Safe Reinforcement Learning with Probabilistic Guarantees Satisfying Temporal Logic Specifications in Continuous Action Spaces

Hanna Krasowski, Prithvi Akella, Aaron D. Ames, and Matthias Althoff

***Abstract*— Vanilla Reinforcement Learning (RL) can efficiently solve complex tasks but does not provide any guarantees on system behavior. To bridge this gap, we propose a three-step safe RL procedure for continuous action spaces that provides probabilistic guarantees with respect to temporal logic specifications. First, our approach probabilistically verifies a candidate controller with respect to a temporal logic specification while randomizing the control inputs to the system within a bounded set. Second, we improve the performance of this probabilistically verified controller by adding an RL agent that optimizes the verified controller for performance in the same bounded set around the control input. Third, we verify probabilistic safety guarantees with respect to temporal logic specifications for the learned agent. Our approach is efficiently implementable for continuous action and state spaces. The separation of safety verification and performance improvement into two distinct steps realizes both explicit probabilistic safety guarantees and a straightforward RL setup that focuses on performance. We evaluate our approach on an evasion task where a robot has to reach a goal while evading a dynamic obstacle with a specific maneuver. Our results show that our safe RL approach leads to efficient learning while maintaining its probabilistic safety specification.**

I. Introduction

Reinforcement Learning (RL) has the potential to solve intricate tasks by learning complex policies efficiently. However, vanilla RL cannot provide (probabilistic) safety guarantees, which is essential for real-world applications. Formal methods can eliminate this problem when integrated into the learning process. The most prominent formal methods approaches achieving safety guarantees for RL with continuous action spaces are control-theoretic methods such as model predictive control [1], [2], control barrier functions [3], or reachability analysis [4]–[6]. However, all these methods only handle reach-avoid safety specifications since they either determine unsafe state sets and prohibit the RL agent from entering them or determine safe state sets and force the RL agent to remain within those. Other methods are needed whenever the safety specification is more complex and cannot be seamlessly translated into a reach-avoid problem.

One way of expressing more complex safety specifications is via temporal logic. Indeed, there has been significant work that combines RL with logical specifications for discrete action spaces [7]–[9]. For example, Alshiekh et al. [7] filter all unsafe actions proposed by the RL agent with a safety shield synthesized from linear temporal logic specifications. Hasanbeig et al. [9] include the temporal logic specifications in the RL process through a pessimistic and an optimistic learner where the pessimistic learner limits the exploration to low-risk actions. Still, applying approaches for discrete action spaces to real-world systems with continuous control inputs requires a low-level controller that converts the discrete actions to continuous inputs.

Other RL approaches realize continuous actions and guide the agent by a temporal logic specification that includes safety and performance objectives [10]–[12], *i.e.*, the temporal logic specification describes the entire task. For example, Cai et al. [12] transform linear temporal logic into a Büchi automaton, which is integrated into RL and yields probabilistic guarantees. Although specifying the task via temporal logic for RL is promising, safety and performance objectives included in the task are potentially not aligned. Possible solutions are to provide feedback to the user whenever the temporal logic specification becomes infeasible [11] or to trade off between safety and performance [12]. The first solution is usually not practical for autonomous real-world systems, and the second one does not provide an explicit probabilistic guarantee for the safety specification, which might be required. Instead, our approach separates safety and performance objectives such that we obtain probabilistic safety guarantees while the feasibility is ensured by utilizing a probabilistically verified safe controller.

To provide probabilistic safety guarantees, we leverage existing work in the probabilistic verification literature taking a scenario approach to risk-aware probabilistic verification [13], [14]. Here, the standard approach as described in [15] is to pose verification as an optimization problem minimizing a quantifiable satisfaction measure provided by either a temporal logic specification or another method.

Contribution: We propose a three-step safe RL approach that improves the performance of a probabilistically verified black-box controller and results in probabilistic safety guarantees for the learned agent. Our key idea is to separate safety and performance objectives in distinct steps (see Fig. 1 with probabilistic verification steps for safety and RL for performance), which leads to efficient RL while providing

The authors gratefully acknowledge the partial financial support of this work by the research training group ConVeY funded by the German Research Foundation under grant GRK 2428, by the project TRAITS funded by the German Federal Ministry of Education and Research, and by an IFI scholarship funded by the DAAD. Prithvi Akella was supported the Air Force Office of Scientific Research, grant FA9550-19-1-0302, and the National Science Foundation, grant 1932091.

H. Krasowski and M. Althoff are with the Technical University of Munich, Munich, Germany {hanna.krasowski, althoff}@tum.de

H. Krasowski, P. Akella and A. D. Ames are with the California Institute of Technology, Pasadena, USA {pakella, ames}@caltech.edu

