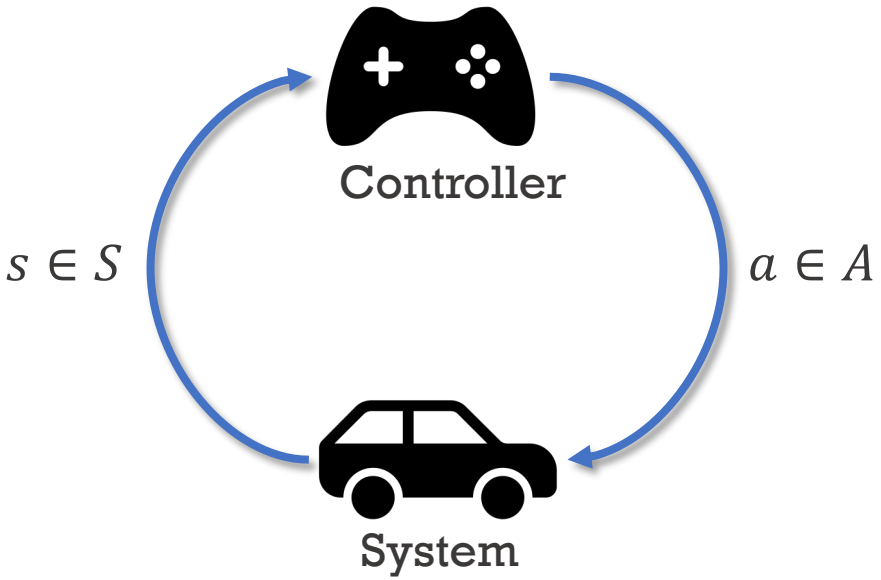
A Composable Specification Language for Reinforcement Learning Tasks

Kishor Jothimurugan, Rajeev Alur, Osbert Bastani

Penn
University of Pennsylvania

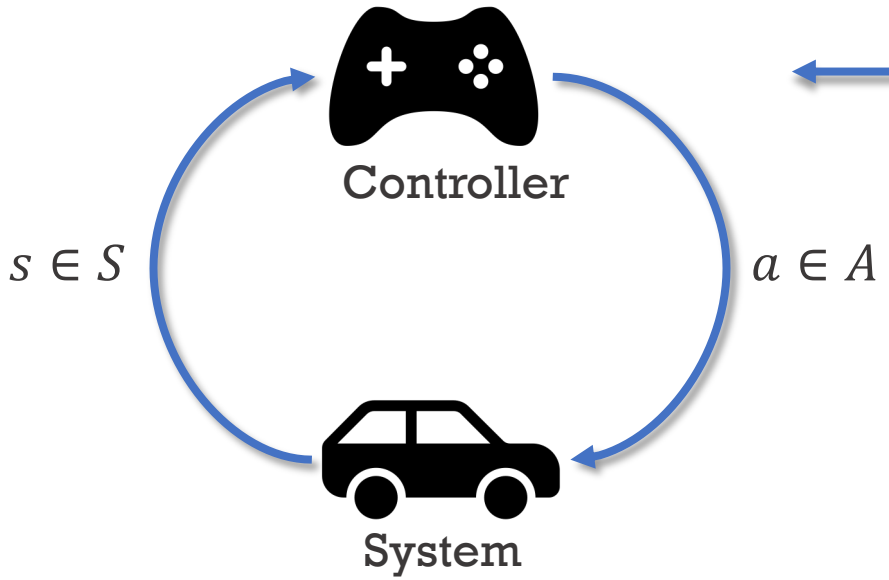
Control System

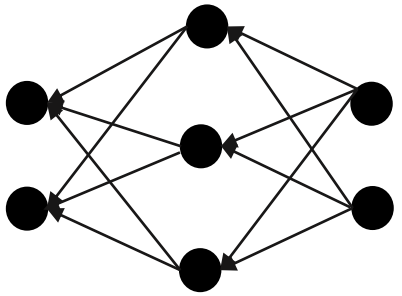


S = Set of System States
A = Set of Control Inputs

- Continuous states and actions
- System can be probabilistic
- Discrete Time
- Finite time horizon - T

