Introduction to MDP Modeling and Interaction via RDDL and pyRDDLGym

Part 2

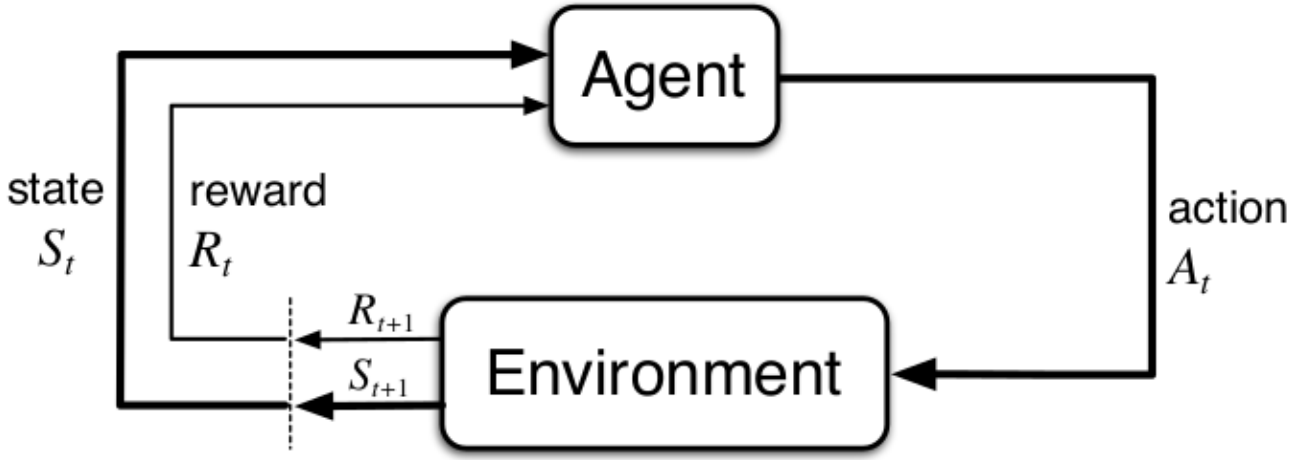
Ayal Taitler and Scott Sanner

University of Toronto

Lab, AAAI

February 20th, 2024

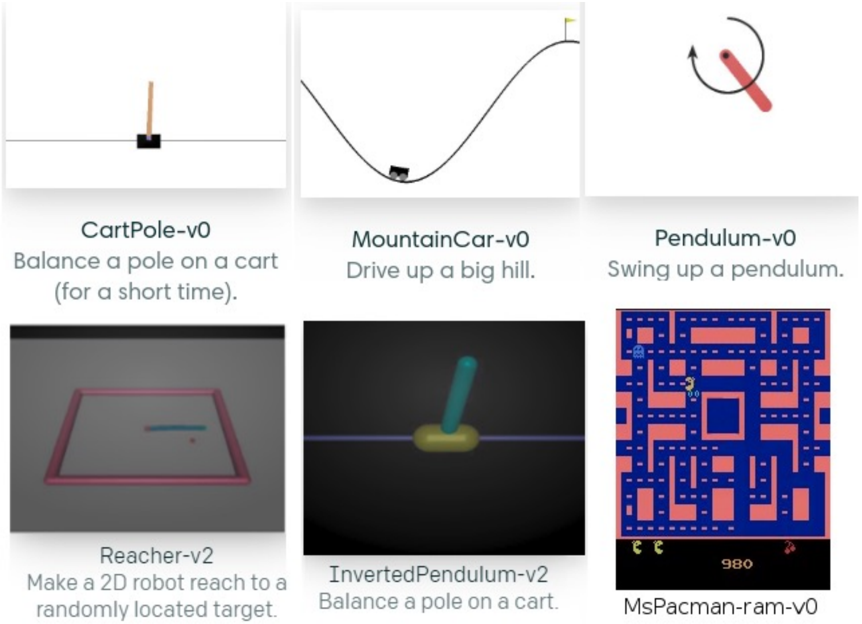
MDP Modeling



- Markov Decision Process (MDP):
  - S – States (discrete/continuous/hybrid)
  - A – Actions (discrete/continuous/hybrid)
  - R – Reward function (scalar)
  - T – Transition function (conditional probability function)

OpenAI Gym

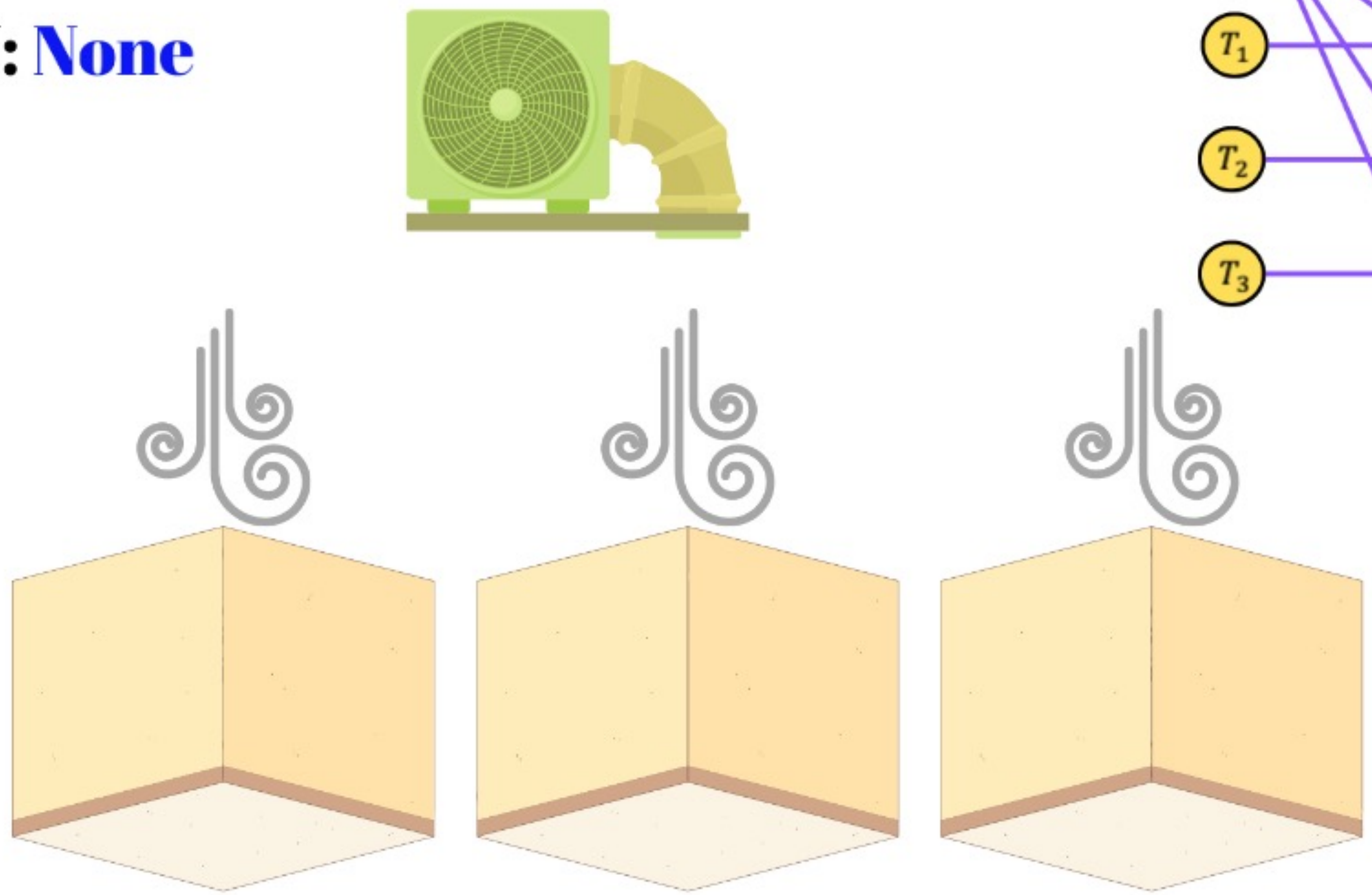
- OpenAI gives an interface to implement MDPs
- Direct environment implementation
  - Python coding of the logic
- Gaps
  - Time consuming
  - Hard coded parameters
  - Minor change = new implementation
  - Infinite implementations
  - No clean way to verify
  - No access to the model

