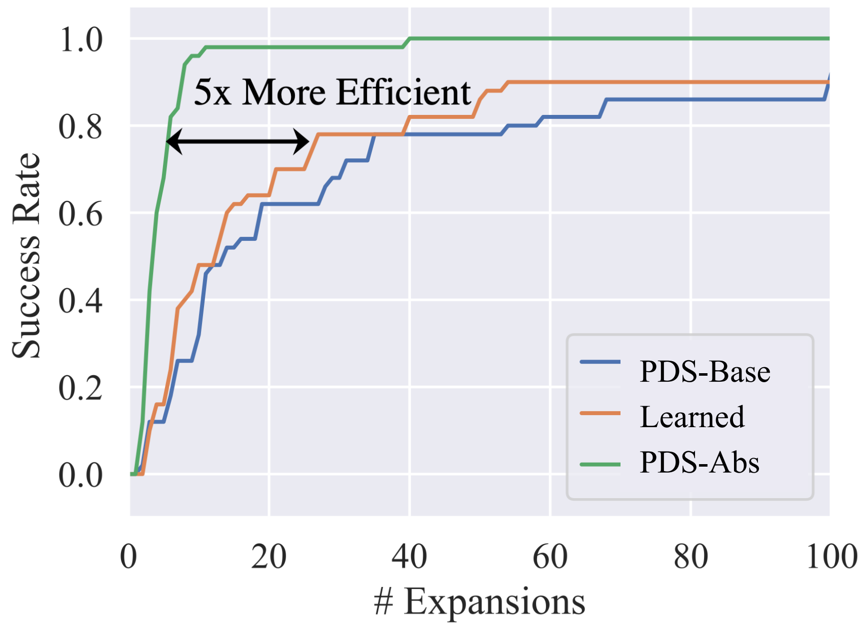
The performance in model learning also translates to *better performance*.

Learning and Planning Efficiency

- Suppose an action has two preconditions.
- E.g., to paint an object, it should be both clean and dry.
- Solve two planning problems separately, and “add” the costs together.
- Such strategy generalizes to neuro-symbolic models of the transition models.