Reinforcement Learning





Neural Network

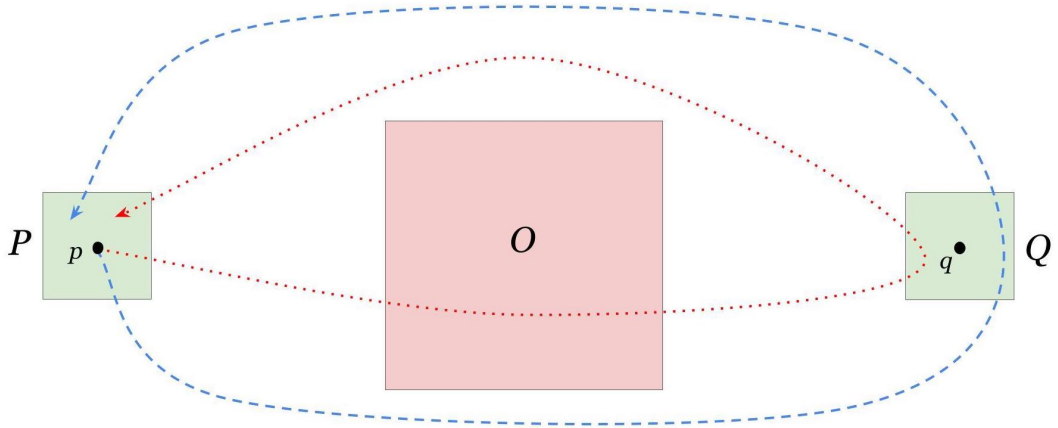
S = Set of System States
A = Set of Control Inputs

- Use neural networks to map states to actions
- Design reward function R mapping runs to rewards
- Learn NN parameters optimizing:

$$\pi^* \in \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{\rho \sim \mathcal{D}_\pi}[R(\rho)]$$

Reward Functions

- Too low-level as compared to logical specification
- No obvious way to compose rewards



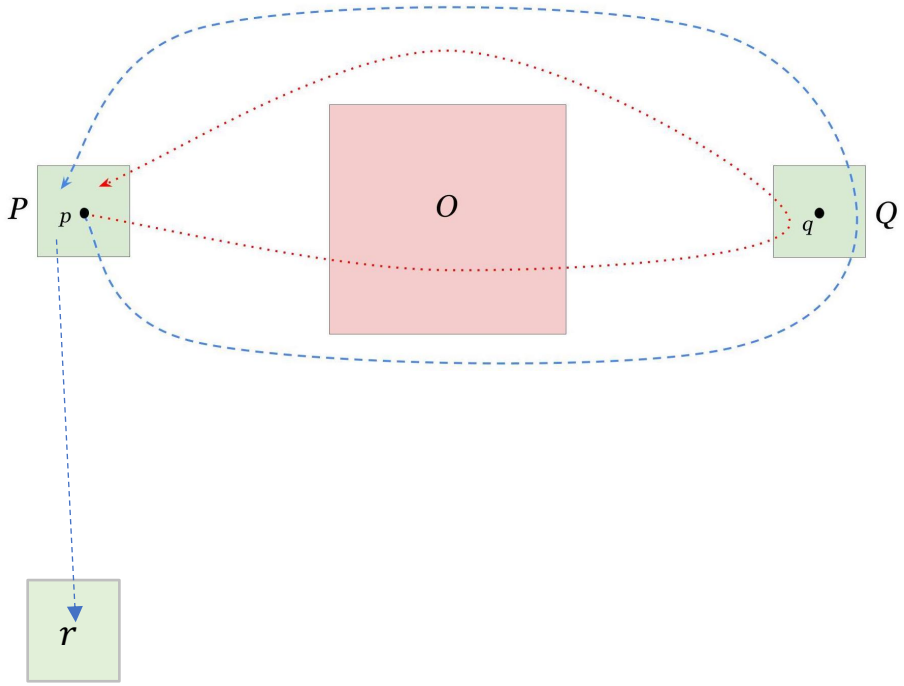
R_1 : Reward function for "Reach q"

R_2 : Reward function for "Reach p"

Reward function for "Reach q and then Reach p" ?