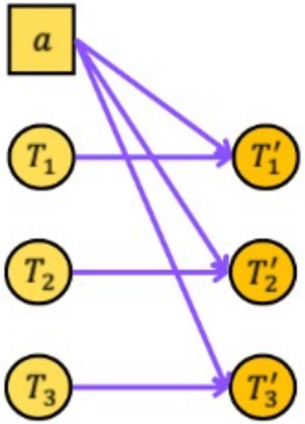


HVAC – scenario 1

ZONES: 3

ADJ: None



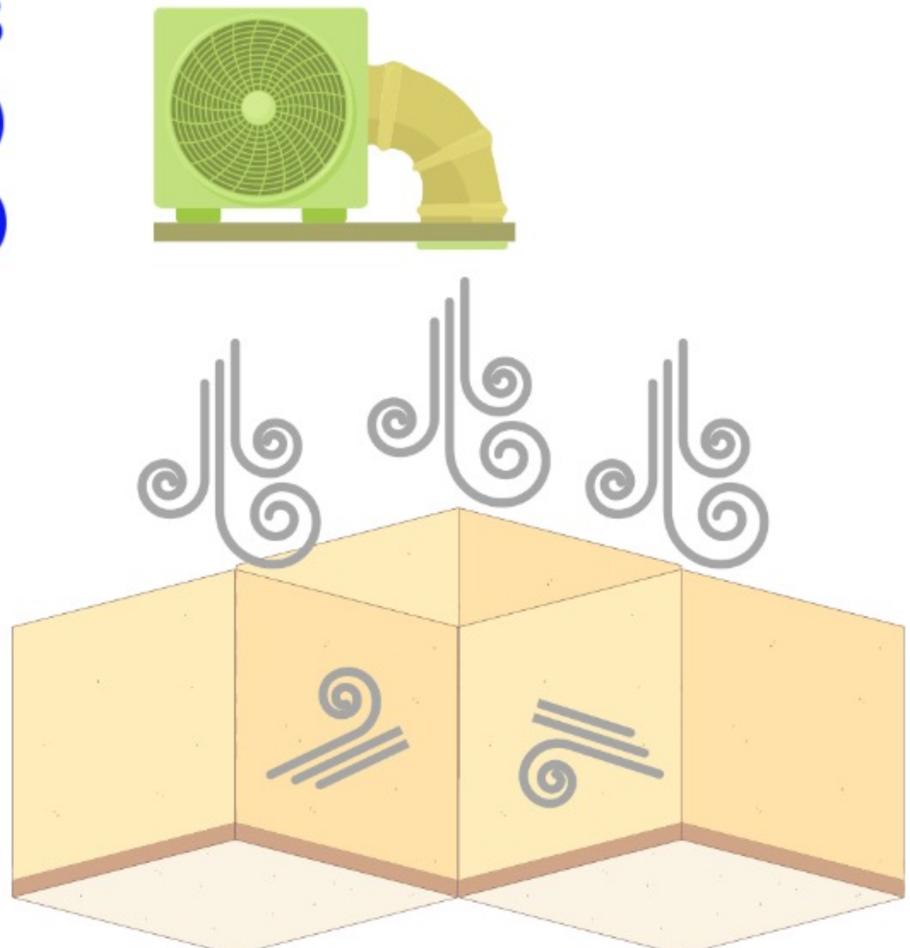


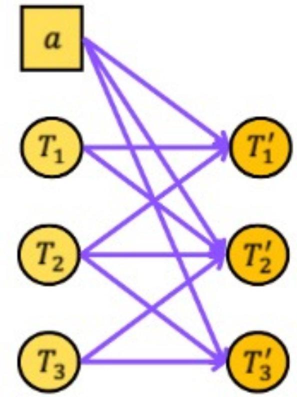
HVAC – scenario 2

ZONES: 3

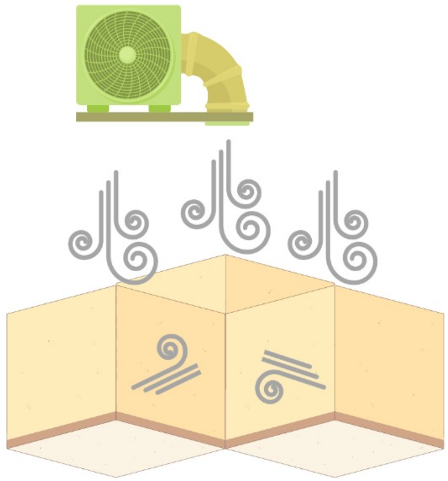
ADJ: (1,2)

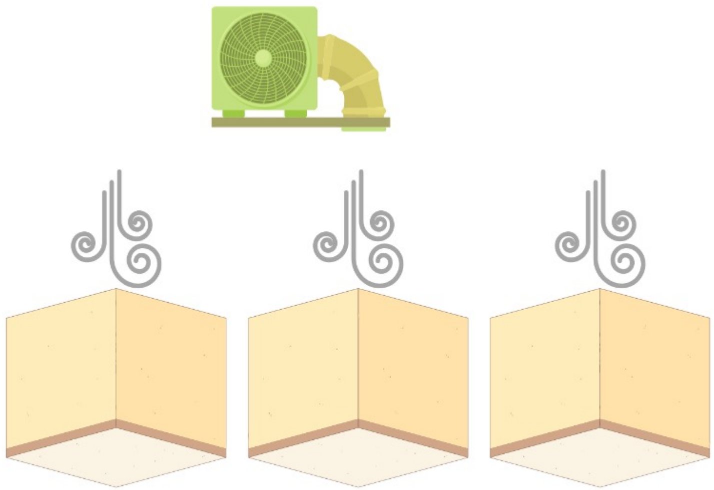
(2,3)





