Structured Reward Shaping using Signal Temporal Logic specifications

Anand Balakrishnan, Jyotirmoy V. Deshmukh

Abstract—**Deep reinforcement learning has become a popular technique to train autonomous agents to learn control policies that enable them to accomplish complex tasks in uncertain environments. A key component of an RL algorithm is the definition of a reward function that maps each state and an action that can be taken in that state to some real-valued reward. Typically, reward functions informally capture an implicit (albeit vague) specification on the desired behavior of the agent. In this paper, we propose the use of the logical formalism of *Signal Temporal Logic* (STL) as a formal specification for the desired behaviors of the agent. Furthermore, we propose algorithms to locally shape rewards in each state with the goal of satisfying the high-level STL specification. We demonstrate our technique on two case studies, a cart-pole balancing problem with a discrete action space, and controlling the actuation of a simulated quadrotor for point-to-point movement.**

The proposed framework is agnostic to any specific RL algorithm, as locally shaped rewards can be easily used in concert with any deep RL algorithm.

I. Introduction

Reinforcement learning (RL) combined with deep learning has been incredibly successful in solving highly complex problems in domains with well-defined reward functions, like maximizing Atari games' high scores [1] and complex cyber-physical problems such as learning gait in simulated bi-pedal robots [2]. To a large extent, this success can be attributed to the ability of deep neural networks to approximate highly non-linear functions that take raw data, like pixel data in [1] and proprioceptive sensor data in [2], as input and output the expected total reward from performing a given action at a specific state.

An important problem to address when designing and training reinforcement learning agents is the design of *reward functions* [3]. Reward functions are a means to incorporate knowledge of the goal and the environment model in the training of an RL agent, using hand-crafted and fine-tuned functions of the current state of the system. Thus, poorly designed reward functions can lead to the reinforcement learning algorithm learning a policy that doesn't exactly accomplish the task that the agent was supposed to learn to complete, as shown in [4]. Moreover, in safety critical systems, the agent can learn a policy that performs unsafe or unrealistic actions, even though it maximizes the expected total reward [4].

The problem of an RL agent learning to maximize the total reward by exploiting the reward function, and thus performing unwanted or unsafe behavior is called *reward hacking* [5] and the study of minimizing reward hacking by designing better reward functions is called *reward shaping* [6].

Meanwhile, research on safety and verification of cyber-physical systems (CPS) has extensively used logical formalisms based on Temporal Logics to define safety specifications. In particular, Signal Temporal Logic (STL) has seen considerable use to define temporal properties of signals in various cyber-physical system applications [7]–[9]. Moreover, there has been work to furnish STL with quantitative semantics, which allow us to quantify how robustly a signal satisfies a given property. The robustness of a signal with respect to an STL formula can be viewed as the distance of the signal to the set of signals satisfying the given formula [10], [11].

Seminal work in [12] explores the idea of using the robust satisfaction semantics of STL to define reward functions for a reinforcement learning procedure. Similar ideas were extended by to a related logic for RL-based control design for Markov Decision Processes (MDP) in [13]. In this paper, we identify certain shortcomings of the previous approaches; in particular, we observe that using reward functions based on traditional definitions of robustness are *global*, i.e. a positive (resp. negative) robustness value translates into a positive (resp. negative) reward that influences all states encountered during a learning episode equally. To address this issue, we adapt the quantitative semantics of STL to be defined over *partial signal traces*. A partial signal trace is a bounded-length segment of the state trajectory of the system being controlled by the RL agent, allowing us to define *locally shaped reward functions* over this partial signal trace. The learning step in the RL agent is then performed in a delayed manner by observing the temporal behavior (over a bounded window) and evaluating the behavior vis-à-vis the given STL formula.

The ultimate objective of this work is to provide a framework for a flexible structured reward function formulation. In this paper, we formulate structured and locally shaped rewards in an expressive manner using STL formulas. We show how locally shaped rewards can be used by any deep RL architecture, and demonstrate the efficacy of our approach through two case studies.