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

Planning Efficiency

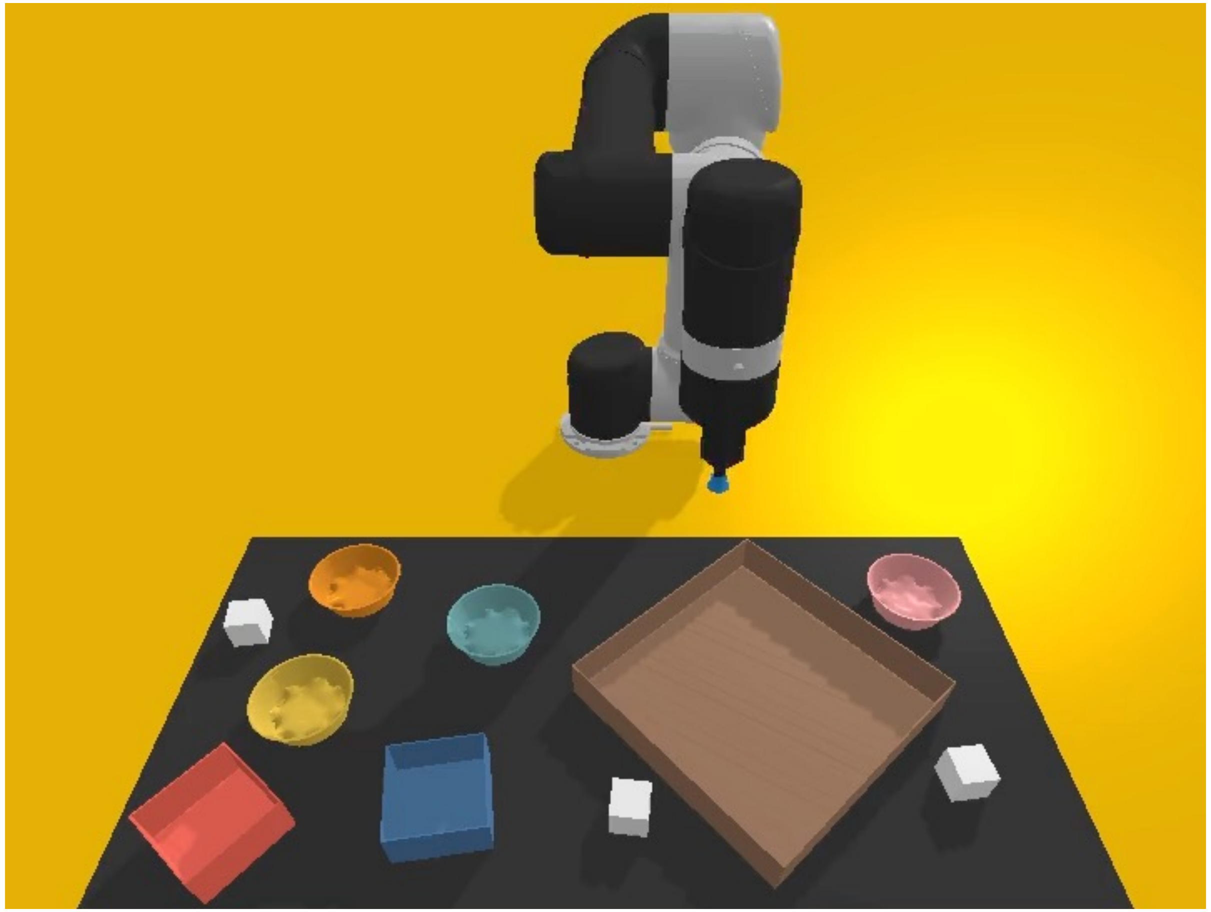


Planning with Learned Models and Samplers

Trained on goals: `∃x.y.`*`color`*`(x)&`*`color`*`(y)&`*`rel`*`(x, y)` Positions, number of objects, colors vary.

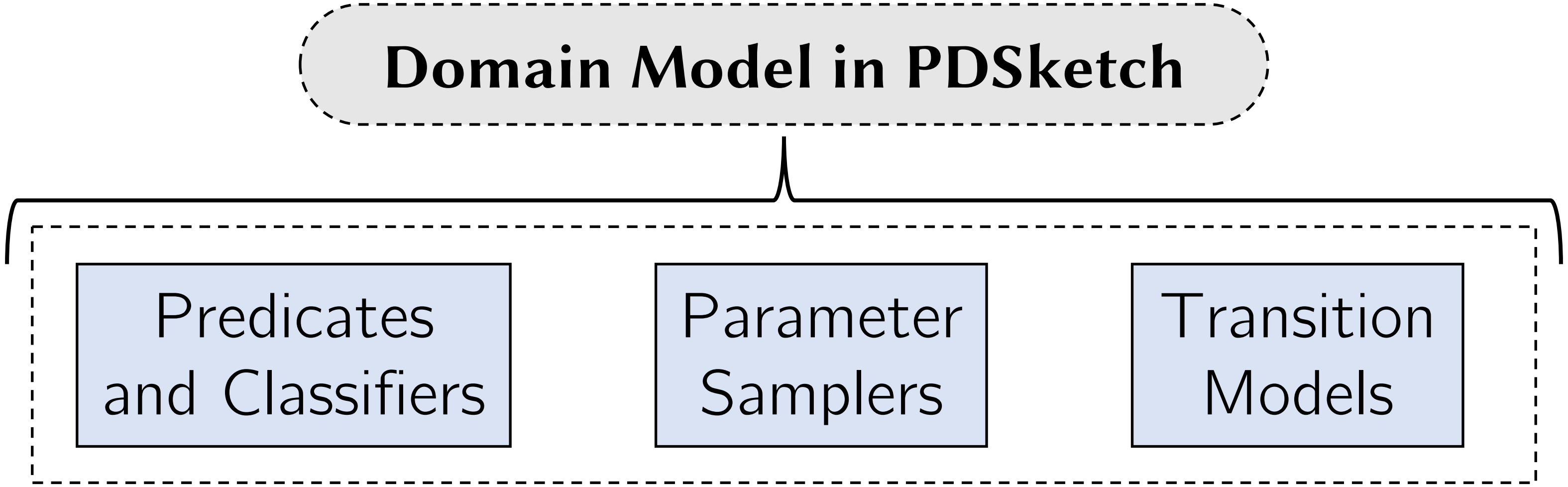
```
∃x.y. purple(x) & yellow(y) &
      inbox(x) & inbox(y) & left-of(x, y)
```

```
∀x. yellow(x) & inbox(x)
```