Motivation



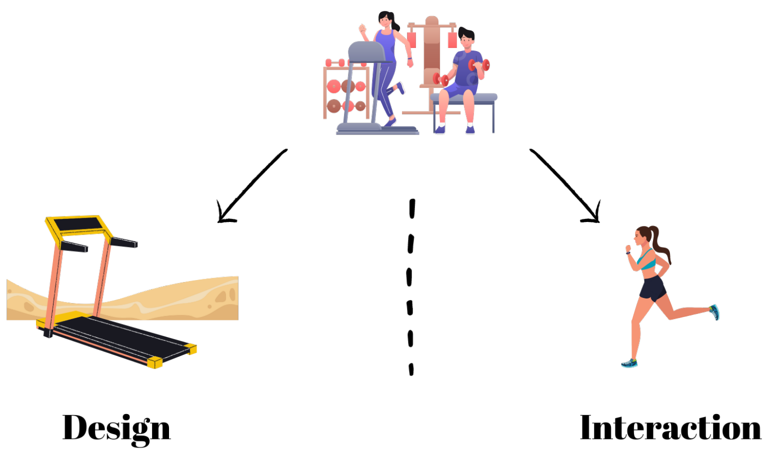


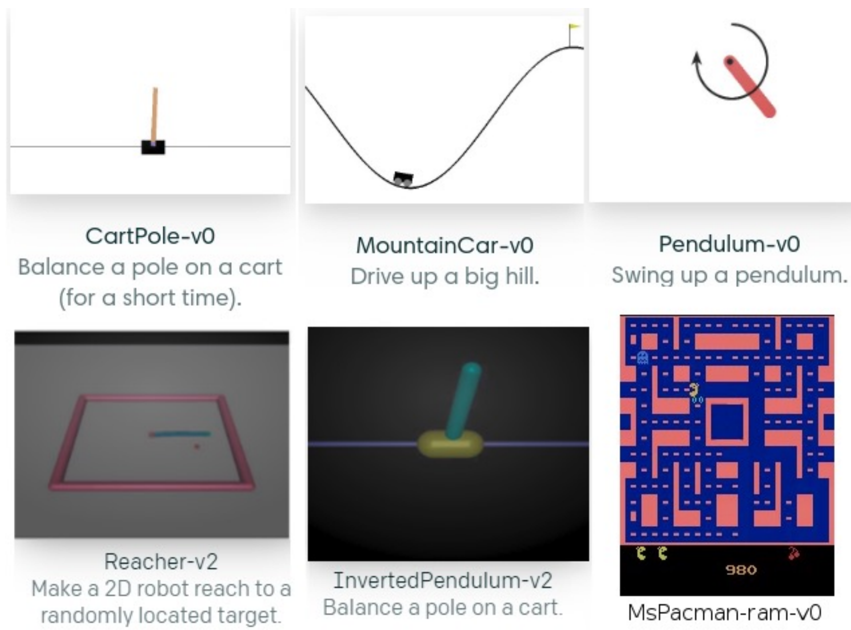
One mathematical model
Two env implementations
With a lot of code duplication

Identical input/output
(actions/states)
Different transition function

Motivation

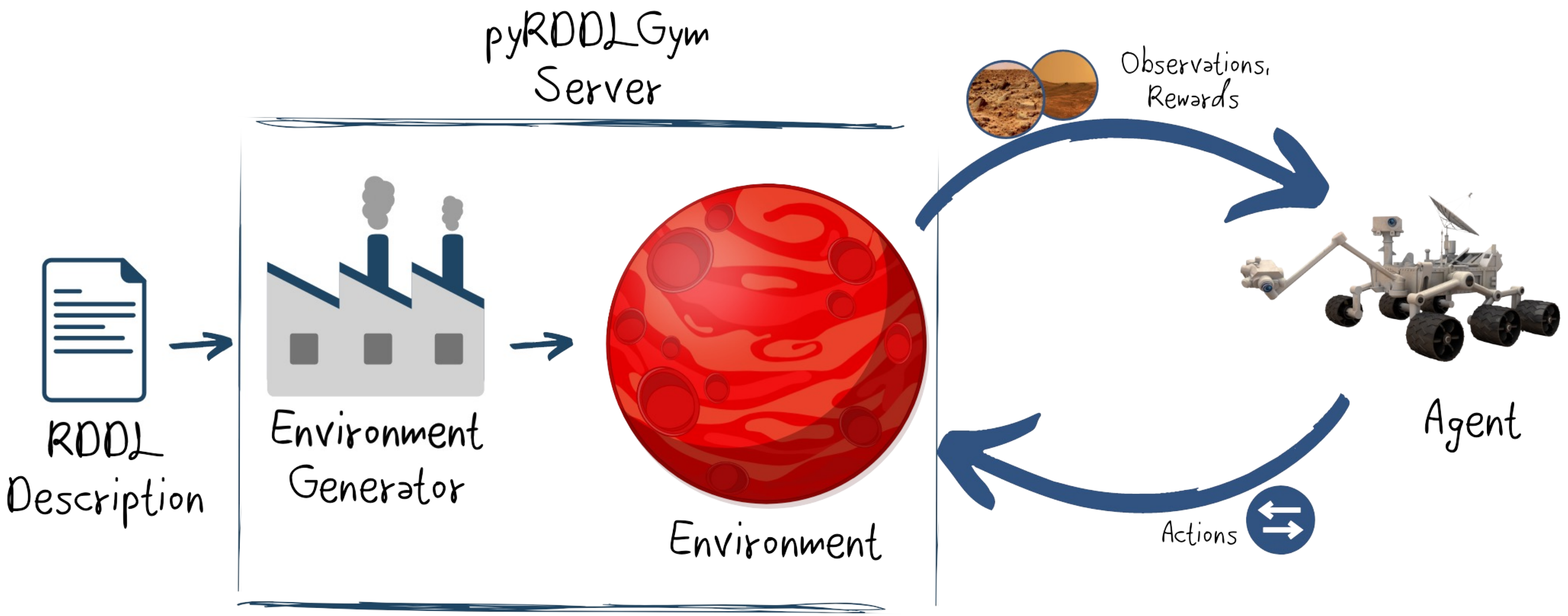
- OpenAI gives an interface to implement MDPs
- Direct environment implementation
  - ➢ Python coding of the logic





Who's doing the implementation? 🤔

pyRDDLGym



RDDL → compiler → Gym environment

- Standard Gym interface and spaces
- Full access to the underlying model
- Differentiable dynamics*

https://pyrddlgym-project.github.io/

Language Variant

Full RDDL support!

New language features:

- Terminal states

$$terminal = cond_1 \vee cond_2 \vee \cdots \vee cond_N$$

- Nested indexing

$$fluent'(?p, ?q) \; = \; NEXT(fluent(?p, ?q))$$

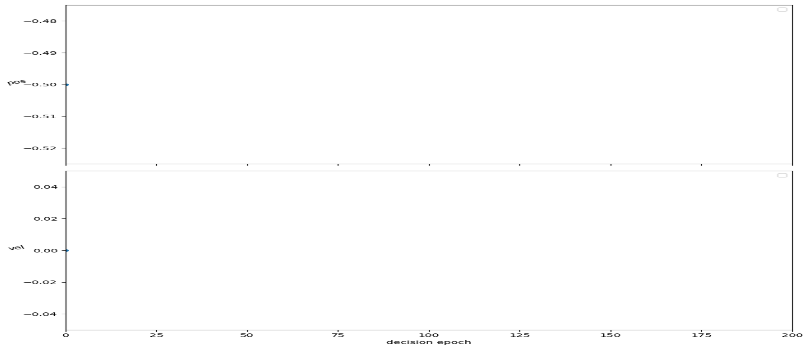
- Lifter parameter (in)equalities

$$?p == ?r$$

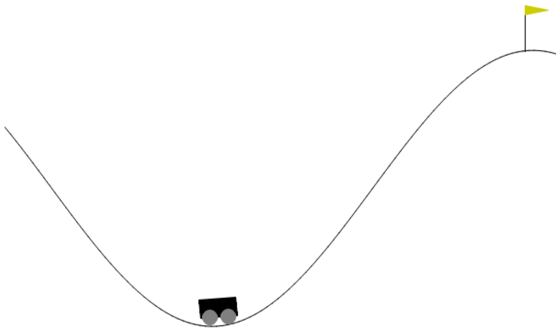
- $argmin$ and $argmax$ for enumerables
- Basic matrix algebra, vectorized distributions, automatic level reasoning and more.

