

# Risk-Awareness in Learning Neural Controllers for Temporal Logic Objectives

Navid Hashemi\*, Xin Qin\*, Jyotirmoy V. Deshmukh\*,  
Georgios Fainekos†, Bardh Hoxha†, Danil Prokhorov†, Tomoya Yamaguchi†  
\*University of Southern California, †Toyota Motor North America R&D.

**Abstract**—In this paper, we consider the problem of synthesizing a controller in the presence of uncertainty such that the resulting closed-loop system satisfies certain hard constraints while optimizing certain (soft) performance objectives. We assume that the hard constraints encoding safety or mission-critical task objectives are expressed using Signal Temporal Logic (STL), while performance is quantified using standard cost functions on system trajectories. In order to prioritize the satisfaction of the hard STL constraints, we utilize the framework of control barrier functions (CBFs) and algorithmically obtain CBFs for STL objectives. We assume that the controllers are modeled using neural networks (NNs) and provide an optimization algorithm to learn the optimal parameters for the NN controller that optimize the performance at a user-specified robustness margin for the safety specifications. We use the formalism of risk measures to evaluate the risk incurred by the trade-off between robustness margin of the system and its performance. We demonstrate the efficacy of our approach on well-known difficult examples for nonlinear control such as a quad-rotor and a unicycle, where the mission objectives for each system include hard timing constraints and safety objectives.

## I. INTRODUCTION

Safety-critical cyber-physical systems typically have hard safety specifications that must be met by all system behaviors to guarantee system safety. Additionally, due to efficiency concerns, system designers often specify performance objectives, and seek controllers to optimize these objectives. For example, consider an autonomous vehicle (AV) following another vehicle. Here, the AV must satisfy the safety specification of maintaining a minimum safe distance ( $d_{safe}$ ) from the lead vehicle. However, the system designer may also want to minimize the travel time for the AV. Clearly, the vehicle can be safe with a high robustness margin by driving slower than required (maintaining distance much greater than  $d_{safe}$ ), but this leads to sub-optimal performance w.r.t. the travel time objective. In many cases, designing for safety and performance objectives may require design trade-offs. While designers must never violate safety requirements in favor of performance, they can trade-off the safety margin against performance. This trade-off thus generates some risk: from a safety perspective how risky is to use a controller that may perform better with a lower safety margin? In this paper, we systematically study this problem.

We assume that safety specifications are provided in a real-time temporal logic such as *Signal Temporal Logic* (STL) [1]. STL has recently emerged as a powerful specification language in the various cyber-physical system applications [2]–[5]. In STL properties, predicates over real-valued signals form atomic subformulae which can be combined using Boolean logic connectives (such as and, or, not), and temporal logic operators (such as eventually, always, until) that

are indexed by time intervals. For example consider a design objective for a quadcopter: “The quadcopter must rendezvous in one of two designated regions  $R_1$  or  $R_2$  exactly 5 to 7 minutes after takeoff before getting as close as possible to a given target destination within 20 mins, while avoiding no-fly zones.” Let  $p(\cdot)$  denote the position of the quadcopter. The hard safety specifications in this objective can be expressed by the following STL formula:

$$\varphi_{qc} \equiv \mathbf{F}_{[5,7]}(p \in R_1 \vee p \in R_2) \wedge \mathbf{G}_{[0,20]}(p \notin R_{nofly}) \quad (1)$$

The soft specification requires us to minimize  $d(p, p_{target})$ , where  $d$  is a distance function and  $p_{target}$  is the target. An advantage of STL is that we can quantify how robustly a given system behavior satisfies an STL property using the notion of a *robustness* value [6]. Given a system behavior and a specification, the robustness can be thought of as a signed distance from the given system behavior to the set of behaviors satisfying the property. We can say that a system has safety robustness margin  $\rho^* > 0$  if the minimum robustness value across all its system behaviors exceeds  $\rho^*$ . We assume that a performance objective is specified as any differentiable, real-valued function of the system behavior.

There has been considerable amount of research on the problem of synthesizing controllers that guarantee STL specifications. For example, using approaches from motion planning [7], [8], model predictive control [3], [9], [10], reactive synthesis [11], [12], reinforcement learning [13], [14], imitation learning [15], [16], and through the use of *control barrier functions* [17], [18]. Of these approaches, the most relevant to our paper is the one based on using control barrier functions (CBFs) [19]. A CBF describes a set  $C$  such that for all system states  $\mathbf{s}_k \in C$ , there exists a control action that ensures that  $\mathbf{s}_{k+1} \in C$ . Control synthesis from CBFs has seen a lot of recent work [19]–[21]. Recent work has focused on CBFs that provide more general classes of invariants such as timed reachability [22], [23] and fragments of STL [24]. *Prima facie*, synthesis of controllers to satisfy STL specifications may look like a well-studied problem, however, several open problems remain:

- 1) Existing work may use hand-crafted CBFs over limited fragments of STL [17]. E.g., existing work does not address disjunctive STL specifications (see Eq. (1)).
- 2) Existing approaches do not consider the trade-off between safety and performance. A naïve encoding of the problem using Lagrange multipliers (as we show in this paper) does not scale, thus demonstrating the need for a more nuanced approach.
- 3) Many existing approaches focus on control of linear

systems or simple nonlinear systems.

4) Existing work does not quantify risk awareness in trading off safety margin versus system performance.

To address all the above challenges, we first formulate an objective function that combines a CBF for STL-based safety specifications with performance objectives using Lagrange multipliers. We then demonstrate that the Lagrangian optimization approach does not scale. We provide an algorithm to automatically generate the CBF for the STL specification directly from its structure. An important consideration in the CBF is our use of the *weighted average* of subformula CBFs to generate the CBF for a disjunctive formula.

Next, we introduce deep neural network (DNN)-based controllers to handle arbitrary nonlinear systems. We train the DNN-based controllers in a model-free fashion using a stochastic gradient optimization method that uses adaptive moments. Our optimization formulation is similar to the problem of training a recurrent neural network (RNN), where a cascade of NNs for a given temporal horizon is trained. A crucial aspect of our optimization algorithm is to explicitly guide the search for DNN parameters using a robustness margin parameter: across iterations, the optimizer alternates between satisfying safety and performance based on the robustness of the DNN controller vis-à-vis the desired robustness margin.

Finally, we evaluate the risk-awareness for each designed controller by picking different robustness margins as design parameters. For this analysis, we utilize the recently formulated risk-aware verification approach [25] that uses risk measures such as value-at-risk and conditional-value-at-risk. We demonstrate the efficacy of our method on several examples of nonlinear systems and disjunctive STL safety specifications.

## II. BACKGROUND

In this section, we provide the mathematical notation and the overall problem definition. We use bold letters to indicate vectors and vector-valued functions, and calligraphic letters to denote sets.

Let  $\mathbf{s}$  and  $\mathbf{a}$  respectively be the variables denoting state and control inputs taking values from compact sets  $\mathcal{S} \subseteq \mathbb{R}^n$  and  $\mathcal{A} \subseteq \mathbb{R}^m$ , respectively. We use the words *action* and control input interchangeably. We consider discrete-time nonlinear feedback control systems of the following form<sup>1</sup>:

$$\mathbf{s}_{k+1} = \mathbf{f}(\mathbf{s}_k, \mathbf{a}_k). \quad (2)$$

Here,  $\mathbf{s}_k$  and  $\mathbf{a}_k$  denote the values of the state and action variables at time  $k$ . We assume that the controller can be expressed as a parameterized function  $\pi_\theta$ , where  $\theta$  is a vector of parameters that takes values in  $\Theta$ . Later in the paper, we instantiate the specific parametric form using a neural network for the controller. Given a fixed vector of parameters  $\theta$ , the parametric control policy  $\pi_\theta$  returns an action  $\mathbf{a}_k$  as a function of the current state  $\mathbf{s}_k \in \mathcal{S}$  and time  $k \in \mathbb{Z}^{\geq 0}$ . Namely,

$$\mathbf{a}_k = \pi_\theta(\mathbf{s}_k, k) \quad (3)$$

<sup>1</sup>Our technique can handle continuous-time nonlinear systems as well. This requires zero-order hold discretization of the dynamics in a sound way to account for system behavior between sample times.