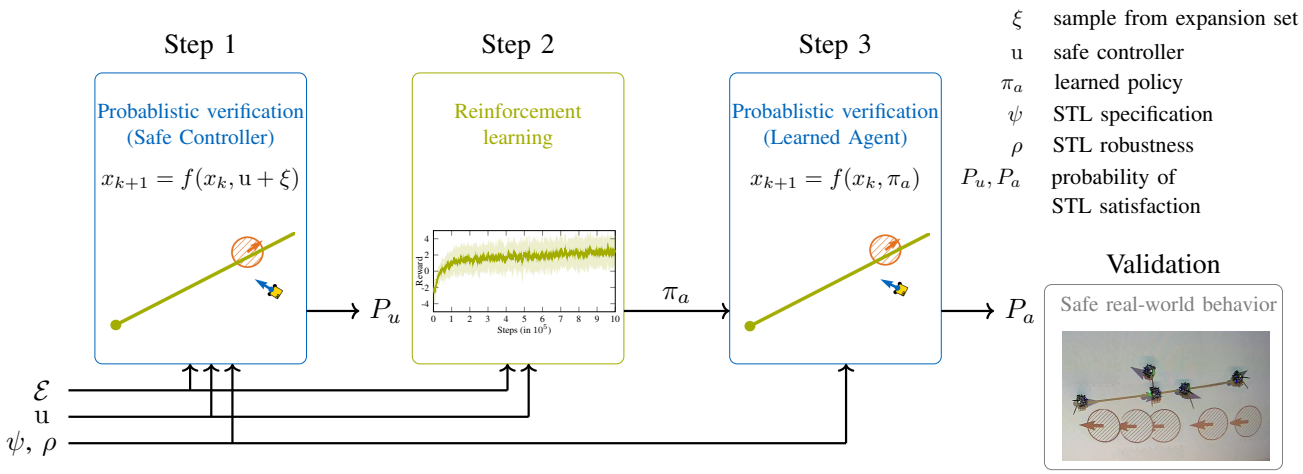


Fig. 1. Safe RL process for STL safety specification ψ with robustness measure ρ using a safe controller u and a system model $x_{k+1} = f(x_k, \cdot)$ where $\cdot$ can be replaced by any controller.

explicit safety guarantees. In contrast to existing methods, our approach is suited for complex real-world systems since it is tailored to continuous action spaces, can probabilistically verify arbitrary Signal Temporal Logic (STL) specifications, and does not require a system model to predict the safety of future actions. We validate our approach on an evasion task in a simulated dynamic environment and show that it can be translated to real-world systems as demonstrated through experiential results on the Robotarium [16].

Structure: The remainder of this paper is organized as follows. First, we introduce preliminary concepts in Sec. II. Second, we present our safe RL approach in Sec. III. Then, we explain the details of our safe evasion task and its experimental validation in Sec. IV. Finally, we discuss our approach in Sec. V and conclude in Sec. VI.

II. Preliminaries

Signal Temporal Logic STL is a language by which rich, time-varying system behavior can be expressed succinctly and concisely. STL is based on predicates τ which are Boolean-valued functions taking a truth value for each state $x \in \mathcal{X}$. Predicates τ and specifications ψ are defined in Backus-Naur notation [17, Section 2.1] with respect to predicate functions h_τ that define subsets of a state space $\mathcal{X}$ where τ evaluates to True:

$$\tau(x) = \text{True} \iff h_\tau(x) \geq 0,\ h_\tau : \mathcal{X} \to \mathbb{R}, \qquad (1)$$
$$\psi \triangleq \tau \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid \psi_1 \,\mathrm{U}_{[a,b]}\, \psi_2,$$

where, $\psi \in \mathbb{S}$, and $a, b \in \mathbb{R}_{\geq 0} \cup \{\infty\},\ b \geq a$. Here, $\mathbb{S}$ is the set of all STL specifications which are evaluated over signals $s : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, and the space of all signals $\mathcal{S}^{\mathrm{n}} = \{s \mid s : \mathbb{R}_{\geq 0} \to \mathbb{R}^n\}$. Finally, we denote that a signal s satisfies ψ at time t via $(s, t) \models \psi$. Furthermore, every STL specification ψ has a *robustness measure* ρ [18]:

Definition 1. *A function $\rho : \mathcal{S}^{\mathrm{n}} \times \mathbb{R}_+ \to \mathbb{R}$ is a robustness measure for an STL specification ψ if it satisfies: $\rho(s, t) \geq 0 \iff (s, t) \models \psi$.*

Example 1. *Let $\psi = \neg(\text{True}\,\mathrm{U}_{[0,2]}\,|s(t)| > 2)$, then any real-valued signal $s : \mathbb{R}_{\geq 0} \to \mathbb{R}$ satisfies ψ at time t, i.e. $(s, t) \models \psi$ if $\forall\, t' \in [t, t{+}2],\ |s(t')| \leq 2$. A possible robustness measure is $\rho(s, t) = \min_{t' \in [t, t+2]}\ 2 - |s(t')|$.*

Note that while defining a robustness measure as per Definition 1 aligns with prior works [19], [20] and our predicate definition in (1), it is not the only way of defining such a measure, *e.g.* see Definition 3 in [21] or Section 2.3 in [22].

Probabilistic Controller Verification As expressed in Section 3 in [15], STL provides a natural way of phrasing black-box controller verification as an optimization problem over a space of parameters $p \in \mathcal{P}$ affecting signal generation. More specifically, let $\mathcal{P}$ be a space of parameters denoting different environmental states in which we expect our closed-loop system to operate. For example, for warehouse robotics, these environmental states could be package and drop-off locations, the floor plan, *etc*. Additionally, for any specific environment parameter p, there may exist disturbances affecting system behavior. Thus, we expect the closed-loop trajectory ϕ_p, which is realized by our system for a specific environment parameter p, to be a sample of a p-parameterized random variable Φ_p with corresponding distribution π_p. To formulate the theorem used in this work, let us introduce $\mathrm{U}[\cdot]$ as uniform distribution and $\mathbb{P}_\triangle$ denoting a probability where $\triangle$ indicates the underlying distribution.