Visualizers

- pyRDDLGym comes with a built-in *ChartVisualizer* class



- It is simple to create customs visualizers

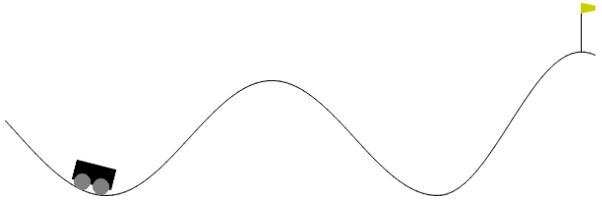


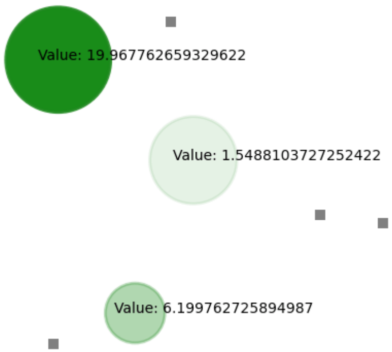
  - Inherit base class *pyRDDLGym.Visualizer.StateViz*
  - One can use his favorite graphical lib, e.g., *matplotlib*, *pygame*, etc…

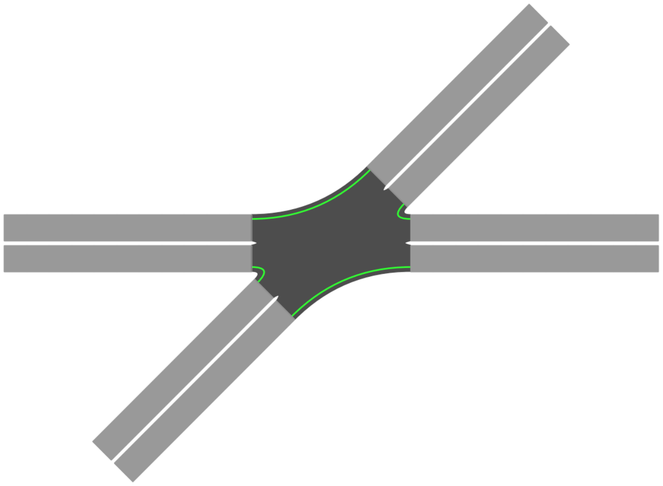
Auxillary Tools

Movie Generator

- Built-in functionality for movie generations of episodes
- Supports GIF and MP4

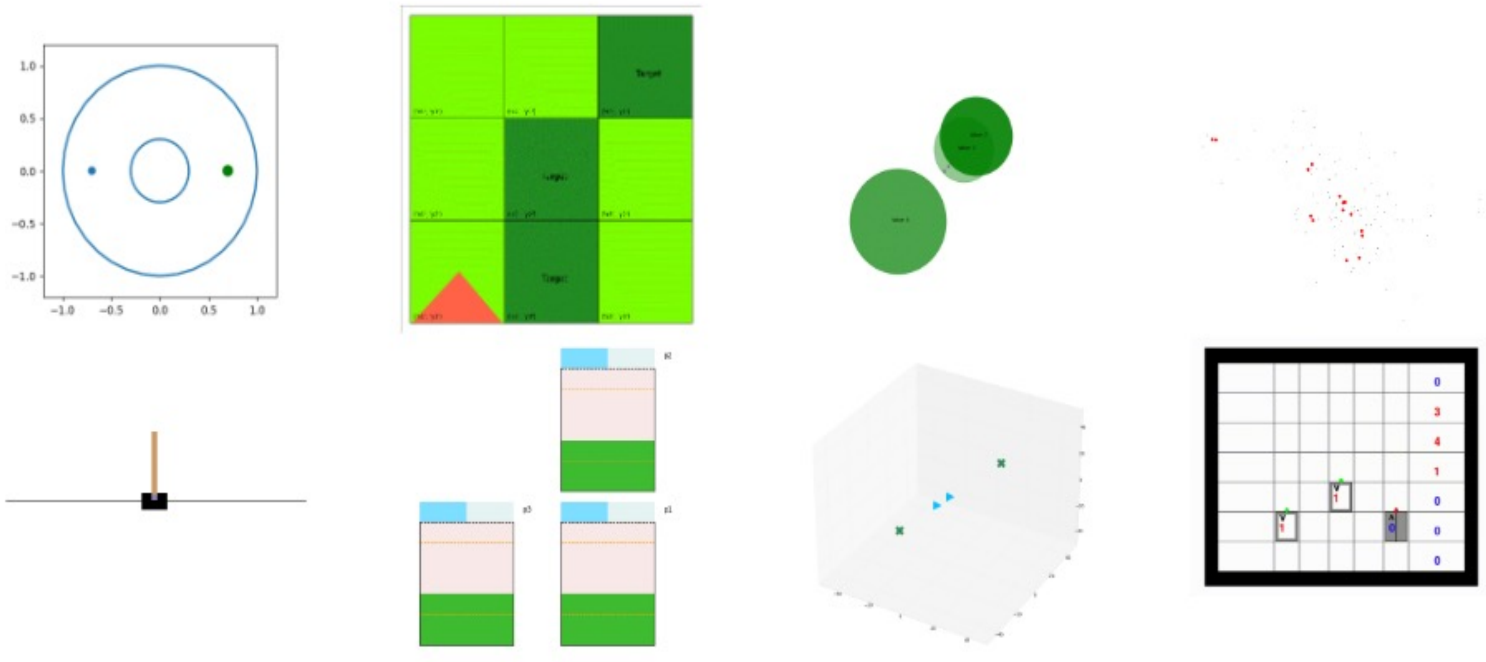






pyRDDLGym Eco-system

Environments Repository



Gym's Classical control environments

All previous RDDL domains

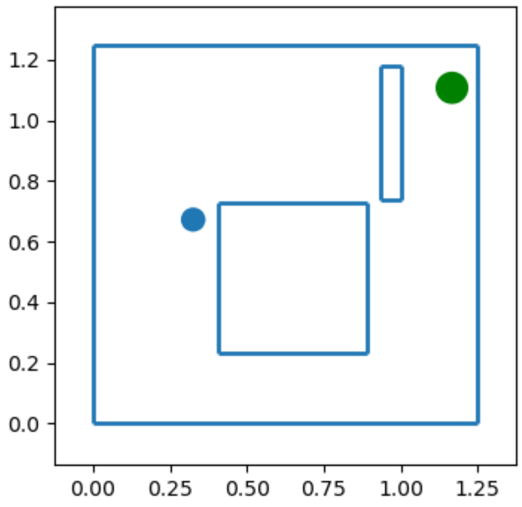
New exciting environments

RaceCar

- Goal oriented problem
- Plan trajectory for a kinematic agent (2nd order) in presence of obstacles
- **Action:** force/acceleration in two axes ($n_a = 2$)
- **Observation:** positions and velocities ($n_s = 4$)
- **Reward:**