We will be using the terms controller and control policy interchangeably. Under a fixed policy, Eq. (2) is an autonomous discrete-time dynamical system. For a given initial state  $\mathbf{s}_0 \in \mathcal{I} \subseteq \mathcal{S}$  and dynamics  $\mathbf{f}$ , a system trajectory  $\sigma_{\mathbf{s}_0, \mathbf{f}}^\theta$  is a function from  $[0, K] \subset \mathbb{Z}^{\geq 0}$  to  $\mathcal{S}$ , where  $\sigma_{\mathbf{s}_0, \mathbf{f}}^\theta(0) = \mathbf{s}_0$ , and for all  $k \in [0, K-1]$ ,  $\sigma_{\mathbf{s}_0, \mathbf{f}}^\theta(k+1) = \mathbf{f}(\mathbf{s}_k, \pi_\theta(\mathbf{s}_k, k))$ . To address modeling inaccuracies, we also consider bounded uncertainty in the model. We denote  $\mathcal{F}$  as the family of possible realizations of the model  $\mathbf{f} \in \mathcal{F}$ . If the policy  $\pi_\theta$  is obvious from the context, we drop the  $\theta$  in the notation  $\sigma_{\mathbf{s}_0, \mathbf{f}}^\theta$ . The main objective of this paper is to formulate algorithms to obtain the optimal policy  $\pi_{\theta^*}$  that guarantees the satisfaction of certain task objectives and safety constraints while optimizing performance rewards. In the rest of the section, we formulate controller synthesis as an optimization problem that we seek to solve. In order to define this formally, we first introduce a performance reward, and then introduce task objectives/safety constraints.

**Performance reward.** In practical control applications, it is common to quantify the control performance using a state-based reward function [26], [27]. Formally,

**Definition 1** (*Performance reward for a trajectory*) Given a reward function  $r : \mathcal{S} \rightarrow \mathbb{R}$ , and discount factor,  $\gamma \in [0, 1]$ , the performance reward of a trajectory initiating in state  $\mathbf{s}_0$ , under policy  $\pi_\theta$  is defined in Eq. (4).

$$\mathcal{J}^{\text{perf}}(\mathbf{s}_0, \mathbf{f}, \theta) = \sum_{k=0}^K \gamma^k r(\sigma_{\mathbf{s}_0, \mathbf{f}}^\theta(k)) \quad (4)$$

**Task Objectives and Safety Constraints.** We assume that task objectives or safety constraints of the system are specified in a temporal logic known as Signal Temporal Logic (STL) [1]. STL formulas are defined using the following syntax:

$$\varphi = h(\mathbf{s}) \bowtie 0 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \mathbf{F}_I \varphi \mid \mathbf{G}_I \varphi \mid \varphi_1 \mathbf{U}_I \varphi_2 \quad (5)$$

Here,  $\bowtie \in \{\leq, <, >, \geq\}$ ,  $h$  is a function from  $\mathcal{S}$  to  $\mathbb{R}$ , and  $I$  is a closed interval  $[a, b] \subseteq [0, K]$ .

**Semantics.** The formal semantics of STL over discrete-time trajectories have been previously discussed in [6]. We denote the formula  $\varphi$  being true at time  $k$  in trajectory  $\sigma_{\mathbf{s}_0, \mathbf{f}}$  by  $\sigma_{\mathbf{s}_0, \mathbf{f}}, k \models \varphi$ . We say that  $\sigma_{\mathbf{s}_0, \mathbf{f}}, k \models h(\mathbf{s}) \bowtie 0$  iff  $h(\sigma_{\mathbf{s}_0, \mathbf{f}}(k)) \bowtie 0$ . The semantics of the Boolean operations ( $\wedge$ ,  $\vee$ ) follow standard logical semantics of conjunctions and disjunctions respectively. For temporal operators, we say  $\sigma_{\mathbf{s}_0, \mathbf{f}} \models \mathbf{F}_I \varphi$  is true if there is a time  $k \in I$  where  $\varphi$  is true. Similarly,  $\sigma_{\mathbf{s}_0, \mathbf{f}} \models \mathbf{G}_I \varphi$  is true iff  $\varphi$  is true for all  $k \in I$ . Finally,  $\sigma_{\mathbf{s}_0, \mathbf{f}} \models \varphi_1 \mathbf{U}_I \varphi_2$  if there is a time  $k \in I$  where  $\varphi_2$  is true and for all times  $k' \in [0, k)$   $\varphi_1$  is true.

In addition to the Boolean satisfaction semantics, STL also permits quantitative satisfaction semantics. These are defined with a *robustness* function  $\rho$  evaluated over a trajectory. We omit the formal definition; it can be found in [1], [6]. Intuitively, the robustness function defines robustness of predicates at a given time  $k$  to be proportional to the

signed distance of the state variable value at  $k$  from the set of values satisfying the predicate. Conjunctions and disjunctions map to minima and maxima of the robustness of their subformulas respectively. Temporal operators can be viewed as conjunctions/disjunctions (or their combinations) over time. We denote  $\rho_\varphi(\mathbf{s}_0, \mathbf{f})$  as the robustness of the trajectory starting in state  $\mathbf{s}_0$  for the dynamics  $\mathbf{f}$ . Note that if  $\rho_\varphi(\mathbf{s}_0, \mathbf{f}) > 0$  then it implies that  $\sigma_{\mathbf{s}_0, \mathbf{f}} \models \varphi$ .

**Risk Measures.** We now introduce two commonly used risk measures that are used to provide probabilistic guarantees of system correctness. We assume that we are provided with a probability distribution  $\mathcal{D}_{\mathcal{F}}$  on model uncertainties (which is a distribution on  $\mathcal{F}$ ), and  $\mathcal{D}_{\mathcal{I}}$  to denote a distribution over the initial states of the system. A risk measure at a given threshold  $\varepsilon$  (denoted  $r_\varepsilon$ ) is a quantity that can be used to provide the following probabilistic guarantee about the robustness of a given STL specification for the system:

$$\mathbf{s}_0 \sim \mathcal{D}_{\mathcal{I}}, \mathbf{f} \sim \mathcal{D}_{\mathcal{F}} \implies \Pr(-\rho_\varphi(\mathbf{s}_0, \mathbf{f}) \leq r_\varepsilon) \geq \varepsilon \quad (6)$$

We now include two of the standard risk measures used in literature from [28].

**Definition 2** (*Value-at-Risk (VaR), Conditional-Value-at-Risk (CVaR) [28]*) Let  $Z$  be shorthand for  $\rho_\varphi(\mathbf{s}_0, \mathbf{f})$ . The Value-at-Risk is defined as follows:

$$\text{VaR}_\varepsilon(-Z) = \inf_{\zeta \in \mathbb{R}} \{\zeta \mid \Pr(-Z \leq \zeta) \geq \varepsilon\} \quad (7)$$

The conditional-value-at-risk is defined as follows:

$$\text{CVaR}_\varepsilon(-Z) = -\mathbb{E}_{-Z \geq \text{VaR}_\varepsilon(-Z)}[-Z] \quad (8)$$

Essentially, both risk measures provide probabilistic upper bounds on the negative of the robustness value, or provide lower bounds on the actual robustness value, as is required in risk-aware verification [25], [28].