II. Related Work

Reward shaping has been addressed in previous work primarily using ideas like inverse reinforcement learning [14], potential-based reward shaping [15], or combinations of the above [16]. Inverse RL, like in [14] involves learning reward functions that best describe a system, typically from

* Cyber-Physical Systems: Verification, Intelligence, Design and Analysis (CPS-VIDA) Group, University of Southern California, Los Angeles, California. Email: {anandbal, jdeshmuk}@usc.edu

several ranked demonstrations, addressing the problem of reward shaping rather than reward maximization. Such an approach typically requires processing several demonstrations and can also result in uninterpretable reward functions that are complex non-linear functions. Similarly, using potential-based reward functions allows us to scale rewards by some *potential function* which can either be hand-crafted or learned using inverse RL, like in [16]. These methods come with a guarantee of invariance from the original reward function while enabling faster convergence to an optimal policy, but come with the same problems as traditional reward functions, that is, the lack of expressiveness and ability to capture complex behavior in a well-structured manner. In contrast to potential-based reward functions, where a potential function is used to alter rewards provided by the environment, the method presented in this paper tackles the issue of *design* of the reward function in a well-structured manner.

To address some of the shortcomings of classical RL, especially regarding the learning of safe policies (or controllers), lot of work has been done in using control theoretic techniques in training and verifying reinforcement learning models.

Temporal logics have been used extensively in the context of cyber-physical systems to encode temporal dependencies in the state of a system in an expressive manner [7]. Thus, high-level temporal logics like LTL have been used to design highly scalable planning algorithms like in [17], where high-level LTL specifications have been used to synthesize decentralized controllers for swarms. In addition to having boolean satisfaction semantics, temporal logics like Signal Temporal Logic (STL) [11] have various proposed quantitative semantics [18]–[23], that establish a robust satisfaction value (or robustness) to quantify how well a trace satisfies a formula. The work in [12] proposes a method to interpret the environment MDP as a temporal abstraction, and thus use the robustness of a trajectory with respect to an STL formula as the reward function for Q-learning. This was further modified in [13] to perform reinforcement learning over Markov Decision Processes for a closely related temporal logic.

In many safety-critical systems, we are interested in acting upon or optimizing behavior in low-level components in a system, thus making STL a better candidate than LTL to specify requirements in, as demonstrated in [7]. The altered Q-learning formulation presented in [12] uses STL robustness of a temporal-MDP transition to compute the reward. But generating this temporal-MDP requires enumeration of possible future states, thus making the size of the state space exponentially proportional to the horizon of the STL specification. In the work presented in [13], the robustness of a trajectory generated by an RL agent is computed after the episode completed, similar to Monte Carlo methods in RL, thus has similar disadvantages to Monte Carlo methods [3].

In this paper, we introduce a method to define well-structured reward functions using STL properties. The presented technique extends previous work on reinforcement learning with temporal logic rewards in two significant directions: (1) we allow STL formulas with (potentially) unbounded time horizons, and (2) we can dynamically alter rewards during a training episode, which gives us significantly better performance and convergence rates in the training procedure. While we elaborate on the key steps of our technique in the subsequent sections, we now introduce the key concepts required for these extensions.

III. Preliminaries

A. Reinforcement Learning (RL)

Definition 1 (Markov Decision Process (MDP))**.** *It is a tuple $M = (S, A, T, R, \gamma)$ where*

- *S is the state space of the system;*
- *A is the set of actions that can be performed on the system;*
- *T is the transition function, where $T(s,a,s') = P(s' \mid s,a)$;*
- *R is a reward function that typically maps either some $s \in S$ or some transition $\delta \in S \times A \times S$ to $\mathbb{R}$;*
- *γ is the discount factor for the MDP.*

In RL, the goal of the learning algorithm is to converge on a policy $\pi : S \to A$ that maximizes the total (discounted) reward from performing actions on a MDP, i.e., the objective is to maximize $\max \sum_{t=0}^{\infty} \gamma^t r_t$, where r_t is the output of the reward function R for the sample at t.

Q-learning [24] is a classic RL algorithm that uses value iteration to train an optimal policy. In Q-learning, we define a function $Q : S \times A \to \mathbb{R}$ as

$$Q(s_t, a_t) = \mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^{\tau} r_{t+\tau} \middle| s_t, a_t \right]$$

and pick action a from a policy $\pi(s) = \max_a Q(s_t, a_t)$. Thus the optimal policy $\pi*$ is the policy that uses the optimal Q-function, Q^* that correctly estimates the expected total reward from an MDP state, using value iteration.

