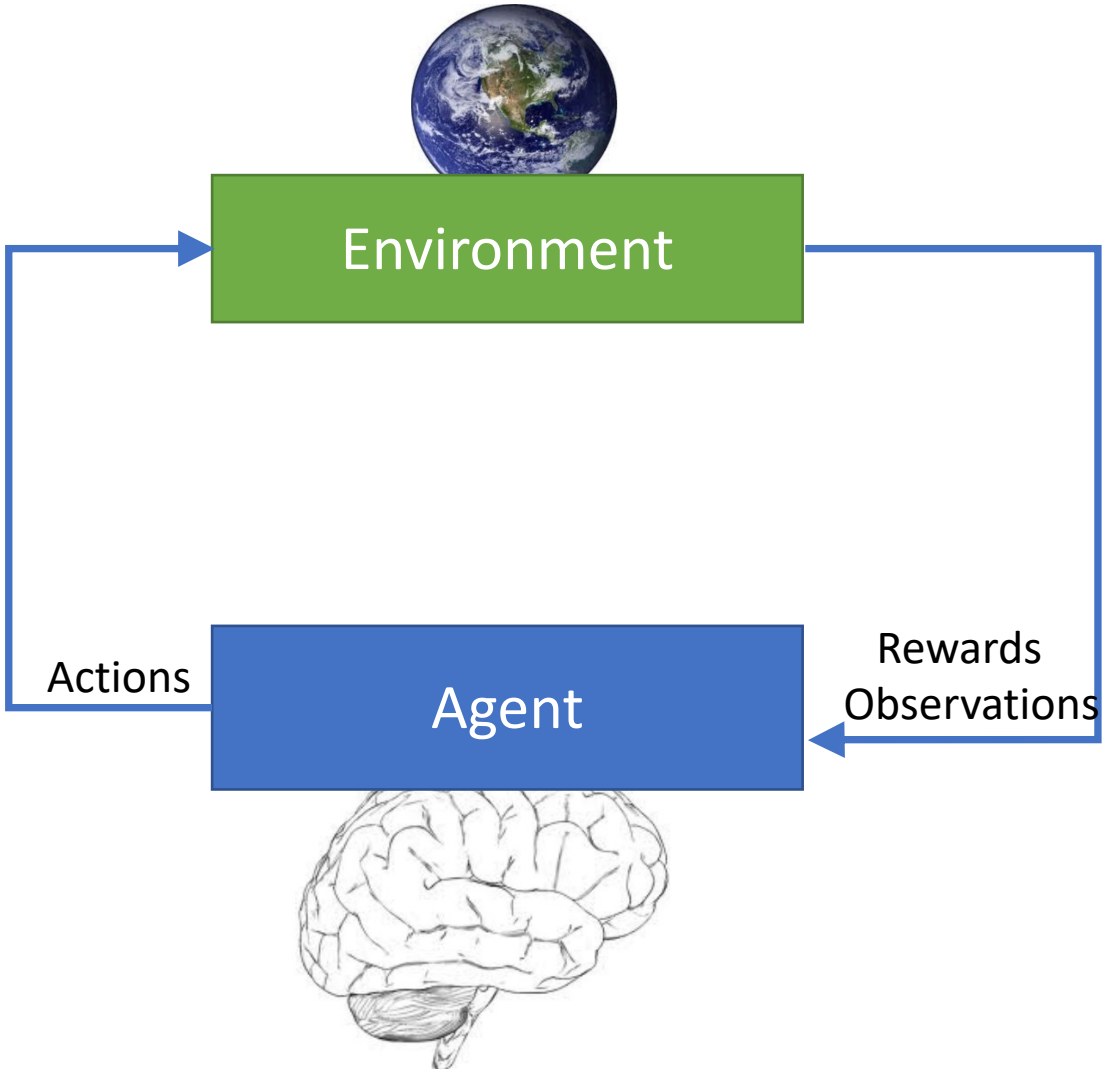
Cyber-Physical Systems

Laura Nenzi

Università degli Studi di Trieste
II Semestre 2022

Lecture 22: RL + TL

Reinforcement Learning



- RL is the theoretical model for learning from interaction with an uncertain environment
- aleatory (intrinsic) or epistemic (knowledge) uncertainty
- **Maximize the average reward function over a given time horizon**
- Very important notion of time horizon, it can change your goal
- There could be different reward to achieve the same goal

- RL Course by David Silver: https://www.youtube.com/watch?v=2pWv7GOvuf0&list=PLzuuYNsE1EZAXYR4FJ75jcJseBmo4KQ9-
- Reinforcement learning tutorial with demo: https://github.com/omerbsezer/Reinforcement_learning_tutorial_with_demo#FunctionApproximation
- **Coursera** https://www.coursera.org/specializations/reinforcement-learning
- **RL Youtube Course DeepMind** https://www.youtube.com/watch?v=TCCjZe0y4Qc&ab_channel=DeepMind (https://deepmind.com/learning-resources/reinforcement-learning-series-2021)

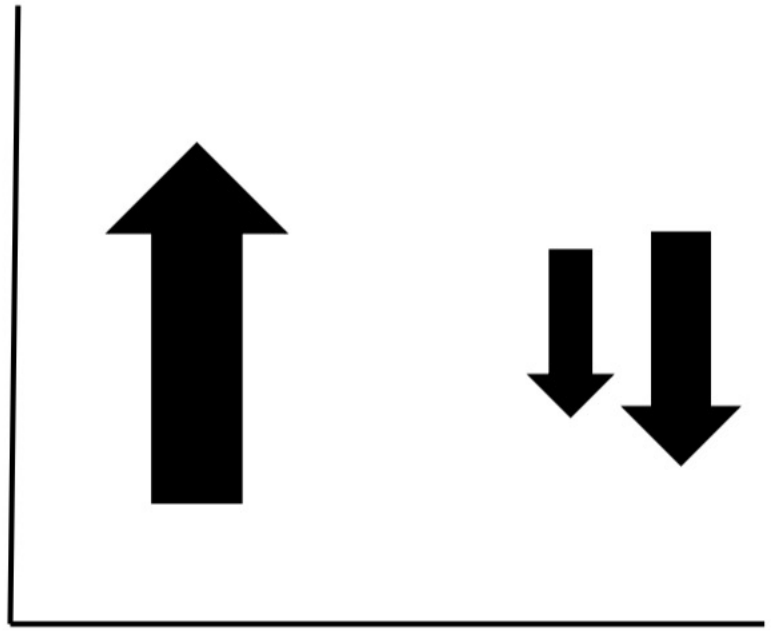
Reinforcement Learning and Temporal Logic

Challenges:

- Safe RL
- Reward Hacking
- Complex/Multi Tasks

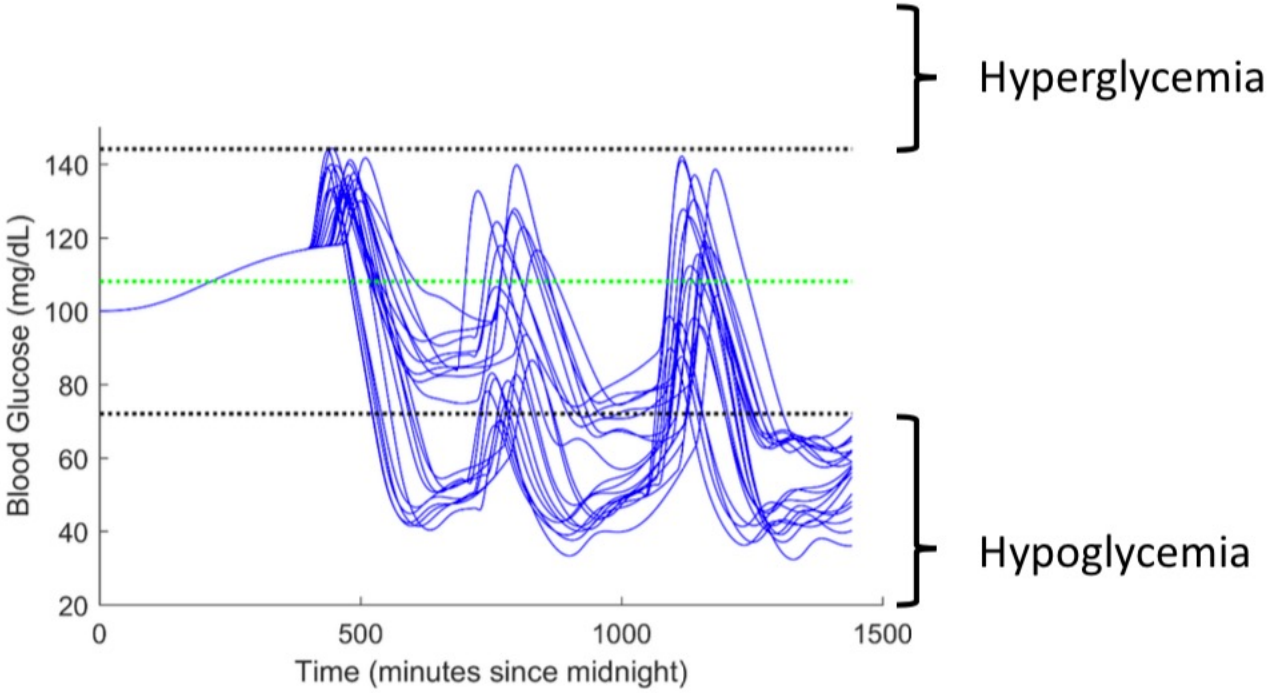
Safe Reinforcement Learning

Blood Glucose (sugar)