**Problem Definition.** (i) Learn an optimal policy  $\pi_\theta(\mathbf{s}_k, k)$  such that it satisfies a given STL formula  $\varphi$  while maximizing the performance reward defined in Eq. (4).

$$\begin{aligned} \theta^* = & \arg \max_{\theta \in \Theta} \mathbb{E}_{\mathbf{s}_0, \mathbf{f} \sim \mathcal{U}} [\mathcal{J}^{\text{perf}}(\mathbf{s}_0, \mathbf{f}, \theta)] \\ \text{s.t.} & \forall \mathbf{s}_0 \in \mathcal{I}, \forall \mathbf{f} \in \mathcal{F} : \sigma_{\mathbf{s}_0, \mathbf{f}} \models \varphi \end{aligned} \quad (9)$$

(ii) Given a confidence threshold  $\varepsilon$ , we will compute the risk measure  $r_\varepsilon$  that guarantees that:  $(\mathbf{s}_0, \mathbf{f}) \sim (\mathcal{D}_{\mathcal{I}} \times \mathcal{D}_{\mathcal{F}}) \implies \Pr(-\rho_\varphi(\mathbf{s}_0, \mathbf{f}) \leq r_\varepsilon) \geq \varepsilon$ .

### III. CONTROL BARRIER FUNCTIONS FOR STL

In [17], the authors introduce *time-varying control barrier functions* that are used to synthesize controllers that are guaranteed to satisfy a given STL specification. We first adapt this notion to discrete-time nonlinear systems.

**Definition 3** (*Discrete-Time Time-Varying Valid Control Barrier Functions (DT-CBF)*) Let  $b : \mathcal{S} \times [0, K] \rightarrow \mathbb{R}$  be a function that maps a state and a time instant to a real value. Let  $B(k) = \{\mathbf{s}_k \mid b(\mathbf{s}_k, k) \geq 0\}$  be a time-varying set. The function  $b$  is a valid, discrete-time, time-varying CBF if it

```

1 Function stl2cbf( $\varphi, \sigma_{\mathbf{s}_0, \mathbf{f}}$ )
2   case  $\varphi = h(\mathbf{s}, k) \geq 0$ 
3     | return  $\mu(h(\mathbf{s}_k, k))$ 
4   case  $\varphi = \varphi_1 \wedge \varphi_2$ 
5     | return
6       |  $\text{softmin}(\text{stl2cbf}(\varphi_1, \sigma_{\mathbf{s}_0, \mathbf{f}}), \text{stl2cbf}(\varphi_2, \sigma_{\mathbf{s}_0, \mathbf{f}}); \eta)$ 
7   case  $\varphi = \varphi_1 \vee \varphi_2$ 
8     | return
9       |  $\text{wavg}(\text{stl2cbf}(\varphi_1, \sigma_{\mathbf{s}_0, \mathbf{f}}), \text{stl2cbf}(\varphi_2, \sigma_{\mathbf{s}_0, \mathbf{f}}); \beta^\varphi)$ 
10  case  $\varphi = \mathbf{G}_{[a, b]} \varphi$ 
11    | return  $\text{softmin}_{k \in [a, b]}(\text{stl2cbf}(\varphi, \sigma_{\mathbf{s}_0, \mathbf{f}}); \eta)$ 
12  case  $\varphi = \mathbf{F}_{[a, b]} \varphi$ 
13    | return  $\text{wavg}_{k \in [a, b]}(\text{stl2cbf}(\varphi, \sigma_{\mathbf{s}_0, \mathbf{f}}); \beta^\varphi)$ 
14  case  $\varphi = \varphi_1 \mathbf{U}_{[a, b]} \varphi_2$ 
15    | for  $k \leftarrow a$  to  $b$  do
16      |
17      |    $\text{stl2cbf}(\varphi, \sigma_{\mathbf{s}_0, \mathbf{f}}) \leftarrow \text{wavg}(\text{stl2cbf}(\varphi, \sigma_{\mathbf{s}_0, \mathbf{f}}),$ 
18      |    $\text{softmin}(\text{stl2cbf}(\varphi_2, \sigma_{\mathbf{s}_0, \mathbf{f}}),$ 
19      |    $\text{stl2cbf}(\mathbf{G}_{[0, k-1]} \varphi_1, \sigma_{\mathbf{s}_0, \mathbf{f}}); \eta); \beta^\varphi)$ 
20    | return  $\text{stl2cbf}(\varphi, \sigma_{\mathbf{s}_0, \mathbf{f}})$ 

```

**Algorithm 1:** Recursive formulation of CBFs based on an STL formula

satisfies the following condition:

The zero levelsets of the CBF are an envelope for any system trajectory, i.e.,

$$\forall \mathbf{s}_0 \in \mathcal{I}, \forall \mathbf{f} \in \mathcal{F} : \forall k \in [0, K] : \sigma_{\mathbf{s}_0, \mathbf{f}}(k) \in B(k) \quad (10)$$

**DT-CBF for STL.** We formulate CBFs in a recursive fashion based on the formula structure. We describe the overall procedure in Algorithm 1. Before we describe the actual algorithm, we introduce some helper functions. The softmin function defined in Eq. (11) has been used in the past by several approaches [16], [29], [30] as a smooth approximation for computing the minimum of a number of real-valued quantities. In our softmin function, we introduce an additional parameter  $\eta > 1$  that is used to control the level of conservatism in the approximation. Intuitively, as larger values of  $\eta$  reduce the conservatism but require greater numeric precision. Later, we discuss how the softmin function appears as a part of a cost function to be optimized; we include  $\eta$  as a part of this optimization process.

$$\text{softmin}(v_1, \dots, v_k; \eta) = -\frac{1}{\eta} \ln \left( \sum_{i=1}^k e^{-\eta v_i} \right) \quad (11)$$

We also define the weighted average function wavg. We

are interested in two different form of  $\text{wavg}$  presented in Eq. (12) and Eq. (13); here  $\beta = (\beta_1, \dots, \beta_k)$ .

$$\text{wavg}_1(v_1, \dots, v_k; \beta) = \sum_{i=1}^k \left( \frac{\beta_i^2}{\sum_{i=1}^k \beta_i^2} \right) v_i, \quad (12)$$

$$\text{wavg}_2(v_1, \dots, v_k; \beta) = \sum_{i=1}^k \left( \frac{\exp(\beta_i)}{\sum_{j=1}^k \exp(\beta_j)} \right) v_i, \quad (13)$$

where the former is more accurate but the latter is more efficient for gradient descent. Finally, we articulate useful properties of  $\text{softmin}$  and  $\text{wavg}$  in Lemma 3.1.