Theorem 1. *(Adapted from [13, Thm. 7]) Let $\mathcal{P}$ be a space that admits a uniform distribution and let $\mathcal{D} = \{r_i = \rho(\phi_{p_i})\}_{i=1}^N$ be a set of N closed-loop system robustnesses r_i, evaluating the robustness of one closed-loop trajectory sample ϕ_{p_i} per i.i.d. sample p_i drawn from the uniform distribution over $\mathcal{P}$. Furthermore, define $\rho_N^* = \min\{r_i \in \mathcal{D}\}$. For any $\epsilon \in [0, 1]$, the probability that ρ_N^* underperforms the $1 - \epsilon$-th quartile of possible robustness values is bounded below by $1 - (1 - \epsilon)^N$, i.e. with $\mu \triangleq \mathrm{U}[\mathcal{P}] \times \pi_p$,*

$$\mathbb{P}_\mu^N \left[\mathbb{P}_\mu[\rho(\phi_p) \geq \rho_N^*] \geq 1 - \epsilon\right] \geq 1 - (1 - \epsilon)^N. \qquad (2)$$

For a more detailed derivation of this theorem and its

implications on dimensional scaling, we refer the interested reader to [13].

Overarching Problem Statement: We assume that we have a safety specification ψ expressed in STL and an associated robustness measure ρ as per Definition 1. We also assume that a model and black-box controller u are available:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k),\ x_k, x_{k+1} \in \mathcal{X} \subseteq \mathbb{R}^n, \\ u_k &\in \mathcal{U} \subseteq \mathbb{R}^m,\ \mathrm{u} : \mathcal{X} \to \mathcal{U}, \end{aligned} \tag{3}$$

where $\mathrm{u}(x_k) = u_k$ and the subscripts denote the time step. Additionally, we assume that the distribution of system behavior π_p is time-invariant. Note that we do not need to explicitly know the model f but can also employ a black-box simulation environment. The problem is to ensure probabilistic safety guarantees specified via STL for the given system while using RL for improving the performance.

III. Safe RL Process

Step One: Our safe RL concept consists of three steps as shown in Fig. 1. Our first step is to follow the probabilistic verification procedure outlined in Sec. II to verify whether the controller u realizes safe behavior when adding the uniformly sampled disturbance ξ:

$$x_{k+1} = f(x_k, \mathrm{u}(x_k) + \xi), \quad \xi \sim \mathrm{U}[\mathcal{E}],\ \mathcal{E} \subseteq \mathbb{R}^m. \tag{4}$$

Here, $\mathrm{U}[\mathcal{E}]$ corresponds to the uniform distribution over the set $\mathcal{E}$, which we assume to be fixed and independent of the system state $x \in \mathcal{X}$. Per Theorem 1, we can determine the following probabilistic lower bound on the robustness measure value achievable by the closed-loop system (4).

Corollary 1. *Let the system dynamics be as per* (4)*, the safety specification ψ have robustness measure ρ as per Definition 1, and $\mathcal{D} = \{r_i = \rho(\phi_{x_0^i})\}_{i=1}^N$ be the robustnesses of N trajectories $\phi_{x_0^i}$ where the initial conditions x_0^i were uniformly sampled over $\mathcal{X}$. Define $\rho_N^* = \min\{r_i \in \mathcal{D}\}$, then for some $\epsilon \in [0,1]$, ρ_N^* underperforms the $1-\epsilon$-th quartile robustnesses achievable by the stochastic closed-loop system in* (4) *with minimum confidence $1-(1-\epsilon)^N$, i.e. with $\mu = \mathrm{U}[\mathcal{X}] \times \mathrm{U}[\mathcal{E}] \times \mathrm{U}[\mathcal{E}] \times \ldots$,*

$$\mathbb{P}_\mu^N \left[\mathbb{P}_\mu \left[\rho(\phi_{x_0}) \geq \rho_N^* \right] \geq 1 - \epsilon \right] \geq 1 - (1-\epsilon)^N.$$

Proof: This is an application of Theorem 1. ∎

Following Corollary 1, we can identify the robustness set $\mathcal{D}$ by sampling N trajectories of the closed-loop system with controller u under bounded input disturbance from $\mathcal{E}$. Given the robustness set $\mathcal{D}$ and a specified ϵ, we can evaluate the probabilities in Corollary 1 and obtain ρ_N^*. We abbreviate this probabilistic verification process with the function $\mathrm{probv}(f, \mathrm{u}, \mathcal{E}, N, \epsilon)$. The first step concludes once we verified that a candidate controller u achieves safe behavior with a high probability, *i.e.* $\rho_N^* \geq 0$ with $1 - \epsilon \approx 1$.

Remark on Maximizing $\mathcal{E}$: The second step in our safe RL process will be to learn a policy that chooses disturbances within the expansion set $\mathcal{E}$. As such, maximizing the size of this set has a direct impact on the ability of our procedure to learn a performant policy. To that end, we propose a possible algorithm to expand $\mathcal{E}$ in Alg. 1.

