2019 IEEE 58th Conference on Decision and Control (CDC)
Palais des Congrès et des Expositions Nice Acropolis
Nice, France, December 11-13, 2019

Control from Signal Temporal Logic Specifications with Smooth Cumulative Quantitative Semantics

Iman Haghighi, Noushin Mehdipour, Ezio Bartocci, Calin Belta

Abstract— **We present a framework to synthesize control policies for nonlinear dynamical systems from complex temporal constraints specified in a rich temporal logic called Signal Temporal Logic (STL). We propose a novel smooth STL quantitative semantics called cumulative robustness, and efficiently compute control policies through a series of smooth optimization problems that are solved using gradient ascent algorithms. Furthermore, we demonstrate how these techniques can be incorporated in a model predictive control framework. The advantages of combining the cumulative robustness function with smooth optimization methods as well as model predictive control are illustrated in case studies.**

I. Introduction

In the last decade, formal methods have become powerful mathematical tools not only for specification and verification of systems, but also to enable control engineers to move beyond classical notions such as stability and safety, and to synthesize controllers that can satisfy much richer specifications [1]. For example, temporal logics such as Linear Temporal Logic (LTL) [2] and Signal Temporal Logic (STL) [3] have been used to define rich time-dependent constraints for control systems in a wide variety of applications, ranging from biological networks to multi-agent robotics [4], [5], [6].

The existing methods for temporal logic control can be divided into two general categories: automata-based [1] and optimization-based [7], [8]. In the first, a finite abstraction for the system and an automaton representing the temporal logic specifications are computed. A controller is then synthesized by solving a game over the product automaton [1]. Even though this approach has shown some promising results, automata-based solutions are generally very computationally expensive. The second approach leverages the definition of quantitative semantics [9], [10], [11] for temporal logics that interpret a formula with respect to a system trajectory by computing a real-value (called *robustness*) measuring how strongly a specification is satisfied or violated. Consequently, the control problem becomes an optimization problem with the goal of maximizing robustness.

In this paper, we propose a novel framework to synthesize cost-optimal control policies for possibly nonlinear dynamical systems under STL constraints. First, we introduce a new quantitative semantics for STL, called *cumulative* quantitative semantics or *cumulative* robustness degree. The traditional STL robustness degree [9] is very conservative since it only considers the robustness at the most critical time (the time that is closest to violation). On the other hand, we cumulate the robustness over the time horizon of the specification. This results in a robustness degree that is more useful and meaningful in many control applications. Specifically, we show that control policies obtained by optimizing the robustness introduced here leads to reaching desired states faster, and the system spends more time in those states.

We show how to extend smooth approximation techniques in [8] to address the novel notion of cumulative robustness introduced here. We then show how to leverage the smooth cumulative robustness function to perform Model Predictive Control (MPC) under STL constraints for nonlinear dynamical systems using gradient ascent algorithms.

II. Related Work and Contributions

In [11], the authors introduce an extension of STL, called AvSTL, and propose a quantitative semantics for it. The AvSTL quantitative semantics has some similarities with the robustness proposed here, but computes the average robustness over specification horizons instead of cumulating the robustness. Moreover, the work in [11] only investigates a falsification problem, while we consider a more general control problem. A similar approach is also presented in [12] in which robustness is defined using arithmetic and geometric averages.

In [7], Karaman et al. demonstrate that temporal logic control problems can be formulated as mixed integer linear problems (MILP), avoiding the issues with state space abstraction and dealing with systems in continuous space. Since this paper, many researchers have adopted this technique and demonstrated Mixed Integer Linear or Quadratic Programs (MILP/MIQP) are often more scalable and reliable than automata-based solutions [13], [14]. However, MILP has an exponential complexity with respect to the number of its integer variables and the computational times for MILP-based solutions are extremely unpredictable. These types of solutions generally suffer when dealing with very large and nested specifications. More recently, Pant et al. in [8] have presented a technique to compute a smooth abstraction for the traditional STL quantitative semantics defined in [9]. The authors show that control problems can be solved using smooth optimization algorithms such as gradient descent in a much more time efficient way than MILP. This technique also works for any smooth nonlinear dynamics while MILP and MIQP require the system dynamics to be linear or quadratic.

This work was partially supported at Boston University by the NSF under grants IIS-1723995 and CMMI-1400167.