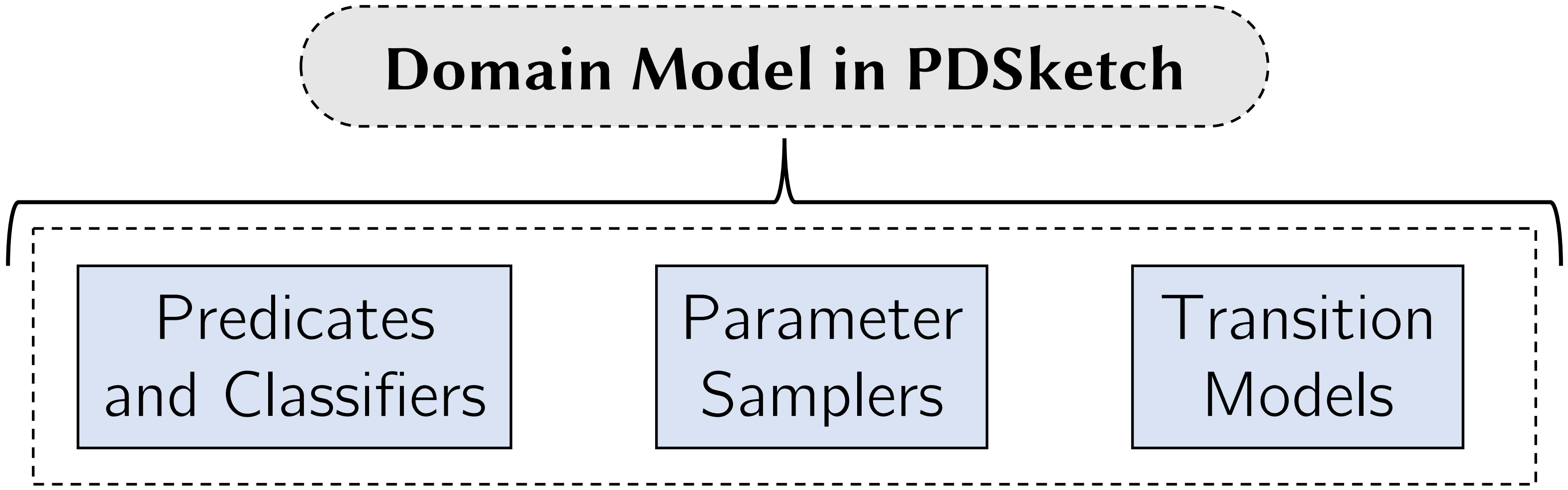
PDSketch

Integrated Domain Programming, Learning, and Planning



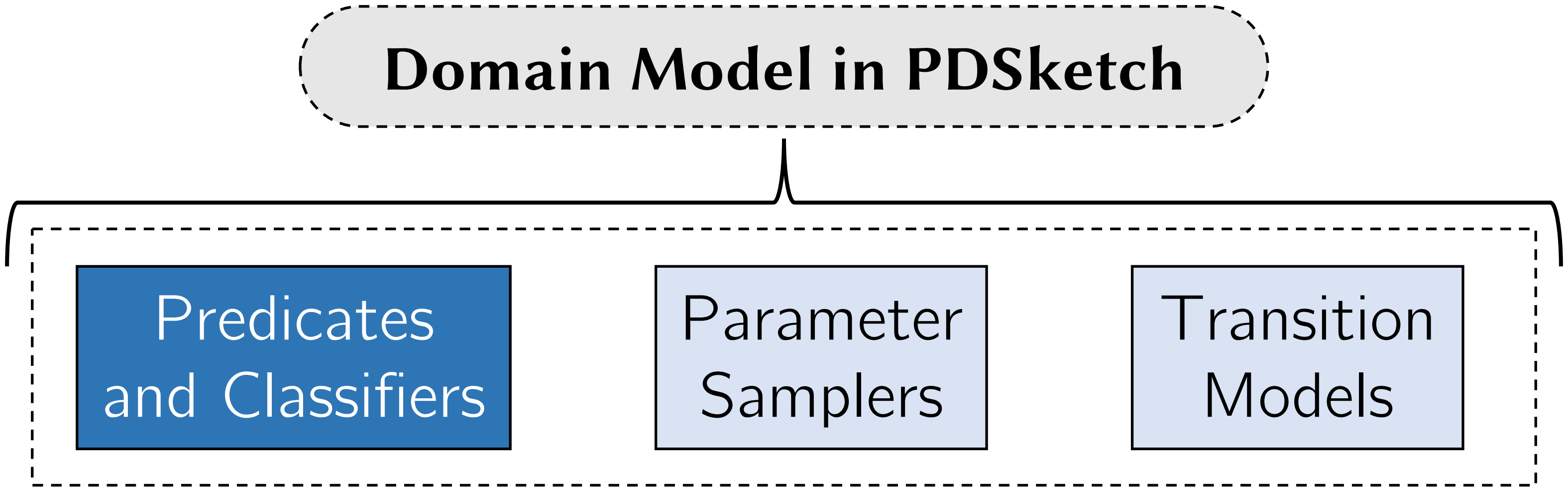
A framework that combines program "sketches" and learning for learning domain models.
It uses *neuro-symbolic representation* to improve data-efficiency in learning.
It leverages symbolic structures of the transition model for faster planning.