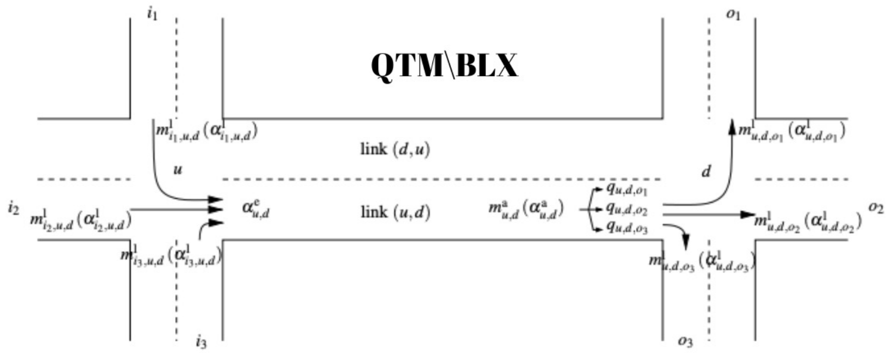
$$R = -\sum_{k=1}^{H} a_x^2[k] + a_y^2[k] + R_G \cdot 1_{\{a_x^2[k]+a_y^2[k]<r_g\}}$$

Termination: $a_x^2[k] + a_y^2[k] < r_g$



Built-in Environments – Traffic

- Traffic network cogestion control



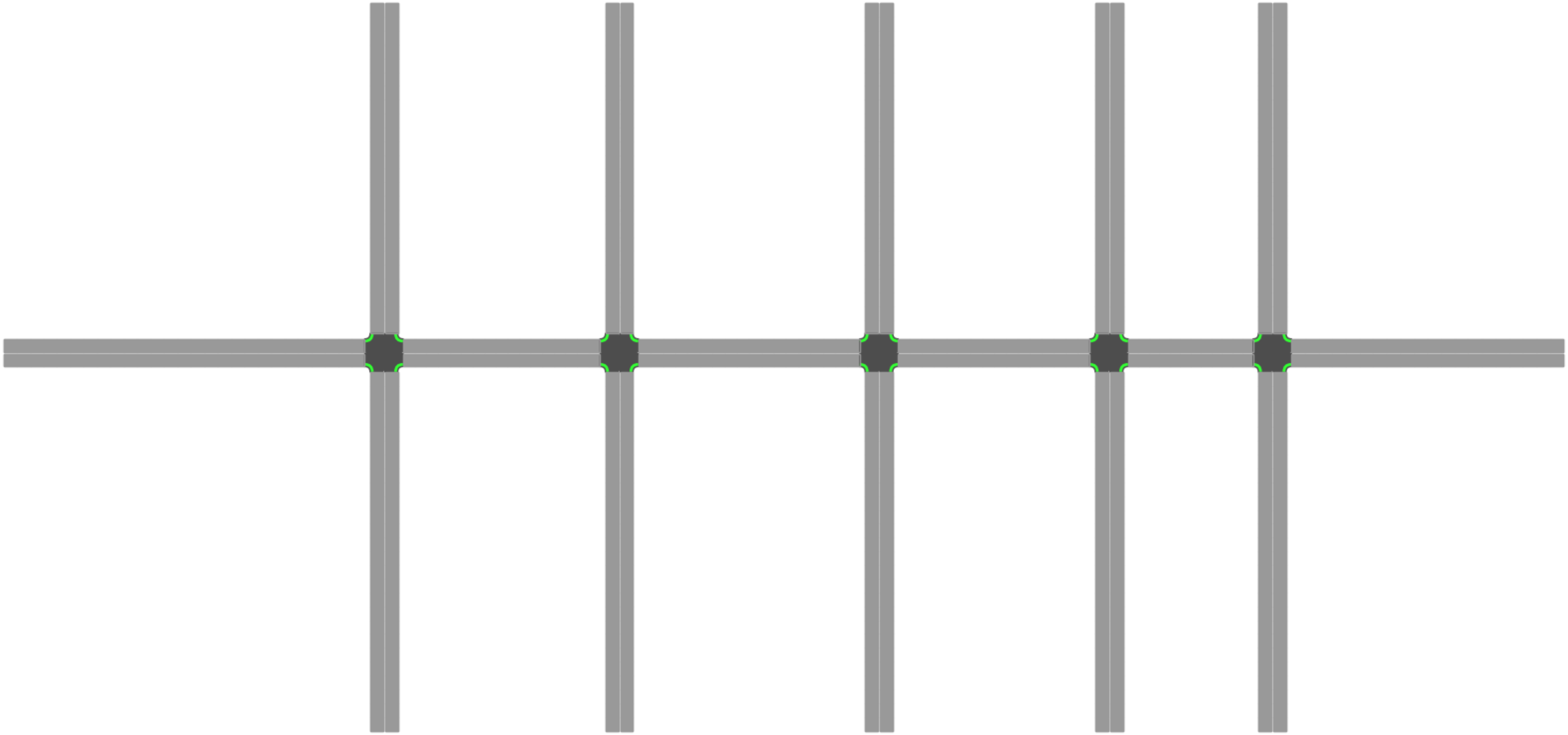
$$q_{u,d,o_m}(k_d+1) = q_{u,d,o_m}(k_d) + \left(\alpha^{\mathrm{a}}_{u,d,o_m}(k_d) - \alpha^{\mathrm{l}}_{u,d,o_m}(k_d)\right)\cdot c_d$$

$$q_{u,d}(k_d) = \sum_{o_m \in O_{u,d}} q_{u,d,o_m}(k_d)$$

$$\vdots$$

- **Action:** Extend/Change for light phases (each intersection)
- **Observation:** Cars in queues, phase, phase time, etc.
- **Reward:** Total travel time (number of cars in the network)
- **Constraints:** Min/max time in phase

Built-in Environments – Traffic



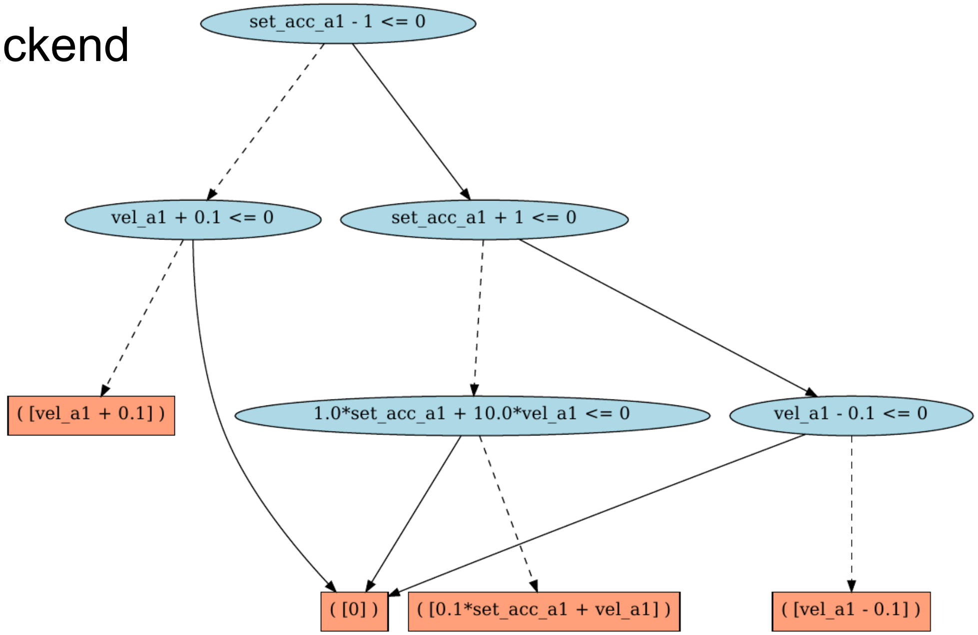
1x5 Network

Symbolic Toolkit (I)

Extended Algebric Decision Diagrams (XADDs)

