Temporal Logic Guided Safe Reinforcement Learning Using Control Barrier Functions

Xiao Li and Calin Belta

***Abstract*— Using reinforcement learning to learn control policies is a challenge when the task is complex with potentially long horizons. Ensuring adequate but safe exploration is also crucial for controlling physical systems. In this paper, we use temporal logic to facilitate specification and learning of complex tasks. We combine temporal logic with control Lyapunov functions to improve exploration. We incorporate control barrier functions to safeguard the exploration and deployment process. We develop a flexible and learnable system that allows users to specify task objectives and constraints in different forms and at various levels. The framework is also able to take advantage of known system dynamics and handle unknown environmental dynamics by integrating model-free learning with model-based planning.**

I. INTRODUCTION

Our aim in this work is to address two challenges in current reinforcement learning (RL) methods - complex task specification and safe exploration. While much effort has been made in the development of sample efficient RL algorithms, the abilities to accurately specify the desired behavior and ensure safety during learning are paramount to the deployment of learning agents on physical systems.

Because reinforcement learning agents aim to maximize the expected return, they often exploit loopholes in the reward definition to gain an edge in achieving this goal and this results in undesired behavior. This phenomenon is often referred to as reward hacking [1] (some examples are provided in [2]). For tasks with higher complexity, an accurate definition of a reward function is challenging. For example, the authors of [3] shaped a reward function for the peg insertion task consisting of nested min/max functions that requires tuning to perform well. In this work, we use the robustness degree of temporal logic formulas as the reward function which is also constructed with nested min/max functions. However, the user only needs to specify the task as a logic formula and the transformation to the reward function is performed automatically. The robustness reward also has rigorous semantics such that a value greater than zero indicates completion of the task.

Safety is a critical aspect of reinforcement learning when physical systems that potentially operate around humans are involved. This issue is often addressed by constructing a similar environment in simulation, perform training in the simulation environment and transfer the learned policy to the real world [4][5]. However, it is difficult for the dynamics of the simulation to exactly resemble that of reality and some fine-tuning is necessary for the sim-to-real transfer to succeed. Therefore, it is often an effective practice to incorporate a shielding controller both during learning and at deployment to ensure safety as well as to guide exploration [6][7].

In this paper, we propose a system that integrates temporal logic guided reinforcement learning with control barrier functions and control Lyapunov functions [8]. Our main contributions are as follows:

- we propose three ways to use temporal logic and its equivalent finite state automata (FSA) in our system - provide reward for the RL agent, perform goal selection for the control Lyapunov function for guided exploration and define safe sets for the control barrier functions.
- we extend the FSA augmented MDP framework [9] to handle hard constraints and violation of specifications.
- we integrate the system into a quadratic program that can be solved efficiently.
- we demonstrate the applicability of our framework in a simulated continuous control task with safety constraints, known system dynamics and unknown environmental dynamics, and study the use and effectiveness of each component in the system.

II. RELATED WORK

Using temporal logic in an RL setting has been explored in the past. The authors of [10] and [11] use temporal logic and automata to solve the non-Markovian reward decision process (NMRD). In [12], the authors take advantage of the robustness degree of signal temporal logic to guide the learning process in discrete MDPs. In [13], the authors propose a reward machine, which in effect is an FSA. However, the user is required to manually design the reward machine. In [14] and [15], linear temporal logic (LTL) and geometric linear temporal logic (GLTL) are combined with MDP to learn policies in discrete environments.

Safe exploration and deployment is critical for applying learning agents on physical systems. The authors of [7] and [16] approach the problem in the context of constrained policy optimization and showed minimum violation of safety constraints during learning. The authors of [17] and [18] utilize control barrier functions to safeguard exploration. Temporal logic is used in conjunction with an abstraction of the system dynamics to shield the learning process from unsafe actions in [6]. The Authors of [19] provides a comprehensive survey on safe reinforcement learning. We incorporate hard constraints in a TL specification with control barriers functions such that these constraints can be strictly enforced (in addition to other user specified constraints) during learning. Compared to previous work, our method

is relatively easy to implement, able to handle continuous state and action spaces, nonlinear constraints and dynamics, and can be executed efficiently.

III. PRELIMINARIES

A. Reinforcement Learning

We start with the definition of a Markov Decision Process.

Definition 1. *An MDP is defined as a tuple* $\mathcal{M} = \langle S, A, p(\cdot|\cdot,\cdot), r(\cdot,\cdot,\cdot)\rangle$*, where* $S \subseteq \mathbb{R}^n$ *is the state space ;* $A \subseteq \mathbb{R}^m$ *is the action space (*S *and* A *can also be discrete sets);* $p : S \times A \times S \to [0, 1]$ *is the transition function with* $p(s'|s, a)$ *being the conditional probability density of taking action* $a \in A$ *at state* $s \in S$ *and ending up in state* $s' \in S$*;* $r : S \times A \times S \to \mathbb{R}$ *is the reward function with* $r(s, a, s')$ *being the reward obtained by executing action* a *at state* s *and transitioning to* s'*.*

We define a task to be the process of finding the optimal policy $\pi^\star : S \to A$ (or $\pi^\star : S \times A \to [0, 1]$ for stochastic policies) that maximizes the expected return, i.e.

$$\pi^\star = \arg\max_\pi \mathbb{E}^\pi[\sum_{t=0}^{T-1} r(s_t, a_t, s_{t+1})], \tag{1}$$

The horizon of a task (denoted T) is defined as the maximum allowable time-steps of each execution of π and hence the maximum length of a trajectory. In Equation (1), $\mathbb{E}^\pi[\cdot]$ is the expectation following π. The state-action value function is defined as

$$Q^\pi(s, a) = \mathbb{E}^\pi[\sum_{t=0}^{T-1} r(s_t, a_t, s_{t+1})|s_0 = s, a_0 = a] \tag{2}$$

i.e. it is the expected return of choosing action a at state s and following π onwards. For off-policy actor critic methods such as deep deterministic policy gradient [20], Q^π is used to evaluate the quality of policy π. Parameterized $Q^\pi_{\theta_Q}$ and π_{θ_π} (θ_Q and θ_π are learnable parameters) are optimized alternately to obtain $\pi^\star_{\theta_\pi}$.