Iman Haghighi, Noushin Mehdipour, and Calin Belta are with Boston University, Boston, MA 02215, USA {haghighi,noushinm,cbelta}@bu.edu

Ezio Bartocci is with TU Wien, Vienna, Austria ezio.bartocci@tuwien.ac.at

We consider the problem of Model Predictive Control (MPC) under temporal logics constraints that is based on iterative, finite-horizon optimization of a plant model and its input in order to satisfy a signal temporal logic specification. The work in [5] employs MPC to control the plant to satisfy a formal specification expressed in a fragment of LTL that can be translated into finite state automata and [15] used it for finite state systems and Büchi automata to synthesize controllers under full LTL specification. MPC has also been used in conjunction with mixed integer linear and quadratic programs for temporal logic control [13], [16].

III. Preliminaries

A. Signal Temporal Logic

Signal Temporal Logic (STL) was introduced in [3]. Consider a discrete unbounded time series $\tau := \{t_k | k \in \mathbb{Z}_{\geq 0}\}$. A signal is a function $\sigma : \tau \to \mathbb{R}^n$ that maps each time point $t_k \in \mathbb{R}_{\geq 0}$ to an n-dimensional vector of real values $\sigma[k]$, with $\sigma_i[k]$ being the ith component. Given a signal σ and $k \in \mathbb{Z}_{\geq 0}$, $\sigma^{t_k} := \{\sigma[k'] | k' \geq k\}$ is the portion of the signal starting at the kth time step of τ.

Definition 1 (STL Syntax):

$$\varphi := \top \mid \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathbf{U}_I \varphi_2, \tag{1}$$

where ϕ, φ_1, φ_2 are formulas and $\top$ stands for the Boolean constant True. We use the standard notation for the Boolean operators. $I = [k_1, k_2]$ denotes a bounded time interval containing all time points (integers) starting from k_1 up to k_2 and $k_2 > k_1 \geq 0$. The building blocks of STL formulas are predicates of the form $\mu := l(\sigma) \geq 0$ where l is a linear or nonlinear combination of the elements of σ. In this paper, we assume that l is smooth. In order to construct a STL formula, different predicates (μ) or STL sub-formulas (φ) are recursively combined using Boolean logical operators ($\neg, \vee, \wedge$) as well as temporal operators. $\mathbf{U}_I$ is the *until* operator. Other temporal operator can also be derived from $\mathbf{U}_I$. $\mathbf{F}_I$ is the *finally* or *eventually* operator and $\mathbf{G}_I$ is the *globally* or *always* operator. For instance, $\mathbf{F}_{[0,5]}\mathbf{G}_{[0,10]}(\sigma_1 > 0)$ means that σ_1 must become positive within 5 units of time in the future and stay positive for 10 steps after that; and $(\sigma_2 > 0)\mathbf{U}_{[0,10]}(\sigma_1 > 0)$ means that σ_1 should become positive at a time point within 10 units of time and σ_2 must be always positive before that.

The authors in [9] introduce a quantitative semantics that can be interpreted as "How much a signal satisfies or violates a formula". The quantitative valuation of a STL formula φ with respect to a signal σ at the kth time step is denoted by $\rho(\varphi, \sigma, t_k)$ and called the *robustness degree*.

Definition 2 (STL robustness):

$$\begin{aligned}
\rho(\top, \sigma, t_k) &:= +\infty, \\
\rho(l(\sigma) \geq 0, \sigma, t_k) &:= l(\sigma[k]), \\
\rho(\neg\varphi, \sigma, t_k) &:= -\rho(\varphi, \sigma, t_k), \\
\rho(\psi \wedge \varphi, \sigma, t_k) &:= \min\{\rho(\psi, \sigma, t_k), \rho(\varphi, \sigma, t_k)\}, \\
\rho(\psi \mathbf{U}_I \varphi, \sigma, t_k) &:= \max_{k' \in I}(\min\{\rho(\varphi, \sigma, t_{k+k'}), \\
&\qquad \min_{k'' \in [k, k+k']} \rho(\psi, \sigma, t_{k''})\}).
\end{aligned} \tag{2}$$