*Lemma 3.1:* For all  $v_1, \dots, v_k \in \mathbb{R}$ , and for  $\eta \in \mathbb{R}_{>1}$ , the following are true:

$$(\min_i v_i) \geq \text{softmin}(v_1, \dots, v_k; \eta) \quad (14)$$

$$(\min_i v_i) \leq \text{wavg}(v_1, \dots, v_k; \beta) \leq (\max_i v_i) \quad (15)$$

We can now describe Algorithm 1. The function  $b_\phi$  computes the CBF w.r.t. either an atomic signal predicate or a conjunction of atomic predicates. The CBF for an atomic predicate  $\phi$  of the form  $h(s_k) > 0$  is defined using a function  $\mu$  that ensures that  $\mu(h(s_k))$  is positive if  $h(s_k)$  is positive, 0 if it is zero and negative otherwise. The CBF of the conjunction of two predicates is simply the  $\text{softmin}$  of the CBFs of the conjuncts. In the function  $\text{stl2cbf}(\varphi, \sigma_{s_0, f})$ , we consider four cases. If  $\varphi$  is a formula of the form  $\mathbf{G}_{[a, b]} \phi$ , then we return the  $\text{softmin}$  of  $b_\phi(s_\ell, \ell)$  for all  $\ell \in [a, b]$ . If  $\varphi$  is of the form  $\mathbf{F}_{[a, b]} \phi$ , then we return the weighted average of the CBFs at all time instants in  $[a, b]$ . For conjunctions of either kinds of temporal formulas, we again return the  $\text{softmin}$  and for disjunctions, we return the weighted average. Note that the function  $\text{stl2cbf}$  can be invoked with a concrete trajectory whereupon it returns a numeric value. It can also be invoked with a symbolic trajectory (where the symbols  $s_k$  indicate the symbolic state at time  $k$ ), whereupon it returns a symbolic candidate CBF that is the smooth robustness of the trajectory and is a guaranteed lower bound for trajectory robustness  $\rho_\varphi(s_0, f)$ .

*Lemma 3.2:* For any formula  $\varphi$  belonging to STL, for a given trajectory  $\sigma_{s_0, f} = s_0, s_1, \dots, s_n$ , if  $\text{stl2cbf}(\varphi, \sigma_{s_0, f}) > 0$ , then  $\sigma_{s_0, f} \models \varphi$ .

*Proof:* We can prove this recursively over the formula structure and from the identities in Lemma(3.1). It is necessary to mention if  $\text{stl2cbf}() < 0$  it does not imply the STL specifications are violated. ■

*Example 1:* Consider the STL specification in Eq. (16).

$$(\mathbf{F}_{[1, 10]}(s \in \mathcal{E}_1) \vee \mathbf{F}_{[1, 10]}(s \in \mathcal{E}_2)) \wedge \mathbf{G}_{[1, 20]}(s \notin \mathcal{E}_3) \quad (16)$$

Let  $c_2 = (2, 8)$ ,  $c_1 = (5, 5)$ ,  $c_3 = (8, 2)$  and  $r = \sqrt{1.5}$ . Then, in Eq. (16), for  $i \in [1, 3]$ :  $\mathcal{E}_i = (s - c_i)^\top (s - c_i) \leq r$ . We define the CBF  $\mu(s_k \in \mathcal{E}_1)$  to be  $(1 - e^{-((s_k - c_1)^\top (s_k - c_1) - r)})$ . For  $j = 2, 3$ , we define  $\mu(s_k \in \mathcal{E}_j)$  to be  $r - (s_k - c_j)^\top (s_k - c_j)$ . Then, the CBF w.r.t. the formula can be computed using Algorithm. 1.

#### IV. LEARNING-BASED CONTROL SYNTHESIS

We remark that the trajectory  $\sigma_{s_0, f}$  is essentially a repeated composition of  $f$  and the neural controller. Thus, we can compute the gradient of the performance costs and

STL objectives (as expressed by the CBF) with respect to the controller parameters  $\theta$  using standard backpropagation methods for neural networks.

##### A. Training Neural Networks to satisfy specifications

We explain the procedure for training a neural controller w.r.t. performance and safety specifications in algorithm 2. The training algorithm aims to approximate the solution of Eq. (9). Thus, the first step is to reformulate it free from constraints. Algorithm 1 provides the smooth trajectory robustness for a given STL formula  $\varphi$ . This robustness is a function of the common variable  $\eta$  that is used by all  $\text{softmin}$  functions, and the tuple of  $\beta$  variables for each subformula  $\psi$  of a disjunctive formula (denoted  $\beta^\psi$ ). In addition to the neural network parameters  $\theta$ , we also treat the variables  $\eta$  and the sets  $\{\beta^\psi\}$  as decision variables in the training process. The  $\beta$  variables are already unconstrained; however,  $\eta$  is constrained:  $\eta > 1$ . To remove this constraint, we introduce  $\eta = \lambda^2 + 1$ , and use  $\lambda$  as a decision variable. We denote the tuple of conjunctive variable ( $\lambda$ ) and all disjunctive variables ( $\beta^\psi$ ) with  $\mathbf{v}$ . We also denote this robustness with  $\mathcal{J}^{\text{STL}}$  that is a function of tuple  $(s_0, f, \theta, \mathbf{v})$ ,

$$\mathcal{J}^{\text{STL}}(s_0, f, \theta, \mathbf{v}) = \text{stl2cbf}(\varphi, \sigma_{s_0, f}), \quad \mathbf{v} = (\{\beta^\psi\}, \lambda). \quad (17)$$

We also sample a batch  $\hat{\mathcal{I}}$  of initial states uniformly from  $\mathcal{I}$  for training purposes.

The training algorithm is primarily inspired by the Lagrange multiplier technique that transforms a constrained optimization to non-constrained,

$$\mathcal{J} = \max_{\theta, \mathbf{v}} \sum_{s_0 \in \hat{\mathcal{I}}, f \in \mathcal{F}} (\mathcal{J}^{\text{perf}}(s_0, \theta) + \omega_{s_0} \mathcal{J}^{\text{STL}}(s_0, \theta, \mathbf{v})). \quad (18)$$

First order optimality conditions guarantee that as long as the Lagrange multipliers are positive, the cost as defined using  $\mathcal{J}^{\text{STL}}$  is positive. This in turn guarantees the satisfaction of STL specifications along the trajectory. Since  $\mathcal{J}^{\text{STL}}(s_0, f, \theta, \mathbf{v})$  is highly non-convex, optimization (18) is quite intractable and the solution may not satisfy the KKT optimality condition [31]. However, the main role of the Lagrange multipliers  $\omega_{s_0, f}$  is to perform a trade-off between  $\mathcal{J}^{\text{STL}}$  and  $\mathcal{J}^{\text{perf}}$  and one of the contributions of this work is to propose a training process that focuses on applying this trade off. The training algorithm utilizes the gradients  $\nabla_\theta \mathcal{J}^{\text{perf}}(s_0, f, \theta)$  and  $\nabla_{\theta, \mathbf{v}} \mathcal{J}^{\text{STL}}(s_0, f, \theta, \mathbf{v})$  (obtainable from Automatic differentiation package [32]). In case, the cost specified with  $\mathcal{J}^{\text{STL}}$  is less than a user-specified threshold, then the algorithm increases this with a wise selection between  $\nabla_\theta \mathcal{J}^{\text{perf}}(s_0, f, \theta)$  and  $\nabla_{\theta, \mathbf{v}} \mathcal{J}^{\text{STL}}(s_0, f, \theta, \mathbf{v})$ . Otherwise, it increases the performance with  $\nabla_\theta \mathcal{J}^{\text{perf}}(s_0, f, \theta)$ . We call the mentioned user specified threshold as *robustness margin* (denoted by  $\rho$ ).

We now describe Algorithm 2. We use the variable  $i$  to denote the iteration number during training. We use the notation  $(\theta, \mathbf{v})^i$  to denote the value of  $\theta$  and  $\mathbf{v}$  at the beginning of iteration  $i$ . We initialize  $(\theta, \mathbf{v})^0$  randomly.

