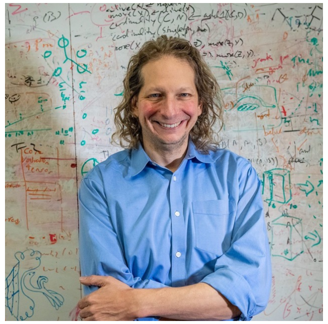
PDSketch

Integrated Domain Programming, Learning, and Planning






Jiayuan Mao Tomás Lozano-Pérez Joshua B. Tenenbaum Leslie Pack Kaelbling

MIT CSAIL


20
Clear Cups
CUT-RITE
Parchment Paper
Slow Cooker
LINERS

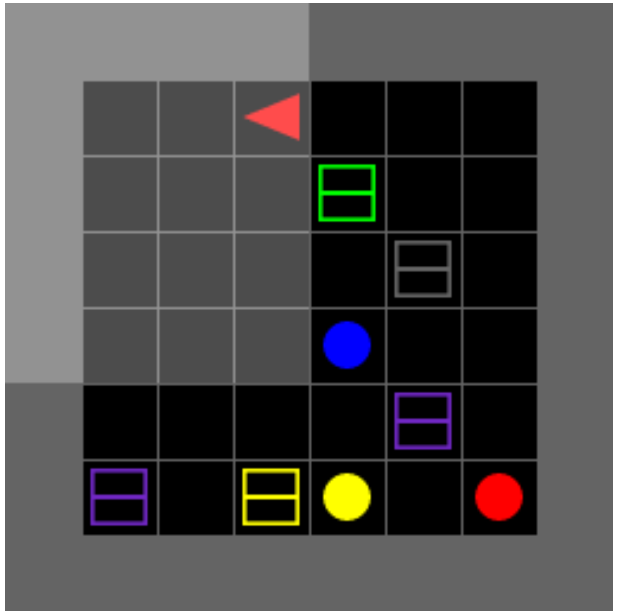
Factored Encodings for Environments

- Human can reason about *factored encodings* of the physical world.

Intuition:

- Factored encodings enable better data efficiency in learning.
- Factored encodings enable better planning efficiency.

MiniGrid Example



State Space:

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

Transition Model

```
def move_forward(s): ...
def rturn(s): ...
def toggle(s): ...
......
```

Predicates

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

Domain Model

State → **Domain-Independent Planner** → **Action** RTurn

Goal **"Go to a red object"**
∃x. is_yellow(x) & next_to(agent, x)

Existing Frameworks

Domain Programming

```
def facing(agent, object): ...

def move_forward(s):
  if not any(
    facing(s.agent, x) and
    is_obstacle(x)
    for x in s.objects
  ):
    if s.agent.facing == 0:
      s.agent.x -= 1
    elif s.agent.facing == 1:
      s.agent.y += 1
    elif ...
```

Neural Network Learning

```
def move_forward(s):
  s.agent = ??(s)
  for i in range(n):
    s.objects[i] = ??(s)
```

??: Trainable Neural Networks.

A lot of prior knowledge.
No/Minimal training data.
Fast planning.

Minimal prior knowledge.
A lot of training data.
Slow planning.

Existing Frameworks

Domain Programming

```
def facing(agent, object): ...

def move_forward(s):
  if not any(
    facing(s.agent, x) and
    is_obstacle(x)
    for x in s.objects
  ):
    if s.agent.facing == 0:
      s.agent.x -= 1
    elif s.agent.facing == 1:
      s.agent.y += 1
    elif ...
```

A lot of prior knowledge.
No/Minimal training data.
Fast planning.

PDSketch (This Work)

```
def move_forward(s):
  if not any(
    ??(s.agent, x)
    for x in s.objects
  ):
    s.agent = ??(s.agent)
```

Small amount of prior knowledge.
Small amount of training data.
Fast planning.

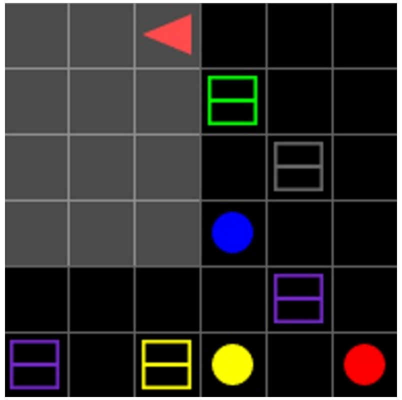
Neural Network Learning

```
def move_forward(s):
  s.agent = ??(s)
  for i in range(n):
    s.objects[i] = ??(s)
```

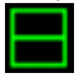
?? : Trainable Neural Networks.

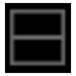
Minimal prior knowledge.
A lot of training data.
Slow planning.