Need to generate reward function from a given logical specification

Need For Memory



- Specification: Reach q, then Reach p, then Reach r
- Controller maps states to actions
- Action at p depends on the history of the run

Solution: Add additional state component to indicate whether q has already been visited

Need to generate reward function from a given logical specification

Need to automatically infer the additional state components from the specification

Our Framework

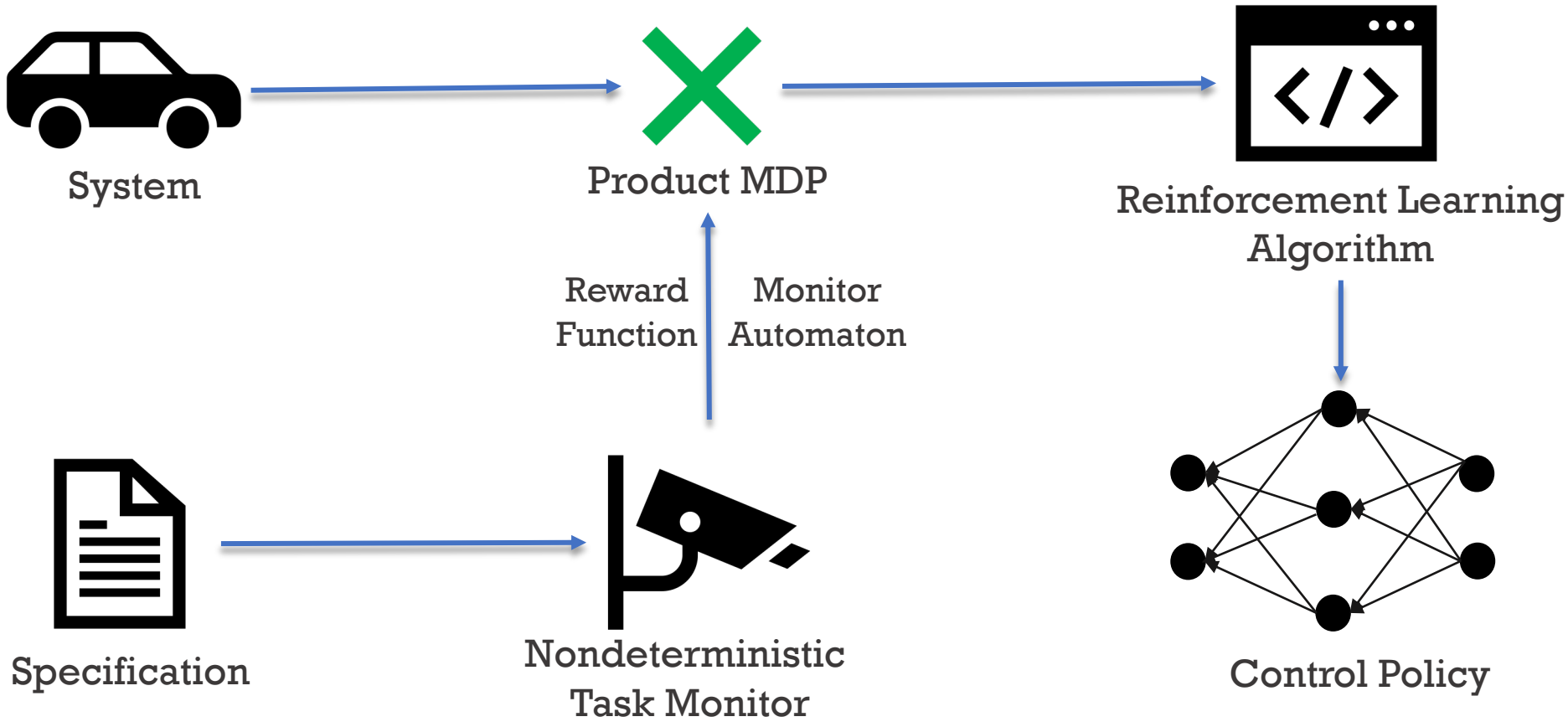
- System MDP $= (S, A, P, T, s_0)$ where $P(s, a, s') = \Pr(s'|s, a)$ given as a black-box forward simulator

- Specification φ given in our task specification language

Synthesizes a control policy π^* such that,

$$\pi^* \in \operatorname{argmax}_\pi \Pr[\rho \vDash \varphi]$$

Our Framework



Task Specification Language

$$\phi := \text{achieve } b \mid \phi_1 \text{ ensuring } b \mid \phi_1; \phi_2 \mid \phi_1 \text{ or } \phi_2$$