At the beginning of each training iteration, in line 3 we sample  $f$  from  $\mathcal{F}$ , then in lines (4-6), for all states in  $\hat{\mathcal{I}}$ , we calculate the gradients

---

**Algorithm 2:** Sampling based algorithm for training the parameterized policy.

---

```

1  $i \leftarrow 0$ , Initialize  $(\theta, \mathbf{v})^0$ , Sample  $\mathcal{I}$  to obtain  $\hat{\mathcal{I}}$ 
2 while true do
3   Sample  $\mathbf{f} \in \mathcal{F}$ 
4   foreach  $\mathbf{s}_0 \in \hat{\mathcal{I}}$  do
5      $\delta_1(\mathbf{s}_0) \leftarrow [\nabla_{\theta} \mathcal{J}^{\text{perf}}(\mathbf{s}_0, \mathbf{f}, \theta), 0]$ 
6      $\delta_2(\mathbf{s}_0) \leftarrow [\nabla_{\theta} \mathcal{J}^{\text{STL}}(\mathbf{s}_0, \mathbf{f}, \theta, \mathbf{v}), \nabla_{\mathbf{v}} \mathcal{J}^{\text{STL}}(\mathbf{s}_0, \mathbf{f}, \theta, \mathbf{v})]$ 
    // get states with the best grad. values
7    $\mathbf{b}1, \mathbf{b}2 \leftarrow \arg \max_{\mathbf{s}_0 \in \hat{\mathcal{I}}} \|\delta_1(\mathbf{s}_0)\|_2, \arg \max_{\mathbf{s}_0 \in \hat{\mathcal{I}}} \|\delta_2(\mathbf{s}_0)\|_2$ 
8    $d_1, d_2 \leftarrow \delta_1(\mathbf{b}1), \delta_2(\mathbf{b}2)$ 
    // candidate parameter updates
9    $(\theta_1, \mathbf{v}_1) \leftarrow (\theta, \mathbf{v})^i + \text{Adam}(d_1)$ 
10   $(\theta_{\text{STL}}, \mathbf{v}_{\text{STL}}) \leftarrow (\theta, \mathbf{v})^i + \text{Adam}(d_2)$ 
11   $(\theta_{1,\text{slow}}, \mathbf{v}_{1,\text{slow}}) \leftarrow (\theta, \mathbf{v})^i + \text{Adam}(d_1)/\tau$ 
12  Sample  $\mathbf{s}_0^i$  from  $\mathcal{I}$ 
    /* Pick update giving best tradeoff
       between perf. and safety */
13  if  $\mathcal{J}^{\text{STL}}(\mathbf{s}_0^i, \mathbf{f}, (\theta, \mathbf{v})^i) \leq \rho$  then
14    if  $\mathcal{J}^{\text{STL}}(\mathbf{s}_0^i, \mathbf{f}, \theta_1, \mathbf{v}_1) \geq \mathcal{J}^{\text{STL}}(\mathbf{s}_0^i, \mathbf{f}, (\theta, \mathbf{v})^i)$ 
      then
15       $(\theta, \mathbf{v})^{i+1} \leftarrow (\theta_1, \mathbf{v}_1)$ 
16    else
17       $(\theta, \mathbf{v})^{i+1} \leftarrow (\theta_{\text{STL}}, \mathbf{v}_{\text{STL}})$ 
18  else
19     $(\theta, \mathbf{v})^{i+1} \leftarrow (\theta_{1,\text{slow}}, \mathbf{v}_{1,\text{slow}})$ 

```

---

$\nabla_{\theta} \mathcal{J}^{\text{perf}}(\mathbf{s}_0, \mathbf{f}, \theta)$ ,  $\nabla_{\mathbf{v}} \mathcal{J}^{\text{perf}}(\mathbf{s}_0, \mathbf{f}, \theta)$  (stored in  $\delta_1(\mathbf{s}_0)$ ), and  $\nabla_{\theta} \mathcal{J}^{\text{STL}}(\mathbf{s}_0, \mathbf{f}, \theta, \mathbf{v})$ ,  $\nabla_{\mathbf{v}} \mathcal{J}^{\text{STL}}(\mathbf{s}_0, \mathbf{f}, \theta, \mathbf{v})$  (stored in  $\delta_2(\mathbf{s}_0)$ ). Of these, note that  $\nabla_{\mathbf{v}} \mathcal{J}^{\text{perf}}(\mathbf{s}_0, \mathbf{f}, \theta)$  is 0. We then compute the state  $\mathbf{b}1$  (resp.  $\mathbf{b}2$ ) for which the 2-norm of  $\delta_1(\mathbf{s}_0)$  (resp.  $\delta_2(\mathbf{s}_0)$ ) is the highest. The gradient values of the states  $\mathbf{b}1$  and  $\mathbf{b}2$  are respectively stored in  $d_1$  and  $d_2$  (Line 8).

The next step is to compute potential updates to the parameter values  $\theta$  and the STL parameters  $\mathbf{v}$  (Lines 9-11). Roughly, the values  $(\theta_1, \mathbf{v}_1)$  represent the update to  $(\theta, \mathbf{v})^i$  using only the inclusion of gradient for performance cost in Adam optimizer. The values  $(\theta_{\text{STL}}, \mathbf{v}_{\text{STL}})$  represent the update only using the gradient of smooth trajectory robustness in Adam optimizer. Finally,  $(\theta_{1,\text{slow}}, \mathbf{v}_{1,\text{slow}})$  represents a slower update with gradient of performance for some  $\tau > 1$ .

Next, we sample a state  $\mathbf{s}_0^i$  uniformly at random and use it for cost computation. If  $\mathcal{J}^{\text{STL}}(\mathbf{s}_0^i, \mathbf{f}, (\theta, \mathbf{v})^i) < \rho$ , i.e., our user-provided robustness margin (Line 13), then we need to take steps to increase the smooth trajectory robustness. We consider two cases: (1) If using the update based on the gradient of the performance cost *improves* the smooth trajectory robustness, we choose this update as it allows us to improve both performance and robustness, i.e., satisfaction robustness (Line 15). (2) Otherwise, we use the update based

on the gradient of the smooth trajectory robustness  $\mathcal{J}^{\text{STL}}$  (Line 17).

If  $\mathcal{J}^{\text{STL}}(\mathbf{s}_0^i, \mathbf{f}, (\theta, \mathbf{v})^i) \geq \rho$ , then we are robustly satisfying our STL constraints. In further quest to improve the performance cost, we need to take care that we do not reduce the robustness margin w.r.t. STL constraints. Hence, we use a slower learning rate that takes smaller steps in trying to improve the performance (Line 19).

*Remark 1:* Considering that this algorithm only focuses on increasing  $\mathcal{J}^{\text{STL}}$  up to  $\rho \geq 0$ , once the STL specification is satisfied then it focuses on optimizing performance. In a sense, this switching strategy plays a role similar to that of Lagrange multipliers: performance cost is optimized only if the robustness is above the user-provided threshold.

### B. Risk estimation

The minimum number of samples to guarantee the confidence on the verification results is proposed in [33]. We generate  $N = 10^6$  samples  $(\mathbf{s}_0, \mathbf{f})$  uniformly from  $(\mathcal{I} \times \mathcal{F})$  and simulate the corresponding trajectories  $\sigma_{\mathbf{s}_0, \mathbf{f}}$ . We compute the robustness  $\rho_{\varphi}(\mathbf{s}_0, \mathbf{f})$  for every single trajectory and calculate *VaR* through obtaining the  $\varepsilon * 100$  percentile of the negation of the robustness values [34] and calculate *CVaR* according to definition 2.