PDSketch: Integrated Programming and Learning



Agent^t
(x=2, y=0,
facing=3)

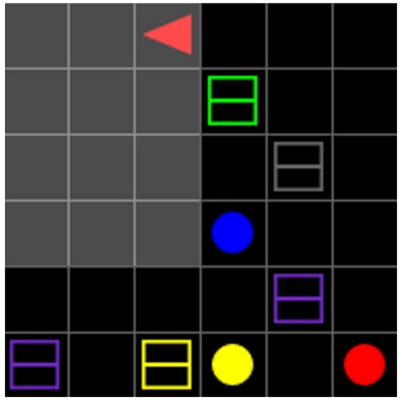
Objects^t[0]
(x=3, y=1,
image=)

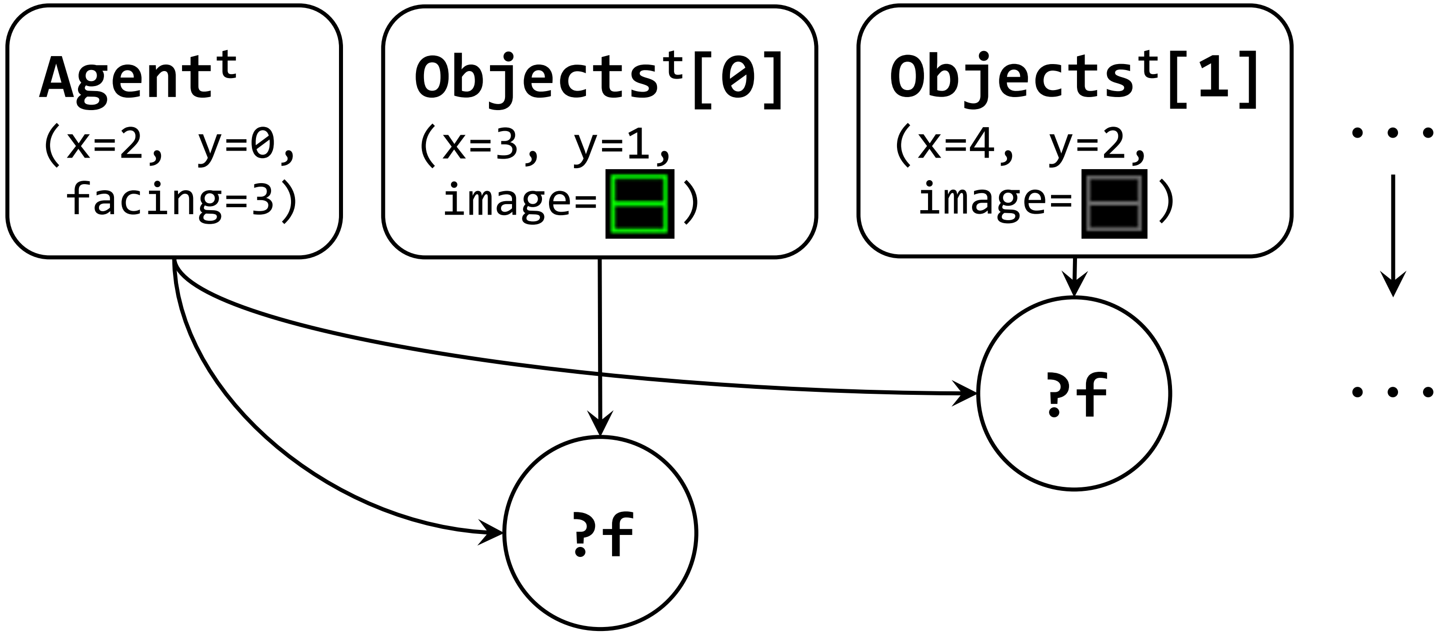
Objects^t[1]
(x=4, y=2,
image=)

...

```
def move_forward(s):
  if not any(
    ?f(s.agent, x)
    for x in s.objects
  ):
    s.agent = ?g(s.agent)
```

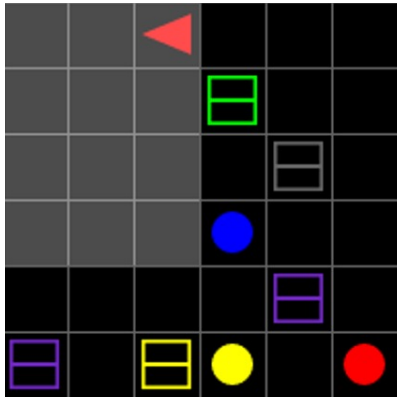
PDSketch: Integrated Programming and Learning