- Example base predicates:
  - $Near_q$ is satisfied if and only if the distance to q is less than 1
  - $Away_O$ is satisfied if and only if there is a positive distance to O
- Specification for navigation example:

$$(\text{achieve } Near_q; \text{achieve } Near_p) \text{ ensuring } Away_O$$

Quantitative Semantics

- Assume each base predicate $b \in P$ is associated with a quantitative semantics, $[\![b]\!]: S \to \mathbb{R}$ such that,

$$s \vDash b \text{ if and only if } [\![b]\!](s) > 0$$

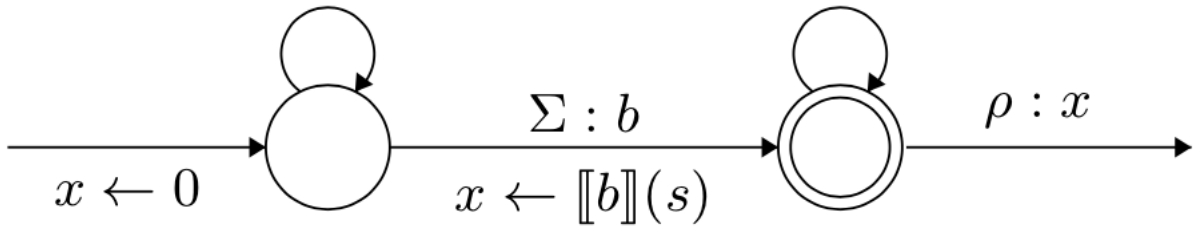
  - $[\![Near_q]\!](s) = 1 - dist(s, q)$
  - $[\![Away_O]\!](s) = dist(s, O)$

- Extend to positive Boolean combinations by,

  - $[\![b_1 \vee b_2]\!] = \max([\![b_1]\!], [\![b_2]\!])$
  - $[\![b_1 \wedge b_2]\!] = \min([\![b_1]\!], [\![b_2]\!])$

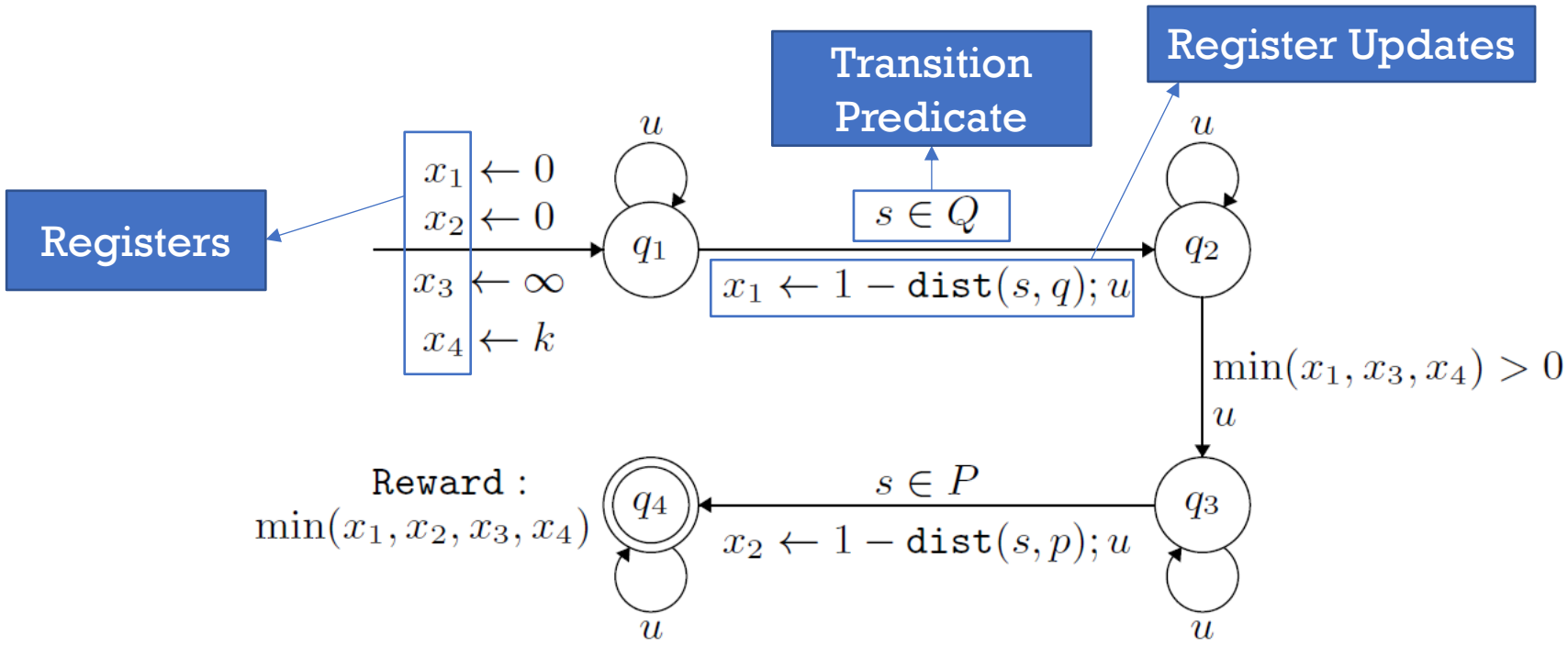
Task Monitor

- Finite State Machine
- Registers that store quantitative information
- Compilation similar to NFA construction from regular expressions



