2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

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

The problem of a reinforcement learning (RL) agent trying to exploit a faulty reward function and find a policy [1] that achieves high returns but against the designer's intentions is referred to as *reward hacking* in [1]. The inability of ad-hoc rewards to capture the semantics of complex tasks has negative repercussions on the learned policies. It is not easy to design and prove that increasing returns translate to better satisfaction of the specifications. [2] provides an illustrative example of reward hacking in RL.

In this paper, we use formal specification languages to capture the designer's requirements of what the robot should achieve. We propose *Truncated Linear Temporal Logic (TLTL)* as a specification language with an extended set of operators defined over finite-time trajectories of a robot's states. TLTL provides convenient and effective means of incorporating complex intentions, domain knowledge, and constraints into the learning process. We define quantitative semantics (also referred to as *robustness degree*) for TLTL. The robustness degree is used to transform temporal logical formulae into real-valued reward functions.

Making good use of the reward function in RL has not been the main focus in modern reinforcement learning research. Combining temporal logic with reinforcement learning to learn logically complex skills has been looked at only very recently. In [3], the authors used the log-sum-exp approximation to adapt the robustness of STL specifications 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.

II. BACKGROUND

A. Policy Search in Reinforcement Learning

We begin the introduction of policy search with the definition of an infinite MDP.

Definition 1. *An infinite MDP is a tuple $\langle S, A, p(\cdot,\cdot,\cdot), R(\cdot)\rangle$, where $S \subseteq \mathbb{R}^n$ is a continuous set of states; $A \subseteq \mathbb{R}^m$ is a continuous set of actions; $p : S \times A \times S \to [0,1]$ is the transition probability function with $p(s,a,s')$ being the probability of taking action $a \in A$ at state $s \in S$ and ending up in state $s' \in S$ (also commonly written as a conditional probability $p(s'|s,a)$); $R : \tau \to \mathbb{R}$ is a reward function where $\tau = (s_0, a_0, ..., s_T,)$ is the state-action trajectory, T is the horizon.*

In RL, the transition $p(s,a,s')$ is unknown to the learning agent. The reward function $R(\tau)$ can be designed or learned (as in the case of inverse reinforcement learning). The goal of RL is to find an optimal stochastic policy $\pi^\star : S \times A \to [0,1]$ that maximizes the expected accumulated reward, i.e.

$$\pi^\star = \arg\max_{\pi} E_{p^\pi(\tau)}\left[R(\tau)\right], \tag{1}$$

$p^\pi(\tau)$ is the trajectory distribution from following policy π. And $R(\tau)$ is the reward obtained given τ.

In policy search methods, the policy is represented by a parameterized model (e.g., neural network, radial basis function) denoted by $\pi(s,a|\theta)$ (also written as π_θ in short) where θ is the set of model parameters. Search is then conducted in the policy's parameter space to find the optimal set of θ that achieves (1)

X. Li and C. Belta are with Boston University, Boston, MA. Email: {xli87,cbelta}@bu.edu. C.-I. Vasile is with Massachusetts Institute of Technology, Cambridge, MA. Email: cvasile@mit.edu

This work is partially supported by the ONR under grants N00014-14-1-0554 and by the NSF under grants NRI-1426907 and CMMI-1400167

[1]We will use the terms controller and policy interchangeably throughout the paper.

978-1-5386-2682-5/17/$31.00 ©2017 IEEE

$$\theta^{\star} = \arg\max_{\theta} E_{p^{\pi_\theta}(\tau)} \left[R(\tau) \right], \tag{2}$$

B. Relative Entropy Policy Search

Relative Entropy Policy Search is an information-theoretic approach that solves the policy search problem. The episode-based version of REPS can be formulated as the following constrained optimization problem

$$\max_{p(\tau)} E_{p(\tau)} \left[R(\tau) \right] \;\; s.t. \; D_{KL}(p(\tau) || q^{\pi_\theta}(\tau)) < \epsilon, \tag{3}$$

where $q^{\pi_\theta}(\tau)$ is the trajectory distribution following the existing policy. $D_{KL}()$ is the KL-divergence between two policies and ϵ is a threshold. The optimization problem in (3) can be solved using the Lagrange multipliers method which results in a closed-form trajectory distribution update equation given by

$$p(\tau_i) \propto \exp\left(\frac{R(\tau_i)}{\eta}\right). \tag{4}$$

where η is the Lagrange multiplier obtained from optimizing the dual function

$$g(\eta) = \eta\epsilon + \eta \log \sum_i \frac{1}{N} \exp\left(\frac{R(\tau_i)}{\eta}\right) \tag{5}$$

We refer interested readers to [6] for detailed derivations.