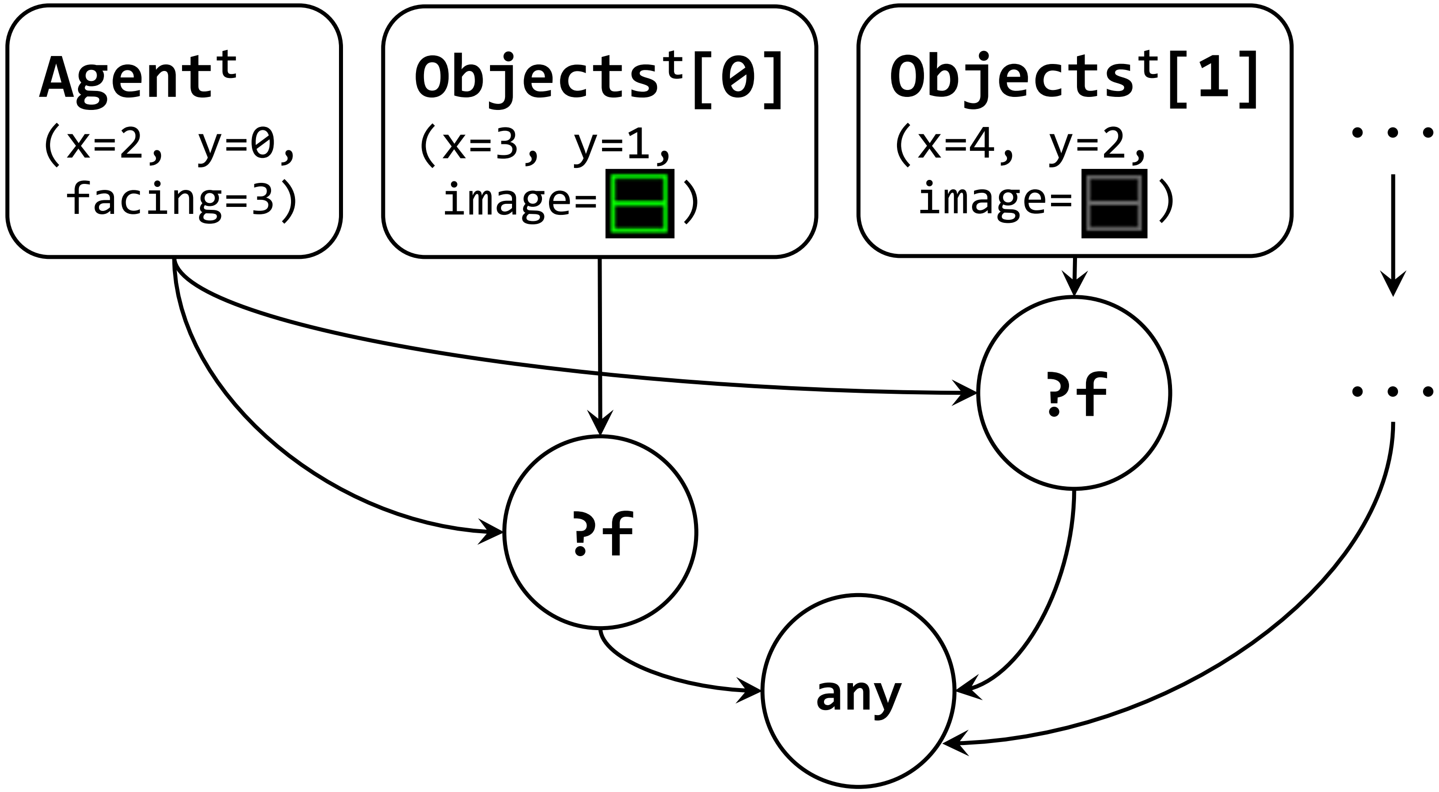





```
def move_forward(s):
  if not any(
    ?f(s.agent, x)
    for x in s.objects
  ):
    s.agent = ?g(s.agent)
```

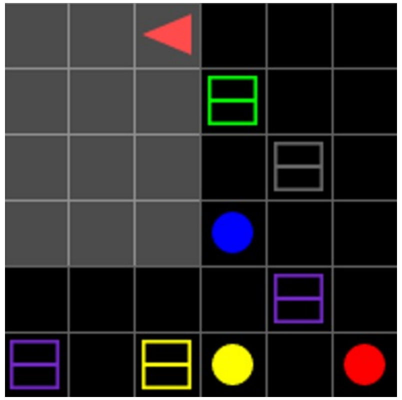
PDSketch: Integrated Programming and Learning