## V. EXPERIMENTAL EVALUATION

### A. Unicycle Dynamics

We demonstrate the efficacy of our technique on a nonlinear unicycle model. We define the uncertainty for the initial condition as:

$$\mathcal{I} = \{\mathbf{s}_0 \mid (x_0, y_0, \alpha_0) \in [0.6, 1.4] \times [0.6, 1.4] \times [\frac{2\pi}{5}, \frac{3\pi}{5}]\}$$

The unicycle dynamics with uncertainties are defined as follows,

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \alpha_{k+1} \end{bmatrix} = \begin{bmatrix} (1 + \delta)x_k + v_k/\omega_k (\sin(\alpha_k + \omega_k) - \sin(\alpha_k)) \\ (1 + \delta)y_k + v_k/\omega_k (\cos(\alpha_k) - \cos(\alpha_k + \omega_k)) \\ (1 + \delta)\alpha_k + \omega_k \end{bmatrix}, \quad (19)$$

where  $\delta \in [-0.01, 0.01]$  and the control inputs,  $v_k, \omega_k$  are bounded:  $v_k \in [0, 1]$ ,  $\omega_k \in [-0.5, 0.5]$ . To restrict the controller in proposed bounds we fix the last hidden layer of neural controller [sigmoid, tanh] and include it to model. Thus, we reformulate the dynamics by replacing the controllers with:

$$\begin{aligned} v_k &\leftarrow \text{sigmoid}(0.5a_1(k)), & a_1(k) &\in \mathbb{R} \\ \omega_k &\leftarrow 0.5 \tanh(0.5a_2(k)), & a_2(k) &\in \mathbb{R} \end{aligned}$$

### B. Quadrotor Dynamics

In another attempt we consider controlling a quadrotor with uncertain dynamics. We define the uncertainty for the initial condition as a spherical set,  $\mathcal{I} = \mathcal{B}_r(c)$  with center,  $c = [0.025, 0.025, 0, 0, 0, 0]^T$  and radius  $r = 0.0125$ . The

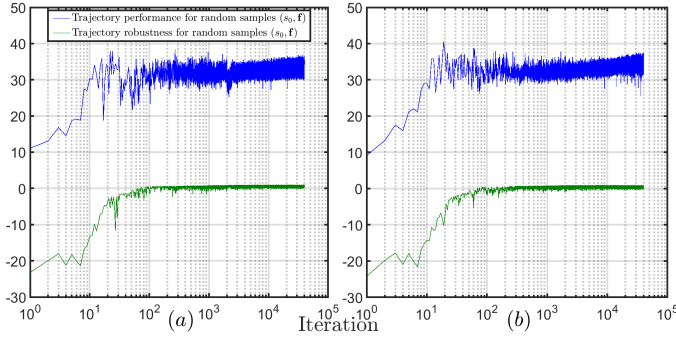


Fig. 1: Figures (a) and (b) present the evolution of performance cost (blue) vs trajectory robustness (green) over the training process of unicycle dynamics for  $\rho = 0.5, 0.3$  respectively. The horizontal axis is presented in log form.

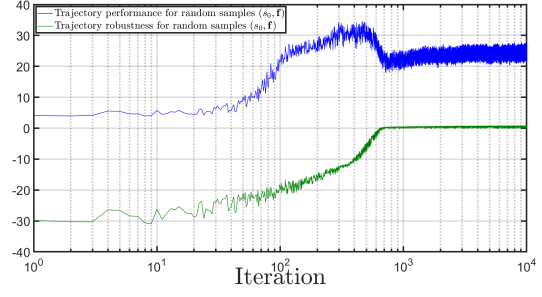


Fig. 2: Presents the evolution of performance cost (blue) vs trajectory robustness (green) over the training process in quadrotor example for  $\rho = 0.1$ . The horizontal axis is in log form.

Example	Training						Validation	
	$\rho$	$\tau$	Controller Dimension	Activation Function	Iterations	Runtime (secs)	Expected value Performance	Expected value $\mathcal{J}^{\text{STL}} / \rho_\varphi$
Unicycle	0.3	1e2	[4,5,2,2]	tanh	40000	1048	35.3430	0.6108 / 0.6109
Unicycle	0.5	1e2	[4,5,2,2]	tanh	40000	1067	33.3528	0.8456 / 0.8518
Quadrotor	0.1	5e4	[7,10,3,3]	tanh	10000	155	24.3024	0.6729 / 0.7516

TABLE I: Training and Validation Results

Example	CBF for atomic propositions, $b_{\phi_i}(s_k, k)$			Reward	discount
	$\phi_1 : \sigma_{s_0}(k) \in \mathcal{E}_1, k \in [1, 10]$	$\phi_2 : \sigma_{s_0}(k) \in \mathcal{E}_2, k \in [1, 10]$	$\phi_3 : \sigma_{s_0}(k) \notin \mathcal{E}_3, k \in [1, 20]$		
Unicycle	$1 - \frac{2}{3}((x_k - 2)^2 + (y_k - 8)^2)$	$1 - \frac{2}{3}((x_k - 8)^2 + (y_k - 2)^2)$	$1 - \exp(1 - \frac{2}{3}((x_k - 5)^2 + (y_k - 5)^2))$	$10 \exp\left(-\frac{(x_k - 8)^2 + (y_k - 8)^2}{36}\right)$	0.9
Quadrotor	$1 - \frac{(x(k) - 0.025)^2 + (y(k) - 0.1)^2 + z(k)^2}{0.00023438}$	$1 - \frac{(x(k) - 0.1)^2 + (y(k) - 0.025)^2 + z(k)^2}{0.00023438}$	$1 - \exp(1 - \frac{(x(k) - 0.0625)^2 + (y(k) - 0.0625)^2 + z(k)^2}{0.00023438})$	$10 \exp\left(-\frac{(x(k) - 0.1)^2 + (y(k) - 0.1)^2 + z(k) + 0.0375)^2}{0.0056}\right)$	0.9

TABLE II: Shows the CBFs and reward functions we utilize in training process.

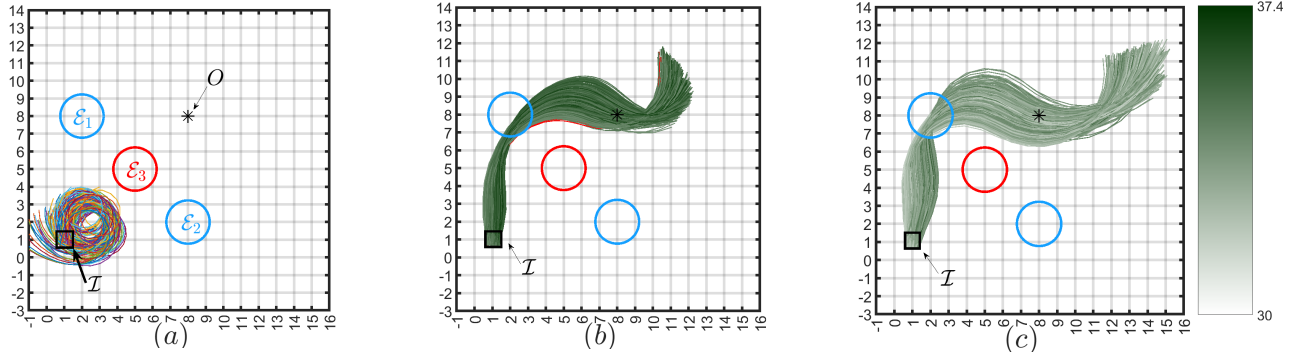


Fig. 3: (a) Represents sample trajectories with the random initial value for  $\theta$ , (b,c) respectively show sample trajectories for trained  $\theta$  with robustness margin  $\rho = 0.3$  and  $0.5$ . This figure clearly shows the trajectories shift towards the center of  $\mathcal{E}_1$  when the robustness margin  $\rho$  increases. For this simulation we sample 500 different  $(s_0, f)$  uniformly at random from  $(\mathcal{I} \times \mathcal{F})$  and simulate the trajectories. The green plots satisfy the STL specifications while its darkness shows the level of performance. There exists 2 red trajectories in (b) that are marginally violating the STL specs.