Algorithm 1 findExpansionSet()

Input: executable system f, controller u, initial expansion interval set $\mathcal{E}_{init}$, vector of fractions to increase set $\Delta\mathbf{f} \in \mathbb{R}^m$, number of samples N, quartile parameter ϵ

Output: Final expansion set $\mathcal{E}$

1: $i = 1$
2: $\rho_N^* = \mathrm{probv}(f, \mathrm{u}, \mathcal{E}_{init}, N, \epsilon)$
3: **while** $\rho_N^* \geq 0$ **do**
4: $\quad \mathcal{E} = (\mathbf{1}_m + (i-1)\,\Delta\mathbf{f})\,\mathcal{E}_{init}$
5: $\quad \mathcal{E}_{temp} = (\mathbf{1}_m + i\,\Delta\mathbf{f})\,\mathcal{E}_{init}$
6: $\quad \rho_N^* = \mathrm{probv}(f, \mathrm{u}, \mathcal{E}_{temp}, N, \epsilon)$
7: $\quad i \leftarrow i + 1$
8: **end while**
9: **if** $i \neq 1$ **then**
10: $\quad$ **return** $\mathcal{E}$
11: **else**
12: $\quad$ **return** Reduce $\mathcal{E}_{init}$ as too large to verify
13: **end if**

In more detail, in line 2 of Alg. 1, the initial expansion set is verified. If this initial set is not verifiable, a smaller initial set must be provided. Otherwise, the expansion set $\mathcal{E}_{temp}$, increased iteratively through $\Delta\mathbf{f}$ (line 5), is subsequently verified (line 6). Once the verification is not successful anymore, the algorithm terminates and returns the largest verified expansion set $\mathcal{E}$ (line 10). Note that we use an interval for $\mathcal{E}$ for simplicity. However, other set representation such as zonotopes would be possible. To identify a more expressive expansion set, conformance checking [23] could be used.

Step Two: The second step in our safe RL process is to learn a controller that improves for performance of the safe controller u we verified in the prior step. To constrain the learning based on the safe controller, we define a state-dependent action space $\mathcal{A}(x)$ around the safe control input $\mathrm{u}(x)$ inspired by continuous action masking [24]:

$$\mathcal{A}(x) = \mathrm{u}(x) \oplus \mathcal{E}, \tag{5}$$

where $\oplus$ denotes the Minkowski sum. Based on Corollary 1, we can compute the probabilistic guarantee (usually ≈ 1) for the closed-loop system when perturbing the safe input $\mathrm{u}(x)$ with uniformly sampled noise within the expansion set $\mathcal{E}$. Therefore, if we continuously choose actions $a \in \mathcal{A}(x)$, our learned agent will with high probability yield safe behavior that also fulfills the probabilistic safety specification. Consequently, the agents can focus on optimizing for performance when learning within $\mathcal{E}$ around the safe controller. Thus, our three-step process delineates the safety (step one and three) and performance aspects (step two). This simplifies the reward and observation definitions as we only need to consider performance. Fig. 2 depicts this learning process and shows how the RL agent can effect a system trajectory in the state space by changing the control input within $\mathcal{A}(x)$.

Step Three: The third step is to verify the learned agent with

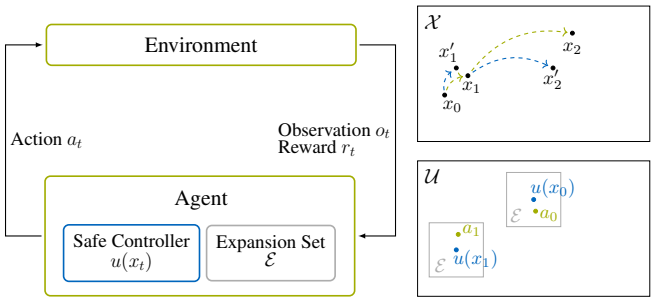


Fig. 2. RL within expansion set $\mathcal{E}$ around the safe controller $u(x)$ with trajectories for state space $\mathcal{X}$ and input space $\mathcal{U}$. The states x are reached by the RL actions, and x' are states that would be reached by the safe controller.

a deterministic policy $\pi_a : \mathcal{X} \to \mathcal{U}$, to ensure the preservation of safety after optimizing for performance through learning. This is necessary since the learned agent (obtained by step two) will be different from the probabilistically verified safe controller with disturbance uniformly sampled from $\mathcal{E}$ (verified in step one). Thus, the probabilistic verification result is likely to hold but still needs to be verified for the learned agent. This verification, however, amounts to one more implementation of Theorem 1:

Corollary 2. *Let the system dynamics be as per* (3) *with a learned RL agent $\pi_a : \mathcal{X} \to \mathcal{U}$ as controller, let the system safety specification ψ have a robustness measure ρ as per Definition 1, and let $\mathcal{D}$ and ρ_N^* be as in Corollary 1. Then for some $\epsilon \in [0,1]$, ρ_N^* underperforms the $1-\epsilon$-th quartile of robustnesses achievable by the learned system with minimum confidence $1-(1-\epsilon)^N$.*

$$\mathbb{P}^N_{\mathrm{U}[\mathcal{X}]}\left[\mathbb{P}_{\mathrm{U}[\mathcal{X}]}\left[\rho(\phi_{x_0}) \geq \rho_N^*\right] \geq 1-\epsilon\right] \geq 1-(1-\epsilon)^N.$$

Proof: This is an application of Theorem 1. ∎

IV. Safe Evasion Task with Experimental Validation

To make our high-level approach more tractable, we define a problem with a safety specification that is more complex than a simple reach-avoid specification. Specifically, the mobile robot's task is to follow an optimal path to the goal while complying with a temporal logic safety specification – whenever a collision is possible with the dynamic obstacle of the environment within the next few time steps, the mobile robot has to evade in a specified manner. Relevant examples for real-world applications where only a specific evasion is safe are: autonomous vehicles that have to overtake another traffic participant in a specific lane [25] or autonomous vessels that have to perform specific collision avoidance maneuvers in order to be predictable for other ships [26]. We first describe the safety specification and RL problem. Our experiments are conducted on the Robotarium [16] and its simulation.