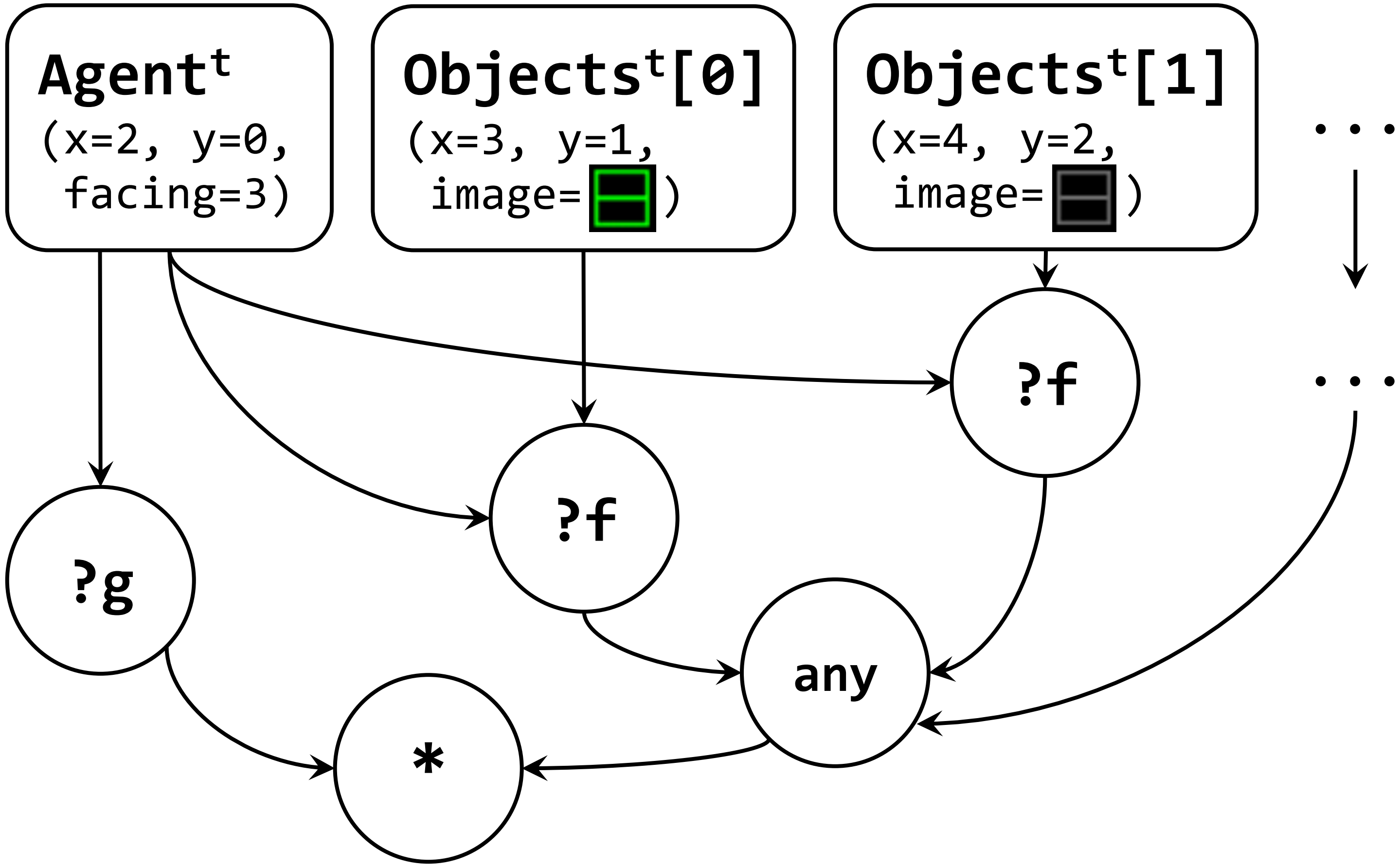



```
def move_forward(s):
  if not any(
    ?f(s.agent, x)
    for x in s.objects
  ):
    s.agent = ?g(s.agent)
```

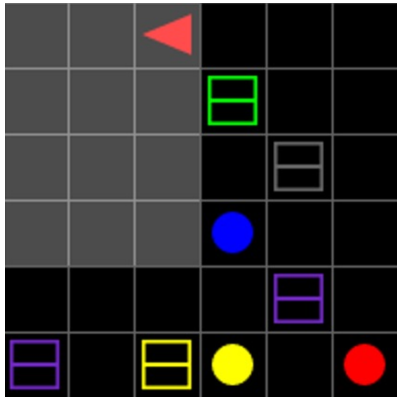
PDSketch: Integrated Programming and Learning