We adopt the time-varying linear-Gaussian policies $\pi_{\theta_t} = \mathcal{N}(K_t s_t + k_t, \Sigma_t)$ (here $\theta_t = (k_t, \Sigma_t)$ for $t = 0, ..., T$) and weighted maximum-likelihood estimation to update the policy parameters (feedback gain K_t is kept fixed to reduce the dimension of the parameter space). This approach has been used in [7]. The difference is that [7] recomputes $p(\tau_i)$ at each step t using cost-to-go before updating θ_i. Since a temporal logic reward (described in the next section) depends on the entire trajectory, it doesn't have the notion of cost-to-go and can only be evaluated as a terminal reward. Therefore $p(\tau_i)$ (written short as p_i) is computed once and used for updates of all θ_t (similar approach used in episodic PI-REPS [8]). The resulting update equations are

$$\begin{aligned} k'_t &= \sum_i^N p_i k_{i,t} \\ \Sigma'_t &= \sum_i^N p_i (k_{i,t} - k'_t)(k_{i,t} - k'_t)^T, \end{aligned} \tag{6}$$

where $k_{i,t}$ is the feed-forward term in the time-varying linear-Gaussian policy at time t and for sample trajectory i.

III. Truncated Linear Temporal Logic(TLTL)

In this section, *we propose TLTL, a new temporal logic that we argue is well suited for specifying goals and introducing domain knowledge for the RL problem.*

A. TLTL Syntax And Semantics

A TLTL formula is defined over predicates of form $f(s) < c$, where $f : \mathbb{R}^n \to \mathbb{R}$ is a function and c is a constant. A TLTL specification has the following syntax:

$$\begin{aligned} \phi := \top \mid f(s) < c \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \\ \Diamond\phi \mid \Box\phi \mid \phi\,\mathcal{U}\,\psi \mid \phi\,\mathcal{T}\,\psi \mid \bigcirc\phi \mid \phi \Rightarrow \psi, \end{aligned} \tag{7}$$

where $f(s) < c$ is a predicate, $\neg$ (negation/not), $\wedge$ (conjunction/and), and $\vee$ (disjunction/or) are Boolean connectives, and $\Diamond$ (eventually), $\Box$ (always), $\mathcal{U}$ (until), $\mathcal{T}$ (then), $\bigcirc$ (next), are temporal operators. Implication is denoted by $\Rightarrow$ (implication). TLTL formulas are evaluated against finite time sequences over a set S. As it will become clear later, such sequences will be produced by the MDP in Definition 1.

We denote $s_t \in S$ to be the state at time t, and $s_{t:t+k}$ to be a sequence of states (state trajectory) from time t to $t+k$, i.e., $s_{t:t+k} = s_t s_{t+1} ... s_{t+k}$. The Boolean semantics of TLTL is defined as:

$$\begin{aligned}
s_{t:t+k} &\models f(s) < c && \Leftrightarrow \quad f(s_t) < c, \\
s_{t:t+k} &\models \neg\phi && \Leftrightarrow \quad \neg(s_{t:t+k} \models \phi), \\
s_{t:t+k} &\models \phi \Rightarrow \psi && \Leftrightarrow \quad (s_{t:t+k} \models \phi) \Rightarrow (s_{t:t+k} \models \psi), \\
s_{t:t+k} &\models \phi \wedge \psi && \Leftrightarrow \quad (s_{t:t+k} \models \phi) \wedge (s_{t:t+k} \models \psi), \\
s_{t:t+k} &\models \phi \vee \psi && \Leftrightarrow \quad (s_{t:t+k} \models \phi) \vee (s_{t:t+k} \models \psi), \\
s_{t:t+k} &\models \bigcirc\phi && \Leftrightarrow \quad (s_{t+1:t+k} \models \phi) \wedge (k > 0), \\
s_{t:t+k} &\models \Box\phi && \Leftrightarrow \quad \forall t' \in [t, t+k) \; s_{t':t+k} \models \phi, \\
s_{t:t+k} &\models \Diamond\phi && \Leftrightarrow \quad \exists t' \in [t, t+k) \; s_{t':t+k} \models \phi, \\
s_{t:t+k} &\models \phi\;\mathcal{U}\;\psi && \Leftrightarrow \quad \exists t' \in [t, t+k) \; s.t. \; s_{t':t+k} \models \psi \\
& && \quad\quad \wedge (\forall t'' \in [t, t') \; s_{t'':t'} \models \phi), \\
s_{t:t+k} &\models \phi\;\mathcal{T}\;\psi && \Leftrightarrow \quad \exists t' \in [t, t+k) \; s.t. \; s_{t':t+k} \models \psi \\
& && \quad\quad \wedge (\exists t'' \in [t, t') \; s_{t'':t'} \models \phi).
\end{aligned}$$

Intuitively, state trajectory $s_{t:t+k} \models \Box\phi$ if the specification defined by ϕ is satisfied for every subtrajectory $s_{t':t+k}$, $t' \in [t, t+k)$. Similarly, $s_{t:t+k} \models \Diamond\phi$ if ϕ is satisfied for at least one subtrajectory $s_{t':t+k}$, $t' \in [t, t+k)$. $s_{t:t+k} \models \phi\;\mathcal{U}\;\psi$ if ϕ is satisfied at every time step before ψ is satisfied, and ψ is satisfied at a time between t and $t+k$. $s_{t:t+k} \models \phi\;\mathcal{T}\;\psi$ if ϕ is satisfied at least once before ψ is satisfied between t and $t+k$. A trajectory s of duration k is said to satisfy formula ϕ if $s_{0:k} \models \phi$.

We equip TLTL with quantitative semantics (robustness degree) , i.e., a real-valued function $\rho(s_{t:t+k}, \phi)$ of state trajectory $s_{t:t+k}$ and a TLTL specification ϕ that indicates how far $s_{t:t+k}$ is from satisfying or violating the specification

ϕ. The quantitative semantics of TLTL is defined as follows:

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

where ρ_{max} represents the maximum robustness value. Moreover, $\rho(s_{t:t+k}, \phi) > 0 \Rightarrow s_{t:t+k} \models \phi$ and $\rho(s_{t:t+k}, \phi) < 0 \Rightarrow s_{t:t+k} \not\models \phi$, which implies that the robustness degree can substitute Boolean semantics in order to enforce the specification ϕ. As an example, consider specification $\phi = \Diamond(s < 10)$, where s is a one dimensional state, and a two step state trajectory $s_{0:2} = s_0 s_1 = [11, 5]$. The robustness is $\rho(s_{0:1}, \phi) = \max_{t \in \{0,1\}} (10 - s_t) = \max(-1, 5) = 5$. Since $\rho(s_t, \phi) > 0$, $s_{0:1} \models \phi$ and the value $\rho(s_t, \phi) = 5$ is a measure of the satisfaction margin (refer to *Example 1* in [3] for a more detail example on task specification using TL and robustness).

B. Comparison With Existing Formal Languages

One of the most important elements in using a formal language in reinforcement learning is the ability to transform a specification into a real-valued function that can be used as reward. This requires quantitative semantics to be defined for the chosen language. One obvious choice is Signal Temporal Logic (STL) [9](related to Metric Temporal Logic (MTL), which is defined over infinite real-valued signals with a time bound required for every temporal operator. While this is useful for analyzing signals, it can cause problems when defining tasks for robots. For example if the goal is to have the robot learn to put a beer in the fridge, the robot only needs to find the correct way to operate a fridge (e.g. open the fridge door, place the beer on a shelf and close the fridge door) and possibly perform this sequence of actions at an acceptable speed. But using STL to specify this task would require the designer to manually put time bounds on how long each action/subtask should take. If this bound is set inappropriately, the robot may fail to find a satisfying policy due to its hardware constraints even though it is capable of performing the task. This is quite common in robotic tasks where we care about the robot accomplishing the given task but don't have hard constraints on when and how fast the task should be finished. In this case mandatory time bounds add unnecessary complexity to the specification and thus the overall learning process.