B. Signal Temporal Logic

Definition 2 (Discrete-Time Signals)**.** *Let $\mathbb{T} = \{t_0, t_1, \ldots\}$ be a finite or infinite set of time-points, where $\forall i, t_i \in \mathbb{R}^{\geq 0}$. For a compact set $\mathscr{D}$, a* discrete-time signal $\mathbf{x}$ *is a function from $\mathbb{T}$ to $\mathscr{D}$. In this paper, we restrict our attention to sets $\mathscr{D}$ that are compact subsets of $\mathbb{R}^m$ for some positive integer m.*

Signal Temporal Logic (STL) is a real-time logic, typically interpreted over a dense-time domain for signals that take values in a continuous metric space (such as $\mathbb{R}^m$). The basic primitive in STL is a *signal predicate* μ that is a formula of the form $f(\mathbf{x}(t)) > 0$, where $\mathbf{x}(t)$ is the value of the signal $\mathbf{x}$ at time t, and f is a function from the signal domain $\mathscr{D}$ to $\mathbb{R}$. STL formulas are then defined recursively using Boolean combinations of sub-formulas, or by applying an interval-restricted temporal operator to a sub-formula. The syntax of STL is formally defined as follows:

$$\varphi ::= \mu \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbf{G}_I \varphi \mid \mathbf{F}_I \varphi \mid \varphi \mathbf{U}_I \varphi \mid \qquad (1)$$

Here, $I = [a,b]$ denotes an arbitrary time-interval, where $a, b \in \mathbb{R}^{\geq 0}$. The semantics of STL are defined over a discrete-time signal $\mathbf{x}$ defined over some time-domain $\mathbb{T}$. The Boolean satisfaction of the primitive is simply $\top$ if the predicate is satisfied and $\bot$ if it is not, and the semantics for the propositional logic operators $\neg, \wedge$ (and thus $\vee, \to$) follow

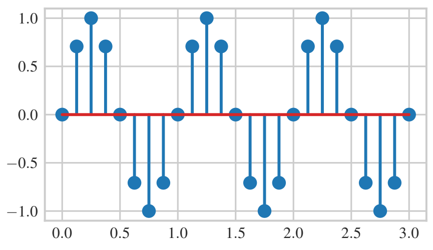

Fig. 1. Properties on a sin wave.

the obvious logical semantics. The temporal operators, thus, model the following behavior:

- $\mathbf{G}_I(\varphi)$ says that φ must hold for all samples in I.
- $\mathbf{F}_I(\varphi)$ says that φ must hold *at least once* for samples in I.
- $\varphi \mathbf{U}_I \psi$ says that φ must hold in I until ψ holds.

Example 1. *Consider the signal $x(t)$ obtained by sampling the function $\sin(2\pi t)$ at times $t_0, t_1, \ldots$, where $t_j = j \times 0.125$ (shown in Fig. 1). Consider the formula $\mathbf{G}(x(t) \geq -1)$, which requires that starting at time 0, $x(t)$ is always greater than -1 (at each sample point). Consider the formula $\mathbf{F}(\mathbf{G}_{[0,1]}(x(t) \geq 0))$. This formula requires that there is some time (say τ) such that between times $[\tau, \tau+1]$, $x(\tau)$ is always greater than 0. Considering that $x(t)$ is a sampling of a sinusoid with period 1, this formula is also satisfied by $x(t)$.*

In addition to the Boolean satisfaction semantics for STL, various researchers have proposed quantitative semantics for STL, [11], [19]–[23] that compute the degree of satisfaction (or *robust satisfaction values*) of STL properties by traces generated by a system. These semantics can be presented in the following general form:

Definition 3 (Quantitative Semantics for Signal Temporal Logic)**.** *Given an algebraic structure $(\oplus, \otimes, \top, \bot)$, we define the quantitative semantics for an arbitrary signal $\mathbf{x}$ against an STL formula φ at time t as follows:*