- Symbolic function representation for Piecewise Linear functions
- Compact representation of the *grounded* cpfs
- Symbolic Dynamic Programming (SDP)
- Representation and framework backend

XADD for the UAV domain



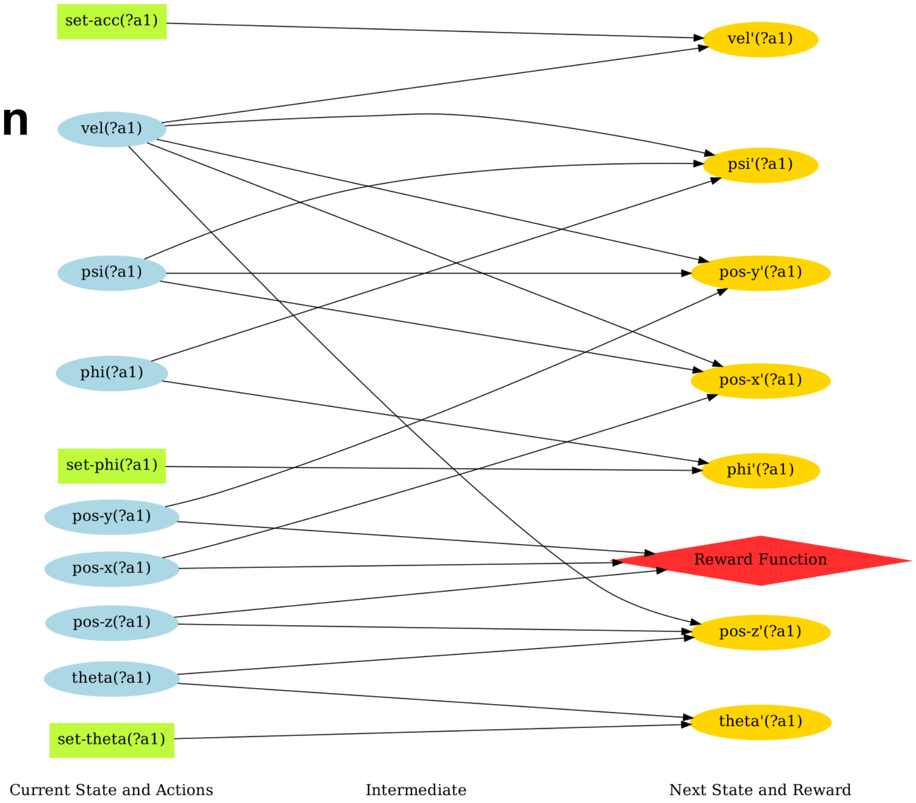
https://github.com/pyrddl-project/pyRDDLGym-symbolic

Symbolic Toolkit (II)

Dynamic Bayes Nets (DBNs) visualization

- Visualization of the causal relations
- Causality inference
- Direct GCN methods

  e.g., SymNets (symbolic Networks)



DBN visualization

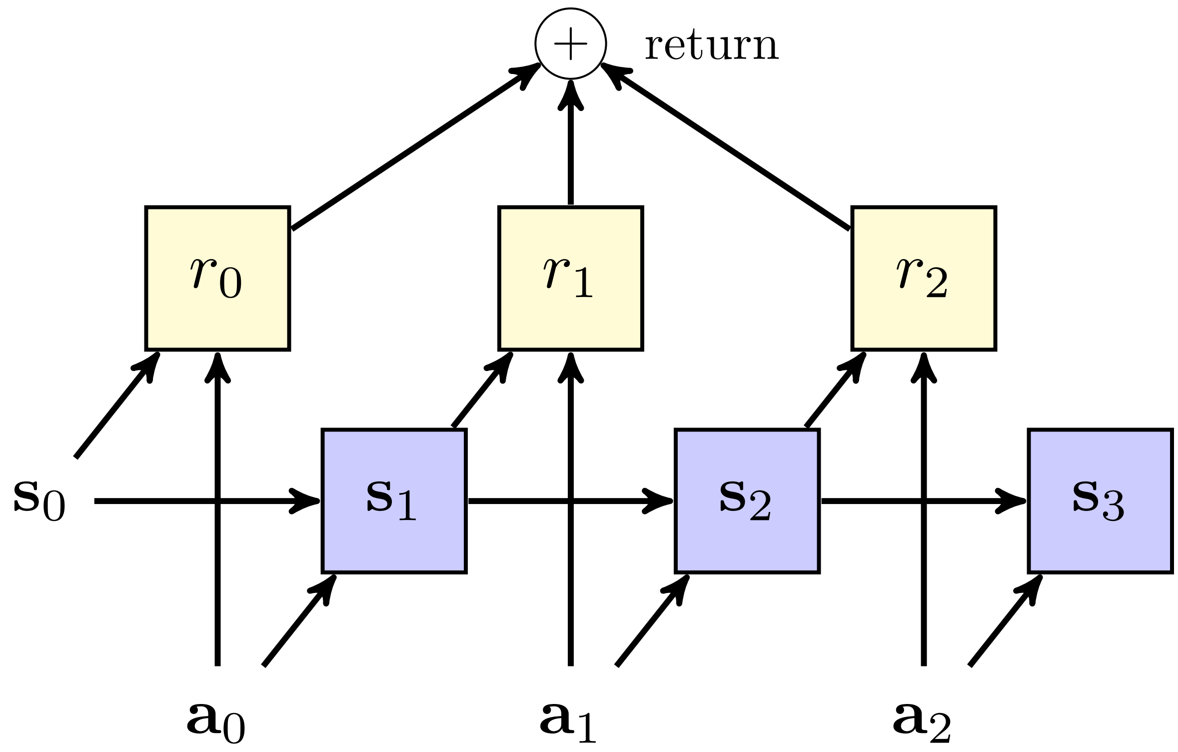
https://github.com/pyrddl-project/pyRDDLGym-symbolic

JaxPlanner

pyRDDLGym-jax

Built-in Model-based Planner

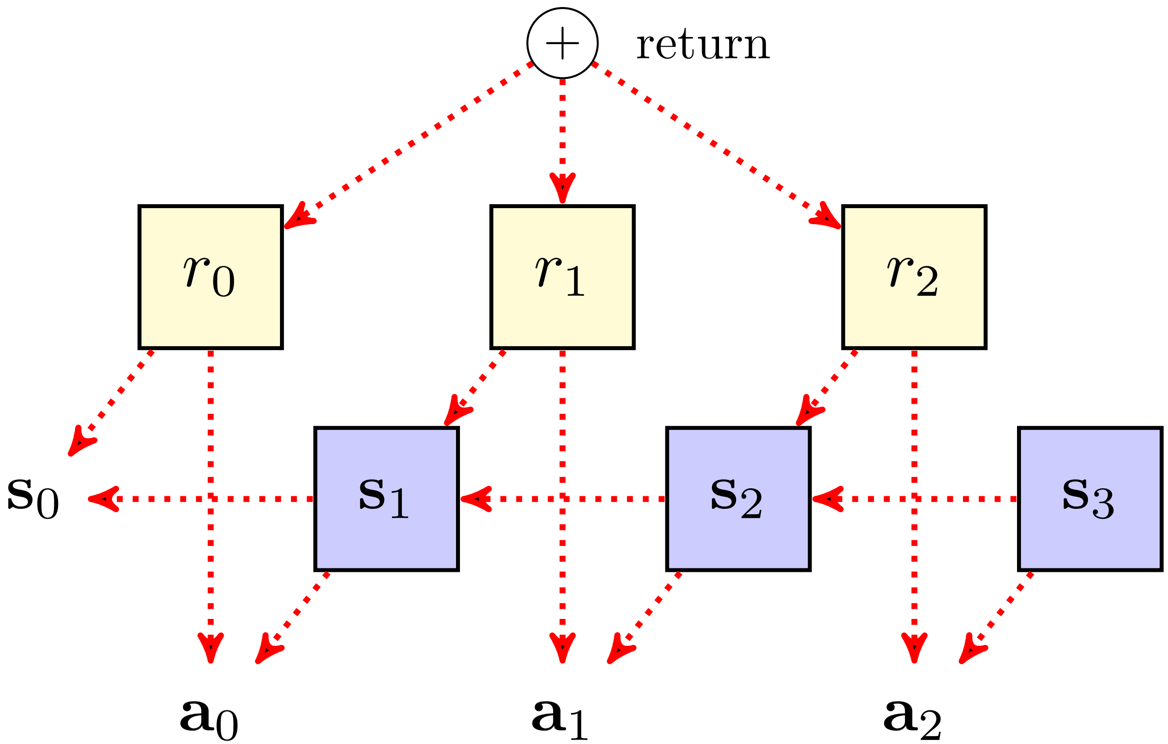
Simulate: Given plan $a_0, a_1, \ldots,$ simulate states s_t and reward r_t