The robustness degree for $\vee$, $\mathbf{F}_I$, and $\mathbf{G}_I$ can be easily derived [9]. The robustness degree is sound, meaning that:

$$\begin{aligned}
\rho(\varphi, \sigma, t_k) > 0 &\Rightarrow \sigma^{t_k} \vDash \varphi, \\
\rho(\varphi, \sigma, t_k) < 0 &\Rightarrow \sigma^{t_k} \nvDash \varphi.
\end{aligned} \tag{3}$$

A formal definition for the *horizon* (h_φ) of a STL formula φ is presented in [17]. Informally, it is the smallest time step in the future for which signal values are needed to compute the robustness for the current time point. For instance, the horizon of the formula $\mathbf{F}_{[0,5]}\mathbf{G}_{[0,10]}(\sigma_1 > 0)$ is $5 + 10 = 15$.

In the rest of this paper, if we do not specify the time of satisfaction or violation of a formula, we mean satisfaction or violation at time 0 (i.e., $\sigma \vDash \varphi$ means $\sigma^0 \vDash \varphi$).

B. Smooth Approximation of STL Robustness Degree

The robustness degree that results from (2) is not differentiable. This poses a challenge in solving optimal control problems using the robustness degree as part of the objective function. The authors in [8] introduce a technique for computing smooth approximations of the robustness degree for Metric Temporal Logic (MTL), which is based on smooth approximations of the max and min functions:

Definition 3 (Smooth Operators):

$$\begin{aligned}
\widetilde{\max}_\beta(a_1, \ldots, a_m) &:= \tfrac{1}{\beta} \ln \textstyle\sum_{i=1}^m e^{\beta a_i}, \\
\widetilde{\min}_\beta(a_1, \ldots, a_m) &:= -\widetilde{\max}(-a_1, \ldots, -a_m).
\end{aligned} \tag{4}$$

The approximation error approaches 0 as β goes to ∞. We denote the smooth approximation of any function ρ by $\tilde{\rho}$.

IV. Problem Statement

Consider a discrete time continuous space dynamical system of the following form:

$$\begin{aligned}
\sigma[k+1] &= f(\sigma[k], u[k]), \\
\sigma[0] &= \gamma,
\end{aligned} \tag{5}$$

where $\sigma[k] \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state of the system at the kth time step of $\tau := \{t_k | k \in \mathbb{Z}_{\geq 0}\}$, $\mathcal{X}$ is the state space, and $\gamma \in \mathcal{X}$ is the initial condition. $u[k] \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input at time step k that belongs to a hyper-rectangle space $\mathcal{U} = [\mathcal{U}_1, \mathcal{U}'_1] \times \ldots \times [\mathcal{U}_m, \mathcal{U}'_m]$. f is a smooth function representing the dynamics of the system. The ith component of σ, u, f, and γ are denoted by σ_i, u_i, f_i, and γ_i, respectively. The system trajectory (n-dimensional signal) produced by applying control policy $u = \{u[k]\}$ is denoted by $\langle \sigma, u \rangle$.

A specification over the state of the system is given as a STL formula φ with horizon h_φ. We also consider a cost function $J : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ where $J(\sigma[k], u[k])$ is a smooth function representing the cost of applying the control input $u[k]$ at state $\sigma[k]$. In the first problem, we intend to determine a control policy $u^* = \{u^*[k] | k = 0, \ldots, h_\varphi - 1\}$ over the time horizon of the specification φ such that φ is satisfied, while optimizing the cumulative cost.

Problem 1 (Finite Horizon Control):

$$\begin{aligned}
u^* = \arg\min & \sum_{k=0}^{h_\varphi - 1} J(\sigma[k], u[k]), \\
\text{s.t.} \quad & \langle \sigma, u \rangle \vDash \varphi.
\end{aligned} \tag{6}$$

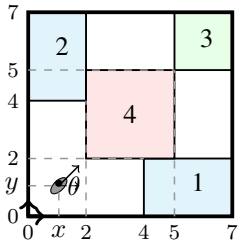

Fig. 1. The workspace for the vehicle in Example 1. The goal is to visit regions 1 or 2 (cyan) in 3 steps and go to region 3 (green) in 7 steps after that and stay there for at least 2 steps, while avoiding region 4 (red).

Furthermore, we intend to find the control policy that results in highest possible robustness degree.