Eat Carbohydrates

Release Insulin

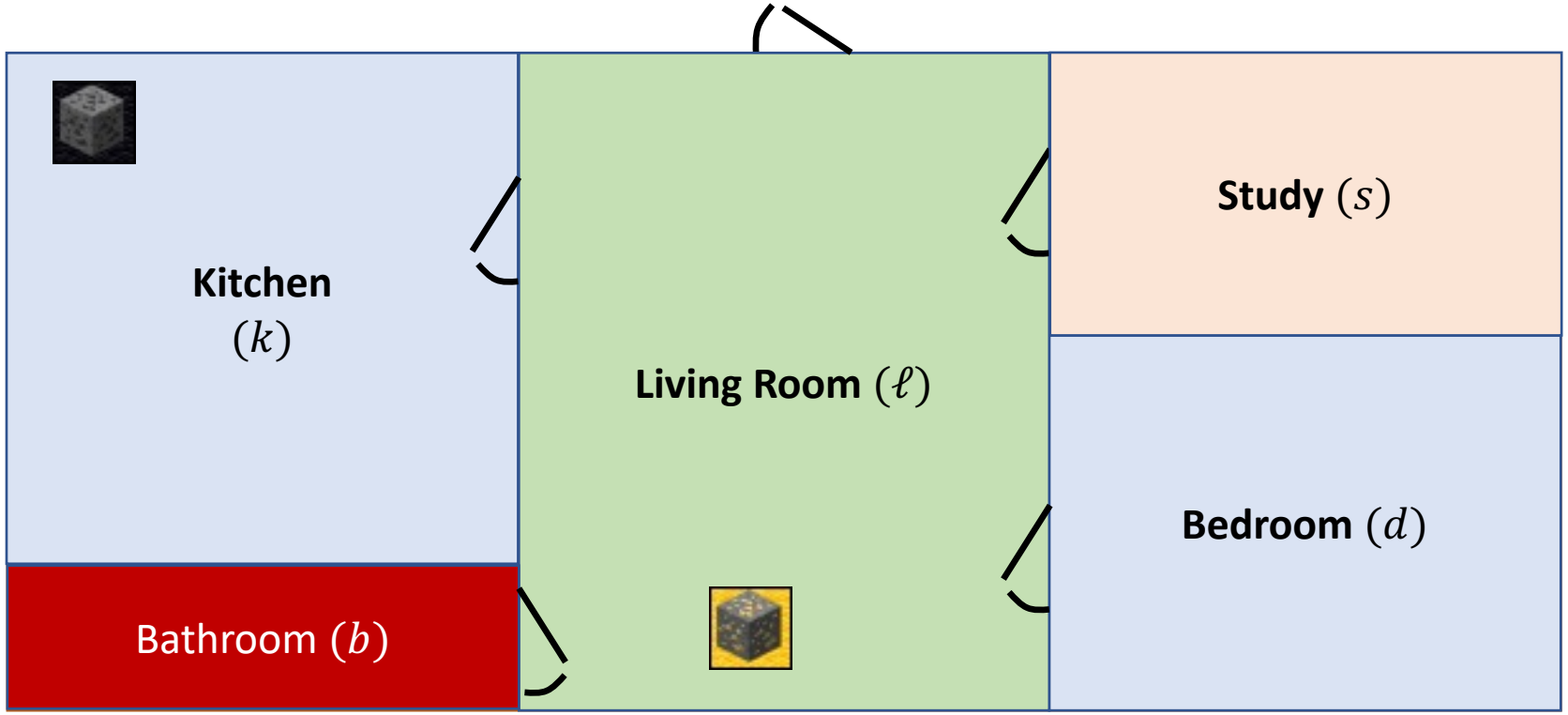


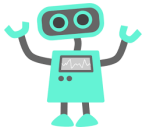
Reward Hacking

A policy that achieves high returns but against the designer's intentions

https://www.youtube.com/watch?v=92qDfT8pENs

Complex Tasks





Several Works with different motivations

- **Reward shaping using robusntess satisfaction**
- LTL constrained, Reward function remained the same
- multi-task-RL

Reward function is not enough

Description using a language can help..

- To define task better
- To learn more efficiently and precisely
- To transfer learning between tasks
- To be “safe”

General Idea

Reward Shaping problem:

Design $R(s, a)$ s.t. I can find $\pi^* \; s.t. \forall \; x, \pi^*(x)$ the "satisfaction" of x is maximised

Why important?

- Poorly design -> poorly convergence
- Learning unsafe or unrealistic action

Model-based Reinforcement Learning from Signal Temporal Logic Specifications

Parv Kapoor[1], Anand Balakrishnan[1] and Jyotirmoy V. Deshmukh[1]

Model-based RL from STL specification

1)

Learning a **deterministic** predictive **model** of the system dynamics using deep neural networks.

Given a state and a sequence of actions, such a predictive model produces a **predicted trajectory** over a user-**specified time horizon**.

Model-based RL from STL specification

2)

We use a **cost function based on** the quantitative semantics of **STL** to evaluate the optimality of the predicted trajectory.

We use a black-box optimizer that uses **evolutionary strategies** to identify the optimal sequence of actions (in an **MPC setting**).

Model-based RL from STL specification

3)

We demonstrate the efficacy of our approach on a number of examples from the robotics and autonomous driving domains.

Model-based RL from STL specification

1. Given a dataset $\mathcal{D}$ on sample transitions (s, a, s') collected from simulations or real-world demonstrations.
2. Fit a model $\hat{F}_\theta(s_t, a_t)$ that takes the current state s_t and an action a_t, and outputs a distribution over the set of possible successor states $s_{t+\delta_t}$
3. MPC: let an action sequence be denoted $A_t^{(H)} = (a_t, \ldots, a_{t+H-1})$
   At every time step t during the execution of the controller for a finite *planning horizon* H we solve the following optimization problem
   - maximize $\rho(\hat{s}_t, a_t, \hat{s}_{t+1}, \ldots, a_{t+H-1}, \hat{s}_{t+H})$
   - where:
     $\hat{s}_t = s_t$ and
     $\hat{s}_{t+i+1} = \hat{F}(\hat{s}_{t+i}, a_{t+i}), \quad \forall\, i \in 0, \ldots, H-1$

ew candidate solutions are sampled according to a multivariate normal distribution in . Recombination amounts to selecting a new mean value for the distribution. Mutation amounts to adding a random vector, a

Non-Linear Optmization Techiniques

- Monte-Carlo methods [25], the Cross-Entropy Method [26], or evolutionary strategies, like CMA-ES [27] and Natural Evolutionary Strategies [28].
- This paper uses CMA-ES: Covariance Matrix Adaptation Evolution Strategy
- In contrast to most classical methods, fewer assumptions on the nature of the underlying objective function are made.

CMA-ES: Covariance Matrix Adaptation Evolution Strategy

- ES:
  - Sampling according to a multivariate normal distribution.
  - Mutation amounts to adding a random vector, a perturbation with zero mean.
  - Recombination amounts to selecting a new mean value for the distribution.
  - Pairwise dependencies between the variables in the distribution are represented by a covariance matrix.
- CMA-ES : Method to update the covariance matrix of this distribution.
- Only the ranking between candidate solutions is exploited for learning the sample distribution and neither derivatives nor even the function values themselves are required by the method.

Robust control Synthesis

A framework that combines the use of:

- model-based reinforcement learning (MBRL)
- sampling-based model predictive control (MPC)
- to maximize the robustness value of a trajectory against a given STL formula via CMA-ES

Learning the System Dynamics

Algorithm 1 Learning the system dynamics model.

Initialize empty dataset, $\mathcal{D}$
for $i \in 1, \ldots, N_{\text{traj}}$ **do**
    for Each time step t **do**
        $\boldsymbol{a}_t \sim \text{Uniform}(\cdot), \boldsymbol{s}_{t+1} \sim \text{Env}(\boldsymbol{s}_t, \boldsymbol{a}_t)$
        $\mathcal{D} \leftarrow (\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$
    end for
end for
$\theta \leftarrow \text{SGD}(\mathcal{D})$

The out put θ is the vector of parameters of the NN

The learned dynamics, $\hat{F}_\theta$, is *deterministic*

Training the model

We train the model by minimizing the sum of squared error loss for each transition in the dataset:

$$\mathcal{L}(\theta) = \sum_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s'})\in\mathcal{D}_{\text{train}}} \left(\boldsymbol{s'} - \hat{F}_\theta(\boldsymbol{s},\boldsymbol{a})\right)^2$$

The loss minimization is carried out using stochastic gradient descent, where the dataset is split into randomly sampled batches and the loss is minimized over these batches.