Two other possible choices are BLTL [10] and LTL_f [11]. Both can be evaluated over finite sequences of states. However, similar to STL, temporal operators in BLTL require time bounds. Both languages are defined over atomic propositions rather than predicates, and do not come with quantitative semantics.

With the above requirements in mind, we design TLTL such that its formulas over state predicates can be evaluated against finite trajectories of any length. In the context of reinforcement learning this can be the length of an execution episode. TLTL does not require a time bound to be specified with every use of a temporal operator. If however the user feels that explicit time bounds are helpful in certain cases, the semantics of STL can be easily incorporated into TLTL.

IV. Experiments

In this section we first use two simulated manipulation tasks to compare TLTL reward with a discrete reward as well as a distance-based continuous reward commonly used in the RL literature. We then specify a toast placing task in TLTL where a Baxter robot is required to learn a combination of reaching policy and gripper timing policy [2].

A. Simulated 2D Manipulation Tasks

Figure 1 shows our 2D simulated environment with a three joint manipulator. The 8 dimensional state feature space includes joint angles, joint velocities and the end-effector position. The 3 dimensional action space includes the joint velocities. Exploration is automatically taken care of by the covariance update in Equation (6).

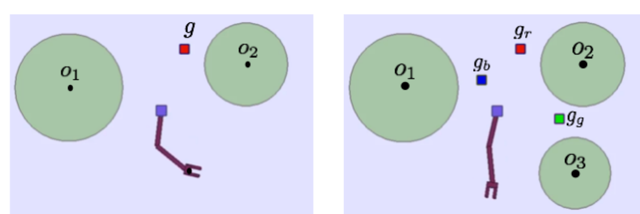

Fig. 1 : 2D manipulation tasks. *left*: Task 1. goal reaching while avoiding obstacles. *right*: Task 2. sequential goal reaching while avoiding obstacles

For the first task, the end-effector is required to reach the goal position $\boldsymbol{g}$ while avoiding obstacles $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$. The discrete and continuous rewards are summarized as follows:

[2]The simulation is implemented in rllab [12] and gym [13]. The experiment is implemented in rllab and ROS