Task Monitor for ϕ = achieve b

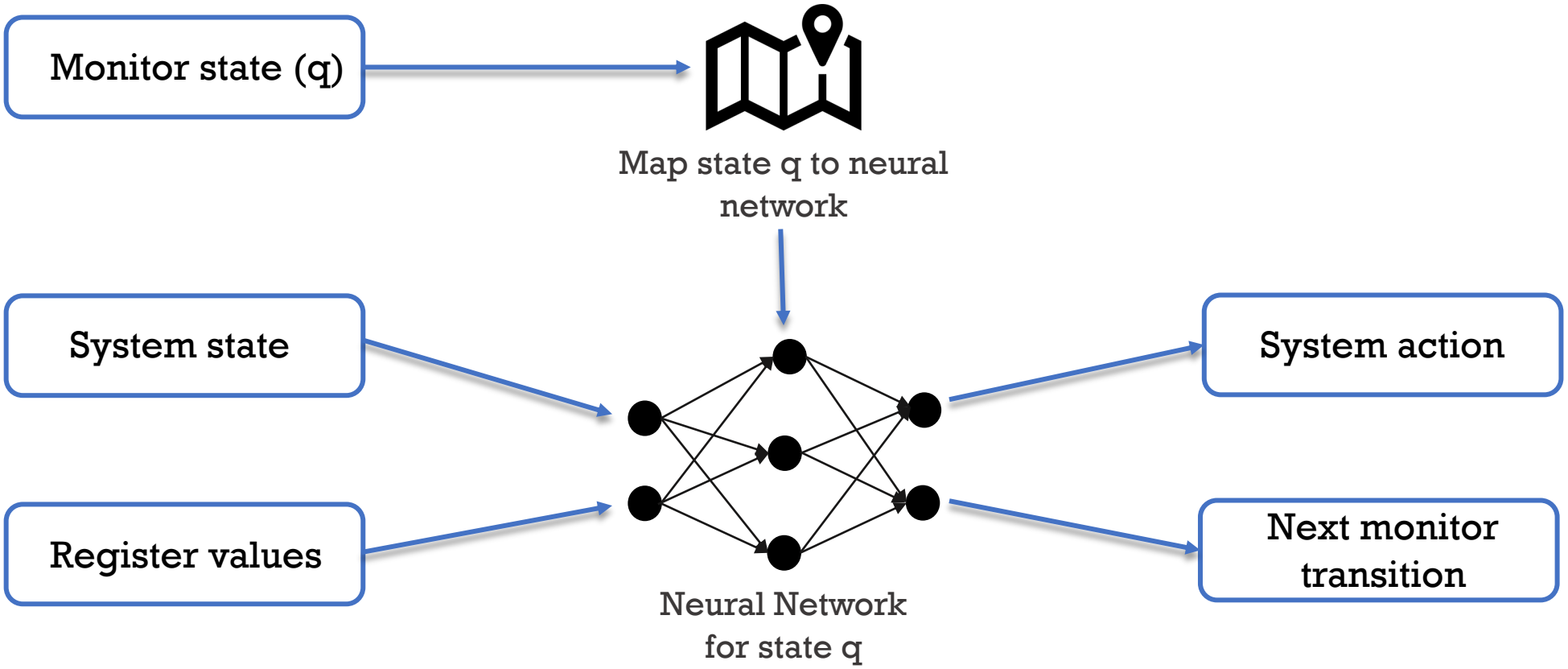
Task Monitor



Task monitor for $\left(\text{achieve } Near_q \text{ ; achieve } Near_p\right)$ ensuring $Away_O$

u: $x_3 \leftarrow \min(x_3, \mathtt{dist}(s, O))$

Extended Policy



Assigning Rewards

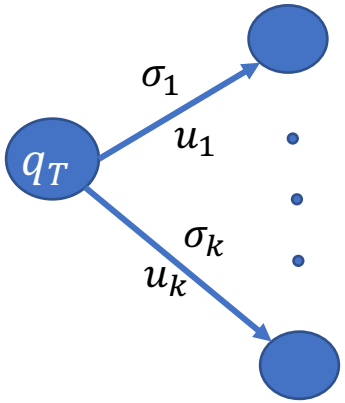
Given a sequence of extended system states, $\rho = (q_0, s_0, v_0) \to \cdots \to (q_T, s_T, v_T)$ what should be its reward?