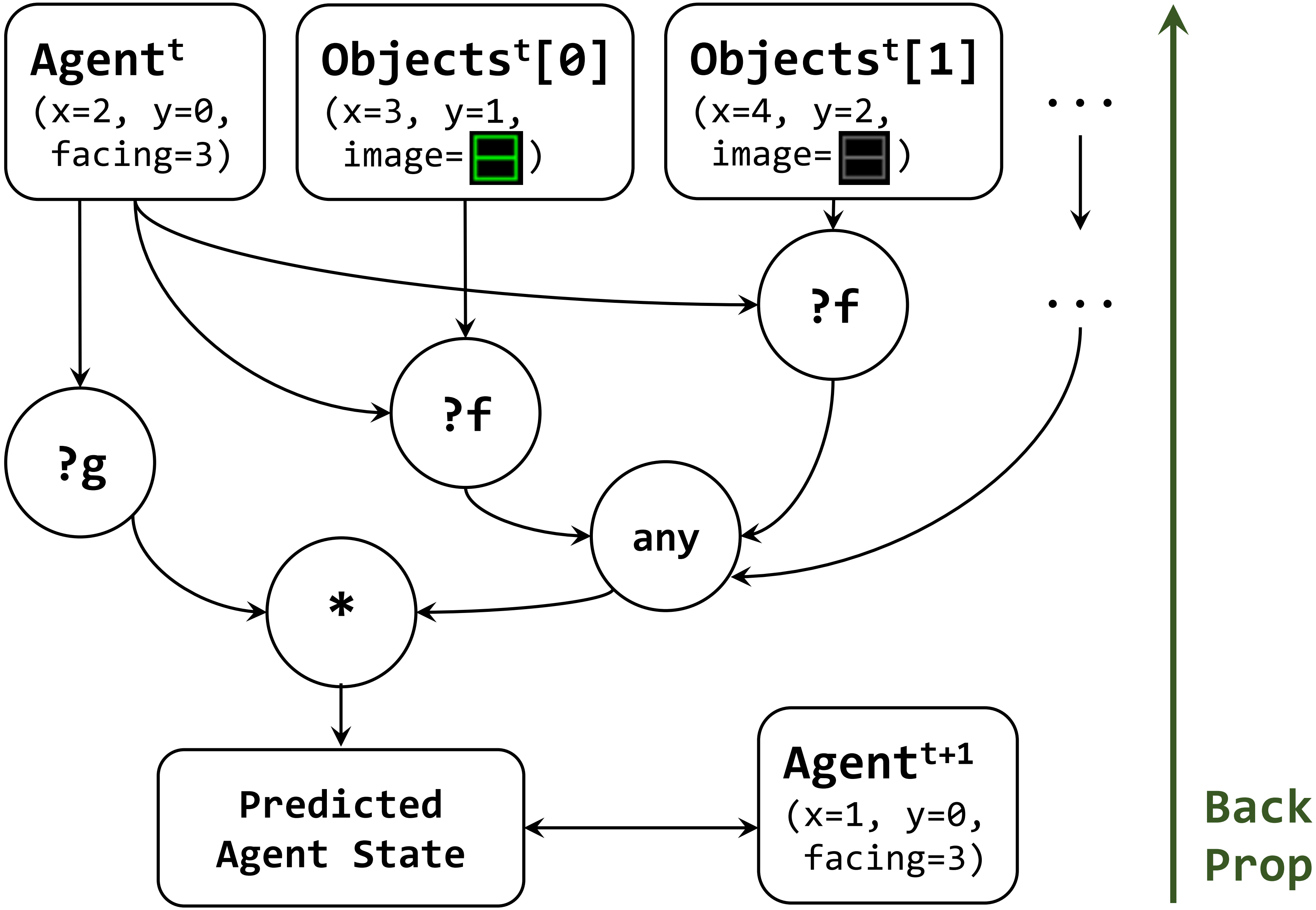
```
def move_forward(s):
  if not any(
    ?f(s.agent, x)
    for x in s.objects
  ):
    s.agent = ?g(s.agent)
```



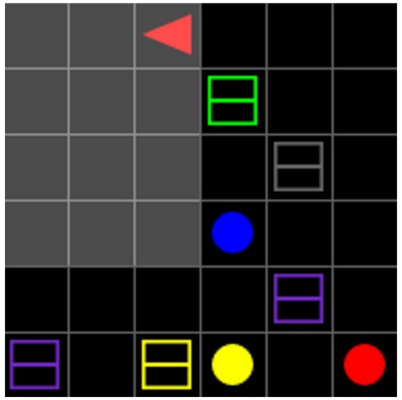
PDSketch: Integrated Programming and Learning