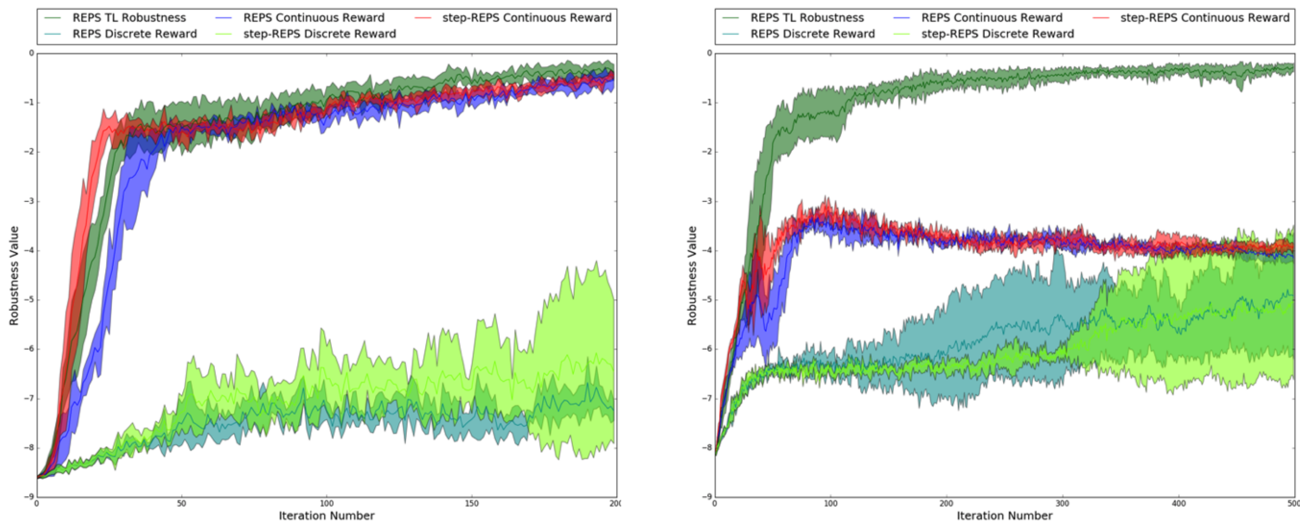

Fig. 2 : Learning curves for TLTL robustness, discrete reward and continuous reward trained with episode based REPS, as well as discrete and continuous rewards trained with step based REPS. *left*: task 1, each episode is 200 time-steps, each iteration uses 20 sample trajectories and trained for 200 iterations *right*: task 2, each episode is 500 time-steps, 20 samples per iteration and trained for 500 iterations

$$r_1^{discrete} = \begin{cases} 5 & d_g \leq 0.2 \\ -2 & d_{o_{1,2}} \leq r_{o_{1,2}} \\ 0 & \text{everywhere else} \end{cases} \tag{8}$$

$$r_1^{continuous} = -c_1 d_g + c_2 \sum_{i=1}^{2} d_{o_i}.$$

In the above rewards, d_g is the Euclidean distance between the end-effector and the goal, d_{o_i} is the distance between the end-effector and obstacle i,r_{o_i} is the radius of obstacle i. The TLTL specification and its resulting robustness function is described as

$$\phi_1 = \Diamond\Box(d_g < 0.2) \wedge \Box(d_{o_1} > r_{o_1} \wedge d_{o_2} > r_{o_2}) \tag{9}$$

$$\rho_1(\phi_1, (x_e, y_e)_{0:T}) = \min\Bigg(\max_{t\in[0,T)}\Big(\min_{t'\in[t,T)}\big(0.2 - d_g^t\big)\Big), \min_{t\in[0,T)}\Big(d_{o_1}^t - r_{o_1}, d_{o_2}^t - r_{o_2}\Big)\Bigg). \tag{10}$$

In English, ϕ_1 describes the task of "*eventually always* stay at goal $\boldsymbol{g}$ *and always* stay alway from obstacles". The user needs only to specify ϕ_1 and the reward function ρ_1 is generated automatically from the quantitative semantics indicated in Section III-A. Here $(x_e, y_e)_{0:T}$ is the trajectory of the end-effector position. d^t is the is distance at time t.

For the second task, the gripper is required to visit goals $\boldsymbol{g}_r$, $\boldsymbol{g}_g$, and $\boldsymbol{g}_b$ in this specific sequence while avoiding the obstacles (one more obstacle is added to further constrain the free space). The discrete and continuous rewards are summarized as

$$r_2^{discrete} = \begin{cases} 5 & \text{goals visited in the right order} \\ -5 & \text{goals visited in the wrong} \\ -2 & d_{o_{1,2,3}} \leq r_{o_{1,2,3}} \\ 0 & \text{everywhere else} \end{cases} \tag{11}$$

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

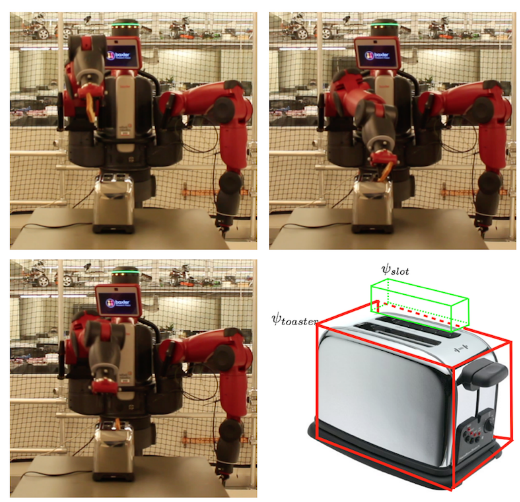

Fig. 3 : *first three*: Experiment execution. The joint states are measured by encoders, the end-effector states are tracked using the motion tracking system (cameras in the back). *last*: Definition of toaster region predicates. Each episode has a horizon of 100 time-steps (around 6 seconds) and each update iteration uses 10 sample trajectories. Episode based REPS is again used as the RL algorithm for this task.