Example 1: Consider an autonomous vehicle on a two dimensional square workspace (Fig. 1). The state of the vehicle consists of the horizontal and vertical position of its center as well as the heading angle ($\sigma = [x, y, \theta]$) The state space is $\mathcal{X} = [0,7] \times [0,7] \times \mathbb{R}$. In Fig. 1, the gray ellipse represents the vehicle. The state evolves according to the following dynamics:

$$\begin{aligned} x[k+1] &= x[k] + \cos\theta[k] v[k] \Delta t, \\ y[k+1] &= y[k] + \sin\theta[k] v[k] \Delta t, \\ \theta[k+1] &= \theta[k] + v[k] \omega[k] \Delta t, \end{aligned} \tag{7}$$

where $u[k] = [v[k], \omega[k]]$ is the control input and belongs to the space $\mathcal{U} = [0,2] \times [-0.75, 0.75]$. Δt is the time step size. We consider the cost function $J(\sigma[k], u[k]) = \|\sigma[k+1] - \sigma[k]\|_2^2$, assigning higher costs to motions over longer distances. Consider the following specification: "Eventually visit region 1 or 2 (cyan) within 6 seconds. Afterwards, move to region 3 (green) in at most 4 seconds and stay there for at least 2 seconds, while always avoiding the unsafe region 4 (red)." Assuming $\Delta t = 0.1$, this translates to:

$$\phi_1 = (\mathbf{G}_{[0,40]} \neg \mu_4) \mathbf{U}_{[0,60]} [(\mu_1 \vee \mu_2) \wedge (\mathbf{F}_{[0,40]} \mathbf{G}_{[0,20]} \mu_3)], \tag{8}$$

where μ_i is the logical formula representing region i:

$$\begin{aligned} \mu_1 &= x > 4 \wedge x < 7 \wedge y > 0 \wedge y < 2, \\ \mu_2 &= x > 0 \wedge x < 2 \wedge y > 4 \wedge y < 7, \\ \mu_3 &= x > 5 \wedge x < 7 \wedge y > 5 \wedge y < 7, \\ \mu_4 &= x > 2 \wedge x < 5 \wedge y > 2 \wedge y < 5. \end{aligned}$$

Note that time step horizon of ϕ_1 is $h_{\phi_1} = 60 + 40 + 20 = 120$.

In the next problem, a time horizon h_M is specified by the user such that $h_M \geq h_\varphi$ and we intend to find the control policy $u^* = \{u^*[k] | k = 0, \ldots, h_M\}$ that results in the satisfaction of a given specification φ at all times.

Problem 2 (Model Predictive Control for Finite Horizon):

$$\begin{aligned} u^* = \arg\min & \sum_{k=0}^{h_\varphi + h_M - 1} J(\sigma[k], u[k]), \\ \text{s.t.} \quad & \langle \sigma, u \rangle \vDash \mathbf{G}_{[0,h_M]} \varphi. \end{aligned} \tag{9}$$

Example 2: Consider a linear system as follows:

$$x[k+1] = \begin{bmatrix} 1 & 0.5 \\ 0 & 0.8 \end{bmatrix} x[k] + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u[k], \tag{10}$$

where $x \in \mathbb{R}^2$, $x_1[0] = x_2[0] = 0$, and $u[k] \in \mathbb{R}$. Consider the following STL specification.

$$\phi_2 = \mathbf{F}_{[0,4]} \mu_5 \wedge \mathbf{F}_{[0,4]} \mu_6, \tag{11}$$

where $\mu_5 = (x_1 > 2 \wedge x_1 < 4)$ and $\mu_6 = (x_1 < -2 \wedge x_1 > -4)$. ϕ_2 requires the value of x_1 (the first component in x) to satisfy both μ_5 and μ_6 within 4 time steps in the future. Note that in this example, $h_{\phi_2} = 4$. Globally satisfying this specification for $h_M = 15$ time steps ($\mathbf{G}_{[0,15]} \phi_2$) means that we require x_1 to periodically alternate between μ_5 and μ_6.

V. Smooth Cumulative Robustness

The robustness degree from Definition 2 has been widely used in the past few years to solve control problems in various applications. Its soundness and correctness properties have enabled researchers to reduce complex control problems to manageable optimization problems. However, this definition is very conservative, since it only considers the system performance at the most critical time. Hence, any information about the performance of the system at other times is lost. Inspired by [11], we introduce an alternative approach to compute the robustness degree, which we call the *cumulative robustness*. This robustness is less conservative than (2), and generally results in better performance if employed to solve control problems such as (6) (see Section VIII).