φ	$\rho(\varphi, \mathbf{x}, t)$
true/false	$\top/\bot$
μ	$f(\mathbf{x}(t))$
$\neg\varphi$	$-\rho(\varphi, \mathbf{x}, t)$
$\varphi_1 \wedge \varphi_2$	$\otimes(\rho(\varphi_1, \mathbf{x}, t), \rho(\varphi_2, \mathbf{x}, t))$
$\varphi_1 \vee \varphi_2$	$\oplus(\rho(\varphi_1, \mathbf{x}, t), \rho(\varphi_2, \mathbf{x}, t))$
$\mathbf{G}_I(\varphi)$	$\otimes_{\tau \in t+I}(\rho(\varphi, \mathbf{x}, \tau))$
$\mathbf{F}_I(\varphi)$	$\oplus_{\tau \in t+I}(\rho(\varphi, \mathbf{x}, \tau))$
$\varphi \mathbf{U}_I \psi$	$\oplus_{\tau_1 \in t+I}(\otimes(\rho(\psi, \mathbf{x}, \tau_1), \otimes_{\tau_2 \in [t, \tau_1)}(\rho(\varphi, \mathbf{x}, \tau_2)))$

The above definition is quite abstract: it does not give specific interpretations to the elements $\top, \bot$ or the operators $\otimes$ and $\oplus$. In the original definitions of robust satisfaction proposed in [11], [18], the interpretation was to set $\top = +\infty$, $\bot = -\infty$, and $\oplus = \max$, and $\otimes = \min$. This algebraic characterization of quantitative robustness, extended from the work proposed in [21] is actually quite general, and allows us to explore different interpretations for the $\bot, \top$ elements and the $\oplus$ and $\otimes$ operators.

IV. Partial Signal Rewarding

Definition 4 (Partial Signal)**.** *Given a trajectory of a system $\mathbf{x}$, a partial signal, $\mathbf{x}[i:j]$, is defined by the slice of the trajectory from the i^{th} sample to the j^{th} sample. That is,*

$$\mathbf{x}[i:j] = (s_i, \ldots, s_j)$$

We define the robust satisfaction values computed for each sample in the partial signal as the *bounded horizon nominal robustness (BHNR)* (denoted $\tilde{\rho}$), so as to not confuse it with the standard definition of robustness. The BNHR values differ from the actual robustness value as the true robust satisfaction value is computed over the entire signal, whereas the BHNR is computed over a partial signal trace, thereby not taking into account samples present outside the slice of the trajectory of the system.

Definition 5 (Bounded Horizon Nominal Robustness (BHNR) values)**.** *For formulas of the type $\psi = \mathbf{G}\varphi$ or $\psi = \mathbf{F}\varphi$ (i.e. with an unbounded outer temporal operator), we define the bounded horizon nominal robustness to be an array of* localized *robust satisfaction values for all sample time steps $t \in i \ldots j$ of a partial signal $\mathbf{x}[i:j]$.*

$$\tilde{\rho}(\psi, \mathbf{x}[i:j], t)[t] = \rho(\varphi, \mathbf{x}[t:j], t), \forall t \in i \ldots j \tag{2}$$

Remark. *The above definition makes sense only when used in the context of signals of the type $\psi = \mathbf{F}\varphi$ or $\psi = \mathbf{G}\varphi$ as the reward function must hold at every time step in the training process and not within a finite time slice.*

By computing the BHNR values of a partial signal, we can compute a reward for each state in the partial signal trace, where the reward encodes information about decisions taken in the future states in an expressive manner. Thus, the *reward vector* can be computed by using the following formula:

$$r_t = \tilde{\rho}(\varphi, \mathbf{x}[i:i+\tau_p], t), \quad \forall t \in [i, i+\tau_p] \tag{3}$$

where, i is the first sample time step in the partial signal, τ_p is the length of the partial signal and φ is the STL specification defined on the system.

We can see that the reward function defined above is *locally shaped*: within a slice of the trajectory of the system, we can define how robust the actions taken by the RL agent are. This is very similar to how N-step learning methods approximate the value function of the current state by performing an N-step simulation, as opposed to computing the true value function using Monte Carlo method [3]: the benefits that BHNR robustness provides to shaping rewards provides a local approximation of the robustness and helps in approximating the "gradient" of the robustness for future actions, thus enabling faster learning. It should be noted that the robustness signal of the true trajectory may not be continuous, and hence a gradient may not exist, but it is more likely to exist in a localized slice of the signal.