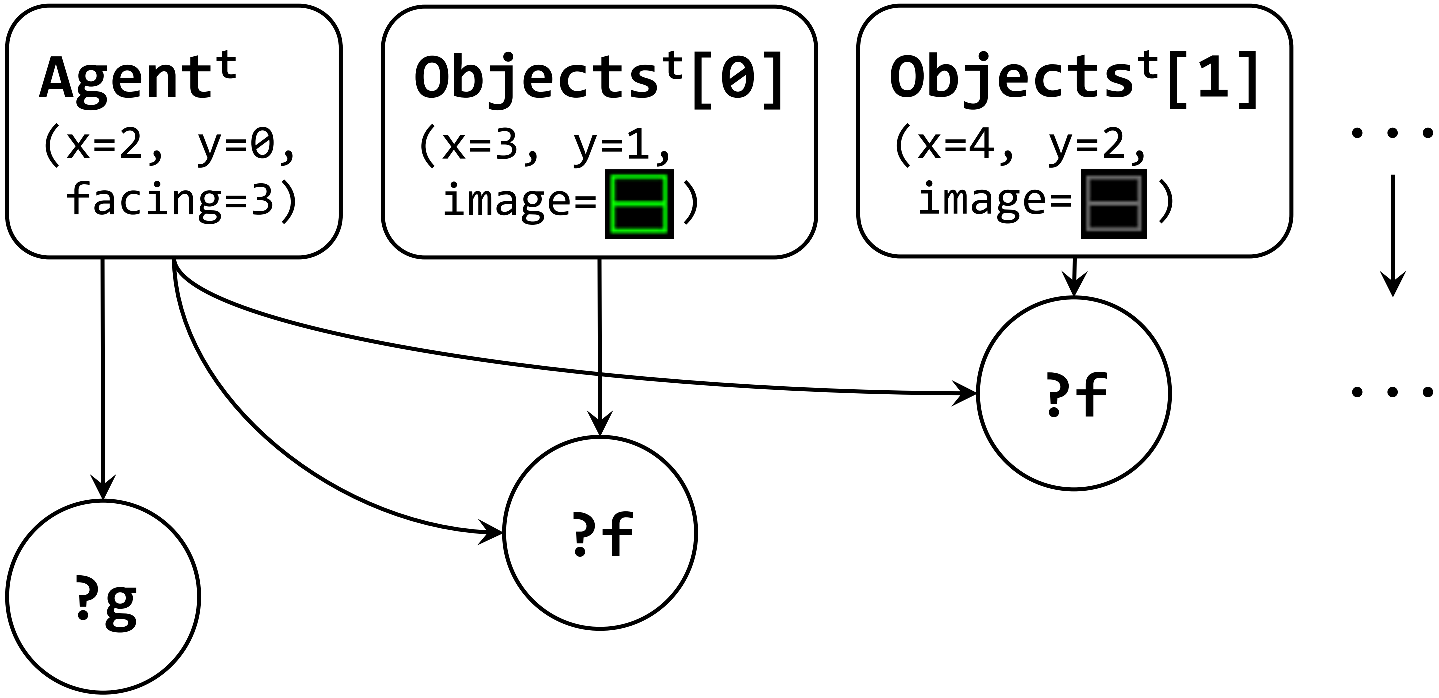


```
def move_forward(s):
  if not any(
    ?f(s.agent, x)
    for x in s.objects
  ):
    s.agent = ?g(s.agent)
```



PDSketch: Integrated Programming and Learning






```
def move_forward(s):
  if not any(
    ?f(s.agent, x)
    for x in s.objects
  ):
    s.agent = ?g(s.agent)
```

Each ?? can be implemented as a neural network module.
The programmatic structures encode

- The sparse and local structures of modules.
- The lifted structures (parameter sharing) of modules.

Learning and Planning Efficiency

PDS-Rob

Full robot movement models.
Need to learn object classifiers.

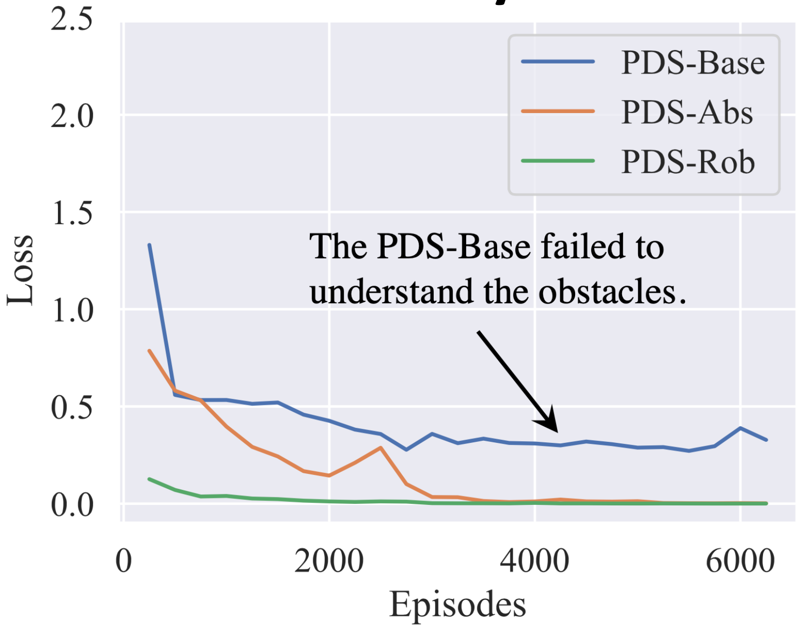
PDS-Abs

Abstract robot models.
(Sparse and local structures)

PDS-Base

Graph neural network.
(Weakest prior)