B. scTLTL And Finite State Automata

We consider tasks specified with *syntactically co-safe Truncated Linear Temporal Logic* (scTLTL) which is a restricted version of truncated linear temporal logic(TLTL) [21]. In particular, we restrict from using the $\square$ (always) operator. By doing so, we can establish a correspondence between an scTLTL formula with a FSA.

Due to space constraints, we do not provide the complete set of definitions for the Boolean and quantitative semantics of scTLTL (refer to [21]). Examples of scTLTL include $\Diamond(\phi_a \wedge \Diamond\phi_b)$ which entails that *eventually* ϕ_a and then *eventually* ϕ_b become true (sequencing). Another example $(\phi_a \Rightarrow \Diamond\phi_b)\ \mathcal{U}\ \phi_c$ means *until* ϕ_c becomes true, ϕ_a is true *implies* that *eventually* ϕ_b is true.

We denote $s_t \in S$ to be the MDP 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., $s_{t:t+k} = s_t s_{t+1} ... s_{t+k}$. scTLTL provides a set of real-valued functions that quantify the degree of satisfaction of a given $s_{0:T}$ with respective to a formula ϕ. This measure is also referred to as robustness degree or simply robustness ($\rho(s_{0:T}, \phi)$ maps a state trajectory and a formula to a real number). For example, $\rho(s_{0:3}, \Diamond(s < 4)) = \max(4 - s_0, 4 - s_1, 4 - s_2)$. Here, if $4 - s_t > 0$, then $s_t < 4$ is satisfied. Because $\Diamond(s < 4)$ requires $s < 4$ to be true at least once in the trajectory, hence we take the max over the time horizon. In general, a robustness of greater than zero implies that $s_{t:t+k}$ satisfies ϕ and vice versa.

Definition 2. *An FSA corresponding to a scTLTL formula* ϕ*. is defined as a tuple* $\mathcal{A}_\phi = \langle \mathbb{Q}_\phi, \Psi_\phi, q_{\phi,0}, p_\phi(\cdot|\cdot), \mathcal{F}_\phi\rangle$*, where* $\mathbb{Q}_\phi$ *is a set of automaton states;* Ψ_ϕ *is the input alphabet (a set of first order logic formulas);* $q_{\phi,0} \in \mathbb{Q}_\phi$ *is the initial state;* $p_\phi : \mathbb{Q}_\phi \times \mathbb{Q}_\phi \to [0, 1]$ *is a conditional probability defined as*

$$p_\phi(q_{\phi,j}|q_{\phi,i}) = \begin{cases} 1 & \psi_{q_{\phi,i},q_{\phi,j}} \text{ is true} \\ 0 & \text{otherwise.} \end{cases}$$
$$\text{or} \tag{3}$$
$$p_\phi(q_{\phi,j}|q_{\phi,i}, s) = \begin{cases} 1 & \rho(s, \psi_{q_{\phi,i},q_{\phi,j}}) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

$\mathcal{F}_\phi$ *is a set of final automaton states.*

Here $q_{\phi,i}$ is the i^{th} automaton state of $\mathcal{A}_\phi$. $\psi_{q_{\phi,i},q_{\phi,j}} \in \Psi_\phi$ is the predicate guarding the transition from $q_{\phi,i}$ to $q_{\phi,j}$. Because $\psi_{q_{\phi,i},q_{\phi,j}}$ is a predicate without temporal operators, the robustness $\rho(s_{t:t+k}, \psi_{q_{\phi,i},q_{\phi,j}})$ is only evaluated at s_t. Therefore, we use the shorthand $\rho(s_t, \psi_{q_{\phi,i},q_{\phi,j}}) = \rho(s_{t:t+k}, \psi_{q_{\phi,i},q_{\phi,j}})$. The FSA corresponding to a scTLTL formula can be generated automatically with available packages like Lomap [22] (refer to [23] for details on the generation procedure).

C. FSA Augmented MDP

The FSA augmented MDP $\mathcal{M}_\phi$ [9] establishes a connection between the TL specification and the standard reinforcement learning problem. A policy learned using $\mathcal{M}_\phi$ has implicit knowledge of the FSA through the automaton state $q_\phi \in \mathcal{Q}_\phi$.

Definition 3. *An FSA augmented MDP corresponding to a scTLTL formula* ϕ *(constructed from FSA* $\langle \mathbb{Q}_\phi, \Psi_\phi, q_0, p_\phi(\cdot|\cdot), \mathcal{F}_\phi\rangle$ *and MDP* $\langle S, A, p(\cdot|\cdot,\cdot), r(\cdot,\cdot,\cdot)\rangle$*) is defined as* $\mathcal{M}_\phi = \langle \tilde{S}, A, \tilde{p}(\cdot|\cdot,\cdot), \tilde{r}(\cdot,\cdot), \mathcal{F}_\phi\rangle$*, where* $\tilde{S} \subseteq S \times \mathbb{Q}_\phi$*,* $\tilde{p}(\tilde{s}'|\tilde{s}, a)$ *is the probability of transitioning to* $\tilde{s}'$ *given* $\tilde{s}$ *and* a*,*

$$\tilde{p}(\tilde{s}'|\tilde{s}, a) = p\big((s', q')|(s, q), a\big) = \begin{cases} p(s'|s, a) & p_\phi(q'|q, s) = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

p_ϕ *is defined in Equation* (3)*.* $\tilde{r} : \tilde{S} \times \tilde{S} \to \mathbb{R}$ *is the FSA augmented reward function, defined by*

$$\tilde{r}(\tilde{s}, \tilde{s}') = \rho(s', D^q_\phi), \tag{5}$$

where $D_\phi^q = \bigvee_{q' \in \Omega_q} \psi_{q,q'}$ represents the disjunction of all predicates guarding the outgoing transitions that originate from q (Ω_q is the set of automata states that are connected with q through outgoing edges).