For any STL formula φ, we define a *positive cumulative robustness* $\rho^+(\varphi, \sigma, t_k) \in \mathbb{R}_{\geq 0}$ and a *negative cumulative robustness* $\rho^-(\varphi, \sigma, t_k) \in \mathbb{R}_{\leq 0}$. For this purpose, we use two functions $\mathfrak{R}^+ : \mathbb{R} \to \mathbb{R}_{\geq 0}$ and $\mathfrak{R}^- : \mathbb{R} \to \mathbb{R}_{\leq 0}$, which we call the positive and negative rectifier, respectively.

Definition 4 (Rectifier Function):

$$\begin{aligned} \mathfrak{R}^+(a) &= \max(0, a), \\ \mathfrak{R}^-(a) &= \min(0, a). \end{aligned} \tag{12}$$

We can use (4) to smoothly approximate both rectifiers.

The positive and negative cumulative robustness are recursively defined as follows:

Definition 5 (Cumulative Robustness):

$$\begin{aligned} \rho^+(l(\sigma) \geq 0, \sigma, t_k) &:= \mathfrak{R}^+(l(\sigma[k])), \\ \rho^-(l(\sigma) \geq 0, \sigma, t_k) &:= \mathfrak{R}^-(l(\sigma[k])), \\ \rho^+(\neg\varphi, \sigma, t_k) &:= -\rho^-(\varphi, \sigma, t_k), \\ \rho^-(\neg\varphi, \sigma, t_k) &:= -\rho^+(\varphi, \sigma, t_k), \\ \rho^+(\psi \vee \varphi, \sigma, t_k) &:= \max\{\rho^+(\psi, \sigma, t_k), \rho^+(\varphi, \sigma, t_k)\}, \\ \rho^-(\psi \vee \varphi, \sigma, t_k) &:= \max\{\rho^-(\psi, \sigma, t_k), \rho^-(\varphi, \sigma, t_k)\}, \\ \rho^+(\psi \wedge \varphi, \sigma, t_k) &:= \min\{\rho^+(\psi, \sigma, t_k), \rho^+(\varphi, \sigma, t_k)\}, \\ \rho^-(\psi \wedge \varphi, \sigma, t_k) &:= \min\{\rho^-(\psi, \sigma, t_k), \rho^-(\varphi, \sigma, t_k)\}, \\ \rho^+(\mathbf{F}_I \varphi, \sigma, t_k) &:= \sum_{k' \in I} \rho^+(\varphi, \sigma, t_{k+k'}), \\ \rho^-(\mathbf{F}_I \varphi, \sigma, t_k) &:= \sum_{k' \in I} \rho^-(\varphi, \sigma, t_{k+k'}), \\ \rho^+(\mathbf{G}_I \varphi, \sigma, t_k) &:= \min_{k' \in I} \rho^+(\varphi, \sigma, t_{k+k'}), \\ \rho^-(\mathbf{G}_I \varphi, \sigma, t_k) &:= \min_{k' \in I} \rho^-(\varphi, \sigma, t_{k+k'}), \\ \rho^+(\psi \mathbf{U}_I \varphi, \sigma, t_k) &:= \sum_{k' \in I} (\min\{\rho^+(\varphi, \sigma, t_{k+k'}), \\ &\qquad \min_{k'' \in [k, k+k']} \rho^+(\psi, \sigma, t_{k''})\}), \\ \rho^-(\psi \mathbf{U}_I \varphi, \sigma, t_k) &:= \sum_{k' \in I} (\min\{\rho^-(\varphi, \sigma, t_{k+k'}), \\ &\qquad \min_{k'' \in [k, k+k']} \rho^-(\psi, \sigma, t_{k''})\}). \end{aligned} \tag{13}$$

An extension to STL, called AvSTL, was introduced in [11]. The authors employed AvSTL to solve falsification problems and did not consider the general control problem. The cumulative robustness for STL as defined in Definition 5 has two main differences from AvSTL. First, we consider signals in discrete time. Second, AvSTL computes the average robustness of $\mathbf{F}_I$ and $\mathbf{U}_I$ over their time intervals, while we cumulate the robustness. Our purpose is to reward trajectories that satisfy the specification in front of $\mathbf{F}_I$ and $\mathbf{U}_I$ for longer time periods.