A. Safety Specification

The state of the robot is $r = [x_r, y_r, \theta_r, v_r] \in \mathbb{R}^4$ where x_r and y_r describe the position of the robot, θ_r is its orientation, and v_r is its velocity aligned with its orientation. There is always one dynamic obstacle present and it is described by the state $o = [x_o, y_o, \theta_o, v_o] \in \mathbb{R}^4$. The time step is Δt. We assume a unicycle model for the robot for which control inputs u are velocity and turning rate. Note that for the Robotarium, we provide the same control inputs and their robot controller generates the corresponding actuator signals.

Our safety specification is as follows: When the projected positions of the robot and any obstacle are closer than $0.4\,\mathrm{m}$ for any time steps within $1\,\mathrm{s}$, then the robot should evade in a specific manner that depends on the relative positions and orientations of the obstacle and agent. To formalize the specification with STL, we first need to define a few predicates and functions.

We can convert heading angles to unit vectors via $\mathrm{a2v}(\phi) = [\cos(\phi), \sin(\phi)]$. The rotation matrix for an angle α is

$$R_\alpha = \begin{bmatrix} \cos(-\alpha) & \sin(-\alpha) \\ -\sin(-\alpha) & \cos(-\alpha) \end{bmatrix}.$$

The function $\mathrm{sgn}(x)$ returns -1 if $x < 0$, 1 if $x > 0$ and 0 otherwise. The minimum distance function $\mathrm{MD}(r, o, \Delta t)$ is defined as:

$$\mathrm{MD}(r,o,\Delta t) = \min_{t \in \{0, \Delta t, \ldots, 1\,\mathrm{s}\}} \Bigg(\Bigg\| \left(\begin{bmatrix} x_r \\ y_r \end{bmatrix} + \mathrm{a2v}(\theta_r) v_r t \right) - \left(\begin{bmatrix} x_o \\ y_o \end{bmatrix} + \mathrm{a2v}(\theta_o) v_o t \right) \Bigg\|_2 \Bigg),$$

where the time step size is $\Delta t = 0.033\,\mathrm{s}$. The minimum distance prediction assumes that the robot and the obstacle will move in their current directions with their current speeds, which is often a reasonable prediction. The predicate $\mathrm{IH}(r, o)$ checks if the obstacle is in the halfspace in front of the vehicle, *i.e.* it evaluates to true iff $[x_o, y_o] \cdot \mathrm{a2v}(\theta_r) - b_r \geq 0$ where b_r is the offset. Safe evasion is specified by the predicate:

$$\mathrm{EV}(\dot\theta_r, \Delta\theta, sign) = \begin{cases} \text{True}, & \text{iff } (|\dot\theta_r| \leq 1.5\,\mathrm{rad\,s^{-1}} \\ & \wedge \mathrm{sgn}(\dot\theta_r) = sign) \vee \\ & ((\Delta\theta \geq 0\,\mathrm{rad} \\ & \vee |\Delta\theta| \leq 0.01\,\mathrm{rad}) \\ & \wedge 0.01\,\mathrm{rad\,s^{-1}} \geq |\dot\theta_r|) \\ \text{False}, & \text{otherwise,} \end{cases}$$

where $\Delta\theta \in [-\pi, \pi]$ is the orientation difference to the orientation perpendicular to the direction of the straight path between the initial state and goal in the direction of turning; $sign$ is -1 if the robot should turn to the left and 1 in case the robot should turn to the right. See Fig. 3 for a depiction of the case-by-case evasion situations. Note that for our task specification with one non-reactive dynamic obstacle, the four cases are exhaustive for identifying the direction of evasion. To give an intuition, a safe evasion maneuver is that the robot turns until its orientation is perpendicular to the straight path between the initial state and goal. If this orientation is reached, the robot is no longer required to turn[1] and continues

[1]No turning would be an impossible requirement due to the stochasticity needed for the input space, which includes the turning rate.

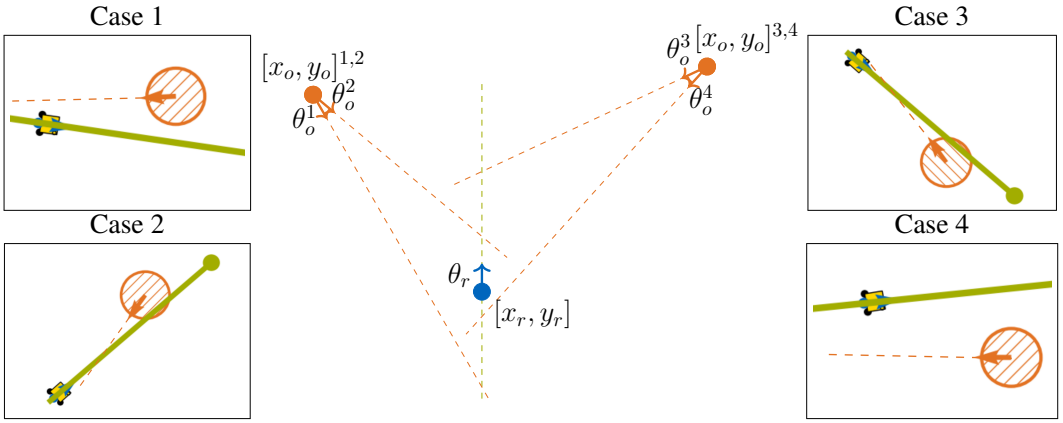


Fig. 3. Visualization of the four different relative orientations and position cases. The obstacle is depicted in orange and the robot in blue, and the case is indicated by the superscript. For cases 1 and 3, the robot should turn right (i.e., $sign = 1$) and for the cases 2 and 4, the robot should turn left (i.e., $sign = -1$).

driving away from the direct path between the initial state and goal to evade further.