Note that in Equation (5), $\tilde{r}$ is calculated from (s', q). It is a measure of the progress of satisfying D_ϕ^q by taking action a in state s (encapsulated by s').

Intuitively, the reward function in Equation (5) encourages the system to exit the current automaton state and move on to the next, and by doing so eventually reach the final state q_f which satisfies the TL specification (property of FSA).

D. Zeroing Control Barrier Functions (CBF) And Control Lyapunov Functions (CLF)

Define an affine control system as

$$\dot{s} = f(s) + g(s)a \tag{6}$$

where $f : S \to S$ and $g : A \to S$ are locally Lipschitz continuous, $s \in S \subseteq \mathbb{R}^n$ is the MDP state, $a \in A \subseteq \mathbb{R}^m$ is the control. Here we used the same notation for state and action as the MDP in Definition 1. As will become clear in the next section, we embed the control system as part of the MDP transition dynamics. Following [24] and [8], we provide the definition of the zeroing control barrier function and the control Lyapunov function.

Definition 4. *Given a set $\mathcal{C} = \{s \in S : h(s) \geq 0\}$ for a continuously differentiable function $h : S \to \mathbb{R}$, the function h is a zeroing control barrier function defined on set $\mathcal{D}$ with $\mathcal{C} \subseteq \mathcal{D} \subset \mathbb{R}$, if there exists an extended class $\mathcal{K}$ function α such that*

$$\sup_s \big(\frac{\partial h(s)}{\partial t} + \alpha h(s) \geq 0 \big). \tag{7}$$

Definition 5. *A continuously differentiable function $V : S \to \mathbb{R}$ is an exponentially stabilizing control Lyapunov function [8] if there exists positive constants $c_1, c_2, c_3 > 0$ such that*

$$\begin{aligned} &c_1 ||s||^2 \leq V(s) \leq c_2 ||s||^2 \\ &\inf_s (\frac{\partial V(s)}{\partial t} + c_3 V(s) \leq 0). \end{aligned} \tag{8}$$

Proposition 1. *A controller that meets the objectives specified by the CLF and renders the safe set $\mathcal{C}$ forward invariant (satisfies the condition in Equation (7)) can be found by solving the quadratic program (CLF-CBF-QP)*

$$\begin{aligned} a^\star(s) = &\arg\min_{a \in A, \delta} a^T a + K\delta \\ s.t. \ &\frac{\partial h(s)}{\partial s} f(s) + \frac{\partial h(s)}{\partial s} g(s) a + \alpha h(s) \geq 0 \\ &\frac{\partial V(s)}{\partial s} f(s) + \frac{\partial V(s)}{\partial s} g(s) a + c_3 V(s) \leq \delta, \end{aligned} \tag{9}$$

where δ is a relaxation variable that grows when there is a conflict between the CBF and CLF constraints. K is chosen to be a large positive constant ($\sim 1e^{10}$).

IV. PROBLEM FORMULATION AND APPROACH

Problem 1. *Given an MDP $\mathcal{M} = \langle S, A, p(\cdot|\cdot,\cdot), r(\cdot,\cdot,\cdot) \rangle$ where $S \subseteq \mathbb{R}^n$ is the state space; $A \subseteq \mathbb{R}^m$ is the action space; $p = \{p_k, p_u\}$ consists of known system dynamics in the form $p_k : \dot{s} = f(s) + g(s)a$ where $s \in S$, $a \in A$ and unknown environmental dynamics p_u; $r : S \times A \times S \to \mathbb{R}$ is the reward function, a safe set $\mathcal{C} = \{s \in S : h(s) \geq 0\}$ and a scTLTL formula ϕ over state predicates, find a policy $\pi_\phi^\star$ such that:*

$$\begin{aligned} \pi_\phi^\star = &\arg\max_{\pi_\phi} \mathbb{E}^{\pi_\phi} [\sum_{t=0}^{T} \gamma^t (\tilde{r} + wr)] \\ s.t. \ &\min_{t \in [0,T]} h(s_t) > 0, \end{aligned} \tag{10}$$

where $\tilde{r}$ is the FSA augmented MDP reward in Equation (5), w is a weighting parameter, T is the time horizon and $0 < \gamma \leq 1$ is the discount factor, h is the control barrier function in Definition 4. It is required that the system trajectory always stays within the safe set defined by h and ϕ is not violated at all times during learning.

Equation (10) requires $\pi_\phi^\star$ to optimize the expectation of a weighted return that combines satisfying ϕ (maximizing $\tilde{r}$) and maximizing the MDP reward r while ensuring safety during learning. Here we distinguish between not satisfying ϕ and violating ϕ. Violation means that a state is entered such that ϕ can never be satisfied afterwards.

To solve Problem 1, we first translate ϕ to its equivalent FSA and construct a FSA augmented MDP using Definition 3. The FSA is used in three ways - provide reward for the RL agent, provide goal states for the CLF to guide exploration and provide the safe set for the CBF. The actions from the RL agent are then integrated with the CLF-CBF-QP to solve for the final safe action that's transmitted to the system and environment. A diagram of our workflow is provided in Figure 1 .