Sampling-based Model Predictive Control

Algorithm 2 Model Predictive Control using $\hat{F}_\theta$.

for Each time step t in episode **do**
  for $1 \dots N_{\text{iter}}$ and $1 \dots N_{\text{samples}}$ **do**
    $\boldsymbol{A}_t^{(H)} = (\boldsymbol{a}_t, \dots, \boldsymbol{a}_{t+H-1}) \sim$ CMA-ES()
    for $i \in 0, \dots, H-1$ **do**
      $\hat{\boldsymbol{s}}_{t+i+1} = \hat{F}_\theta(\hat{\boldsymbol{s}}_{t+i}, \hat{\boldsymbol{a}}_{t+i})$
    end for
    Compute cost $\rho(\hat{\boldsymbol{s}}_t, \boldsymbol{a}_t, \hat{\boldsymbol{s}}_{t+1}, \dots, \boldsymbol{a}_{t+H-1}, \hat{\boldsymbol{s}}_{t+H})$
    Update CMA-ES()
  end for
  Execute first action $\boldsymbol{a}^*$ from optimal sequence $\boldsymbol{A}_t^{(H)} \sim$ CMA-ES()
end for

Cartpole

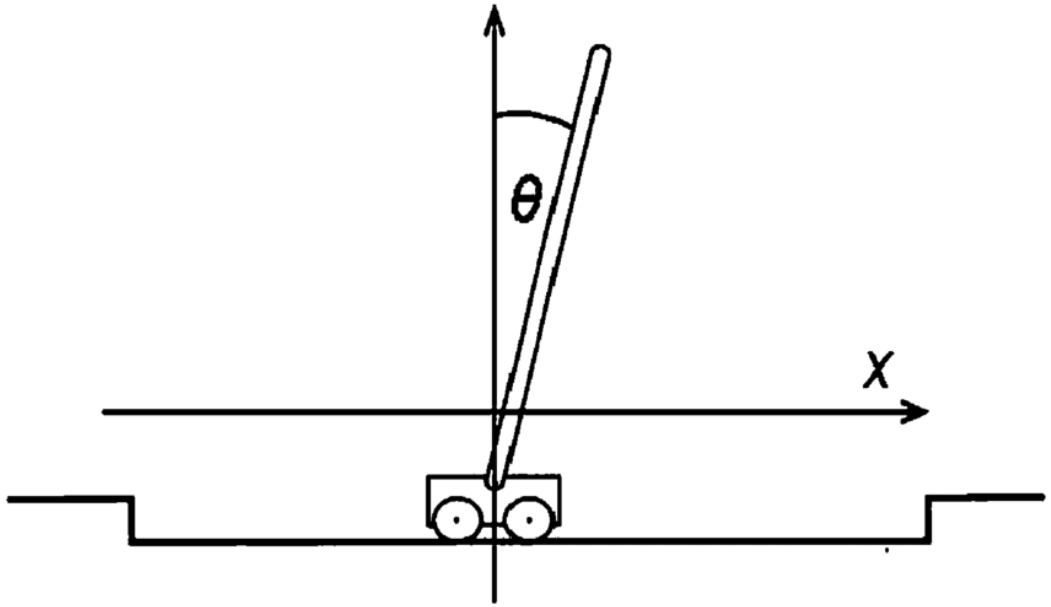
A pole is attached to a cart on an unactuated joint

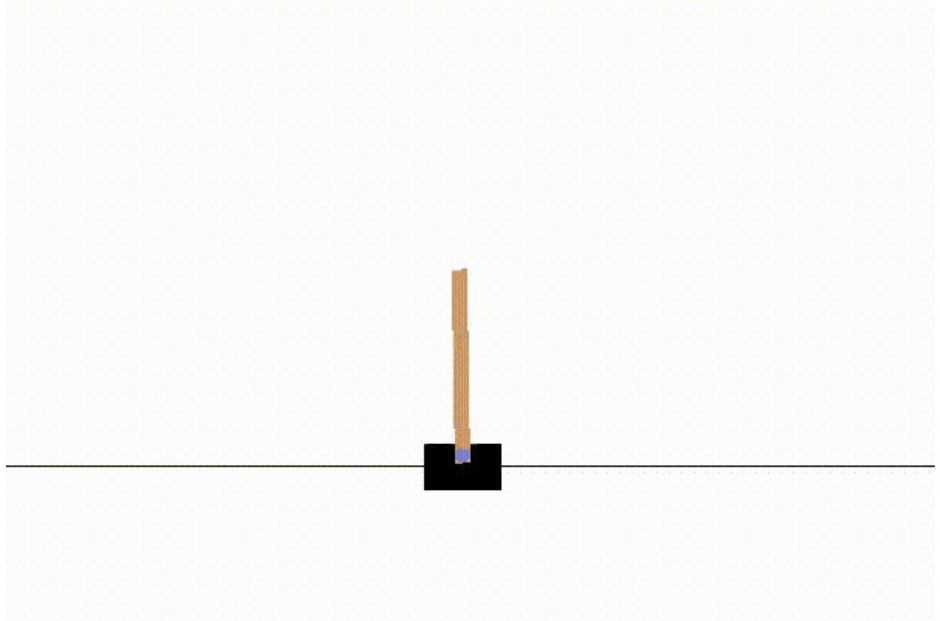
The cart moves on a frictionless track

It is controlled by applying a force to push it left or right on the track.

$$s = (\theta, x, \theta', x')$$

$$\varphi \coloneqq \mathrm{G}(|x| < 0.1 \wedge |\theta| < 12^{\circ})$$





Mountain Car

A car stuck at the bottom of a valley between two mountains.

The car can be controlled by applying a force pushing it left or right

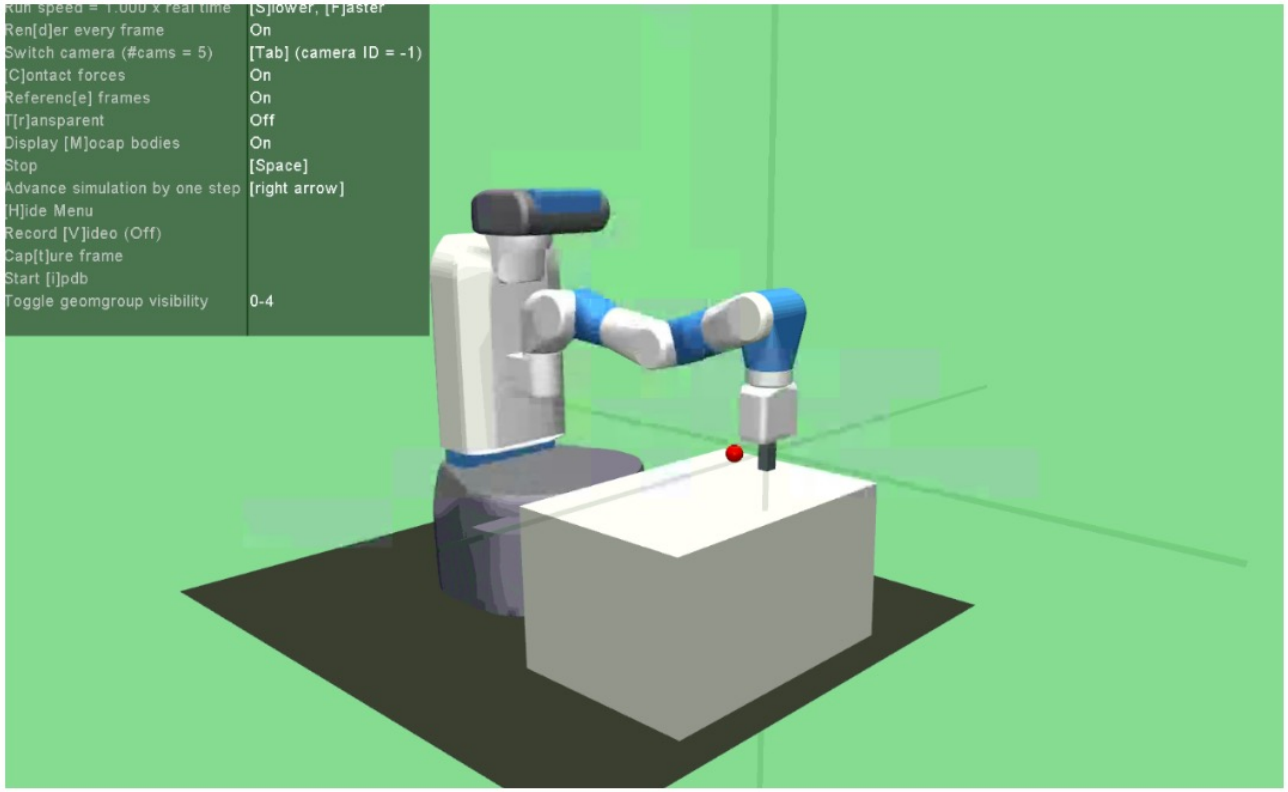
$$s = (x, x')$$

$$\varphi := F(x > 0.4)$$

Fetch robot

The goal of the environment is design a controller to move the arm of a simulated Fetch manipulator robot to a region in 3D space

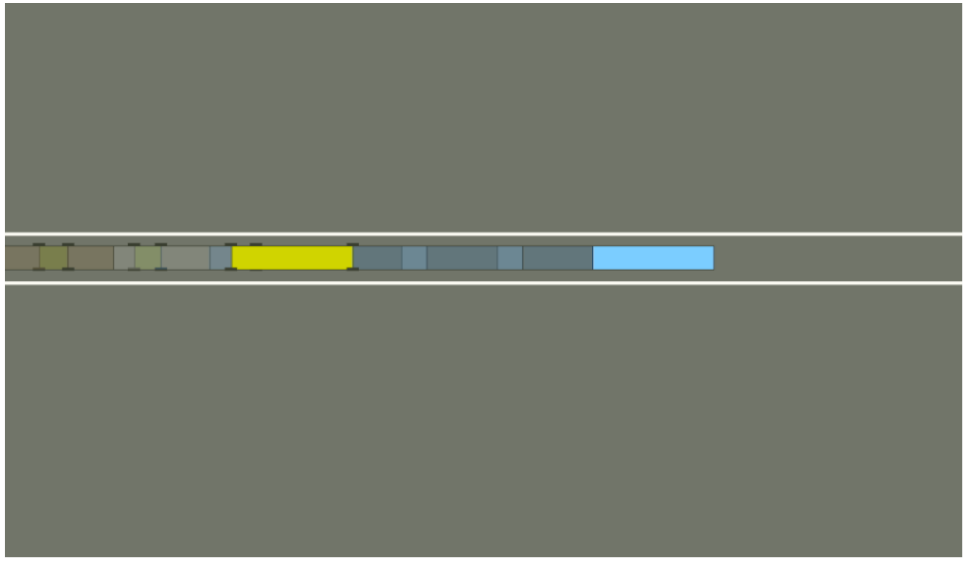
The state vector of the environment is a 16 dimensional vector containing the position and orientation of the robots joints and end-effector



$$\varphi := F(|x_g - x| < 0.1 \wedge |y_g - y| < 0.1 \wedge |z_g - z| < 0.1)$$

Adaptive Cruise-Control

The goal is to synthesize a controller for a car (the *ego* car) that safely does cruise control in the presence of an adversarial (or *ado*) car ahead of it.