Learning Everything from Scratch Is Unscalable

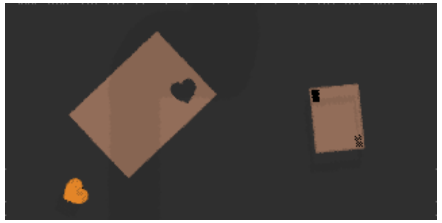


The neuro-symbolic, modularized system enables learning from different data streams.

Leveraging “Foundation Models” for Predicates



Modular Integration with "Foundation Models"



‘put the heart in the hole’

Predicates

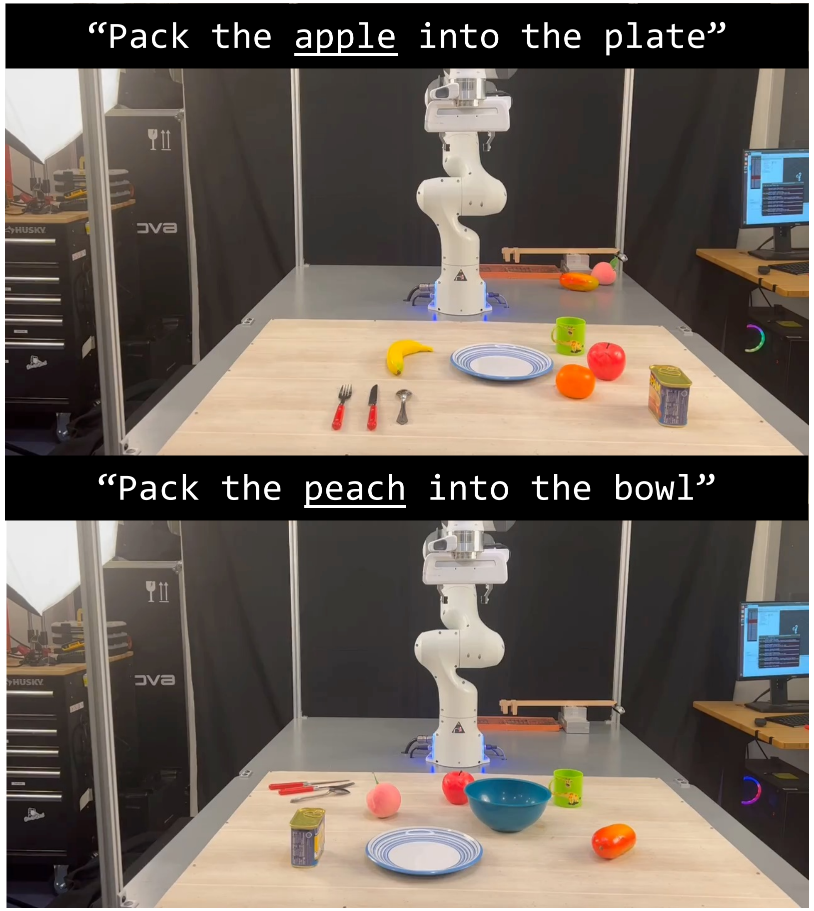
```
in(object, object)
is_heart(object)
is_hole(object)
```

Object Recognition

Directly leverage pretrained CLIP.