With these predicates and functions, the STL formula is:

$$G\Big((\mathrm{IH}(r, o) \wedge \mathrm{MD}(r, o, \Delta t) \leq 0.4\,\mathrm{m}) \implies \mathrm{EV}(\dot{\theta}_r, \Delta\theta, sign)\Big). \tag{6}$$

The robustness measure for this STL formula is:

$$\rho = \begin{cases} -1, & \text{iff eq. (6)} = False \\ \sum_{k=1}^{K} \mathrm{prfm}(r_K, r_0, goal, K), & \text{otherwise,} \end{cases}$$

where K is the length of the trajectory, $goal$ is the position of the goal, r_K is the position of the robot at the end of the trajectory, r_0 is the initial position of the robot, and the performance objective is defined by

$$\mathrm{prfm}(r_K, r_0, goal, K) = \max\left(\left(1 - \frac{\|r_K - goal\|_2}{\|r_0 - goal\|_2}\right), 0\right) + \left(1 - \frac{K}{K_{max}}\right),$$

with the maximal length of a trajectory $K_{max} = 300$. Note that although the objective of this safety specification is collision avoidance, which can also be achieved with other formal methods such as control barrier functions, these methods cannot easily be used to guarantee a specific avoidance behaviour as defined in the predicate EV. Additionally, note that this particular safety specification can also be expressed with Linear Temporal Logic (LTL). However, as STL is more expressive than LTL, STL can be utilized to express more complex specifications, *e.g.* for marine traffic rules [26].

B. Reinforcement Learning Problem

We obtain the action space as described in (5). The reward function R is the difference between the achieved state with the RL control input and only with the safe control input

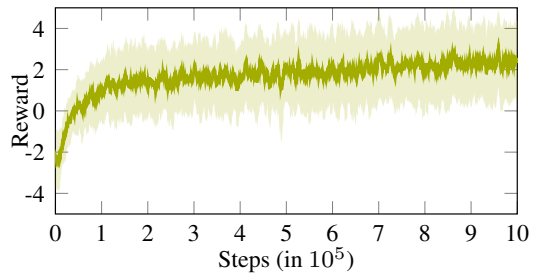
$$\mathrm{R} = r_{\mathrm{diff}} \cdot (\|goal - [x_{sc}, y_{sc}]\|_2 - \|goal - [x_r, y_r]\|_2),$$

where $r_{\mathrm{diff}} \in \mathbb{R}_+$ scales the reward, and $[x_{sc}, y_{sc}]$ is the agent's position if the input from the safe controller would be used in the previous state, *i.e.* $a_{t-1} = u_{t-1}$. Our agent can observe its relative position to the goal, the relative position to the closest point on the optimal path, the orientation difference to the optimal orientation calculated from the initial position to the goal, and the relative position to the obstacle. Note that this observation differs from the state.

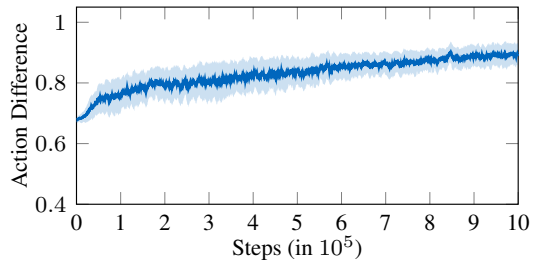
We used Proximal Policy Optimization (PPO) [27] as our RL algorithm and scaled the reward R with the factor r_{diff} such that the reward is approximately between -5 and 5 per episode. We identified the best hyperparameters for PPO with a small search of eight different configurations and three random seeds. However, all tested hyperparameters showed a converging behavior. The hyperparameters that differ from the default PPO configuration of stable-baselines3 [28] are the network architecture (two-layer multi-layer perceptron with 128 neurons in each layer), the value function coefficient (0.05), and the entropy coefficient (0.01). We train the agent for one million training steps.

Verification of Safe Controller (Step 1): We want to verify whether our safe controller with uniformly sampled perturbations from the expansion set $\mathcal{E}$ achieves safe behavior with 95% probability ($\epsilon = 0.05$), *i.e.* has a probabilistic cutoff $\rho_N^* > 0$ as per Theorem 1 with $N = 50$ samples. We apply Alg. 1 with $\mathcal{E}_{init} = [-0.0002, 0.0002]\mathrm{m\,s^{-1}} \times [-0.005, 0.005]\mathrm{rad\,s^{-1}}$ and $\Delta\mathbf{f} = [10, 1.0]$ and obtain $\mathcal{E} = [-0.002, 0.002]\mathrm{m\,s^{-1}} \times [-0.01, 0.01]\mathrm{rad\,s^{-1}}$. Verifying the safe controller on the unicycle model with $N = 50$ samples, yielded a probabilistic cutoff $\rho_{50}^* = 0.276$ (see Fig. 5). This indicates that our chosen safe controller successfully exhibits safe behavior with 95% probability, and we are 92.3% confident in this statement — probabilities were calculated via substitution in (2).

