PDDLGym: Gym Environments from PDDL Problems

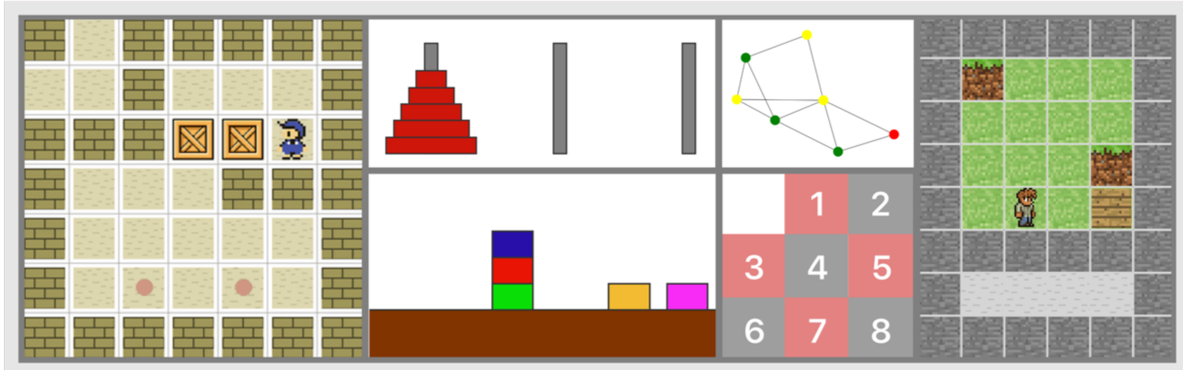
Tom Silver and Rohan Chitnis
MIT Computer Science and Artificial Intelligence Laboratory
{tslvr, ronuchit}@mit.edu

Massachusetts Institute of Technology

CSAIL
MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

PDDLGym Overview

- *Open-source Github repository available at*: **tinyurl.com/pddlgym**.
- Python library that creates Gym environments from PDDL files.
- Useful for research in **relational reinforcement learning**.
- 20+ domains, covering most features of PDDL 1.2, and PPDDL.



Code Example

```
(:action put-down
      :parameters (?x - block)
      :precondition (and
         (holding ?x)
      )
      :effect (and
         (not (holding ?x))
         (clear ?x)
         (handempty)
         (ontable ?x)
      )
)
```
A

```
(:init
   (clear B1)
   (clear B2)
   (ontable B1)
   (ontable B2)
   (handempty)
)

(:goal
   (on B1 B2)
)
```
B

```
import pddlgym

env = pddlgym.make("PDDLEnvBlocks-v0")
obs, info = env.reset()
img = env.render()
# We can take random actions
action = env.action_space.sample(obs)
obs, reward, done, info = env.step(action)
# Or execute a plan
plan = run_planner(...)
for action in plan:
   env.step(action)
```
C

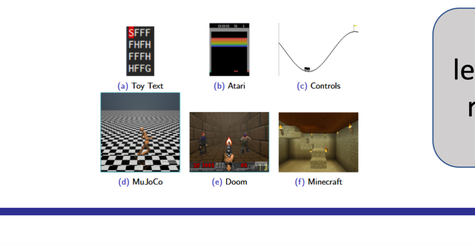
(A) You provide a PDDL domain file (example shown).
(B) You provide a set of PDDL problem files (one example shown).
(C) Then, you can interact with these files as a Gym environment!

Motivation

OpenAI Gym: a minimal environment API for reinforcement learning.
Two main methods in the API:

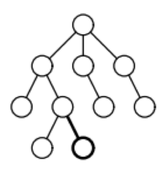
- `reset()` gives an initial observation.
- `step(action)` transitions state; gives next observation & reward.

Reinforcement learning: episodic, closed-loop interaction with an environment



PDDL: structured, relational representation of states, actions, and transition model

- Initial State (:init …)
- Goal (:goal …)
- Actions (:action name
  :parameters (?from ?to ?dir)
  :preconditions (…)
  :effects (…)
)



(a) Toy Text (b) Atari (c) Controls

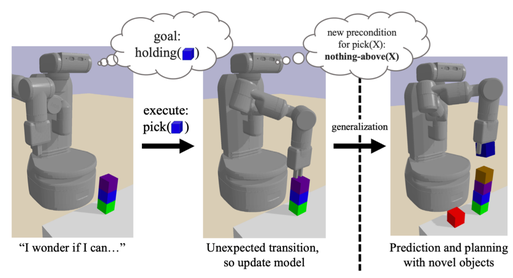
(d) MuJoCo (e) Doom (f) Minecraft

Together: learning symbolic reasoning from interaction!