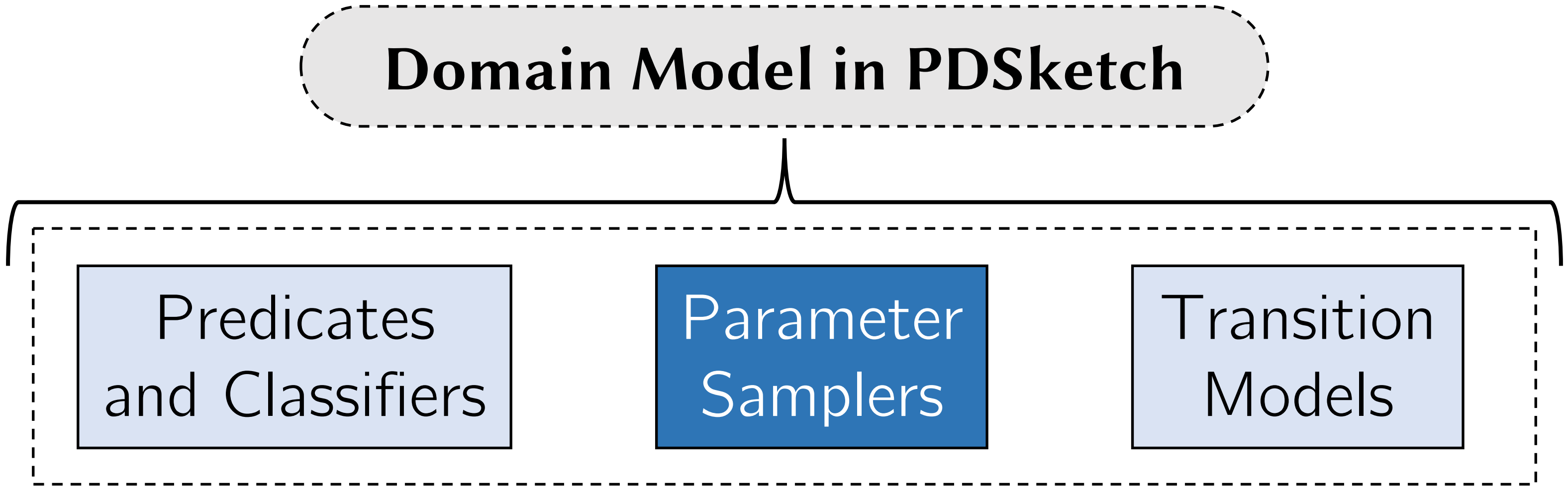
Object Relations

Learn classifiers + samplers.



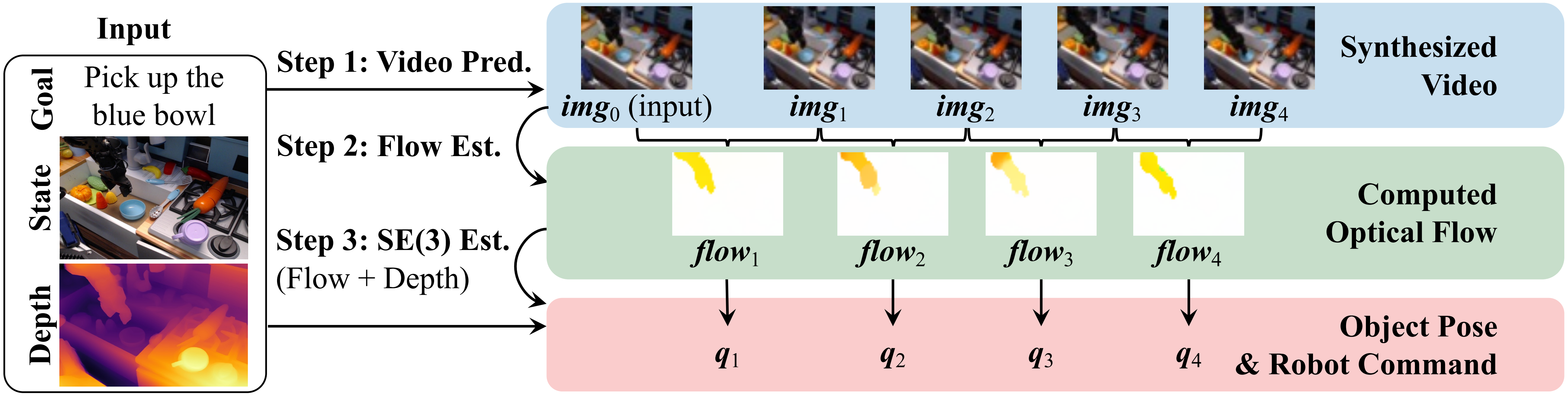
ProgPort: Programmatically Grounded, Compositionally Generalizable Robotic Manipulation
Wang*, **Mao***, Hsu, Zhao, Wu, Gao. In *ICLR* 2023.