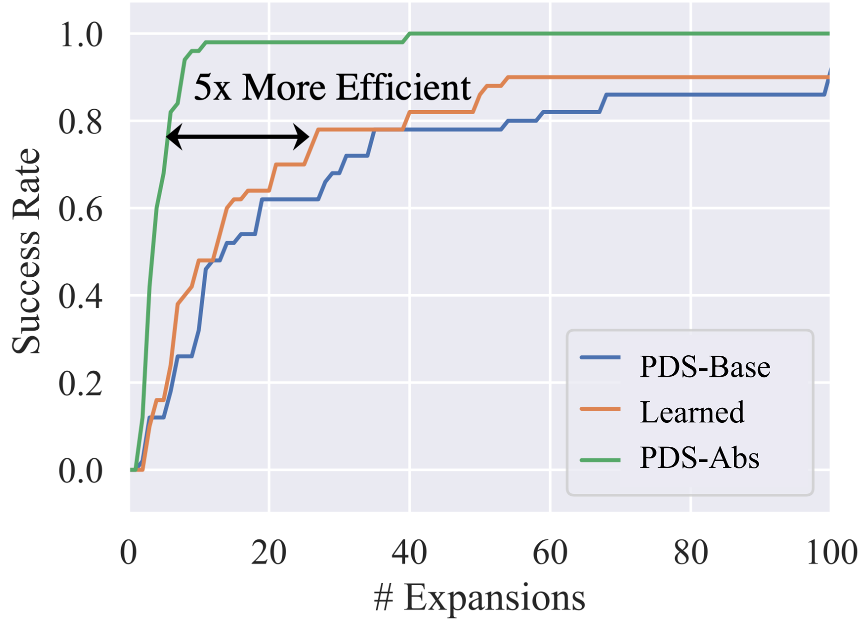
Data Efficiency



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



Learning and Planning Efficiency

These sparsity and locality structures can be *easily specified* using a First-Order-Logic language (derived from PDDL).

PDS-Abs

Abstract robot models.
(Sparse and local structures)

Data Efficiency

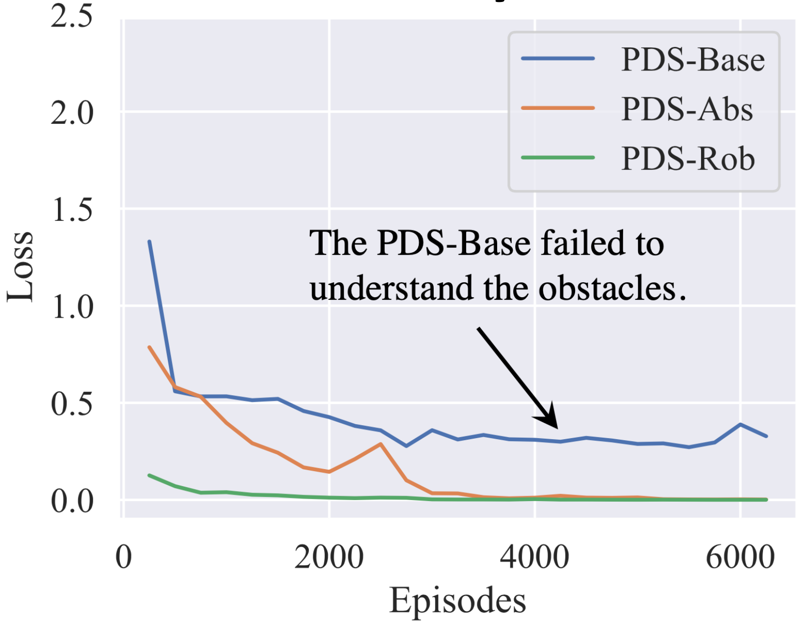
Success Rate

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

Learning and Planning Efficiency

Data Efficiency



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

Learning and Planning Efficiency

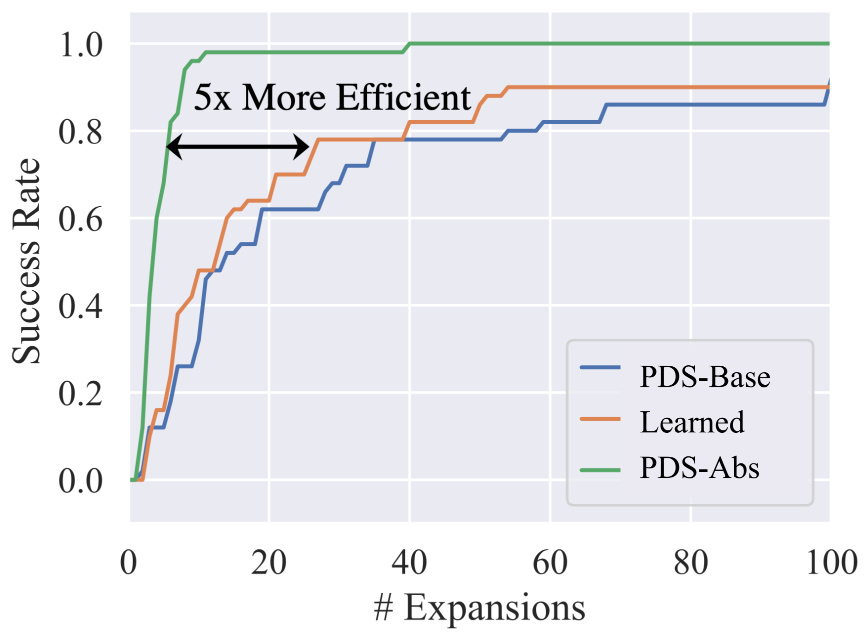
Success Rate	
Behavior Cloning	0.79
Decision Xformer	0.82
DreamerV2	0.79
PDS-Base	0.62
PDS-Abs	0.98
PDS-Rob	1.00