- Case 1: (q_T is a final state) Reward is given by monitor
- Case 2: (q_T not a final state) Not all tasks have been completed
  - Suggestion 1: $R(\rho) = -\infty$
  - Suggestion 2: Find a reward function R' that preserves ordering of runs w.r.t. R,

$$R(\rho) > R(\rho') \text{ implies } R'(\rho) > R'(\rho')$$

Reward Shaping

Given $\rho = (q_0, s_0, v_0) \to \cdots \to (q_T, s_T, v_T)$ with q_T non-final,



Higher reward for states farther from start

$$R''(q_T)\,(s,v) = C_l + 2C_u\left(d_{q_T} - D\right) + \max_i \llbracket \sigma_i \rrbracket (s,v)$$

$$R'(\rho) = \max_{t:q_t = q_T} R''(q_T)(s_t, v_t)$$

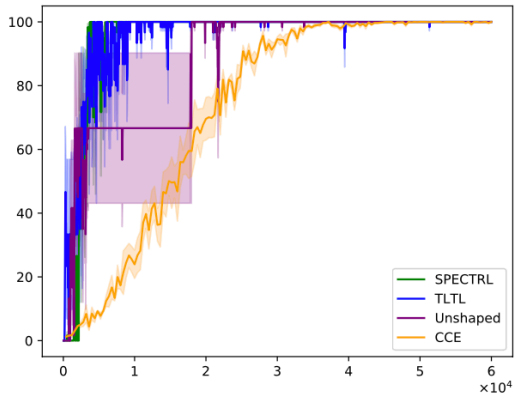
Prefer runs that get close to satisfying some predicate on transitions that make progress

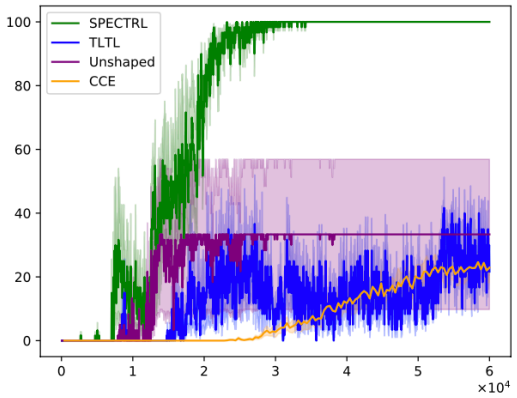
- d_q : Length of the longest path from q_0 to q without using self loops
- C_l : Lower bound for possible reward in any final state
- C_u : Upper bound on the third term for all q

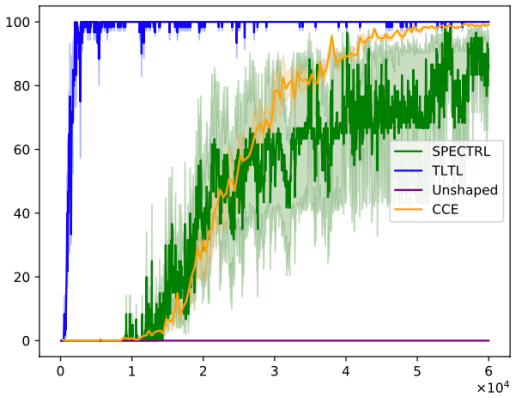
Experiments

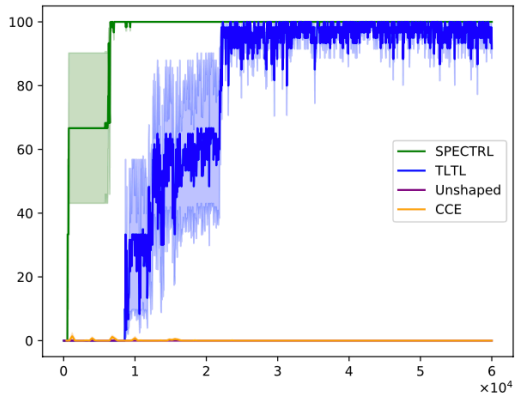
- Implemented our approach in a tool called SPECTRL (SPECifying Tasks for RL)
- Case study in the 2D navigation setting:
  - $S = \mathbb{R}^2$ and $A = \mathbb{R}^2$
  - Transitions given by $s_{t+1} = s_t + a_t + \varepsilon$ where ε is a small gaussian noise