The environment itself is a single lan. The *ego* car is controlled only by the acceleration applied to the car.

The *ado* agent ahead of the car accelerates and decelerates in an erratic manner, in an attempt to cause a crash.

$$s = (x_{ego}, v_{ego}, a_{ego}, d_{rel}, v_{rel})$$

$$\varphi := FG(50 > drel > 15)$$

TABLE I
HYPERPARAMETERS AND RESULTS.

Environment	**Hyperparameters**				**Results** (averaged across 30 runs)	
	No. of training trajectories N_{traj}	No. of optimizer iterations N_{iter}	No. of optim. samples per iteration N_{samples}	Horizon H	Trajectory Robustness ρ	Vanilla rewards[a]
Cartpole	2000	5	1000	10	$0.069 \pm 8\text{e}{-3}$	200.0 ± 0
Mountain Car	2000	2	1000	50	$0.047 \pm 1\text{e}{-3}$	89.06 ± 4.57
Fetch	2000	7	500	10	$0.067 \pm 6\text{e}{-3}$	–
ACC	400	7	500	2	7.679 ± 5.47	–
Parking Lot	400	5	5	5	$1.69\text{e}{-2} \pm 3.8\text{e}{-3}$	–

Q-learning RL with TL

- D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, “Q-Learning for robust satisfaction of signal temporal logic specifications,” in *2016 IEEE, CDC*, Dec. 2016, pp. 6565–6570.

  An extension to Q-learning where STL robustness is directly used to define reward functions over trajectories in an MDP.

- X. Li, C.-I. Vasile, and C. Belta, “Reinforcement learning with temporal logic rewards,” in *2017 IEEE/RSJ International Conference on Intelli- gent Robots and Systems (IROS)*, Sept. 2017, pp. 3834–3839.

  Propose a method that augments an MDP with finite trajectories, and defines reward functions for a truncated form of Linear Temporal Logic.

- A. Balakrishnan and J. V. Deshmukh, “Structured Reward Shaping using Signal Temporal Logic specifications,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 3481–3486.

  Translate STL specifications into locally shaped reward functions using a notion of “bounded horizon nominal robustness”

STL and discrete space

Q-Learning for Robust Satisfaction of Signal Temporal Logic Specifications

Derya Aksaray, Austin Jones, Zhaodan Kong, Mac Schwager, and Calin Belta

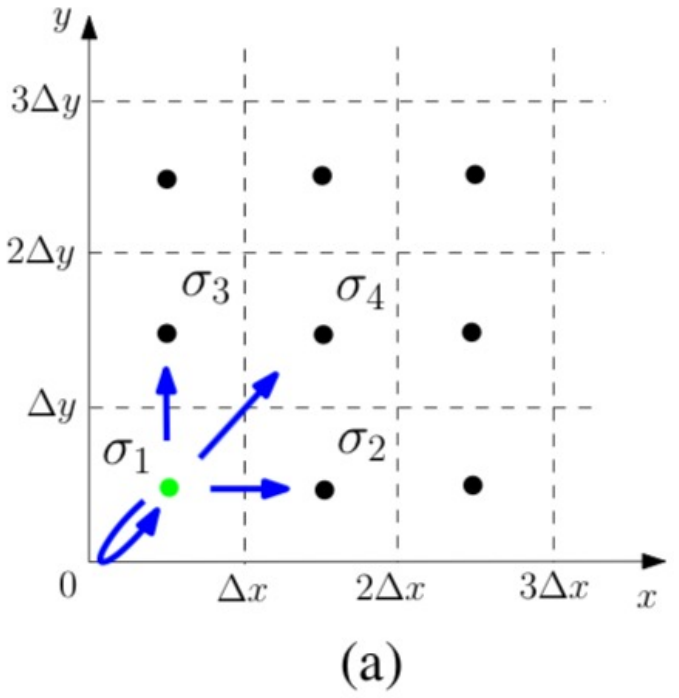
***Abstract*— In this paper, we address the problem of learning optimal policies for satisfying signal temporal logic (STL) specifications by agents with unknown stochastic dynamics. The system is modeled as a Markov decision process, in which the states represent partitions of a continuous space and the transition probabilities are unknown. We formulate two synthesis problems where the desired STL specification is enforced by maximizing 1) the probability of satisfaction, and 2) the expected robustness degree, i.e., a measure quantifying the quality of satisfaction. We discuss that Q-learning is not directly applicable to these problems because, based on the quantitative semantics of STL, the probability of satisfaction and expected robustness degree are not in the standard objective form of Q-learning (i.e., the sum of instantaneous rewards). To resolve this issue, we propose an approximation of STL synthesis problems that can be solved via Q-learning, and we derive some performance bounds for the policies obtained by the approximate approach. Finally, we present simulation results to demonstrate the performance of the proposed method.**

to describe tasks involving bounds on physical parameters and time intervals [8]. An example STL specification is "Within t_1 seconds, a region in which y is less than p_1 is reached, and regions in which y is larger than p_2 are avoided for t_2 seconds." STL is also endowed with a metric called *robustness degree* that quantifies how strongly a given trajectory satisfies an STL formula as a real number rather than just providing a *yes* or *no* answer [10], [8]. This measure enables the use of continuous optimization methods to solve inference (e.g., [14], [15], [18]) or formal synthesis problems (e.g., [22]) involving STL.

In this paper, we formulate two problems that enforce a desired STL specification by maximizing 1) the probability of satisfaction and 2) the expected robustness degree. One of the difficulties in solving these problems is the history-dependence of the satisfaction. For instance, if the specifi-

General idea

- States: partition of a Continuous Space
  Unknown stochastic dynamics
- Goal: Maximizing $\Pr[s \vDash \Phi]$ or $E[r(s, \Phi)]$
- Issue: $\Pr[s \vDash \Phi]$ or $E[r(s, \Phi)]$ are not in the standard objective form of *Q*- learning (i.e., the sum of instantaneous rewards)
- Solution: approximation of STL synthesis problems that can be solved via *Q*-learning,

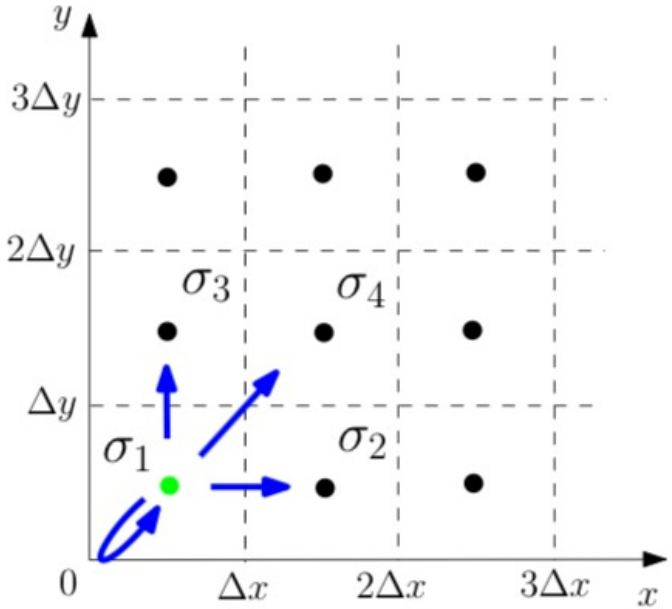


(a)

Fig. 2. (a) Discretized state-space,

System Model

- Set of partitions with centroid $\sigma_1 = \left[\frac{\Delta x}{2}, \frac{\Delta y}{2}\right]$
- Motion primitives $a \in A$, blue arrow
- $s_{t_1:t_2}$: state trajectory of the system within $[t_1, t_2]$
  e.g. $s_0 = \sigma_1$. If the system visits σ_3 and returns to σ_1, its state trajectory can be written as $s_{0:2\Delta t} = \sigma_1\sigma_3\sigma_1$
- probability distribution for s_{t+1} is unknown



(a)

Fig. 2. (a) Discretized state-space,

Problem: history- dependence of the satisfaction

- Let Φ be an STL specification with *hrz*(Φ) = *T* . Given a stochastic model *M* = ⟨Σ,*A*,*P*,*R*⟩ with unknown *P* and an initial partial state trajectory *s*0:τ for some 0 ≤ τ < *T* , find a control policy π such that

$$\pi_1^* = \arg\max_{\pi} Pr^{\pi}[s_{0:T} \models \Phi]$$

- Fragment o $\pi_2^* = \arg\max_{\pi} E^{\pi}[r(s_{0:T}, \Phi)]$ ction is checked with a sufficient nu