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

Learning and Planning Efficiency

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

Planning Efficiency

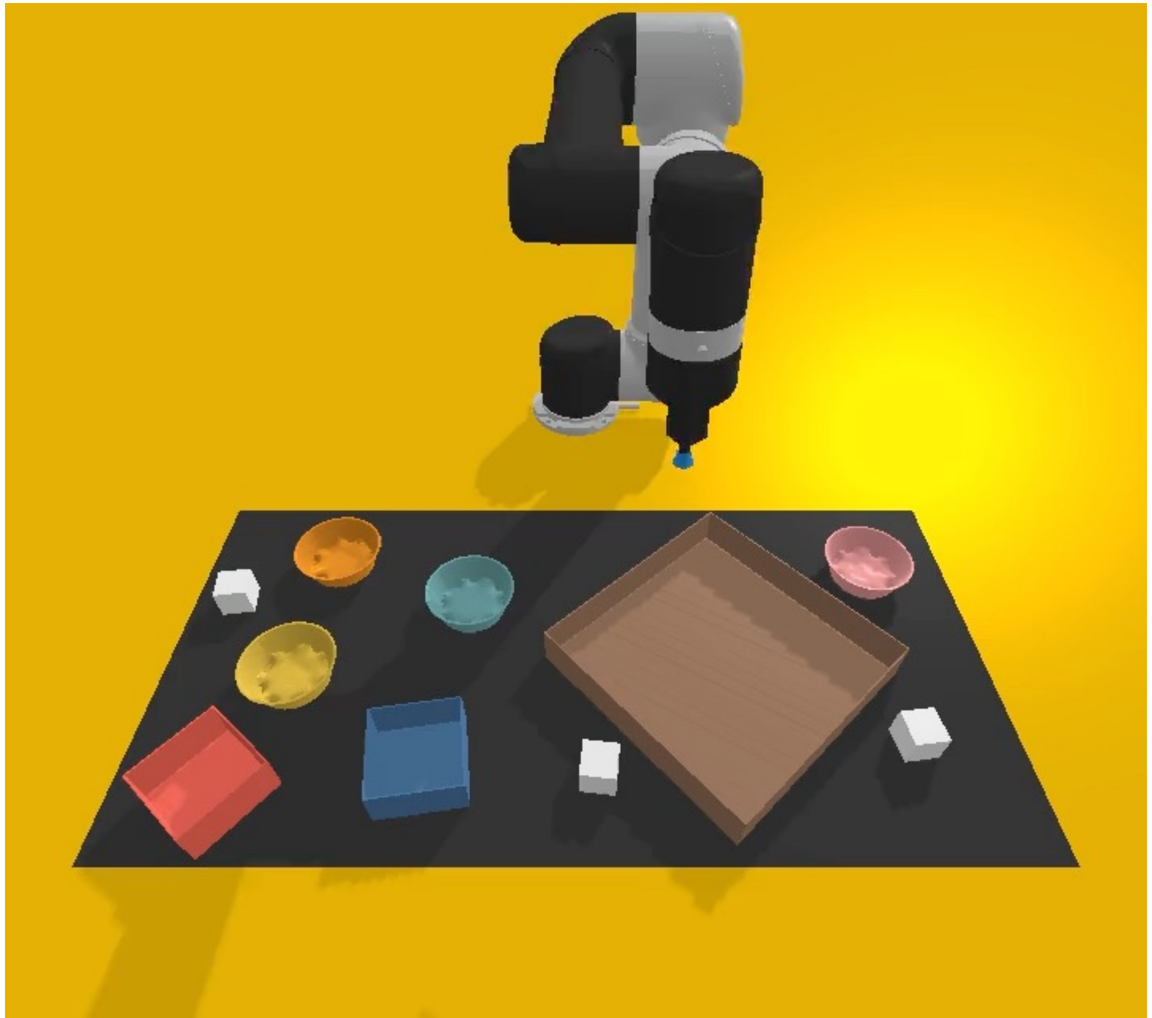


Generalization to Continuous Domains and Unseen Goals

Trained on goals: $\exists$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

Mao, Lozano-Pérez, Tenenbaum, Kaelbling. In *NeurIPS* 2022.

- A framework for combining programmatic structures and learning for model-based planning.
- Such structural priors can be flexibly specified and matches the structures in the physical world.
- Leveraging factored representations improves data efficiency.
- Factored representation supports automatically derived planning heuristics.
- https://pdsketch.csail.mit.edu