V. Experimental Results

To evaluate the new framework, we run experiments that compare the performance between a RL agent that is rewarded using a traditionally defined reward function, and another RL agent rewarded with a the bounded horizon nominal robustness $\tilde{\rho}$ defined in (3). These RL agents are trained with the same learning algorithm and hyper parameters, and are hence identical. The experiments were chosen to demonstrate the generality with which we can apply this rewarding framework to different RL algorithms, especially in the context of deep RL. Specifically, we choose the following two tasks:

1) *Cart-Pole problem* [25]: In this classic control environment, we train two double deep Q-network (Double DQN) agents to learn to maximize the amount of time a pole is balanced on a cart. This experiment is simulated in the OpenAI Gym reinforcement learning environment [26].
2) *Quadrotor Position Control* [27]: Here, we simulate a quadrotor and train a deep RL agent to perform attitude control for the quadrotor and reach a goal position. In our experiments, we use the PPO algorithm to train the agent.

We also perform each experiment multiple times with different random seeds, and average the training data across the multiple trials with error computed using a 95% confidence interval.

A. Case 1: Cart-Pole Problem

This environment is the classical cart-pole balancing problem described in [25], where a cart is controlled with a simple bang-bang controller that pushes it left or right, and the goal is to balance a pole on the cart for as long as possible. This environment is very useful to demonstrate the effectiveness of various algorithms due to is simplicity, as a good controller can be learned from using algorithms like tableau Q-learning and traditional actor-critic methods.

In this problem, the state space of the environment is $\langle x, \dot{x}, \theta, \dot{\theta} \rangle$, where x is the displacement of the cart from the origin and θ is the angle by which the pole is displaced from the upright position. Moreover, the reward function is traditionally defined as follows:

$$r_t = \begin{cases} 1 & \text{if } \theta \in \pm 12^\circ \wedge x \in \pm 2 \text{ units} \\ -10 & \text{otherwise} \end{cases} \tag{4}$$

This essentially rewards the controller for every time step that the cart is balanced and punished it if the pole falls of the cart or the cart is "driven off the table". The -10 in the other case can also be replaced by either 0 or some large negative number, to act as negative reinforcement for the RL agent.

Using this reward function, several algorithms can be used too train a policy that maximizes the total reward. But, the controller *game* the reward function by having a $|\dot{x}| <= \delta$ until the end of the episode, where δ is some small positive value. This essentially introduces a drift in the control system that the learned policy cannot learn to compensate for using the current reward function. While an argument can be made that the reward function can have stricter constraints, doing so causes the agent to have too few positively rewarded states, causing an increase in the duration that the agent has to be trained.

In our reformulated reward function, we use the following STL formula to monitor the BNHR of the partial signal traces generated by the environment:

$$\varphi = \mathbf{G}\Big(\mathbf{F}\left(|\dot{x}| < 0.01\right) \wedge \left(|\theta| < 2^\circ\right) \wedge \left(|x| < 0.5\right)\Big) \tag{5}$$

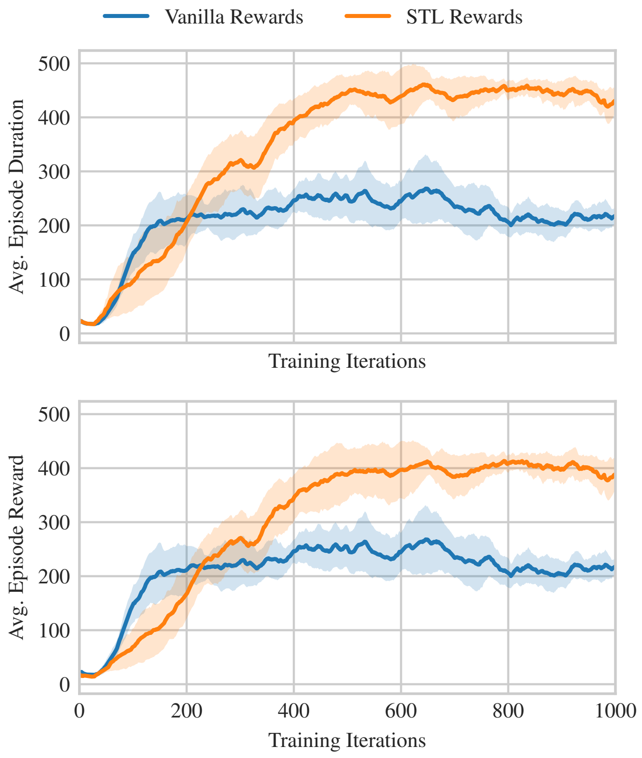

Fig. 2. Double DQN training data for cart-pole problem

The above equation is essentially incorporates a stricter version of the reward function defined in (4), but also incorporates the $\mathbf{G}$ and $\mathbf{F}$ temporal operators, thus describing the following behavior:

> For the entirety of the duration of operation, the pole must be constrained to a $\pm 2^\circ$ angle and the cart must not be moved more than 0.5 units. The cart must also eventually have a velocity with magnitude less than 0.01.