Reinforcement Learning (Step 2): We expect that if we define our action space as in (5), the previous verification result will most likely hold for the learned agent. Fig. 4 depicts the episode reward and action difference to the safe control input over the training steps. A reward above zero indicates that the agent performs better with respect to goal reaching than the safe controller. This is the case after approximately



(a)



(b)

Fig. 4. Training curves for (a) reward function and (b) Euclidean norm difference between the learned agent's action and the safe control input scaled between 0 and 1. The action difference of a step is one if the agent selects an action on the border of the expansion set and zero if the agent does not alter the safe control input. The curves depict the mean and standard deviation for five different random seeds with the same hyperparameters.

50000 training steps. Additionally, the difference between the safe control input and the agent's action increases over the training towards the agent mainly selecting actions close to its action space boundary.

Verification of Learned Agent (Step 3): Following the same verification procedure as for the stochastic safe controller yielded probabilistic cutoffs $\rho^*_{50} = 0.276$ for the safe controller without perturbations and $\rho^*_{50} = 0.221$ for the learned agent. Additionally, Fig. 5 shows the robustness measure histograms for 200 samples. For the learned agent, the distribution is shifted to slightly higher robustness values (robustness measure mean is 0.78 and standard deviation is 0.32) in comparison to the deterministic safe controller (robustness measure mean is 0.76 and standard deviation is 0.35). Since positive robustness measure values encode faster goal reaching, our numerical results imply that the learned agent improves the performance of the deterministic safe controller, while still exhibiting the same probabilistic level of STL specification compliance. The histogram for the stochastic safe controller has the same range and a similar shape as the histogram for the learned agent, which indicates that the verification result of the perturbed safe controller is, in fact, a good approximator of learned system behavior.

Testing on Robotarium Robot: Although our implementation is in Python and did not focus on efficiency, it was not necessary to improve the computational efficiency to be real-time capable for the Robotarium robot [16]. This is because we mainly run the forward evaluation of the policy network and safe controller online, which are computationally lightweight, and there is no need to predict and incorporate the safety of actions online which is often computation heavy. Fig. 5 shows example trajectories from the Robotarium experiments[2]. We observe that for high robustness values the robot only has to evade shortly or not at all. The main reason for low robustness values is that the robot has to evade and the final time for our experiment is reached before the robot can return to the optimal path and reach the goal.

V. Discussion

Our implementation is a proof of concept for our probabilistically safe RL approach and shows that for the safe evasion task probabilistic safety guarantees are not jeopardized by improving the performance with RL. Since other approaches using RL with temporal logic for safety specifications tackle a different problem (see Sec. I), we compare our approach only with the baseline of the safe controller's performance. A more complex problem would be autonomous driving on highways, where it is very hard to fulfill performance and safety specifications simultaneously during controller synthesis but often a safe controller exists. In the future such more complex problems should be investigated but are out of scope for this paper.

An important assumption of our approach is that a safe (black-box) controller is available. This assumption can be satisfied through the utilization of existing methods to synthesize safe controllers from temporal logic specifications [29], [30] or from expert data via imitation learning [31]. Note that, we only require this controller to satisfy a safety specification, *e.g.* avoid obstacles, and operate within safe bounds. Furthermore, as in the case of our experiment, it can be relatively easy to implement a safe controller. In particular, our safe controller consists of a high-level algorithm that provides waypoints to prevent a collision or track the optimal path as the situation requires. A low-level controller then tracks these waypoints and provides control inputs for the unicycle model. Note that we do not need to know the architecture of the safe controller as our approach regards it as a black-box.

VI. Conclusion

The proposed three-step safe RL approach achieves effective behavior that satisfies a desired probabilistic STL specification. Our results on a safe evasion task show that the separation of safety and performance leads to lean and efficient learning and the probabilistic STL guarantees can easily be verified. The architecture of our approach removes the necessity of online predictions for the safety of actions, which, as a consequence, reduces the computation time and allows us to effortlessly run the controller in real-time.

References

[1] S. Gros, M. Zanon, and A. Bemporad, "Safe reinforcement learning via projection on a safe set: How to achieve optimality?" *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 8076–8081, 2020.

[2] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, no. 1, pp. 109 597–109 614, 2021.