Here an addition state vector is maintained to record which goals have already been visited in order to know what the next goal is. In $r_2^{continuous}$, g_i is the correct next goal to visit and g_j, g_k are the goals to avoid. The TLTL specification is defined as

$$\begin{aligned}\phi_2 =&(\psi_{g_r}\ \mathcal{T}\ \psi_{g_g}\ \mathcal{T}\ \psi_{g_b}) \wedge (\neg(\psi_{g_g} \vee \psi_{g_b})\ \mathcal{U}\ \psi_{g_r}) \wedge \\ &(\neg(\psi_{g_b})\ \mathcal{U}\ \psi_{g_g}) \wedge (\bigwedge_{i=r,g,b} \Box(\psi_{g_i} \Rightarrow \bigcirc\Box\neg\psi_{g_i})) \wedge \Box\psi_o,\end{aligned} \tag{12}$$

where $\psi_{g_i} : d_{g_i} < 0.2$ is the predicate for goal g_i, $\psi_o : \bigwedge_{j=1,2,3} d_{o_j} > r_{o_j}$ is the obstacle avoidance constraint ($\bigwedge$ is a shorthand for a sequence of conjunction). In English, ϕ_2 states "visit g_r *then* g_g *then* g_b, *and don't* visit g_g *or* g_b *until* visiting g_r, *and don't* visit g_b *until* visiting g_g, *and always* if visited g_i *implies next always don't* visit g_i (don't revisit goals), *and always* avoid obstacles" . Due to space constraints the robustness of ϕ_2 will not be explicitly presented, but it will also be a complex function consisted of nested $\min()/\max()$ functions that would be difficult to design by hand but can be generated from the quantitative semantics of TLTL.

To compare the influence of reward functions on the learning outcome, we first fix the learning algorithm to be the episode based REPS and compare the average return per iteration for TLTL robustness reward, discrete reward and continuous reward. However it is meaningless to compare returns on different scales. We therefore take the sample trajectories learned with $r^{discrete}$ and $r^{continuous}$ and calculate their corresponding TLTL robustness return for comparison. The reason for choosing TLTL robustness as the comparison measure is that both the discrete and continuous rewards have semantic ambiguity depending on the choices of the discrete returns and coefficients c_i. TLTL is rigorous in its semantics and a robustness greater than zero guarantees satisfaction of the task specification.

In addition, since $r^{discrete}$ and $r^{continuous}$ can provide a immediate reward per step (as oppose to TLTL robustness which requires the entire trajectory to produce a terminal reward), we also used a step based REPS[7] that updates at each step using the cost-to-go. This is a common technique used to reduce the variance in the Monte Carlo return estimate. For continuous rewards, a grid search is performed on the coefficients c_i and the best outcome is reported. We train each comparison case on 4 different random seeds. The mean and variance of the average returns are illustrated in Figure 2.

It can be observed that in both tasks TLTL robustness reward resulted in the best learning outcome in terms of convergence rate and final return. For the level of stochasticity presented in the simulation, step based REPS showed only minor improvement in the rate of convergence and variance reduction. For the simpler case of task 1, a well tuned continuous reward achieves comparable learning performance with the TLTL robustness reward. For task 2, the TLTL reward outperforms competing reward functions by a considerable margin. Discrete reward fails to learn a useful policy due to sparse returns. A video of the learning process is provided.