Research We've Used PDDLGym For

PDDLGym offers a useful substrate for research that connects learning algorithms to relational, predicate-based domains. Examples:

Exploration for lifted operator learning [1]



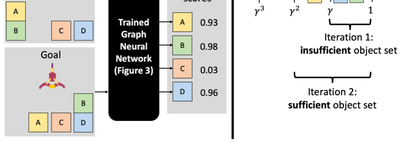
Learning goal-conditioned policies [2]

Policy for `Holding`

```
¬Holding(V0) ∧
¬Holding(V1) ∧
  At(V0, V2) ∧  ⟹ MoveTo(V2)
 IsRobot(V3) ∧
  At(V3, V4)
```

```
¬Holding(V0) ∧
¬Holding(V1) ∧
  At(V0, V2) ∧  ⟹ Pick(V0)
 IsRobot(V3) ∧
  At(V3, V2)
```

Learning state abstractions for planning [3]



[1] Chitnis, Silver, Tenenbaum, Kaelbling, Lozano-Perez. *GLIB: Exploration via goal-literal babbling for lifted operator learning.* Under review.
[2] Silver, Chitnis, Ajay, Tenenbaum, Kaelbling. *Learning skill hierarchies from predicate descriptions and self-supervision.* In GenPlan workshop @ AAAI 2020.
[3] Silver, Chitnis, Curtis, Tenenbaum, Lozano-Perez, Kaelbling. *Planning with learned object importance in large problem instances using graph neural networks.* Under review.

Environment Details

- States have 3 components: a goal, a set of objects, a set of true facts.
- Rewards are sparse: +1 if the goal is achieved, 0 otherwise.
- `step(action)` checks action applicability against preconditions.
- Inference backends: typed SLD resolution in Python; SWI-Prolog.

Actions vs. Operators

We distinguish between *operators* (in planning) and *actions* (in RL).
Issue: In PDDL operators, only some parameters are "free".
Our solution: Introduce new "action predicates" defining action space.

```
(:action move
   :parameters (?p - thing ?from - loc
                ?to - loc ?dir - dir)
   :precondition (and
      (is-player ?p)
      (at ?p ?from)
      (clear ?to)
      (move-dir ?from ?to ?dir)
   )
   :effect (and
      (not (at ?p ?from))
      (not (clear ?to))
      (at ?p ?to)
      (clear ?from)
   )
)
```

```
(:action move
   :parameters (?p - thing ?from - loc
                ?to - loc ?dir - dir)
   :precondition (and
      (move-action-selected ?dir)
      (is-player ?p)
      (at ?p ?from)
      (clear ?to)
      (move-dir ?from ?to ?dir)
   )
   :effect (and
      (not (at ?p ?from))
      (not (clear ?to))
      (at ?p ?to)
      (clear ?from)
   )
)
```

Left: Classic Sokoban `move` operator. Only `?dir` is a free parameter. Other parameters are forced by current state or choice of `?dir`.
Right: In PDDLGym, the Sokoban `move` operator includes an action predicate, `move-action-selected`, parameterized only by the free parameter `?dir`. This lets us sample random actions and learn policies.

Feature Requests? Email Us!

- We can add new environments.
- We can add support for more PDDL features.
- We can help you start using PDDLGym for your own research.

Currently implemented domains:

Domain Name	Rendering Included	Probabilistic	Average FPS
Baking	No	No	5897
Blocks	Yes	No	7064
Casino	No	No	7747
Crafting	Yes	No	4568
Depot	No	No	97
Doors	Yes	No	917
Elevator	No	No	3501
Exploding Blocks	Yes	Yes	6260
Ferry	No	No	1679
Gripper	Yes	No	319
Hanoi	Yes	No	4580
Meet-Pass	No	No	7380
Rearrangement	Yes	No	3808
River	No	Yes	18632
Search and Rescue	Yes	No	3223
Slide Tile	Yes	No	3401
Sokoban	Yes	No	155
Triangle Tireworld	No	Yes	6491
TSP	Yes	No	1688
USA Travel	No	No	1251