Dynamic Bayes' Net (DBN)

Built-in Model-based Planner

Optimize: Adjust a_t based on the return gradient

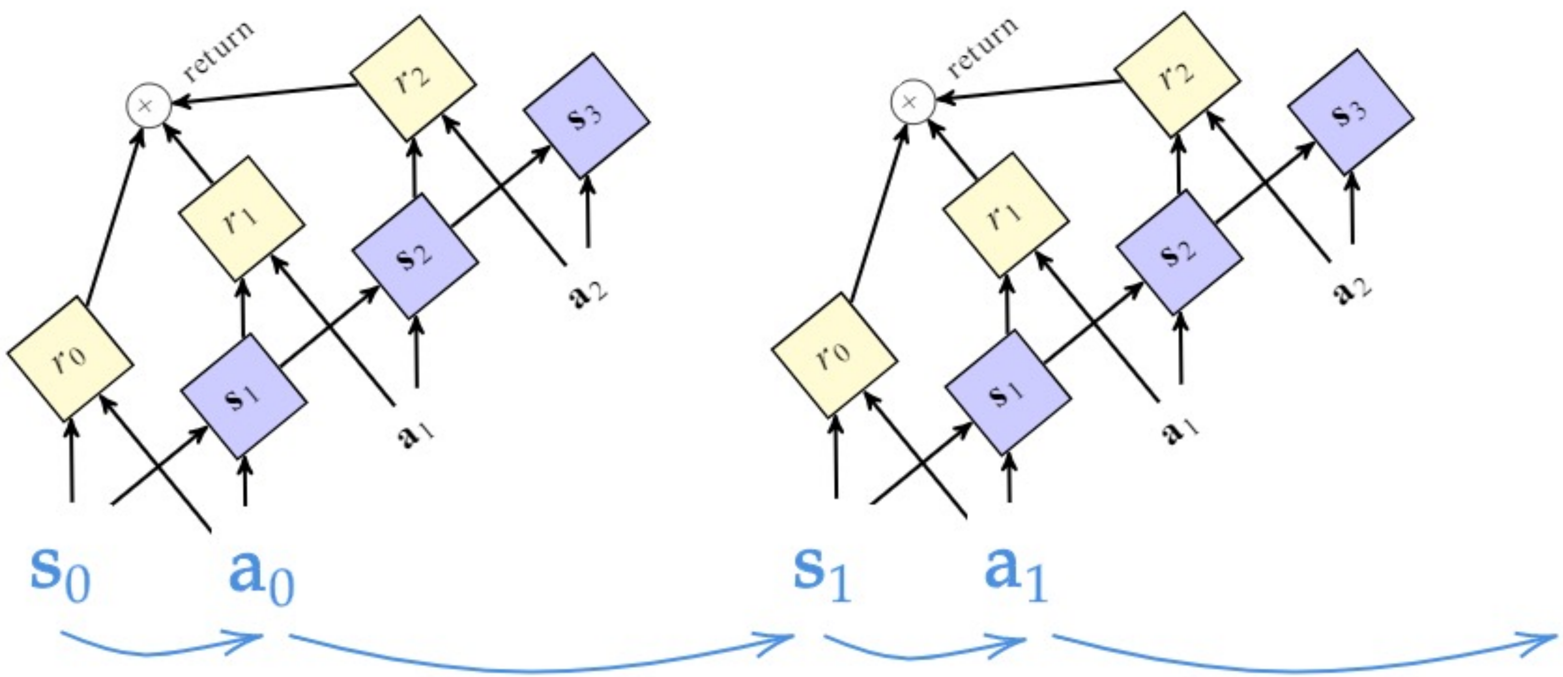


$$a'_t = a_t + \eta \nabla_{\mathbf{a}_t} \sum_{\tau} r_\tau$$

Wu, Ga, Buser Say, and Scott Sanner. "Scalable planning with tensorflow for hybrid nonlinear domains." *NeurIPS* (2017).

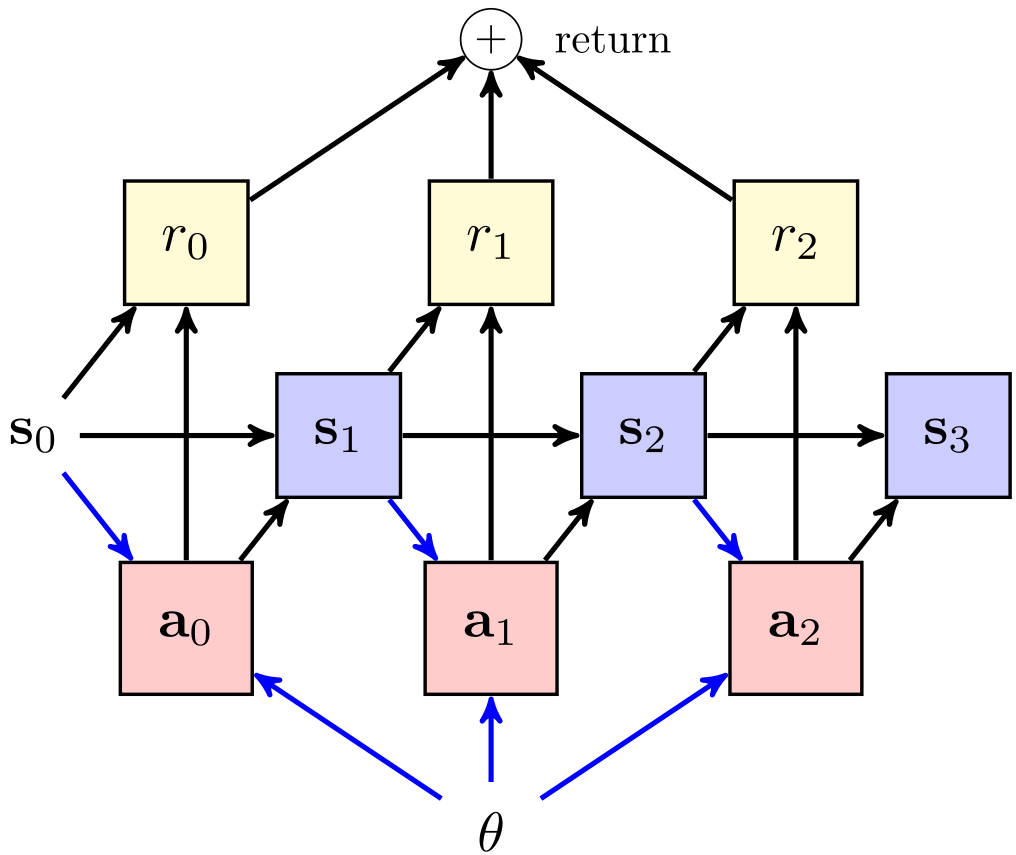
Built-in Model-based Planner

Closed-loop plan: Periodic re-planning (rolling horizon)