V. TL GUIDED SAFE REINFORCEMENT LEARNING

In this section, we discuss in detail the components of Figure 1 . We will use a running example for explanation purposes. We define three circular regions a, b, c each of form $|s - s_i| < th$, $i \in \{a, b, c\}$ where s_i is the center coordinate of the region, th is the radius. A specification $\phi = (\Diamond a \vee \Diamond b) \wedge \Diamond c \wedge ((\neg a \wedge \neg b) \ \mathcal{U} \ c)$ entails that regions a *or* b *and* c are to be *eventually* visited *and* a *and* b should *not* be visited before c. The FSA for ϕ is shown in Figure 2 .

One drawback of the FSA definition (Definition 2) is that it does not explicitly handle spec violation. This problem can be solved by adding an additional set of trap states Q_{trap} such that if $q_{trap} \in Q_{trap}$, then no path exists on the FSA that connects q_{trap} to q_f (i.e. entering q_{trap} violates the spec). This can be seen in Figure 2 when b is visited before a.

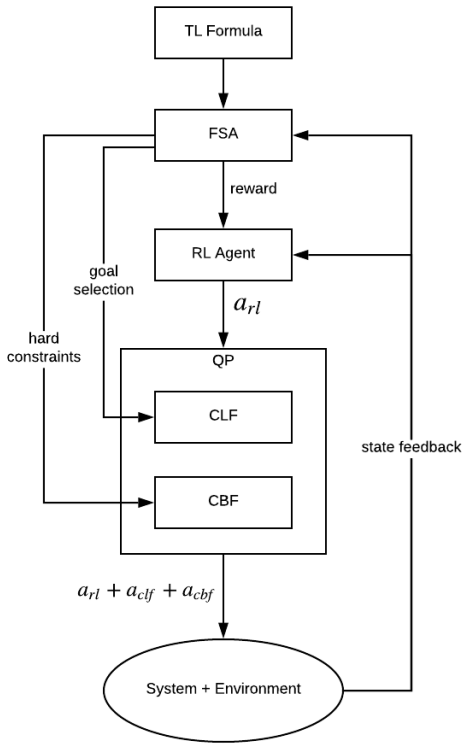

Fig. 1 : Flow diagram for TL guided safe reinforcement learning

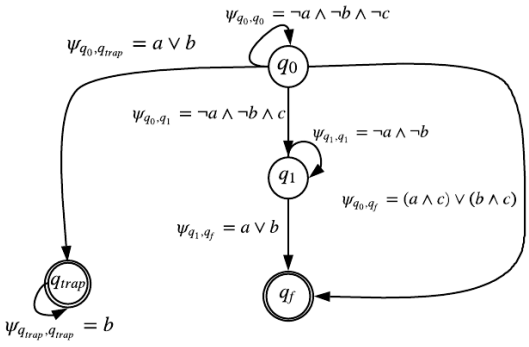

Fig. 2 : FSA for formala $(\Diamond a \vee \Diamond b) \wedge \Diamond c \wedge (\neg a \wedge \neg b \, \mathcal{U} \, c)$

A. FSA as RL Reward

Here we extend the reward definition in Equation (5) for the FSA augmented MDP to handle the trap state. The modified reward is as follows:

$$\tilde{r} = \tilde{r}(\tilde{s}, \tilde{s}') = \min\left(\rho(s', D_\phi^q), c_r \rho(s', \neg\psi_{q,q_{trap}})\right), \quad (11)$$

where $D_\phi^q = \bigvee_{q' \in \Omega_q, q' \neq q_{trap}} \psi_{q,q'}$. As an example, in Figure 2 $D_\phi^{q_0} = \psi_{q_0,q_1} \vee \psi_{q_0,q_f}$.

$\rho(s', D_\phi^q)$ in Equation (11) promotes exiting the current FSA state and enter the next non-trap state. $\rho(s', \neg\psi_{q,q_{trap}})$ penalizes entering the trap state. the min() function penalizes the more violating term with lower robustness. $c_r > 0$ is a weighting coefficient. It is important to note that $\tilde{r}$ in Equation (11) does not prevent the RL agent from violating ϕ during learning. In fact, the agent needs to explore q_{trap} in order to learn and hence the self-loop. During testing, the user can choose to terminate once the a trap state is entered

B. FSA For CLF Goal Selection

The intuition behind goal selection using the FSA is to find the outgoing edge (leading to a non-trap state) that is easiest for the agent to activate at its current state. Then select the MDP state that maximizes the robustness of the guarding predicate of that edge as the goal. This can be formalized by

$$\begin{aligned} s_g(q) &= \arg\max_{s \in S} \rho(s, \psi_{q,q'_{\rho max}}), \text{ where} \\ q'_{\rho_{max}}(s,q) &= \arg\max_{q' \in \Omega_q, q' \neq q_{trap}} \rho(s, \psi_{q,q'}), \end{aligned} \quad (12)$$

where Ω_q is the set of automata states that are connected with q through outgoing edges. Finding s_g using Equation (12) requires solving two optimization problems. The second one is relatively easy since there are only a finite number of outgoing edges to choose from. The first problem can be difficult if the state space is large. One useful heuristic is that if the predicates are defined as regions in the state space, the state with max robustness is the center of that region or at the center of intersection of multiple regions. For the example in Figure 2 , $s_g(s, q_1)$ would be the center of region a or b whichever is closer to state s.

After finding s_g, a simple CLF can be defined as

$$V_\phi(s,q) = (s - s_g)^T (s - s_g). \quad (13)$$

Other CLFs that meet the criteria in Equation (8) can also be used. For simplicity, the above CLF will be adopted in our case studies.