Q-learning

Algorithm 1: Q-learning

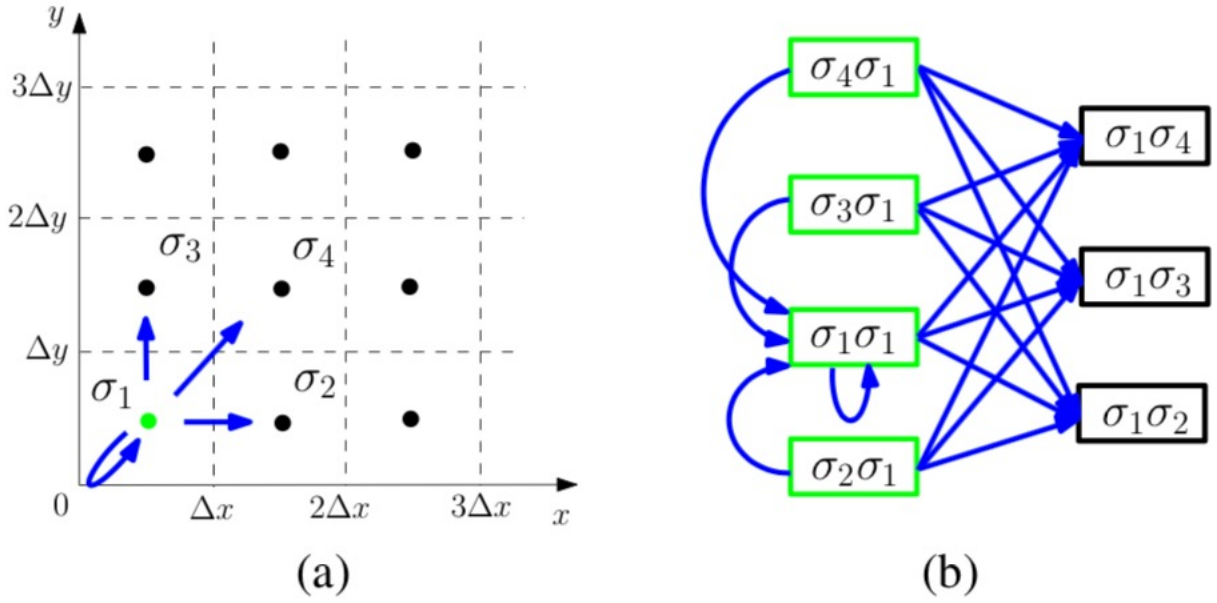
Input: s - current state

Output: π- control policy maximizing the sum of (discounted) rewards

initialization: Arbitrary $Q(s,a)$ and π;
  for $k = 1 : N_{episode}$
    Initialize s
    for $t = 1 : T$
      Select an action a (via ε-greedy or π);
      Take action a, observe r and s';
      $Q(s,a) \leftarrow (1-\alpha_k)Q(s,a) + \alpha_k\left[r + \gamma^t \max_{a'} Q(s',a')\right]$;
      $\pi(s) \leftarrow \arg\max_{a} Q(s,a)$;
      $s \leftarrow s'$;
    end for
  end for

Problem: history- dependence of the satisfaction

- The policies should be defined as $\pi : \Sigma^{\tau} \times N_{\geq 0} \rightarrow A$ where $\Sigma^{\tau} = \Sigma \times \cdots \times \Sigma$ for τ times. Hence, the state-space of the system needs to be redefined as $\Sigma^{\tau} \times N_{\geq 0}$.
- τ −MDP where $\tau = [\frac{hrz(\psi)}{\Delta t}] + 1$ for $F_{[0,T]}\psi, G_{[0,T]}\psi$
- Each state corresponds to a τ-length trajectory



(a) (b)

Fig. 2. (a) Discretized state-space, (b) Representation of σ_1 over $2-MDP$.

Problem: robustness shape

$$\max_{\pi} E^{\pi}[r(s_{0:T}, \Phi)] = \begin{cases} \max_{\pi} E^{\pi}\left[\max_{\tau-1 \le t \le T}\left(r(s_t^{\tau}, \phi)\right]\right., & \text{if } \Phi = F_{[0,T]}\phi \\ \max_{\pi} E^{\pi}\left[\min_{\tau-1 \le t \le T}\left(r(s_t^{\tau}, \phi)\right]\right., & \text{if } \Phi = G_{[0,T]}\phi \end{cases}$$

- log-sum-exp approximation to adapt the Robustness of Q-learning

$$\max(x_1, \ldots, x_n) \sim \frac{1}{\beta} \log \sum_{i=1}^{n} e^{\beta x_i},$$

Finally…

$$\pi^*_{2A} = \begin{cases} \arg\max_{\pi} E^{\pi}\left[\sum_{t=\tau-1}^{T} e^{\beta r(s_t^{\tau},\phi)}\right], & \text{if } \Phi = F_{[0,T]}\phi \\ \arg\max_{\pi} E^{\pi}\left[-\sum_{t=\tau-1}^{T} e^{-\beta r(s_t^{\tau},\phi)}\right], & \text{if } \Phi = G_{[0,T]}\phi \end{cases}$$

The immediate reward is :

$$R = \begin{cases} e^{\beta I(r(s_j^{\tau},\phi))}, & \text{if Problem 1A with } \Phi = F_{[0,T]}\phi \\ -e^{-\beta I(r(s_j^{\tau},\phi))}, & \text{if Problem 1A with } \Phi = G_{[0,T]}\phi \\ e^{\beta r(s_j^{\tau},\phi)}, & \text{if Problem 2A with } \Phi = F_{[0,T]}\phi \\ -e^{-\beta r(s_j^{\tau},\phi)}, & \text{if Problem 2A with } \Phi = G_{[0,T]}\phi \end{cases}$$

Experiments

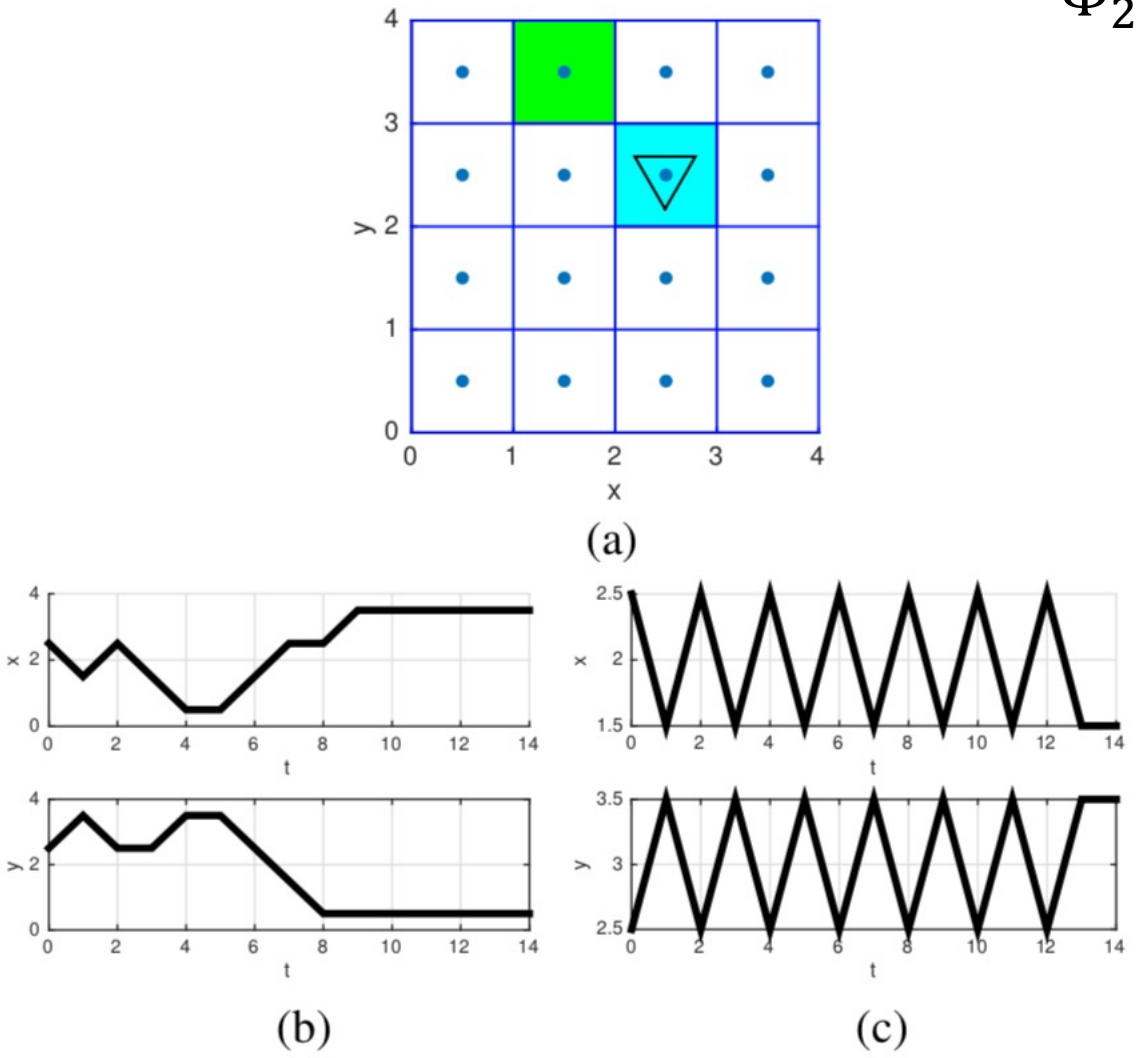


Fig. 5. (a) The initial state and the desired regions in case study 2 for which a sample trajectory by (b) π^*_{1A} and (c) π^*_{2A}.

$$\Phi_2 = G_{[0,12]}(F_{[0,2]}(region\ A) \wedge F_{[0,2]}(region\ B))$$

$|S| = 19$, $|S^\tau| = 676$ and $\tau = 3$

the robustness degree gives "partial credit" for trajectories that are close to satisfaction

For the prop satisfaction, instead, Q-learning algorithm is essentially performing a random search

Reinforcement Learning With Temporal Logic Rewards

Xiao Li, Cristian-Ioan Vasile and Calin Belta.