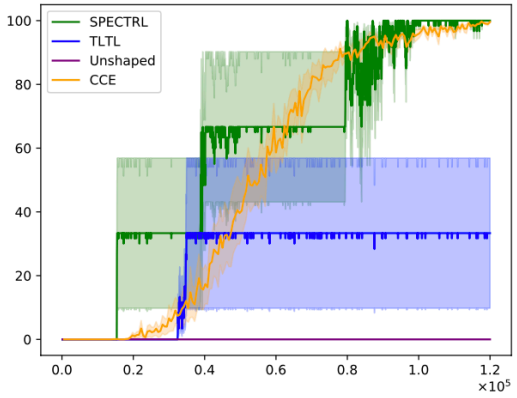
2D Navigation Tasks

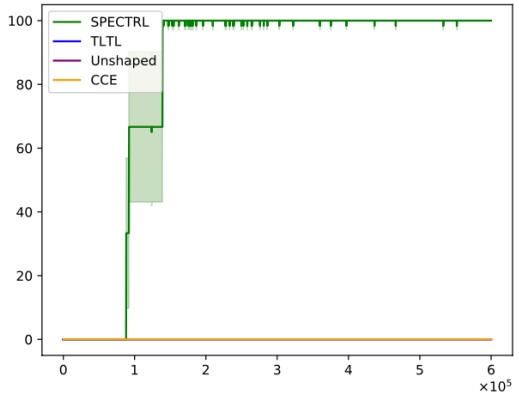


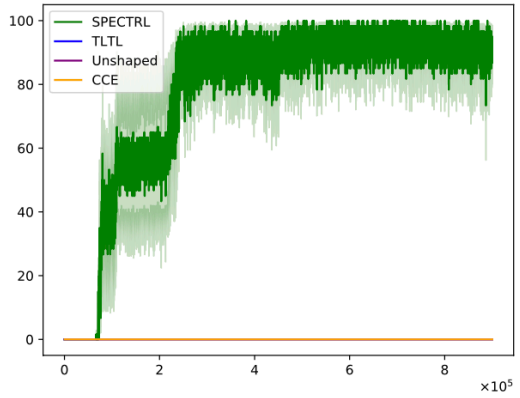








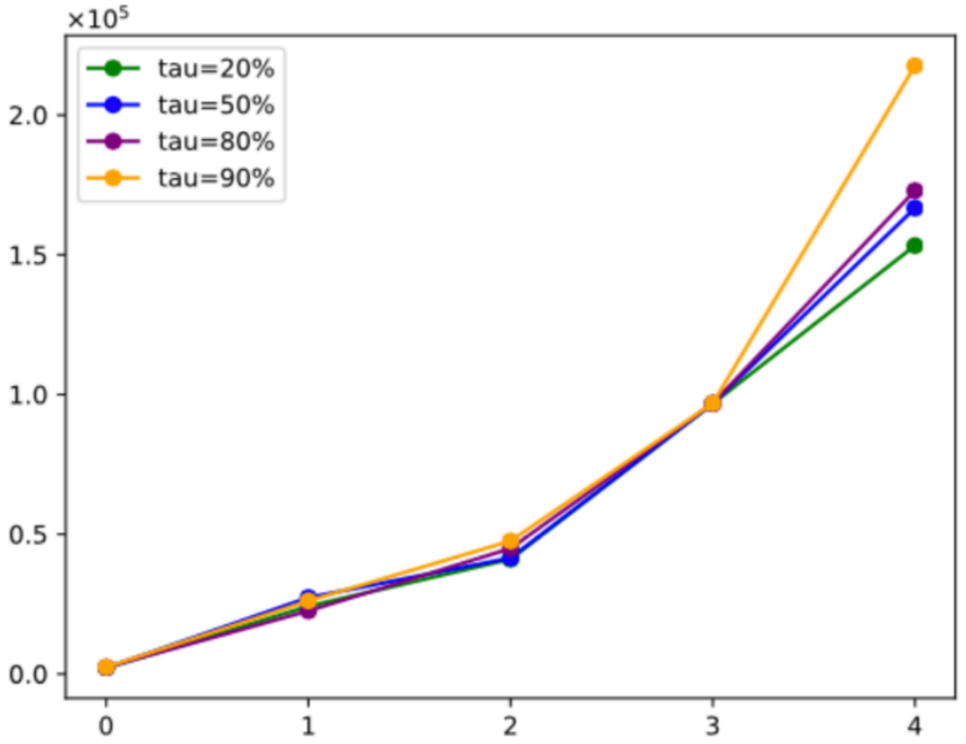




SPECTRL TLTL CCE

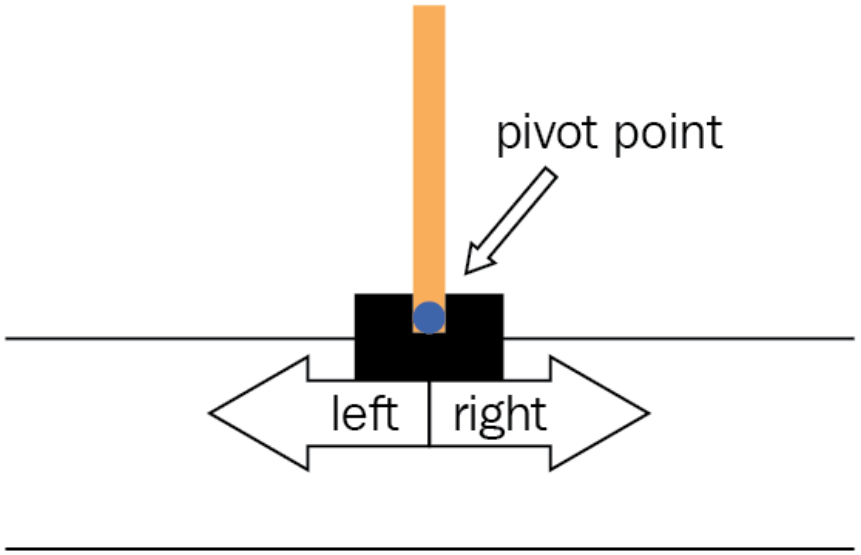
Learning curves for different tasks