C. FSA for CBF Constraints

We exploit the fact that the guarding predicates ψ in an FSA are propositional logic formulas of inequality constraints over states. They can be naturally interpreted as safe sets and incorporated into the CBF. At state (s, q), if there is a trap state q_{trap} connected to q, we would like to avoid activating ψ_{q,q_trap}. That is we need to ensure that $\psi_{q,q_{trap}}$ is always false or $\neg\psi_{q,q_{trap}}$ is always true. We first restructure $\neg\psi_{q,q_{trap}}$ to its conjunctive normal form (CNF) i.e. $\neg\psi_{q,q_{trap}} = \bigwedge_i (\bigvee_j \psi_{ij})$ [25] - the conjunction of a set of disjunctive formulas. An example of a CNF is $\bigwedge_i (\bigvee_j \psi_{ij}) = (a) \wedge (b \vee \neg c)$ where $\phi_{00} = a$, $\phi_{10} = b$, $\phi_{11} = \neg c$.

We define the FSA-based safe set at state (s, q) with $\neg\psi_{q,q_{trap}} = \bigwedge_i (\bigvee_j \psi_{ij})$ as

$$\begin{aligned} &\mathcal{C}_\phi(s,q) = \{h_\phi^0(s,q), \ldots, h_\phi^i(s,q)\}. \text{ where} \\ &h_\phi^i(s,q) = \psi_{ij_{\rho_{min}}}, \; j_{\rho_{min}} = \arg\min_j \rho(s, \psi_{ij}) \end{aligned} \quad (14)$$

Here we assume each predicate ψ_{ij} is written in the form $f(s) > 0$. The goal of the CBF is to keep the agent from entering the trap state during learning

For the example in Figure 2 , at state (s, q_0), $\psi_{q_0,q_{trap}} = a \vee b$, $\neg\psi_{q_0,q_{trap}} = \neg a \wedge \neg b$. Therefore, $\mathcal{C}_\phi(s, q_0) = \{\neg a, \neg b\} = \{|s - s_a| > th, |s - s_b| > th\}$.

D. TL-CLF-CBF QP

We modify the known system dynamics p_k to

$$\begin{aligned} &f(s) + g(s)(a_{rl}(s) + a_{cbf} + a_{clf}) \\ &= \underbrace{f(s) + g(s)a_{rl}(s)}_{\hat{f}(s)} + g(xs)(a_{cbf} + a_{clf}), \end{aligned} \tag{15}$$

a_{rl} is the action provided by the RL agent. a_{cbf} and a_{clf} can be found by solving the following quadratic program

$$\begin{aligned} \hat{a}^\star(s, q) = \arg\min_{\hat{a},\delta}\ & \hat{a}^T\hat{a} + K\delta \\ s.t.\ & \frac{\partial h(s)}{\partial s}\hat{f}(s) + \frac{\partial h(s)}{\partial s}g(s)\hat{a} + \alpha h(s) \geq 0 \\ & \frac{\partial h^0_\phi(s,q)}{\partial s}\hat{f}(s) + \frac{\partial h^0_\phi(s,q)}{\partial s}g(s)\hat{a} + \alpha h^0_\phi(s,q) \geq 0 \\ & \vdots \\ & \frac{\partial h^n_\phi(s,q)}{\partial s}\hat{f}(s) + \frac{\partial h^n_\phi(s,q)}{\partial s}g(s)\hat{a} + \alpha h^n_\phi(s,q) \geq 0 \\ & \frac{\partial V_\phi(s,q)}{\partial s}\hat{f}(s) + \frac{\partial V(s,q)}{\partial s}g(s)\hat{a} + c_3V(s,q) \leq \delta \\ & a^{min}_{cbf} \leq a_{cbf} \leq a^{max}_{cbf} \\ & a^{min}_{clf} \leq a_{clf} \leq a^{max}_{clf} \end{aligned} \tag{16}$$

where $\hat{a} = [a_{cbf}, a_{clf}]$, $h(s)$ captures extra safety constraints defined by the user, $h^i_\phi \in \mathcal{C}_\phi$, $i \in \{1, ..., n\}$ are TL spec related constraints constructed with Equation (14), V_ϕ is constructed with Equation (13). A summary of our algorithm is provided in Algorithm 1.

This framework provides much flexibility in defining a task. It incorporates prior knowledge about the task domain in the form of temporal logic specifications as well as regular reward functions. The user can also specify hard and soft constraints (encapsulated in the CBF and CLF). The system also combines model-based planning and model-free learning so that knowledge about the system dynamics can be taken advantage of while unknown dynamics can also be handled.

Algorithm 1 TL Guided Safe RL

Inputs: scTLTL task specification ϕ, MDP $\mathcal{M}$, episode horizon T, RL agent $\mathcal{RL}$, parameterized policy π_θ, action bounds $b = [a^{max}_{cbf}, a^{min}_{cbf}, a^{max}_{clf}, a^{min}_{clf}]$, QP parameters c_r, K, α, c_3, reward weighting parameter w
Construct the FSA augmented MDP $\mathcal{M}_\phi$ ▷ using Definition 3
Initialize policy $\theta \leftarrow \theta_0$, empty episode buffer $\mathcal{B} \leftarrow []$
for i = 0 to number of training iterations **do**
  Reset environment
  Receive initial observation s_0
  for $t = 0$ to T **do**
    $a^t_{rl} = \pi_{\theta_i}(s_t)$
    Construct Lyapunov function V_ϕ and safe set $\mathcal{C}_\phi$ ▷ Equation (14), (13)
    $a^t_{cbf}, a^t_{clf} = QP(s_t, q_t, a^t_{rl}, \mathcal{C}_\phi, V_\phi)$ ▷ Equation (16)
    $s_{t+1} \leftarrow step(s_t, q_t, a_{rl} + a_{cbf} + a_{clf})$ ▷ step the environment to get next observation
    Construct reward $\tilde{r} + wr$ ▷ $\tilde{r}$ from Equation (11)
    $\mathcal{B}.append\big((s_t, q_t a_r l, s_{t+1}, \tilde{r} + wr)\big)$
  end for
  $\theta_i \leftarrow \mathcal{RL}(\theta_{i-1}, \mathcal{B})$ ▷ update the policy with chosen RL agent
end for
return optimal policy parameters $\theta^\star$

VI. CASE STUDY

A. Experiment Setup

Figure 3 shows our simulation environment. It consists of one agent with known dynamics, safety constraints in the form of two straight lines forming a channel that the agent has to stay within, three circular goal regions whose positions are kept fixed in a episode but are randomized between episodes, and two obstacles that move in the vicinity of the channel and whose dynamics are unknown.