***Abstract*— *Reinforcement learning* (RL) depends critically on the choice of reward functions used to capture the desired behavior and constraints of a robot. Usually, these are handcrafted by a expert designer and represent heuristics for relatively simple tasks. Real world applications typically involve more complex tasks with rich temporal and logical structure. In this paper we take advantage of the expressive power of *temporal logic (TL)* to specify complex rules the robot should follow, and incorporate domain knowledge into learning. We propose *Truncated Linear Temporal Logic* (TLTL) as a specification language,We propose *Truncated Linear Temporal Logic* (TLTL) as a specification language,that is arguably well suited for the robotics applications, We show in simulated trials that learning is faster and policies obtained using the proposed approach outperform the ones learned using heuristic rewards in terms of the robustness degree, i.e., how well the tasks are satisfied. Furthermore, we demonstrate the proposed RL approach in a toast-placing task learned by a Baxter robot.**

I. Introduction

to Q-learning on τ-MDPs in discrete spaces. Authors of [4] and [5] has also taken advantage of automata-based methods to synthesize control policies that satisfy LTL specifications for MDPs with unknown transition probability.These methods are constrained to discrete state and action spaces, and a somewhat limited set of temporal operators. To the best of our knowledge, this paper is the first to apply TL in reinforcement learning on continuous state and action spaces, and demonstrates its abilities in experimentation.

We compare the convergence properties and the quality of learned policies of RL algorithms using temporal logic (i.e., robustness degree) and heuristic reward functions. In addition, we compare the results of a simple TL algorithm against a more elaborate RL algorithm with heuristic rewards. In both cases better quality policies were learned faster using the proposed approach with TL rewards than with the heuristic reward functions.

Truncated Linear Temporal Logic (TLTL)

- Specifically for robots
- Unbounded
- Atomic propositions
- Evaluated against finite time sequences

$$s_{t:t+k} = s_t\, s_{t+1} \dots s_{t+k}$$

-

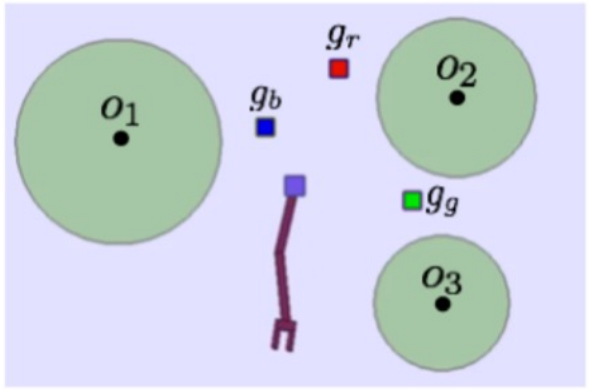
$$
\begin{aligned}
\rho(s_{t:t+k}, \top) &= \rho_{max}, \\
\rho(s_{t:t+k}, f(s_t) < c) &= c - f(s_t), \\
\rho(s_{t:t+k}, \neg\phi) &= -\rho(s_{t:t+k}, \phi), \\
\rho(s_{t:t+k}, \phi \Rightarrow \psi) &= \max(-\rho(s_{t:t+k}, \phi), \rho(s_{t:t+k}, \psi)) \\
\rho(s_{t:t+k}, \phi_1 \wedge \phi_2) &= \min(\rho(s_{t:t+k}, \phi_1), \rho(s_{t:t+k}, \phi_2)), \\
\rho(s_{t:t+k}, \phi_1 \vee \phi_2) &= \max(\rho(s_{t:t+k}, \phi_1), \rho(s_{t:t+k}, \phi_2)), \\
\rho(s_{t:t+k}, \bigcirc\phi) &= \rho(s_{t+1:t+k}, \phi) \ (k > 0), \\
\rho(s_{t:t+k}, \Box\phi) &= \min_{t' \in [t, t+k)} (\rho(s_{t':t+k}, \phi)), \\
\rho(s_{t:t+k}, \Diamond\phi) &= \max_{t' \in [t, t+k)} (\rho(s_{t':t+k}, \phi)), \\
\rho(s_{t:t+k}, \phi\ \mathcal{U}\ \psi) &= \max_{t' \in [t, t+k)} (\min(\rho(s_{t':t+k}, \psi), \\
&\quad \min_{t'' \in [t, t')} \rho(s_{t'':t'}, \phi))), \\
\rho(s_{t:t+k}, \phi\ \mathcal{T}\ \psi) &= \max_{t' \in [t, t+k)} (\min(\rho(s_{t':t+k}, \psi), \\
&\quad \max_{t'' \in [t, t')} \rho(s_{t'':t'}, \phi))),
\end{aligned}
$$

STL and continuous space

- Policy parametrization $\pi(s, a|\theta)$ where θ is the set of model parameters
- $\theta^* = argmax_\theta \, E_{p^{\pi_\theta}(\tau)}[R(\tau)]$,
  where $p^{\pi_\theta}(\tau)$ is trajectory distribution from following policy π
- Relative Entropy Policy Search (REPS) :
  constrained optimization problem that can be solved by Lagrange multipliers method
- Time-varying linear Gaussian policies and weighted maximum-likelihood estimation to update the policy parameters

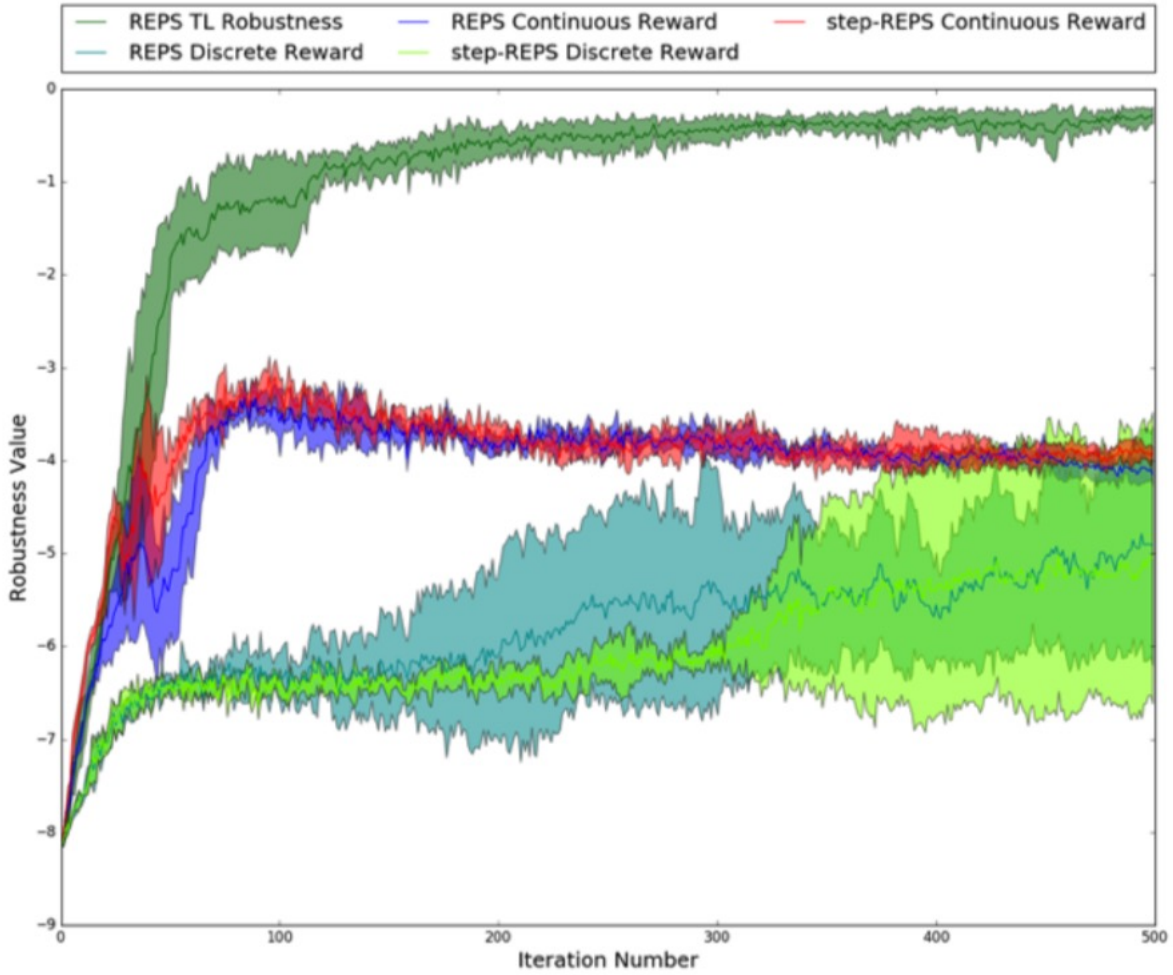
Experiments

$$\phi_2 = (\psi_{g_r} \mathcal{T} \psi_{g_g} \mathcal{T} \psi_{g_b}) \wedge (\neg(\psi_{g_g} \vee \psi_{g_b}) \mathcal{U} \psi_{g_r}) \wedge (\neg(\psi_{g_b}) \mathcal{U} \psi_{g_g}) \wedge \left(\bigwedge_{i=r,g,b} \Box(\psi_{g_i} \Rightarrow \bigcirc\Box\neg\psi_{g_i}) \right) \wedge \Box\psi_o$$



$$r_2^{discrete} = \begin{cases} 5 & \text{goals visited in the right order} \\ -5 & \text{goals visited in the wrong} \\ -2 & d_{o_{1,2,3}} \leq r_{o_{1,2,3}} \\ 0 & \text{everywhere else} \end{cases}$$

$$r_2^{continuous} = -c_1 d_{g_i} + c_2(d_{g_j} + d_{g_k}) + c_3 \sum_{i=1}^{3} d_{o_i}.$$