The use of the temporal operators allows us to define a behavior that we want our policy to *converge upon* as opposed to follow it instantaneously. The reason the temporal constraint is added to the velocity of the cart is because, typically, when training an RL agent to "solve" the cart-pole environment, the final policy results in a drifting behaviour. This is because the agent essentially has to learn to keep that cart within bounds for a sufficient amount of time, as the episode is set to terminate after a fixed number of steps. This is an example of "reward hacking", as gradual drift can be considered unsafe in the context of cyber-physical systems.

Moreover, we use the the quantitative semantics defined in [21] where the quantitative semantics are defined over the algebraic structure $(+, \times, 1, 0)$, effectively bounding the robustness values in $[0, 1]$, i.e. providing a normalized robustness value. For further information on how the filter semantics of STL works, refer to [21].

In our experiments, we used a Double DQN learning algorithm [28], where the neural network is defined to be a fully-connected, multi-layer perceptron with the *ReLU* activation function. The hyper-parameters for the training are

TABLE I
DOUBLE DQN HYPER-PARAMETERS

Discount factor	0.95
Learning Rate	0.001
Uniform Sample Memory Size	2000
Target Update	15
Partial Signal Length	15

TABLE II
PPO HYPER-PARAMETERS

Discount factor	0.95
Learning Rate	0.001
Value Loss, Entropy Coefficients	0.5, 0.01
Max. gradient norm	0.5
Number of workers	4
Rollout length	100
Mini-batch Size	32
Clipping parameter	± 0.2
BHNR Partial Signal Length	100

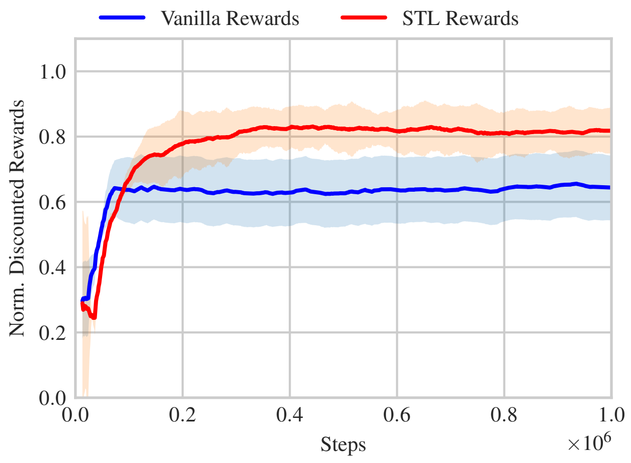

Fig. 3. PPO training data for quadrotor position control

described in the TABLE I, including the length of the partial signal buffer. The training results can be seen in Fig. 2

In Fig. 2, we can see that the BHNR rewarded agent learns a policy that outperforms the vanilla agent very early on in the training by increasing the duration for which the pole is balanced (max of 500 time-steps), while continuing to maximize the total expected reward.

B. Case 2: Quadrotor Position Control

For this environment, simulate a quadrotor environment in which the goal is to control the drone by directly sending commands to the actuators (propellers) on the drone and get it from its position to a randomly generated goal point. The controller and the simulation operate at a frequency of 200 Hz, thus each step in the simulation is 5 milliseconds long. This is a combination of the environment described in [27] and the OpenAI Roboschool environment, *RoboschoolHumanoidFlagrun*.[1]

The state space in this problem is the vector $(x, \theta, \dot{x}, \dot{\theta}, g, collision)$, where x is the position of the quadrotor in Cartesian space, θ is the Euler angle orientation of the quadrotor, g is the position of the goal point, and *collision* is a Boolean flag associated with the collision state of the drone, thus $|S| = 16$. As mentioned earlier, the action space of the environment is the direct actuation of the motors of the quadrotor, thus $|A| = 4$.