B. Learning Toast-Placing Task With A Baxter Robot

In this experiment, a Baxter robot is used to perform the task of placing a piece of bread in a toaster (as shown in Figure (3)).The robot will simultaneously learn to reach the specified region and a gripper timing policy that releases the object at the right instant (as oppose to directly specifying the point of release). The 21 dimensional state feature space includes 7 joint angles and joint velocities, the xyz-rpy pose of the end-effector and the gripper position. The end-effector pose is tracked using the motion tracking system as an additional source of information. The gripper position ranges continuously from 0 to 100 with 0 being fully closed. The 8 dimensional action space includes 7 joint velocities and the desired gripper position. Actions are sent at 20hz.

The placing task is specified by the TLTL formula

$$\begin{aligned}\phi =&\Box(\neg(\psi_{table} \vee \psi_{toaster})) \wedge \Diamond(\psi_{slot}) \wedge \\ &(\psi_{gc}\ \mathcal{U}\ \psi_{slot}) \wedge \Box(\psi_{slot} \Rightarrow \bigcirc\Box(\psi_{go})),\end{aligned} \tag{13}$$

where $\psi_{table}, \psi_{toaster}, \psi_{slot}$ are predicates describing spatial regions in the form $(x_{min} < x_e < x_{max}) \wedge (y_{min} < y_e < y_{max}) \wedge (z_{min} < z_e < z_{max})$ ((x_e, y_e, z_e) is the position of the end-effector). Orientation constraints are specified in a similar way to ensure the correct pose is reached at the position of release. The regions for $slot$ and $toaster$ are depicted in Figure 3. $\psi_{gc} : p_g < \delta_{close}$ and $\psi_{go} : p_g > \delta_{open}$ describe the conditions for gripper open/close. In English, the specification describes the process of "*always don't* hit the table *or* the toaster, *and eventually* reach the slot, *and* keep gripper closed *until* slot is reached, *and always* if slot is reached *implies next always* keep gripper open". The robustness of ϕ ($\rho(\phi, p^e_{0:T})$) is again generated from the TLTL quantitative semantics and is satisfied when $\rho(\phi, p^e_{0:T}) > 0$. Due to space constraints $\rho(\phi, p^e_{0:T})$ will not be explicitly shown. The robustness for ψ_{gc} and ψ_{go} are normalized to the same scale as that of the other predicates. This is to ensure that all sub-formulas are treated equally during learning.

For a comparison case, we design the following reward function

$$r_t = \begin{cases} -c_1 d^t_{slot} + c_2 d^t_{toaster} - c_3|p^t_g| & \min\limits_{t' \in [0,t)} d^t_{slot} > 0.03 \\ -c_1 d^t_{slot} + c_2 d^t_{toaster} - c_3|100 - p^t_g| & \min\limits_{t' \in [0,t)} d^t_{slot} < 0.03. \end{cases} \tag{14}$$

In the above equation, d^t_{slot} and $d^t_{toaster}$ are the Euclidean distances between the end-effector and the center of the toaster regions defined in Figure 3 (at time t). p^t_g is the gripper position at time t. The coefficients $c_{1,2,3}$ are manually tuned and the best outcome is reported.

In Figure 4, trajectories learned from r_t at each iteration are used to calculated their corresponding robustness

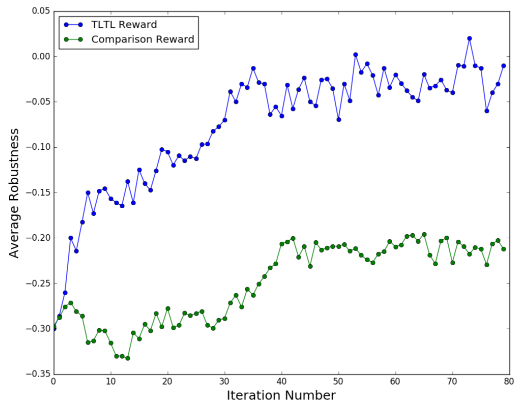

Fig. 4 : Training curves for Baxter toast-placing task. An episode is 100 time-steps long (around 6 seconds). Each update iteration uses 10 sample trajectories. Trained for 80 iterations