LTL constrained to discrete state and action

Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints

Jie Fu and Ufuk Topcu
Department of Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, Pennsylvania 19104
Email: jief, utopcu@seas.upenn.edu

***Abstract*—We consider synthesis of controllers that maximize the probability of satisfying given temporal logic specifications in unknown, stochastic environments. We model the interaction between the system and its environment as a Markov decision process (MDP) with initially unknown transition probabilities. The solution we develop builds on the so-called model-based probably approximately correct Markov decision process (PAC-MDP) method. The algorithm attains an ε-approximately optimal policy with probability $1-\delta$ using samples (i.e. observations), time and space that grow polynomially with the size of the MDP, the size of the automaton expressing the temporal logic specification, $\frac{1}{\varepsilon}$, $\frac{1}{\delta}$ and a finite time horizon. In this approach, the system maintains a model of the initially unknown MDP, and constructs a product MDP based on its *learned model* and the specification automaton that expresses the temporal logic constraints. During execution, the policy is iteratively updated using observation of the transitions taken by the system. The iteration terminates in finitely many execution steps. With high probability, the resulting policy is such that, for any state, the difference between the probability of satisfying the specification under this policy and the optimal one is within a predefined bound.**

arrived positions differ
of different grounds.
terrain can be modele
probabilities are unkno
observations of robot'
number of samples.
may not be affordable
amount of samples, we
MDP and reason abou
respect to the underl
policies synthesized us

We develop an alg
updates the controller
for an unknown MDP.
method [4, 5] to maxin
temporal logic specific
tion probabilities. In th
a model of the MDP

A Learning Based Approach to Control Synthesis of Markov Decision Processes for Linear Temporal Logic Specifications

Dorsa Sadigh, Eric S. Kim, Samuel Coogan, S. Shankar Sastry, Sanjit A. Seshia

***Abstract*— We propose to synthesize a control policy for a Markov decision process (MDP) such that the resulting traces of the MDP satisfy a linear temporal logic (LTL) property. We construct a product MDP that incorporates a deterministic Rabin automaton generated from the desired LTL property. The reward function of the product MDP is defined from the acceptance condition of the Rabin automaton. This construction allows us to apply techniques from learning theory to the problem of synthesis for LTL specifications even when the transition probabilities are not known *a priori*. We prove that our method is guaranteed to find a controller that satisfies the LTL property with probability one if such a policy exists, and we suggest empirically that our method produces reasonable control strategies even when the LTL property cannot be satisfied with probability one.**

practical contexts where we start from a partial model with unspecified probabilities.

Our approach is based on finding a policy that maximizes the expected utility of an auxiliary MDP constructed from the original MDP and a desired LTL specification. As in the above mentioned existing work, we convert the LTL specification to a *deterministic Rabin automaton* (DRA) [11], [12], and construct a product MDP such that the states of the product MDP are pairs representing states of the original MDP in addition to states of the DRA that encodes the desired LTL specification. The novelty of our approach is that we then define a state based reward function on this product MDP based on the *Rabin* acceptance condition of

LTL constrained to discrete state and action

- select the reward function on the product MDP so it corresponds to the *Rabin* acceptance condition of the LTL specification.

$$s_{\mathcal{P}} = (s, q) \in S_{\mathcal{P}}$$

$$W^i_{\mathcal{P}}(s_{\mathcal{P}}) = \begin{cases} w_G & \text{if } s_{\mathcal{P}} \in \mathcal{G}_i \\ w_B & \text{if } s_{\mathcal{P}} \in \mathcal{B}_i \\ 0 & \text{if } s_{\mathcal{P}} \in S \backslash (\mathcal{G}_i \cup \mathcal{B}_i) \end{cases} \tag{4}$$

where $w_G > 0$ *is a positive reward,* $w_B < 0$ *is a negative reward.*

- Prove convergence if policy exist s.t. it satisfies property with probability 1

- 1) Learn the transition probabilities and
  2) Optimize the expected utility.
  E.g. with a modified active temporal difference learning algorithm

Several Works with different motivations

- Reward shaping using probability of average robusntess satisfaction
- **LTL constrained, Reward function remained the same**
- multi-task-RL

The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Safe Reinforcement Learning via Shielding

Mohammed Alshiekh,[1] Roderick Bloem,[2] Rüdiger Ehlers,[3]
Bettina Könighofer,[2] Scott Niekum,[1] Ufuk Topcu [1]
[1]University of Texas at Austin, 210 East 24th Street, Austin, Texas 78712, USA
[2]Graz University of Technology, Rechbauerstraße 12, 8010 Graz, Austria
[3]University of Bremen and DFKI GmbH, Bibliothekstraße 1, 28359 Bremen, Deutschland
{malshiekh, sniekum, utopcu}@utexas.edu, {roderick.bloem, bettina.koenighofer}@iaik.tugraz.at, rehlers@uni-bremen.de

Abstract

Reinforcement learning algorithms discover policies that maximize reward, but do not necessarily guarantee safety during learning or execution phases. We introduce a new approach to learn optimal policies while enforcing properties expressed in temporal logic. To this end, given the temporal logic specification that is to be obeyed by the learning system, we propose to synthesize a reactive system called a *shield*. The shield monitors the actions from the learner and corrects them only if the chosen action causes a violation of the specification. We discuss which requirements a shield must meet to preserve the convergence guarantees of the learner. Finally, we demonstrate the versatility of our approach on several challenging reinforcement learning scenarios.

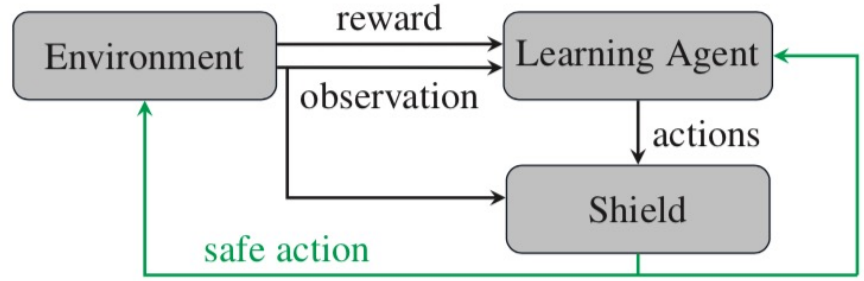


Figure 1: Shielded reinforcement learning

its operation whenever absolutely needed in order to ensure safety?"

In this paper, we introduce *shielded learning*, a framework that allows applying machine learning to control sys-

Safe RL via Shield

How can we let a learning agent do whatever it is doing, and also monitor and interfere with its operation whenever absolutely needed in order to ensure safety?

- The shield is computed upfront from the safety part of the given system specification and an abstraction of the agent's environment dynamics
- ***Minimum interference:*** monitors the actions selected by the learning agent and corrects them if and only if the chosen action is unsafe.
- Boundary helps to separate the concerns, e.g., safety and correctness on one side and convergence and optimality on the other
- Compatible with mechanisms such as function approximation, employed by learning algorithms in order to improve their scalability

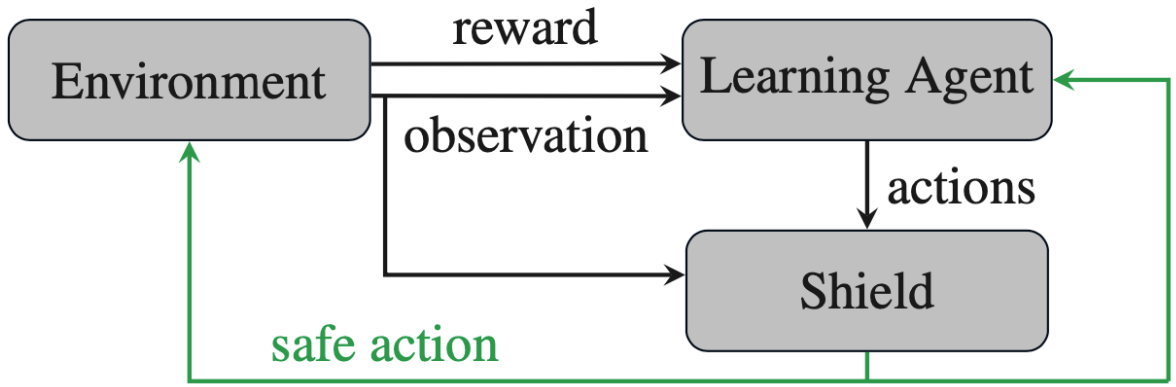


Figure 1: Shielded reinforcement learning

Safe RL via Shield

- Safety fragment of LTL
  (something bad should never happen, e.g. no safety G(r → Fg), every request is eventually granted)
- A faithful, yet precise enough, abstraction of the physical environment is required
- Independent of the state space components of the system to be controlled
- The shield is the product between specification automaton and the MDP abstraction

Grid world Example

With tabular Q-learning with an ε-greedy explorer

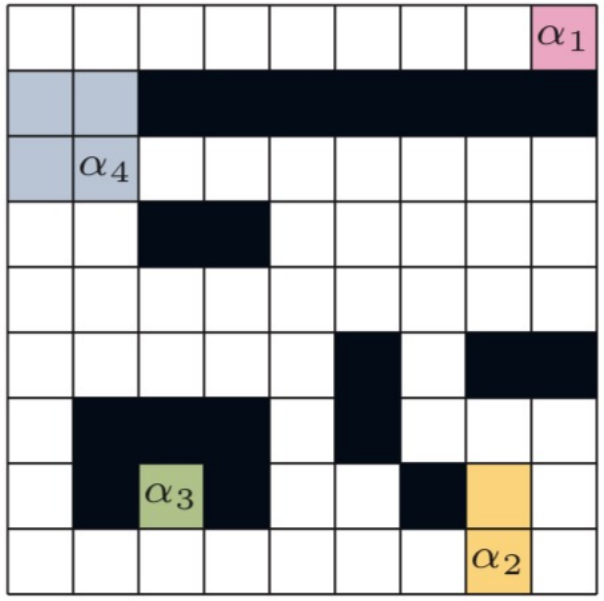


Figure 6: 9x9 grid world.