We train two RL agents using PPO [29], and similar to Sec.V-A, we reward one with a vanilla reward function and the other with the BHNR-based reward function. The list of hyper-parameters for the training can be seen in TABLE II.

For the vanilla rewards, we use the reward function defined in [27], that is:

$$r_t = -\left(4\mathrm{e}{-3}\,\|x_t - g_t\| + 2\mathrm{e}{-4}\,\|\theta_t\| + 3\mathrm{e}{-4}\,\|\dot{\theta}_t\| + 5\mathrm{e}{-4}\,\|\dot{x}_t\|\right) \tag{6}$$

The reward function described in (6) essentially gives the RL agent a higher reward when the agent is closer to the goal position and with an extra incentive for the agent for being closer to hovering while minimizing drift. But, it can also be seen that (6) was also hand-engineered and fine-tuned for the coefficients for the error terms, hinting at extensive experimentation by the authors in [27] to decide upon these terms.

We show that computing the locally shaped rewards from the BHNR signal of the trajectory of the simulation works just as well as the reward function described in (6) if we use the following property:

$$\varphi = \mathbf{G}\left(\mathbf{F}_{[0,5]}(\|x_t - g_t\| \le 0.05) \wedge (\|\dot{\theta}_t\| \le 5^\circ/s)\right) \tag{7}$$

This property specifies the following behavior:

> The drone must get within 5cm of the goal within 5 seconds, while the angular velocity is always within $\pm 5^\circ/s$.

We can argue that (7) is a combination of a goal specification and a safety specification, where the angular constraints are the safety specifications and the expression depending on position error is the goal specification. We monitor the trajectory using the classical quantitative semantics of STL defined in [30]. It should be noted that if the quadrotor reaches the position within 5 seconds and hovers, the reward it obtains will be positive, but if it fails to satisfy the property, the rewards will be large and negative.

In Fig. 3, we see similar results as in Fig. 2: the BHNR-rewarded agent learns faster and achieves a higher normalized reward than the vanilla agent. This shows that an intuitive, structured reward function defined using STL properties performs much better than a hand-engineered reward function.

Remark. *The results are presented in a normalized fashion as the scale of the two reward functions are significantly different. In (6), we see that reward function has a maximum value of 0, but the reward itself is very small. Meanwhile, the robust satisfaction value against (7) can have a max of ≈ 0 (like in the vanilla reward function) but the reward values have relatively large magnitude.*

VI. CONCLUSION

In this work, we presented a framework for designing *locally shaped reward functions* by defining STL specifications

[1] https://gym.openai.com/envs/RoboschoolHumanoidFlagrun-v1/

that models the desired behavior of the system being controlled by a (deep) RL controller. We do this by introducing the notion of *bounded horizon nominal robustness* of a partial signal trace of a system, with respect to a STL property, and demonstrate the potential of using temporal logic as a means to train deep RL controllers using two examples, a cart-pole example and a quadrotor control example. While defining logical specifications for RL tasks is useful, using quantitative semantics for temporal logics in training RL agents has one drawback: if different state variables operate on different scales, they contribute disproportionately to the robustness computation. This can cause training to become erratic when there are multiple sub-formulas on the different state variables. In future work, we would like to extend the use of temporal logics in training RL agents to accomplish multiple goals, and also study its use in defining hierarchical learning models.

Acknowledgment

This work was funded in part by a grant from Toyota Motors North America R&D. Computation was supported by the University of Southern California's Center for High-Performance Computing (hpc.usc.edu).

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[2] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv:1509.02971 [cs, stat]*, Sept. 2015. [Online]. Available: http://arxiv.org/abs/1509.02971

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., ser. Adaptive Computation and Machine Learning Series. Cambridge, MA: The MIT Press, 2018.