quadrotor also follows the following uncertain dynamics,

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ z(k+1) \\ v_x(k+1) \\ v_y(k+1) \\ v_z(k+1) \end{bmatrix} = \begin{bmatrix} (1 + \delta)x(k) + 0.05v_x(k) \\ (1 + \delta)y(k) + 0.05v_y(k) \\ (1 + \delta)z(k) + 0.05v_z(k) \\ (1 + \delta)v_x(k) + 0.4905 \tan(u_1(k)) \\ (1 + \delta)v_y(k) - 0.4905 \tan(u_2(k)) \\ (1 + \delta)v_z(k) + 0.05(g - u_3(k)) \end{bmatrix},$$

discretized with ZOH for timestep  $T = 0.05$  sec. Here  $\delta \in [-0.01, 0.01]$  and the control inputs,  $u_1(k) \in [-0.1, 0.1]$ ,  $u_2(k) \in [-0.1, 0.1]$ ,  $u_3(k) \in [7.81, 11.81]$ . The parameter  $g = 9.81$  is the gravity. To impose bounds on the controller, like the Unicycle example, we fix the last hidden layer of the neural controller,  $[\tanh, \tanh, \tanh]$  and include

Confidence Threshold	Unicycle Dynamics				Quadrotor Dynamics	
	$\rho = 0.3$		$\rho = 0.5$		$\rho = 0.1$	
$\varepsilon$	$-VaR_\varepsilon$	$-CVaR_\varepsilon$	$-VaR_\varepsilon$	$-CVaR_\varepsilon$	$-VaR_\varepsilon$	$-CVaR_\varepsilon$
0.95	0.246	0.132	0.540	0.417	0.527	0.455
0.98	0.133	0.036	0.421	0.311	0.452	0.395
0.99	0.059	-0.027	0.336	0.239	0.406	0.360
0.999	-0.133	-0.191	0.121	0.067	0.305	0.284

TABLE III: Risk measures with one million data points.

it in the model,

$$\begin{aligned} u_1(k) &\leftarrow 0.1 \tanh(0.1a_1(k)), & a_1(k) &\in \mathbb{R} \\ u_2(k) &\leftarrow 0.1 \tanh(0.1a_2(k)), & a_2(k) &\in \mathbb{R} \\ g - u_3(k) &\leftarrow 2 \tanh(0.1a_3(k)), & a_3(k) &\in \mathbb{R} \end{aligned}$$

### C. Results

The STL specifications for both examples are adopted from [15] and are introduced in Eq. (16) (Example 1). Regions  $\mathcal{E}_1, \mathcal{E}_2$  and  $\mathcal{E}_3$  for unicycle and quadrotor examples are introduced in Fig 3 and Fig. 4 respectively. The unicycle and quadrotor approaches to the target  $O = [8, 8]^\top$ ,  $O = [0.1, 0.1, -0.0375]^\top$  respectively. They are planned to approach  $O$  with the highest possible level of performance (fast and close) within  $K = 20$  time steps. The reward function and CBFs are defined in table II for both examples. We train a controller to satisfy the performance and STL task for unicycle and quadrotor dynamics. Table I shows the training result for  $\rho = 0.3, 0.5$  in unicycle and  $\rho = 0.1$  in quadrotor example. This table shows the trade-off between performance and STL robustness for the unicycle example.

We utilized (12) and (13) in training the disjunctive parameters  $\beta$  for unicycle and quadrotor respectively. Fig. 5 shows the evolution of disjunctive parameters over the training process. Fig. 1 and 2 present the trade-off between the performance cost  $\mathcal{J}^{\text{perf}}$  and trajectory robustness  $\mathcal{J}^{\text{STL}}$  over the training process for both examples. Fig. 3 and Fig. 4 present the simulation of trajectories for unicycle and quadrotor examples, respectively.

Table III presents the results on probabilistic verification or risk-analysis for the controllers. For the unicycle dynamics, we can see that increasing the robustness margin parameter  $\rho^*$  leads to an increase in the (probabilistic) lower bound on the robustness. Increasing the confidence level reduces the probabilistic lower bound. In fact, at 99.9% confidence, there is a risk of seeing system behaviors that violate the specifications by a margin of 0.133. Similar risks can be seen at the 99% and 99.9% confidence in the  $CVaR$  values. Intuitively, table III matches our expectation that controllers designed with higher robustness margin should have lower risk of violating specifications (at the cost of performance).

## VI. CONCLUSION AND FUTURE WORK

In this work we propose the weighted average, a useful tool to include disjunctive STL formula in the existent soft constrained policy optimization techniques [17]. We also utilize time dependent feedback policies that facilitates control in presence of STL specifications. This enables us to

control the model with smaller neural networks. Non-convex optimizations may be intractable for Lagrange multiplier techniques. We address this problem with proposition of a training algorithm that simulates the trade off between objective and its constraints. We finally utilize this training algorithm for non-convex policy optimization with respect to STL specifications.

In the future, we will focus on improving the scalability of the training process. The proposed recurrent structure for feedback models suffers from vanishing or exploding gradient issue. This results in inefficient training for long trajectories and is due to its resemblance to RNN structures. Thus we plan to include LSTM structure with introduction of hidden states between feedback blocks in the recurrent dynamic structure.

## REFERENCES

- [1] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Proc. of FORMATS*, 2004, pp. 152–166.
- [2] Y. V. Pant, H. Abbas, R. A. Quay, and R. Mangharam, "Fly-by-logic: control of multi-drone fleets with temporal logic objectives," in *Proc. of ICCPS*, 2018, pp. 186–197.
- [3] 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 CDC*. IEEE, 2014, pp. 81–87.
- [4] E. Bartocci, J. V. Deshmukh, A. Donzé, G. E. Fainekos, O. Maler, D. Nickovic, and S. Sankaranarayanan, "Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications." Springer, 2017.
- [5] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," in *Proc. of CDC*. IEEE, 2016, pp. 6565–6570.
- [6] G. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications," in *Formal Approaches to Testing and Runtime Verification*, ser. LNCS, vol. 4262. Springer, 2006, pp. 178–192.
- [7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.
- [8] Y. Shoukry, P. Nuzzo, I. Saha, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "Scalable lazy smt-based motion planning," in *Proc. of CDC*. IEEE, 2016, pp. 6683–6688.
- [9] S. S. Farahani, V. Raman, and R. M. Murray, "Robust model predictive control for signal temporal logic synthesis," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 323–328, 2015.
- [10] E. A. Gol, M. Lazar, and C. Belta, "Temporal logic model predictive control," *Automatica*, vol. 56, pp. 78–85, 2015.
- [11] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *Proc. of HSCC*, 2015, pp. 239–248.
- [12] L. Lindemann, G. J. Pappas, and D. V. Dimarogonas, "Reactive and risk-aware control for signal temporal logic," *IEEE Transactions on Automatic Control*, 2021.
- [13] L. Berducci, E. A. Aguilar, D. Ničković, and R. Grosu, "Hierarchical potential-based reward shaping from task specifications," *arXiv e-prints*, pp. arXiv-2110, 2021.
- [14] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *Proc. of IROS*. IEEE, 2017, pp. 3834–3839.
- [15] W. Liu, N. Mehdipour, and C. Belta, "Recurrent neural network controllers for signal temporal logic specifications subject to safety constraints," *IEEE Control Systems Letters*, vol. 6, pp. 91–96, 2021.
- [16] S. Yaghoubi and G. Fainekos, "Worst-case satisfaction of stl specifications using feedforward neural network controllers: A lagrange multipliers approach," *ACM Transactions on Embedded Computing Systems*, vol. 18, no. 5S, 2019.
- [17] 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.
- [18] —, "Robust control for signal temporal logic specifications using discrete average space robustness," *Automatica*, vol. 101, pp. 377–387, 2019.