[2]Video of example trajectories: `https://youtu.be/8WWmKfh6WSM`

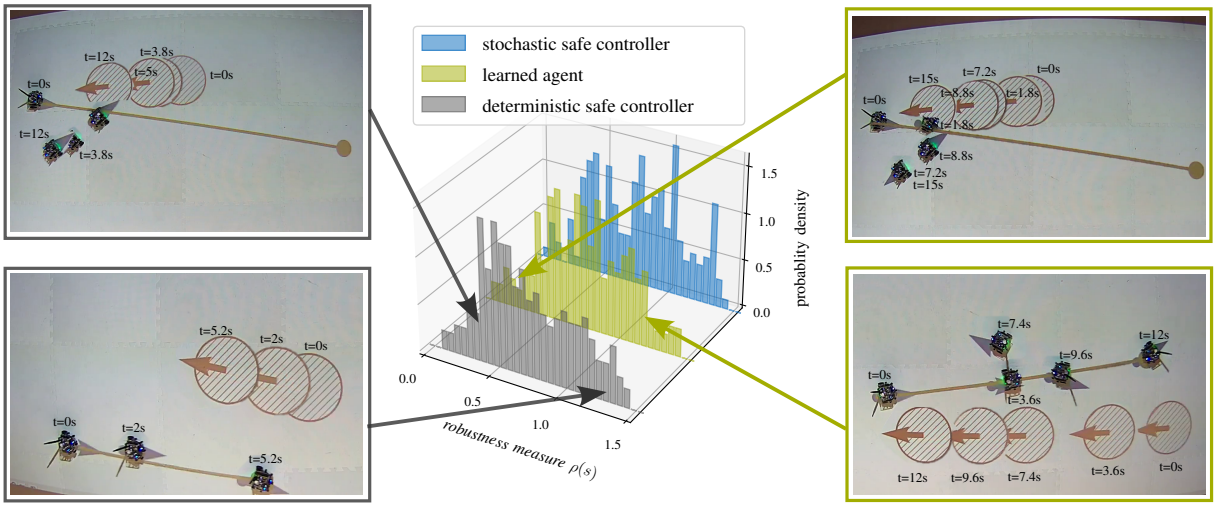


Fig. 5. Center: Robustness measure $\rho(s)$ histograms for 200 samples; Robotarium trajectories for different robustness values for learned agent (left top 0.114, bottom 1.62) and for safe controller (right top: 0.089, bottom: 1.21).

[3] 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 *Proc. of the AAAI Conf. on Artificial Intelligence*, 2019, pp. 3387–3395.

[4] A. K. Akametalu, S. Kaynama, J. F. Fisac, M. N. Zeilinger, J. H. Gillula, and C. J. Tomlin, "Reachability-based safe learning with Gaussian processes," in *Proc. of the IEEE Conference on Decision and Control*, 2014, pp. 1424–1431.

[5] N. Kochdumper, H. Krasowski, X. Wang, S. Bak, and M. Althoff, "Provably safe reinforcement learning via action projection using reachability analysis and polynomial zonotopes," *IEEE Open Journal of Control Systems*, vol. 2, pp. 79–92, 2023.

[6] Y. S. Shao, C. Chen, S. Kousik, and R. Vasudevan, "Reachability-Based Trajectory Safeguard (RTS): A safe and fast reinforcement learning safety layer for continuous control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3663–3670, 2021.

[7] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proc. of the AAAI Conf. on Artificial Intelligence*, 2018, pp. 2669–2678.

[8] B. Könighofer, F. Lorber, N. Jansen, and R. Bloem, "Shield synthesis for reinforcement learning," in *Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles*, 2020, pp. 290–306.

[9] M. Hasanbeig, A. Abate, and D. Kroening, "Cautious reinforcement learning with logical constraints," in *Proc. of the International Conference on Autonomous Agents and Multiagent Systems*, 2020, pp. 483–491.

[10] A. Lavaei, F. Somenzi, S. Soudjani, A. Trivedi, and M. Zamani, "Formal controller synthesis for continuous-space MDPs via model-free reinforcement learning," in *Proc. of the ACM/IEEE International Conference on Cyber-Physical Systems*, 2020, pp. 98–107.

[11] D. Gundana and H. Kress-Gazit, "Event-based signal temporal logic synthesis for single and multi-robot tasks," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3687–3694, 2021.

[12] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan, "Modular deep reinforcement learning for continuous motion planning with temporal logic," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7973–7980, 2021.

[13] P. Akella, A. Dixit, M. Ahmadi, J. W. Burdick, and A. D. Ames, "Sample-based bounds for coherent risk measures: Applications to policy synthesis and verification," *arXiv preprint arXiv:2204.09833*, 2022.

[14] B. Weng, G. A. Castillo, W. Zhang, and A. Hereid, "On safety testing, validation, and characterization with scenario-sampling: A case study of legged robots," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 5179–5186.

[15] A. Corso, R. Moss, M. Koren, R. Lee, and M. Kochenderfer, "A survey of algorithms for black-box safety validation of cyber-physical systems," *Journal of Artificial Intelligence Research*, vol. 72, pp. 377–428, 2021.

[16] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The Robotarium: A remotely accessible swarm robotics research testbed," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2017, pp. 1699–1706.

[17] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Nickovi ć, and S. Sankaranarayanan, *Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications*. Springer, 2018.

[18] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[19] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 96–101, 2018.

[20] ——, "Barrier function based collaborative control of multiple robots under signal temporal logic tasks," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 4, pp. 1916–1928, 2020.

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

[22] 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.

[23] H. Roehm, J. Oehlerking, M. Woehrle, and M. Althoff, "Model conformance for cyber-physical systems: A survey," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 3, pp. 1–26, 2019.

[24] H. Krasowski, J. Thumm, M. Müller, L. Schäfer, X. Wang, and M. Althoff, "Provably safe reinforcement learning: A theoretical and experimental comparison," *arXiv preprint arXiv:2205.06750*, 2022.

[25] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff, "Formalization of interstate traffic rules in temporal logic," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2020, pp. 752–759.

[26] T. R. Torben, J. A. Glomsrud, T. A. Pedersen, I. B. Utne, and A. J. Sørensen, "Automatic simulation-based testing of autonomous ships using Gaussian processes and temporal logic," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 2022.

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

[28] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[29] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An

optimization perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 115–140, 2019.

[30] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *Proc. of the IEEE Conference on Decision and Control*, 2014, pp. 81–87.

[31] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys*, vol. 50, no. 2, 2017.