Built-in Model-based Planner

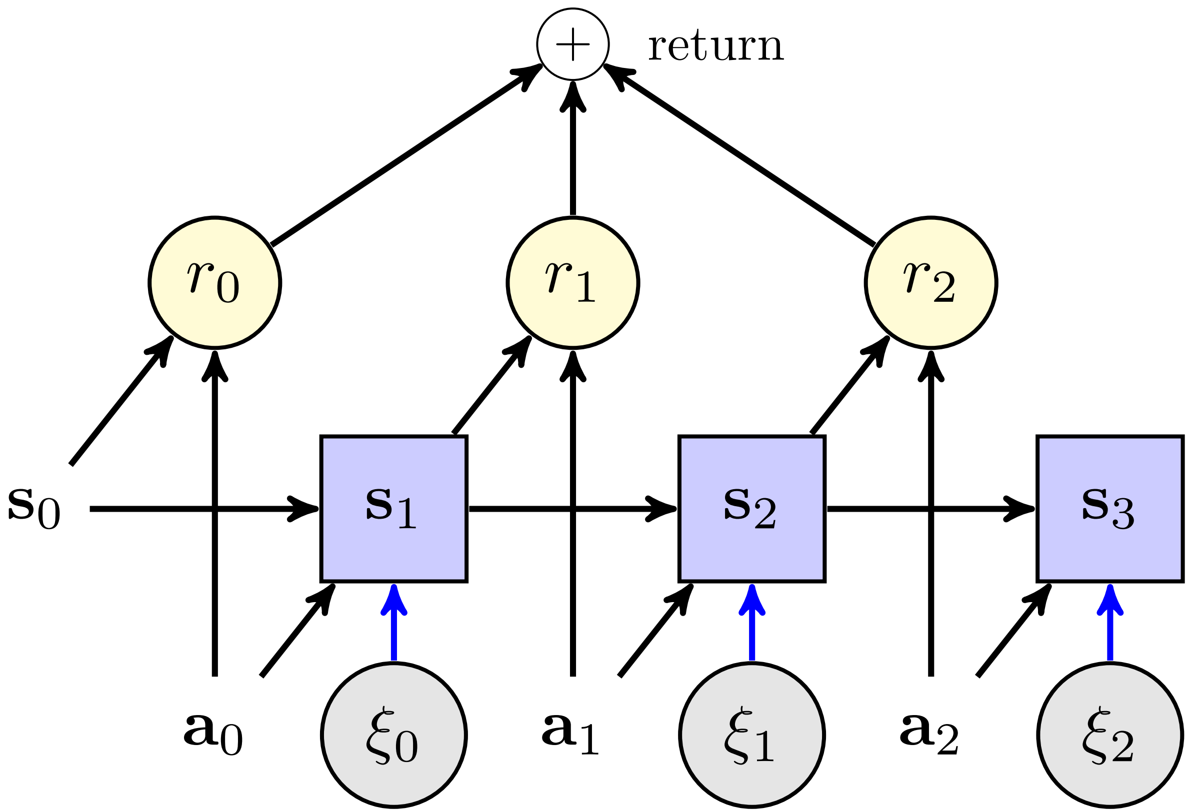
Closed-loop plan: Deep reactive policy



Bueno, T. P., de Barros, L. N., Mauá, D. D., and Sanner, S. Deep Reactive Policies for Planning in Stochastic Nonlinear Domains. *AAAI* (2019).

Built-in Model-based Planner

Stochastic domains: Use the reparameterization trick



$$\mathcal{N}(\mathbf{s}+\mathbf{a}, \sigma^2\mathbf{I})$$

⬇

$$\mathbf{s}+\mathbf{a}+\sigma\mathcal{N}(\mathbf{0},\mathbf{I})$$

$$\mathbf{s}_{t+1} \sim p(\cdot\,|\mathbf{s}_t, \mathbf{a}_t)$$

⬇

$$\mathbf{s}_{t+1} = \phi(\mathbf{s}_t, \mathbf{a}_t, \xi_t)$$

Bueno, T. P., de Barros, L. N., Mauá, D. D., and Sanner, S. Deep Reactive Policies for Planning in Stochastic Nonlinear Domains. *AAAI* (2019).

Built-in Model-based Planner

“Not all domains are born continuous”
– Anonymous

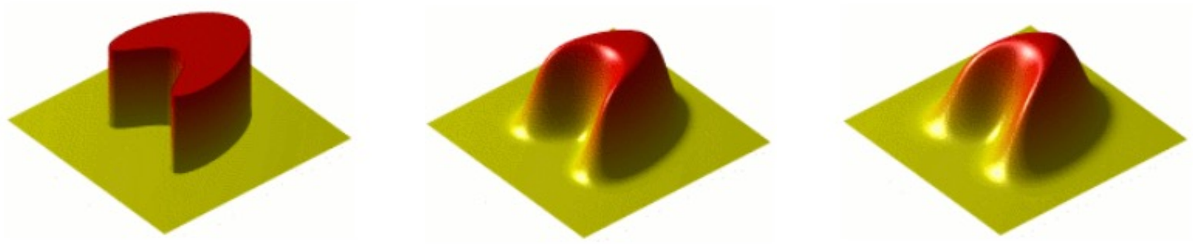
```
cpfs {
  burning'(?x, ?y) = if ( put-out(?x, ?y) )  // Intervention to put out fire?
    then false
      else if (~out-of-fuel(?x, ?y) ^ ~burning(?x, ?y))  // Ignition of a new fire? Depends on neighbors.
        then [if (TARGET(?x, ?y) ^ ~exists_{?x2: x-pos, ?y2: y-pos} (NEIGHBOR(?x, ?y, ?x2, ?y2) ^ burning(?x2, ?y2)))
          then false
          else Bernoulli( 1.0 / (1.0 + exp[4.5 - (sum_{?x2: x-pos, ?y2: y-pos} (NEIGHBOR(?x, ?y, ?x2, ?y2) ^ burning(?x2, ?y2)))]) ) ]
        else
          burning(?x, ?y);  // State persists

  out-of-fuel'(?x, ?y) = out-of-fuel(?x, ?y) | burning(?x, ?y) | (~TARGET(?x, ?y) ^ cut-out(?x, ?y));
};
```

Built-in Model-based Planner

T-norm Fuzzy logic

$$f_c: \{0,1\}^n \rightarrow [0,1]$$

RDDL Operation	Continuous Expression
$a \wedge b$	$a * b$
$\neg a$	$1 - a$
IF c THEN a ELSE b	$c * a + (1 - c) * b$
forall_{?p : type} x(?p)	$\prod_{?p} x(?p)$
a > b	$sigmoid\left(\frac{a - b}{\tau}\right)$



Hands-on

Colab notebook

- Basic pyRDDLGym usage
- Modeling and execution
- JaxPlanner