value (as explained in the previous section) for a reasonable comparison. We can observe that training with TLTL reward has reached a significantly better policy than that with the comparison reward. One important reason is that the semantics of r_t in Equation (14) relies heavily on the relative magnitudes of the coefficients $c_{1,2,3}$. For example if c_1 is much higher than c_2 and c_3, then r_t will put most emphasis on reaching the slot and pay less attention on learning the correct gripper timing policy or obstacle avoidance. An exhaustive hyperparameter search on the physical robot is infeasible. In addition, r_t expresses much less information than $\rho(\phi, p^e_{0:T})$. For example, penalizing collision with the toaster is necessary only when the gripper comes in contact with the toaster. Otherwise the agent should focus on the other subtasks (reaching the slot, improving the gripper policy). For reward r_t, this logistics is again achieved only by obtaining the right combination of hyperparameters. However, because the robustness function is made up of a series of embedded $\min()/\max()$ functions, at any instant the agent will be maximizing only a set of active functions. These active functions represent the bottlenecks in improving the overall return. By adopting this form, the robustness reward effectively focuses the agent's effort in improving the most critical set of subtasks at any time so to achieve an efficient overall learning progress. However, this may render the TLTL robustness reward susceptible to scaling Therefore, proper normalization is required. Currently this normalization process is achieved manually, future work can include automatic or adaptive normalization of predicate robustnesses.

To evaluate the resulting behavior, 10 trials of the toast-placing task is executed with the policy learned from each reward. The policy from the TLTL reward achieves 100% success rate while the comparison reward fails to learn the task (due to its inability to learn the correct gripper time policy). The specification in Equation (14) does not impose constraints on joint efforts, resulting in some minor quivering motion. This can be alleviated by setting action bounds in the TLTL formula. A video of the learning process is provided.

V. Conclusion

In this paper we proposed TLTL, a formal specification language with quantitative semantics that is designed for convenient robotic task specification. We compare learning performance of the TLTL reward with two of the more commonly used forms of reward (namely a discrete and continuous form of reward functions) in a 2D simulated manipulation environment by fixing the RL algorithm. We also compare the outcome of TLTL reward trained using a relatively inefficient episode based method with the discrete/continuous rewards trained using a lower variance step based method. Results show that TLTL reward not only outperformed all of its comparison cases, it also enabled a non-hierarchical RL method to successfully learn to perform a temporally structured task. Furthermore, We used TLTL to express a toast-placing task and demonstrated successful learning on a Baxter robot.

References

[1] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete Problems in AI Safety," pp. 1–29, 2016. [Online]. Available: http://arxiv.org/abs/1606.06565

[2] A. Dario and J. Clark. Faulty reward functions in the wild. [Online]. Available: https://blog.openai.com/faulty-reward-functions/

[3] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q - Learning for Robust Satisfaction of Signal Temporal Logic Specifications," 2016.

[4] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, S. Seshia, and Others, "A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications," *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pp. 1091–1096, 2014.

[5] J. Fu and U. Topcu, "Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints," 2014. [Online]. Available: http://arxiv.org/abs/1404.7073

[6] M. P. Deisenroth, "A Survey on Policy Search for Robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1, pp. 1–142, 2011. [Online]. Available: http://www.nowpublishers.com/articles/foundations-and-trends-in-robotics/ROB-021

[7] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine, "Path integral guided policy search," *arXiv preprint arXiv:1610.00529*, 2016.

[8] V. Gómez, H. J. Kappen, J. Peters, and G. Neumann, "Policy search for path integral control," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 482–497.

[9] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6246 LNCS, pp. 92–106, 2010.

[10] T. Latvala, A. Biere, K. Heljanko, and T. Junttila, "Simple bounded LTL model checking," *Formal Methods in Computer-Aided Design*, vol. 3312, no. LCNS, pp. 186–200, 2004. [Online]. Available: http://www.springerlink.com/index/A1JNFCB7Q9KNC1Q1.pdf

[11] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and Linear Dynamic Logic on finite traces," *IJCAI International Joint Conference on Artificial Intelligence*, pp. 854–860, 2013.

[12] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

[13] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.