[4] J. Leike, M. Martic, V. Krakovna, P. A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, and S. Legg, "AI Safety Gridworlds," *arXiv:1711.09883 [cs]*, Nov. 2017. [Online]. Available: http://arxiv.org/abs/1711.09883

[5] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete Problems in AI Safety," *arXiv:1606.06565 [cs]*, June 2016. [Online]. Available: http://arxiv.org/abs/1606.06565

[6] M. Grześ, "Reward Shaping in Episodic Reinforcement Learning," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '17. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2017, pp. 565–573. [Online]. Available: http://dl.acm.org/citation.cfm?id=3091125.3091208

[7] A. Donzé, X. Jin, J. V. Deshmukh, and S. A. Seshia, "Automotive systems requirement mining using breach," in *2015 American Control Conference (ACC)*, July 2015, pp. 4097–4097.

[8] J. Kapinski, X. Jin, J. Deshmukh, A. Donze, T. Yamaguchi, H. Ito, T. Kaga, S. Kobuna, and S. Seshia, "ST-Lib: A Library for Specifying and Classifying Model Behaviors," SAE International, Warrendale, PA, SAE Technical Paper 2016-01-0621, Apr. 2016.

[9] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Nĭcković, and S. Sankaranarayanan, "Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications," in *Lectures on Runtime Verification: Introductory and Advanced Topics*, E. Bartocci and Y. Falcone, Eds. Cham: Springer International Publishing, 2018, pp. 135–175.

[10] J. V. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. A. Seshia, "Robust online monitoring of signal temporal logic," *Formal Methods in System Design*, vol. 51, no. 1, pp. 5–30, Aug. 2017.

[11] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[12] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-Learning for robust satisfaction of signal temporal logic specifications," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec. 2016, pp. 6565–6570.

[13] X. Li, C. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2017, pp. 3834–3839.

[14] P. Abbeel and A. Y. Ng, "Apprenticeship Learning via Inverse Reinforcement Learning," in *Proceedings of the Twenty-First International Conference on Machine Learning*, ser. ICML '04. New York, NY, USA: ACM, 2004, pp. 1–.

[15] A. Y. Ng, D. Harada, and S. J. Russell, "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping," in *Proceedings of the Sixteenth International Conference on Machine Learning*, ser. ICML '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 278–287. [Online]. Available: http://dl.acm.org/citation.cfm?id=645528.657613

[16] H. B. Suay, T. Brys, M. E. Taylor, and S. Chernova, "Learning from Demonstration for Shaping Through Inverse Reinforcement Learning," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, ser. AAMAS '16. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 429–437. [Online]. Available: http://dl.acm.org/citation.cfm?id=2936924.2936988

[17] S. Moarref and H. Kress-Gazit, "Decentralized control of robotic swarms from high-level temporal logic specifications," in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, Dec. 2017, pp. 17–23.

[18] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2010, pp. 92–106.

[19] S. Jakšić, E. Bartocci, R. Grosu, T. Nguyen, and D. Ničković, "Quantitative monitoring of STL with edit distance," *Formal Methods in System Design*, vol. 53, no. 1, pp. 83–112, Aug. 2018.

[20] Y. V. P. Houssam Abbas and R. Mangharam, "Temporal Logic Robustness for General Signal Classes," in *To Appear in the Proc. of Hybrid Systems: Computation and Control*, 2019.

[21] A. Rodionova, E. Bartocci, D. Nickovic, and R. Grosu, "Temporal Logic as Filtering," *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control - HSCC '16*, pp. 11–20, 2016.

[22] T. Akazaki and I. Hasuo, "Time Robustness in MTL and Expressivity in Hybrid System Falsification," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, D. Kroening and C. S. Păsăreanu, Eds. Springer International Publishing, 2015, pp. 356–374.

[23] S. Silvetti, L. Nenzi, E. Bartocci, and L. Bortolussi, "Signal Convolution Logic," in *Automated Technology for Verification and Analysis*, ser. Lecture Notes in Computer Science, S. K. Lahiri and C. Wang, Eds. Springer International Publishing, 2018, pp. 267–283.

[24] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, 1989.

[25] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 834–846, Sept. 1983.

[26] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016. [Online]. Available: http://arxiv.org/abs/1606.01540

[27] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[28] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," *arXiv:1509.06461 [cs]*, Sept. 2015. [Online]. Available: http://arxiv.org/abs/1509.06461

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347 [cs]*, July 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[30] A. Donzé, T. Ferrère, and O. Maler, "Efficient Robust Monitoring for STL," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, N. Sharygina and H. Veith, Eds. Springer Berlin Heidelberg, 2013, pp. 264–279.