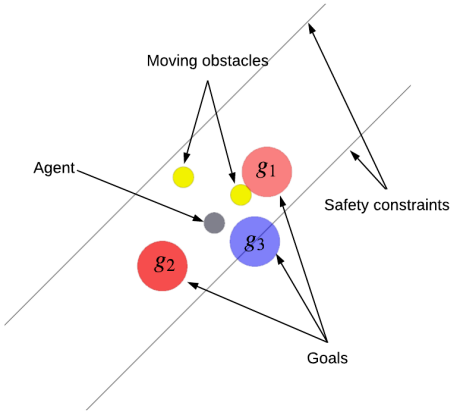

Fig. 3 : Simulation environment setup.

B. Agent Dynamics

The MDP state of the agent consists of its position and orientation $[x, y, \theta]$ as well as the position of all goals and moving obstacles (13 continuous dimensions). Combined with 1 discrete dimension for the FSA state (from construction of the FSA augmented MDP), the total state space is 14 dimensional. The agent's controls are the forward velocity a_v and steering speed a_θ.

The agent moves according to the unicycle model

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos(\theta) & 0\\ \sin(\theta) & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}a_v\\ a_\theta\end{bmatrix} \tag{17}$$

This underactuated system is uncontrollable. As in [26][27], we use a reference point different from the robot center with coordinates $[\epsilon, 0]$, $\epsilon > 0$ in the robot frame. It is shown that the relationship below holds

$$\begin{bmatrix}\dot{x}_\epsilon\\ \dot{y}_\epsilon\end{bmatrix} = \begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & \epsilon\end{bmatrix}\begin{bmatrix}a_v\\ a_\theta\end{bmatrix}, \tag{18}$$

where $[x_\epsilon, y_\epsilon]$ are the coordinates of the reference point (orientation is the same as the center). With $a_i = a_{i,rl} + a_{i,cbf} + a_{i,clf}$, $i \in \{v, \theta\}$, we can rewrite the above fully actuated dynamics in the form in Equation (15). Having obtained $[x_\epsilon, y_\epsilon]$, the robot's center coordinates can be calculated via a simple transformation.

C. Task Definition

The task is defined as

$$\begin{aligned}\phi = &(\Diamond\psi_{g_1} \vee \Diamond\psi_{g_2}) \wedge \Diamond\psi_{g_3} \wedge \\ &(\neg\psi_{g_3}\ \mathcal{U}\ (\psi_{g_1} \vee \psi_{g_2})) \wedge \\ &((\psi_{c_1} \wedge \psi_{c_2})\ \mathcal{U}\ \psi_{g_3})\end{aligned} \tag{19}$$

In English, ϕ entails that the agent is to *eventually* visit g_1 *or* g_2 *and eventually* g_3 *and* g_3 is not to be visited before g_1 *or* g_2 *and* constraints c_1 and c_2 are to be satisfied at all times before g_3 is visited. The predicates are defined as $\psi_{g_i} = |\boldsymbol{p}_a - \boldsymbol{p}_{g_i}| < \eta_g$, $i \in \{1, 2, 3\}$ where $\boldsymbol{p}_a = [x, y]$ is the coordinate of the agent, $\boldsymbol{p}_{g_i}$ is the center coordinate of goal region g_i. $\psi_{c_1} = -y + x + w > 0$, $\psi_{c_2} = y - x + w > 0$ define the safety constraints as channel of width $2w$ that the agent needs to navigate within.

We define an additional reward $r = \min_i(|\boldsymbol{p}_a - \boldsymbol{p}_{o_i}|)$, $i \in \{1, 2\}$ as the minimum distance between the agent and the moving obstacles. This reward is part of the objective in Equation (10).

D. Implementation Details

For the RL component, we use proximal policy optimization (PPO) [28] as the learning algorithm. The policy is represented by a feed-forward neural network with 3 hidden layers of 300, 200, 100 ReLU units respectively. The value function is of the same architecture. Each episode has horizon $T = 200$ steps and the positions of the goal regions are randomized between episodes (goals may initiate outside the safe channel). We collect a batch of 5 trajectories for each update iteration. During learning, an episode terminates only when the horizon is reached or the task is completed. The agent is allowed to travel outside the safety channel and collide with the moving obstacles during learning (to receive a penalty). However, this is not allowed during evaluation.

E. Results and Discussion

In this section, we study the effect of each component in our system in Figure 1 on the learning progress. Specifically, we train the system in four configurations - only RL, RL with CBF, RL with CLF and RL with CBF and CLF.

Figure 4 shows the learning curve in terms of the discounted return. We can see that all configurations are able to achieve similar end performance. The configurations with CLF exhibit faster rising time at the beginning of the learning process. This implies that the RL agent is able to find a useful policy faster with the help of CLF. However, it can also be observed that the learning curves with CLFs show higher variances and volatility. This is mainly due to the fact that the CLF is not aware of moving obstacles or safety constraints, its only job is to guide the agent to the nearest next goal that promotes progress on the FSA according to Equation (12). Therefore, at moments of near collision and safety violations, the RL agent will need to fight the actions provided by the CLFs and hence adding difficulty to the learning problem. This is most apparent with the configuration RL+CLF+CBF where the RL agent will need to learn to work in harmony with the CLF and CBF controls.