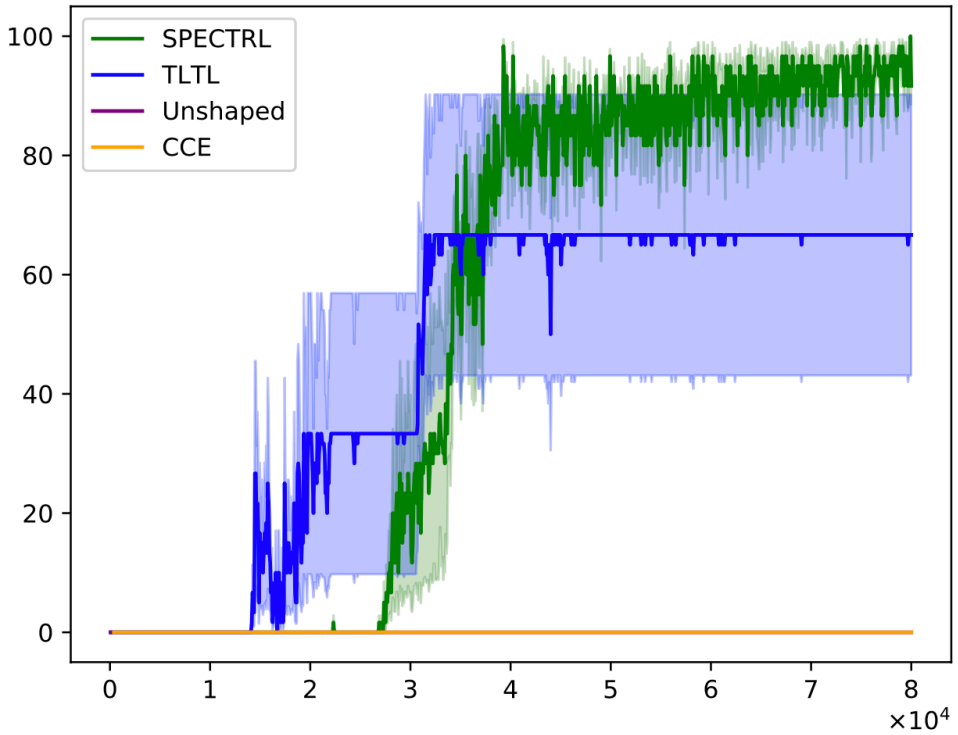
2D Navigation Tasks



Sample Complexity Curve
Y-axis denotes number of sample trajectories needed to learn
X-axis denotes number of nested goals

Cartpole





Learning Curve for Cartpole

Spec: Go to the right and return to start position without letting the pole fall

THANK YOU!