Learning Samplers from Videos



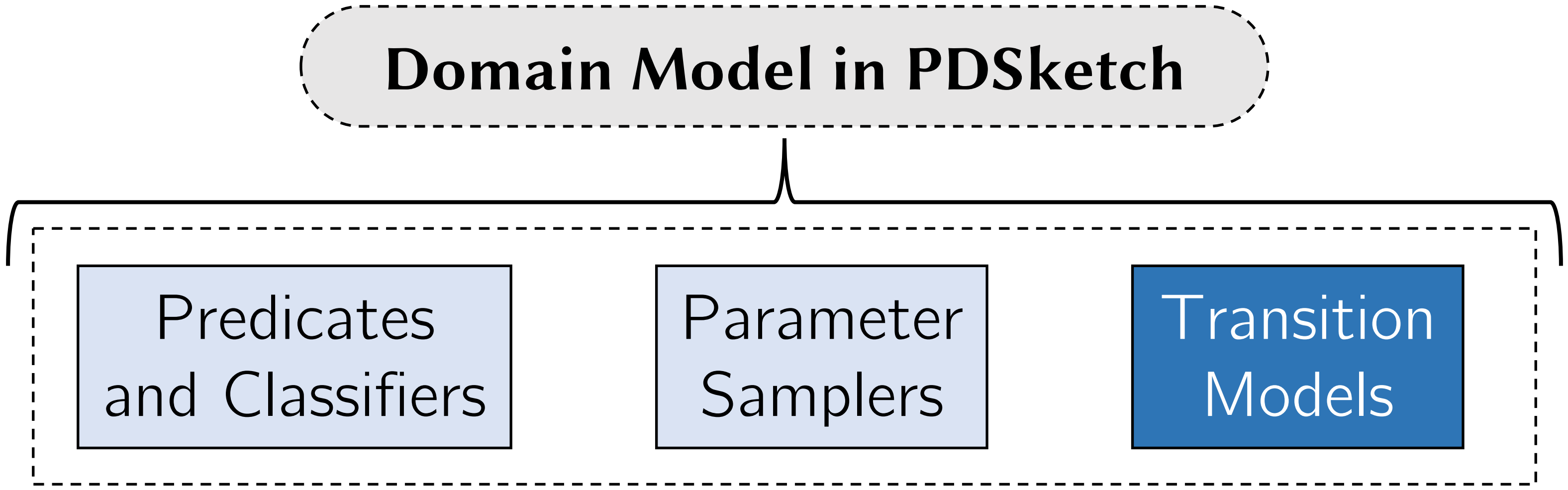
Learning Samplers from Videos

- Leveraging video prediction and flow estimation to reconstruct object motion in manipulation videos.
- Enables learning from video datasets for samplers for articulated objects, tool using, etc.



Learning to Act from Actionless Video through Dense Correspondences
Ko*, **Mao***, Du, Sun, Tenenbaum. In Submission 2023.

Learning Transition Models from LLMs



Learning Transition Models from LLMs

- PDSketch leverages symbolic structures in transition models.
- We can leverage large language models propose those structures for us, and perform learning for details and samplers.

Inst: *Get me a sliced piece of bread.* → → To **slice** something, you should use a knife. →

```
action slice(o, k, ...)
 pre: holding(k)
      is-knife(k),
      ...
 eff: sliced(o)
```

Learning Grounded Hierarchical Planning Skills From Language
Wong*, **Mao***, Sharma*, et al. In preparation 2023.

Towards Generalist Robots

Visual representation

object_1

object_2

Word	Syntax	Semantics	Concept Representations
orange	*set/set*	$\lambda x.\ \mathit{filter}(x,$ `orange`$)$	ORANGE
filter(object_1, orange) = TRUE			
left	*set\set/set*	$\lambda x \lambda y.\ \mathit{relate}(x, y,$ `left`$)$	LEFT
relate(object_1, object_2, left) = FALSE			
move	*action\set/set*	$\lambda x \lambda y.\ \mathit{action}(x, y,$ `move`$)$	MOVE

Precondition: relate(cylin, hand, holding)

Postcondition: not(relate(cylin, hand, holding)) relate(cylin, bottle, left)

Neuro-Symbolic Concepts

Neuro-symbolic concepts can be combined through reasoning and planning algorithms to solve tasks across domains and modalities.

Its modular nature enables data-efficient learning from various data streams.

Its symbolic structure enables interpretable, and also, faster reasoning and planning.