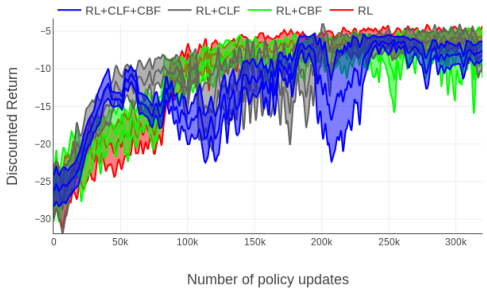

Fig. 4 : Discounted return distribution for different experiment configurations. The figure reports the mean and 1 standard deviation calculated from a batch of 5 trajectories at each update iteration.

Figure 5 shows the maximum violation of the safety constraints ψ_{c1} and ψ_{c2} (a value smaller than zero indicates violation). We can see that the CBF is able to prevent the agent from violating the safety constraints at all times during learning. For configurations without CBF, because a violation penalty is imposed on the agent (Equation (11)), it also gradually learns to minimize violation. Figure 5 (a) shows that the RL+CLF configuration exhibits a sudden decrease in performance (same time as in Figure 4). This occurs when the nearest next goal happens to be outside the safety region and the RL agent has not learned enough to counter the guiding actions of the CLF.

Figure 6 shows the minimum distance between the agent and the moving obstacles as a function of policy updates. As learning progresses, the agent learns to stay away from the obstacles. The agent is able to keep a further distance from the obstacles in configurations without CBF as this configuration allows the agent to travel outside the safe zone.

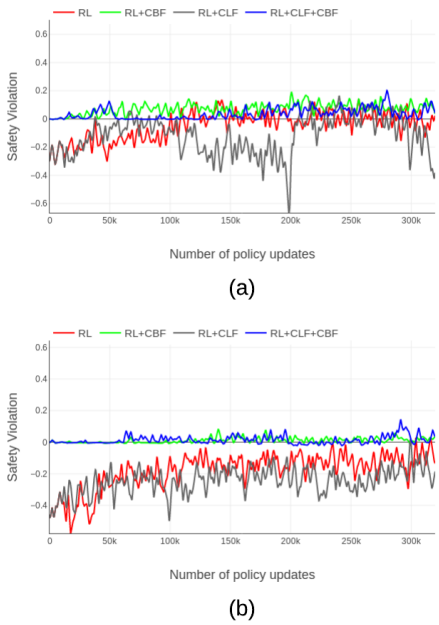

Fig. 5 : Minimum value of the safety constraints for (**a**) ψ_{c_1} and (**b**) ψ_{c_2} calculated from a batch of 5 trajectories at each update iteration. A value smaller than 0 indicates violation of the constraints.

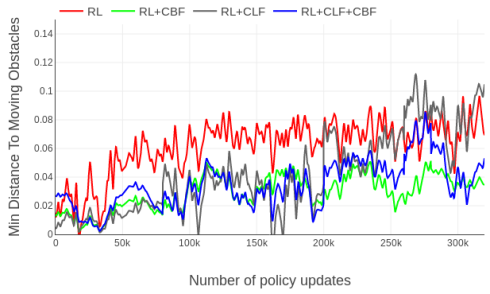

Fig. 6 : Minimum agent distance to moving obstacles.

Figure 7 shows sample trajectories of the agent in various configurations. The upper sub-figures (a) and (b) illustrate trajectories for pure RL and pure CBF. We can see that both agents behave similarly by going to g_2 (the nearer of g_1 and g_2) then g_3. The difference is that the RL agent chooses an obstacle free path and tries to trade off between accomplishing the task, avoiding obstacles and minimizing safety violation (controlled by the weights introduced in the previous section). Figure 7 (c) shows the agent trajectory with CLF and CBF. CLF still commands the agent to g_2, however, CBF stops the agent at the border of the safety zone and the agent is stuck. Collision with moving obstacles occurs in this case because nor the CLF or the CBF know the obstacle dynamics which is required for avoidance. Figure 7 (d) shows the agent trajectory with RL, CLF and CBF. The agent starts close to and tries to move towards g_2. However, it has learned that if it keeps trying to get to g_2 it will get stuck at the border and receive a low return. Therefore, near the border it chooses to instead move towards g_1 and eventually finishes the task. An agent trained with RL and CBF (not shown here) can potentially also obtain similar behavior. As discussed previously, adding CLF promotes exploration at the beginning of learning. A video can be accessed at https://youtu.be/lUsE3hGpLAk.

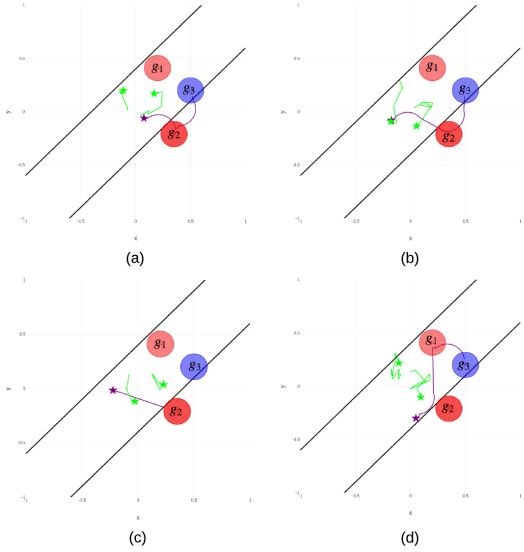

Fig. 7 : Sample trajectories for agents with configuration (**a**) RL (**b**) CLF (**c**) CLF with CBF (**d**) RL, CLF and CBF. Agent trajectory is in purple, moving obstacle trajectories are in green. The stars represent starting positions.