ϕ_s: the robot must not crash into walls or the moving opponent agent.

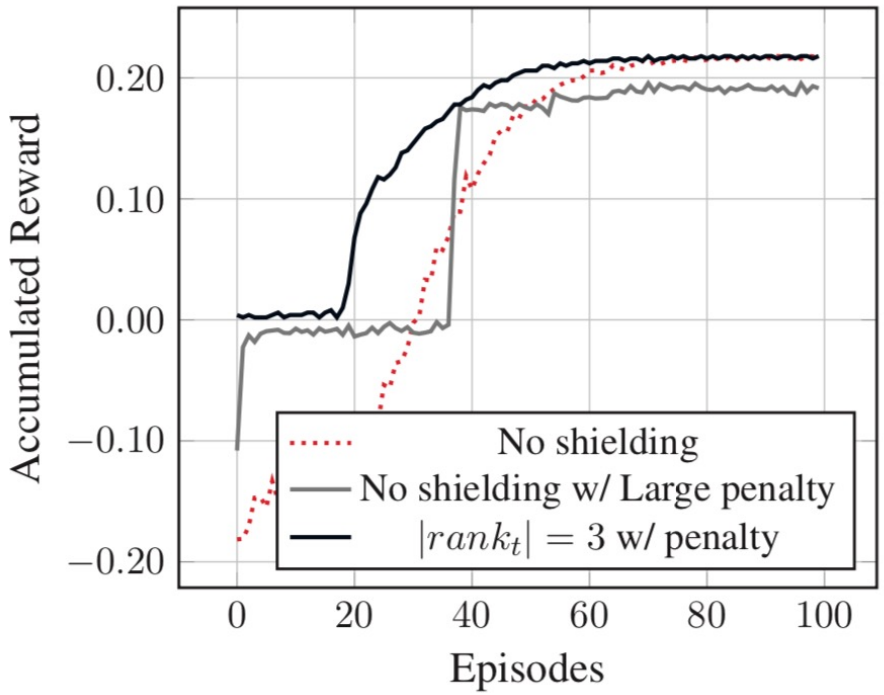


Figure 7: The accumulated reward per episode for the 9x9 grid world example.

The PacMan Example

Approximate Deep Q-learning agent

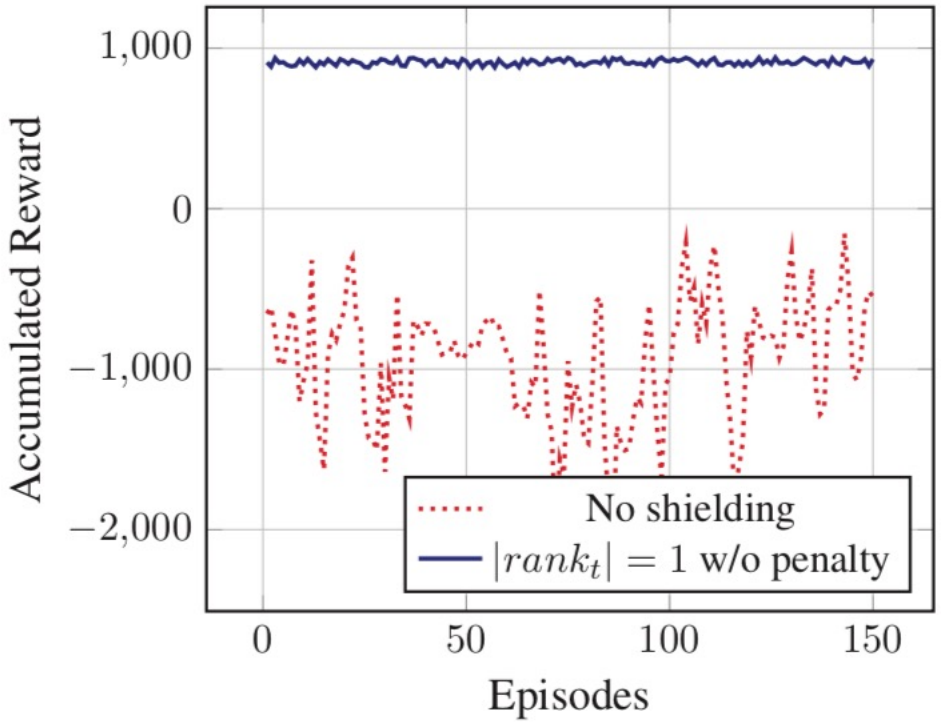


Figure 13: The accumulated reward per episode for the pac-man example.

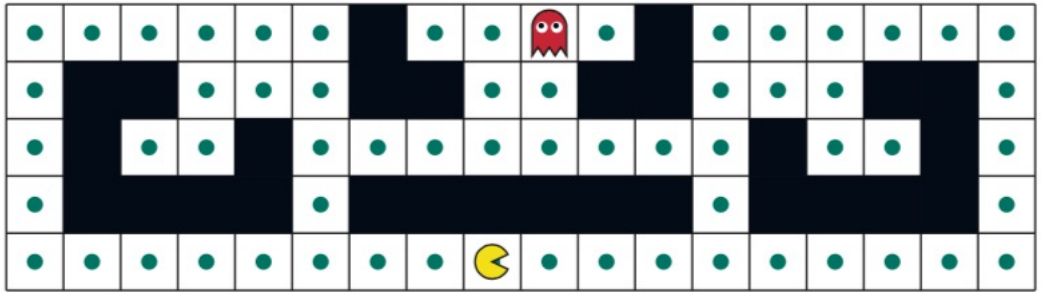


Figure 12: The 5x18 grid world of the pacman example.

The safety specification in this example is to avoid crashing into a wall.

Several Works with different motivations

- LTL constrained, Reward function remained the same
- Reward shaping using probability of average robusntess satisfaction
- **Multi-task-RL**

Multi-task-RL

Teaching Multiple Tasks to an RL Agent using LTL

Rodrigo Toro Icarte
University of Toronto
Department of Computer Science & Vector Institute
rntoro@cs.toronto.edu

Toryn Q. Klassen
University of Toronto
Department of Computer Science
toryn@cs.toronto.edu

Richard Valenzano
Element AI
rick.valenzano@elementai.com

Sheila A. McIlraith
University of Toronto
Department of Computer Science
sheila@cs.toronto.edu

ABSTRACT

This paper examines the problem of how to teach multiple tasks to a *Reinforcement Learning (RL)* agent. To this end, we use Linear Temporal Logic (LTL) as a language for specifying multiple tasks in a manner that supports the composition of learned skills. We also propose a novel algorithm that exploits LTL progression and off-policy RL to speed up learning without compromising convergence guarantees, and show that our method outperforms the state-of-the-art approach on randomly generated Minecraft-like grids.

Linear Temporal Logic (LTL) and then defining reward functions that provide positive reward for their successful completion. LTL is a propositional, modal temporal logic first developed for the verification of reactive systems [35]. It augments propositional logic with modalities such as ◇ (*eventually*), □ (*always*), and U (*until*) in support of expressing statements such as *"Always if clothes are on the floor, put them in the hamper"* or *"Eventually make dinner."* Such statements can be combined via logical connectives and nesting of modal operators to provide task specifications. The syntax is natural and compelling and, as a formal language, it has a well-defined

Decompose tasks into subtasks with LTL progression

LTL progression

Given an LTL formula φ and state s, we can *progress* φ using s:

- $\text{prog}(s, p) = \text{true}$ if $p \in L(s)$, where $p \in \mathcal{P}$
- $\text{prog}(s, p) = \text{false}$ if $p \notin L(s)$, where $p \in \mathcal{P}$
- $\text{prog}(s, \neg\varphi) = \neg\, \text{prog}(s, \varphi)$
- $\text{prog}(s, \varphi_1 \wedge \varphi_2) = \text{prog}(s, \varphi_1) \wedge \text{prog}(s, \varphi_2)$
- $\text{prog}(s, \bigcirc\varphi) = \varphi$
- $\text{prog}(s, \Diamond\varphi) = \text{prog}(s, \varphi) \vee \Diamond\varphi$
- $\text{prog}(s, \varphi_1 \,\mathsf{U}\, \varphi_2) = \text{prog}(s, \varphi_2) \vee (\text{prog}(s, \varphi_1) \wedge \varphi_1 \,\mathsf{U}\, \varphi_2)$

Task with finite-episode -> restriction to co-safe properties