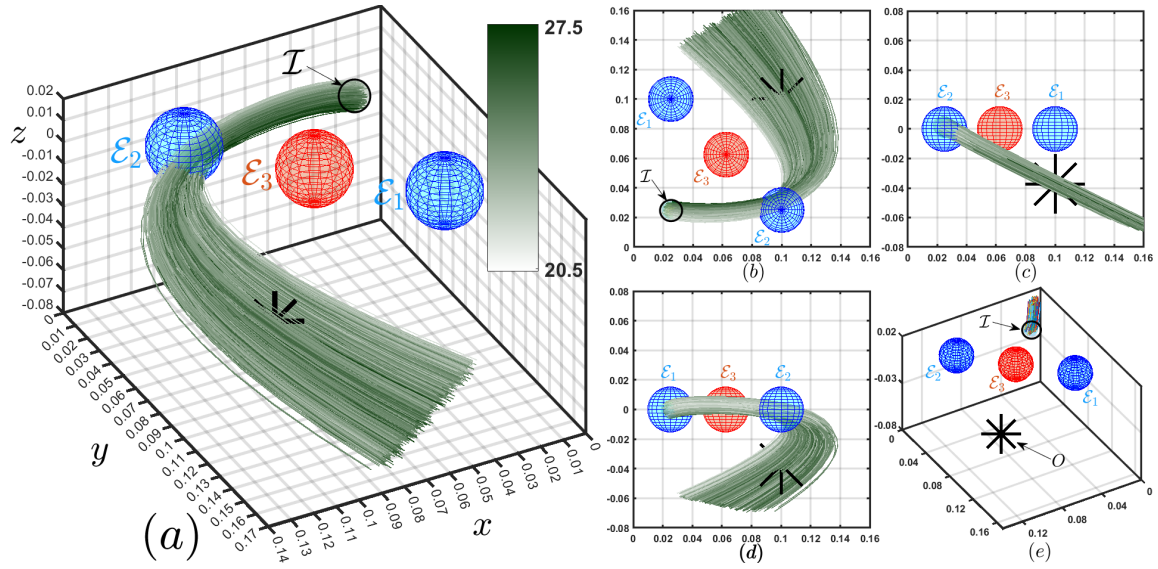


Fig. 4: (a) Represents 500 trajectories generated with trained controller parameters  $\theta$  for  $\rho = 0.1$ . For this simulation, we sample 500 different  $(s_0, f)$  uniformly at random from  $(\mathcal{I} \times \mathcal{F})$  and simulate the trajectories. The darkness of trajectories is corresponding to their level of performance. There is no trajectory violating the STL specification. (b,c,d) shows the projection of trajectories on **X-Y**, **Y-Z** and **X-Z** planes respectively. (e) Represents simulated sampled trajectories for the initial value of  $\theta$  that we utilized in training process.

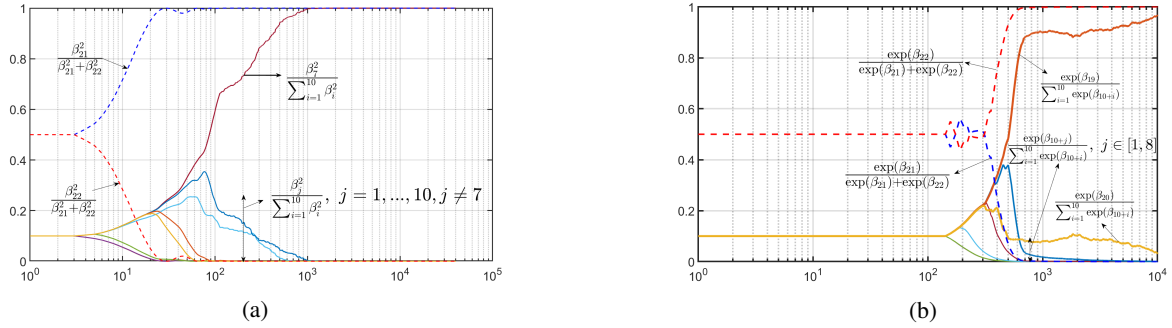


Fig. 5: (a,b) shows the evolution of disjunctive parameters over the training process in unicycle ( $\rho = 0.3$ ) and quadrotor ( $\rho = 0.1$ ) examples, respectively. The log-scale horizontal axis indicates number of training iterations. There are three disjunctive formulas in (16):  $\mathbf{F}_{[1,10]}(s \in \mathcal{E}_1)$ , (that needs parameters  $\beta_1, \dots, \beta_{10}$ ) in its CBF,  $\mathbf{F}_{[1,10]}(s \in \mathcal{E}_2)$  (using parameters  $\beta_{11}, \dots, \beta_{20}$  and the disjunction between these formulas that uses parameters  $\beta_{21}$  and  $\beta_{22}$ ). (a) Parameter  $\beta_{22}$  converging to zero indicates that the system chooses to satisfy the first subformula thus the variables  $\beta_{11}, \dots, \beta_{20}$  are not relevant and not plotted. The  $\beta_7$  parameter has the largest value, indicating the majority of the trajectories are in region  $\mathcal{E}_1$  at time  $k = 7$ . (b) Here,  $\exp(\beta_{21})$  converging to zero implies that  $\beta_1, \dots, \beta_{10}$  are not relevant. As the parameters  $\beta_{19}, \beta_{20}$  are nonzero, the majority of trajectories are in  $\mathcal{E}_2$  at  $k = 9$  and the others at  $k = 10$ .

- [19] 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.
- [20] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," *arXiv preprint arXiv:1903.11199*, 2019.
- [21] P. Nilsson and A. D. Ames, "Barrier functions: Bridging the gap between planning from specifications and safety-critical control," in *Proc. of CDC*, 2018, pp. 765–772.
- [22] K. Garg, E. Arabi, and D. Panagou, "Prescribed-time convergence with input constraints: A control lyapunov function based approach," in *Proc. of ACC*, pp. 962–967.
- [23] K. Garg and D. Panagou, "Control-lyapunov and control-barrier functions based quadratic program for spatio-temporal specifications," in *Proc. of CDC*, 2019, pp. 1422–1429.
- [24] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE Control. Syst. Lett.*, vol. 3, no. 1, pp. 96–101, 2019. [Online]. Available: <https://doi.org/10.1109/LCSYS.2018.2853182>
- [25] P. Akella, M. Ahmadi, and A. D. Ames, "A scenario approach to risk-aware safety-critical system verification," *arXiv preprint arXiv:2203.02595*, 2022.
- [26] B. D. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Courier Corporation, 2007.
- [27] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [28] L. Lindemann, L. Jiang, N. Matni, and G. J. Pappas, "Risk of stochastic systems for temporal logic specifications," 2022. [Online]. Available: <https://arxiv.org/abs/2205.14523>
- [29] Y. V. Pant, H. Abbas, and R. Mangharam, "Smooth operator: Control using the smooth robustness of temporal logic," in *Proc. of CCTA*. IEEE, 2017, pp. 1235–1240.
- [30] K. Leung, N. Aréchiga, and M. Pavone, "Backpropagation for parametric stl," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 185–192.
- [31] S. Boyd and L. Vandenberghe, "Convex optimization," 2004.
- [32] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.
- [33] M. C. Campi and S. Garatti, "The exact feasibility of randomized solutions of uncertain convex programs," *SIAM Journal on Optimization*, vol. 19, no. 3, pp. 1211–1230, 2008.
- [34] R. T. Rockafellar and S. Uryasev, "Optimization of conditional value-at-risk," *Journal of Risk*, vol. 2, pp. 21–41, 2000.