The positive cumulative robustness can be interpreted as robustness for portions of the signal that satisfy the formula and the negative cumulative robustness can be interpreted as robustness for portions of the signal that violate the formula. Our motivation for defining this alternative robustness degree was to modify the robustness of the *finally* operator in order to cumulate the robustness for all the times in which the formula is true, whereas the traditional robustness degree only considers the most critical time point and does not take other portions of the signal into account. However, this cannot be done in one robustness degree, since the positive and negative values of robustness may cancel each other in time. We divide the robustness into two separate positive and negative values to avoid this issue.

The following example demonstrates the advantages of the cumulative robustness (13) in comparison with (2).

Example 3: Consider $\varphi_e = \mathbf{F}_{[0,10]}(\sigma > 1 \wedge \sigma < 3)$ over a one dimensional signal σ in a discrete time space $\tau = \{0, 1, 2, \dots\}$. Fig. 2 demonstrates two different instances of this signal $\sigma^{(1)}$ and $\sigma^{(2)}$ starting at $\sigma[0] = 0$, both satisfying the formula. φ_e has the same robustness score with respect to $\sigma^{(1)}$ and $\sigma^{(2)}$ at time 0, $\rho(\varphi_e, \sigma^{(1)}, 0) = \rho(\varphi_e, \sigma^{(2)}, 0) = 1$. This is because the traditional robustness degree of the finally operator $\mathbf{F}_I\psi$ only returns the robustness of ψ at the most critical time point in I, which is the same for both instances in this case. However, the positive cumulative robustness of $\mathbf{F}_I\psi$ adds the robustness at all times in which ψ is satisfied, hence rewarding signals that reach the condition specified by ψ faster and stay in the satisfactory region longer. In this example, $\rho^+(\varphi_e, \sigma^{(1)}, 0) = 2.5$ while $\rho^+(\varphi_e, \sigma^{(2)}, 0) = 7.5$. This is a desirable property in many control applications since by synthesizing controls that optimize the cumulative robustness for the finally operator, we are producing a trajectory of the system that reaches the specified condition as soon as possible and holds it true for as long as possible.

By comparing Definition 5 with Definition 2, it is easy to see that a STL formula is satisfied if the corresponding positive robustness is strictly positive.

Proposition 1:

$$\rho^+(\varphi, \sigma, t_k) > 0 \Leftrightarrow \rho(\varphi, \sigma, t_k) > 0 \Rightarrow \sigma^{t_k} \vDash \varphi. \tag{14}$$

Remark 1: The causes of non-smoothness in (13) are the $\max$, $\min$, $\mathfrak{R}^+$, and $\mathfrak{R}^-$ functions. Therefore, any cumulative robustness function can be smoothly approximated by replacing any appearance of these terms with their corresponding smooth approximations from (4). The smooth approximation of ρ^+ and ρ^- are denoted by $\tilde{\rho}^+$ and $\tilde{\rho}^-$.

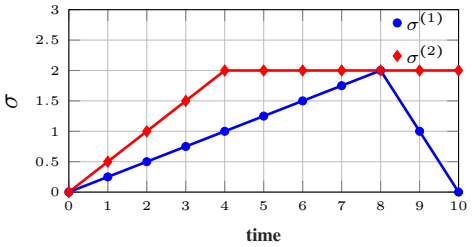

Fig. 2. Two discrete time signals that satisfy $\varphi_e = \mathbf{F}_{[0,10]}(\sigma > 1 \wedge \sigma < 3)$ with similar robustness $\rho(\varphi_e, \sigma, 0)$ but different positive cumulative robustness $\rho^+(\varphi_e, \sigma, 0)$

VI. Smooth Optimization

In this section, our approach to solve Problem 1 is briefly explained. A more detailed explanation is provided in the long version of the paper [1].