Figure 8 shows the evaluation success rate over 20 trials. Different from the training process, during evaluation an episode terminates in three circumstances - horizon is reached, task accomplished and the agent comes in collision with the moving obstacles (defined by a minimum threshold on relative distance). To ensure safety, CBF is always turned on during evaluation. The results show that the agents trained with CBF exhibits higher success rates. This is because agents trained without CBF sometimes rely on traveling outside the safe zone to avoid obstacles and get to goals. During evaluation, this is not an option and therefore these agents are more prone to getting stuck at the safe zone boundaries.

Lastly, we want to discuss some practical issues of our approach and a possible area of future work. The system presented in Figure 1 integrates multiple components together. Central to controlling how the components interact with each other is a set of hyperparameters. This includes c_r in Equation (11), w in Equation (10), the coefficients K, α, c_3 and the control bounds in Equation (16). We find that the learning process and the final performance of the

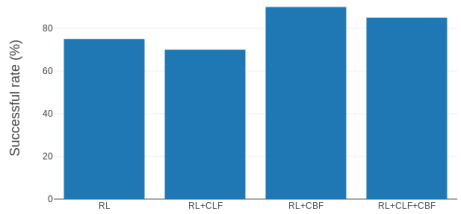

Fig. 8 : Success rates over 20 evaluation trials.

policy is noticeably influenced by these parameters. Taking the control bounds as an example, a good practice is to set a wide bound for the CBF actions as these actions will need to dominate at the boundary of the safety constraints. The bounds for the CLF actions should be set narrower than that of the RL actions. This is to ensure that the RL agent is able to overcome the CLF actions when it needs to. A better strategy is to shrink the CLF action bounds as learning progresses such that it can guide exploration at the beginning of training and diminishes its effect as the agent learns. This type of guidance scheduling is also adopted in [29] with demonstrations. An important future work is therefore to alleviate the burden of parameter tuning from the user.

VII. CONCLUSIONS

In this paper, we explore the combination of temporal logic, reinforcement learning, control Lyapunov functions and control barrier functions to form an expressive, safe and learnable control system. We propose techniques that allow an FSA to simultaneously provide rewards, objectives and safety constraints to each component in the system. We analyzed the effect of each component on the learning process and evaluate the resultant policy. In a simulation experiment with known agent dynamics and unknown environmental dynamics, we showed that our system is able to learn a policy that accomplishes the given task expressed as a logic formula while ensuring safety during learning.

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] J. Lehman, J. Clune et al., "The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities," CoRR, vol. abs/1803.03453, 2018.

[3] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. A. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," CoRR, vol. abs/1707.08817, 2017.

[4] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," CoRR, vol. abs/1812.07252, 2018.

[5] Y. Chebotar, A. Handa, D. Khayyat, M. Macklin, J. Issac, N. D. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," CoRR, vol. abs/1810.05687, 2018.

[6] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in AAAI, 2018.

[7] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," CoRR, vol. abs/1801.08757, 2018.

[8] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in 53rd IEEE Conference on Decision and Control. IEEE, 2014, pp. 6271–6278.

[9] X. Li, Y. Ma, and C. Belta, "Automata guided reinforcement learning with demonstrations," arXiv preprint arXiv:1809.06305, 2018.

[10] G. D. Giacomo, L. Iocchi, M. Favorito, and F. Patrizi, "Reinforcement learning for ltlf/ldlf goals," CoRR, vol. abs/1807.06333, 2018.

[11] A. Camacho, O. Chen, S. Sanner, and S. A. McIlraith, "Decision-making with non-markovian rewards: From ltl to automata-based reward shaping," in Proceedings of the Multi-disciplinary Conference on Reinforcement Learning and Decision Making (RLDM), 2017, pp. 279–283.

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

[13] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith, "Using reward machines for high-level task specification and decomposition in reinforcement learning," in International Conference on Machine Learning, 2018, pp. 2112–2121.

[14] 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.

[15] M. L. Littman, U. Topcu, J. Fu, C. L. Isbell, M. Wen, and J. MacGlashan, "Environment-independent task specifications via gltl," CoRR, vol. abs/1704.04341, 2017.

[16] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in ICML, 2017.

[17] M. Ohnishi, L. Wang, G. Notomista, and M. Egerstedt, "Barrier-certified adaptive reinforcement learning with applications to brushbot navigation," 2018.

[18] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in AAAI, 2019.

[19] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," Journal of Machine Learning Research, vol. 16, pp. 1437–1480, 2015.

[20] 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," CoRR, vol. abs/1509.02971, 2015.

[21] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," arXiv preprint arXiv:1612.03471, 2016.

[22] C. Vasile, Github repository, 2017.

[23] K. Y. Rozier, "Explicit or symbolic translation of linear temporal logic to automata," Ph.D. dissertation, Rice University, 2013.

[24] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," IFAC-PapersOnLine, vol. 48, no. 27, pp. 54–61, 2015.

[25] P. Jackson and D. Sheridan, "Clause form conversions for boolean circuits," in International Conference on Theory and Applications of Satisfiability Testing. Springer, 2004, pp. 183–198.

[26] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot motion planning and control in polygonal environments," IEEE Transactions on Robotics, vol. 21, no. 5, pp. 864–874, 2005.

[27] J. P. Desai, J. Ostrowski, and V. Kumar, "Controlling formations of multiple mobile robots," in Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146), vol. 4. IEEE, 1998, pp. 2864–2869.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017.

[29] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," CoRR, vol. abs/1709.10087, 2018.