To ensure that the state σ always remains in the state space $\mathcal{X}$, we add the following requirement to specification φ:

$$\varphi \leftarrow \varphi \wedge \mathbf{G}_{[0,h_\varphi]}(\sigma \in \mathcal{X}). \tag{15}$$

We aim to use $\tilde{\rho}^+$ in a gradient ascent setting similar to [8] to solve this problem. However, note that $\rho^+ = 0$ any time that a formula is violated. Therefore, $\nabla\rho^+ = 0$ when $\rho^+ = 0$ unless the formula is on the boundaries of violation and very close to being satisfied. As a result, $\tilde{\rho}^+$ is almost guaranteed to fall in its local minimum when one initializes the gradient ascent algorithm for $\tilde{\rho}^+$, unless there is significant a priori knowledge about the system. The negative cumulative robustness ρ^- is not helpful either since there is no soundness theorem associated with it (see Section V).

We can circumvent this setback if we initialize the gradient ascent algorithm such that the specification is already satisfied and we only intend to maximize the level of satisfaction and minimize the cost. Hence, we propose a three-stage algorithm. The first stage aims to find a control policy that minimally satisfies φ, the second stage aims to maximize $\tilde{\rho}^+$, and the third stage aims to minimize the cumulative cost. In each stage, one optimization problem is solved.

Given a generic smooth objective function $Q(\sigma, u)$, an initial control policy $u^\iota = \{u^\iota[k] | k = 0, \dots, h_\varphi - 1\}$, and a termination condition $\mathcal{T}$, we use gradient ascent [18] to find the optimal control policy u^Δ. We start by initializing a control policy u^ι for every time step $k \in \{0, \dots, h_\varphi - 1\}$, and computing the corresponding system trajectory $\langle \sigma, u \rangle$ from the system dynamics (5). At each gradient ascent iteration i, we update the control policy for every time step according to $u \leftarrow u + \alpha_i \nabla Q$ where $\alpha_i > 0$. In the case studies presented in this paper, we used decreasing gradient coefficients $\alpha_{i+1} < \alpha_i$. However, one can employ other strategies that might work more efficiently depending on the application [18]. Each component of ∇Q can be computed as:

$$\nabla_p Q[k] = \frac{\partial Q}{\partial u_p[k]} + \sum_{q=1}^{n} \Big(\frac{\partial Q}{\partial \sigma_q[k+1]} . \frac{\partial f_q[k+1]}{\partial u_p[k]} \Big), \tag{16}$$

[1] https://arxiv.org/abs/1904.11611

for $p = 1, \ldots, m$ and $k = 0, \ldots, h_\varphi - 1$. We continue this process until we find a control policy u^{Δ} that satisfies $\mathcal{T}$.

Recall that the problem setup in Section IV included constraints for both state ($\sigma[k] \in \mathcal{X}$) and control ($u[k] \in \mathcal{U}$). we have already dealt with state constraints as part of the specification (15). Considering the fact that $\mathcal{U} = [\mathcal{U}_1, \mathcal{U}'_1] \times \ldots \times [\mathcal{U}_m, \mathcal{U}'_m]$ is assumed to be a hyper-rectangle, the input constraints are included using a projected gradient.

$$u_p[k] \leftarrow \max\{\min\{u_p[k], \mathcal{U}'_p\}, \mathcal{U}_p\}. \tag{17}$$

At any point in the optimization process, if a component of the control input $u_p[k]$ falls outside of the admissible interval $[\mathcal{U}_p, \mathcal{U}'_p]$, we simply project it into the interval.

In the first stage, we randomly initialize a control policy $u^0 = \{u^0[k] | k = 0, \ldots, h_\varphi - 1\}$ and use the smooth approximation for traditional robustness $\tilde{\rho}$ as the objective function in gradient ascent, only to find a control policy that minimally satisfies the specification (Hence, in this stage $\mathcal{T}$ is $\tilde{\rho} > 0$). In the second stage, we use the policy that we find in stage 1 to initialize a new gradient ascent with the objective function $\tilde{\rho}^+$ and termination condition $\nabla\rho^+ < \epsilon$ where ϵ is a small positive number. Note that $\tilde{\rho}^+$ is guaranteed to be strictly positive at the first iteration ($i = 0$) due to (14). Therefore, it will remain strictly positive as we proceed in the gradient ascent algorithm since this algorithm is designed such that the objective function only increases at each iteration [18]. This prohibits the algorithm to ever fall in a local minimum at $\tilde{\rho}^+ = 0$. At the end of the second stage, we have a control policy u^{Δ_2} with a maximal level of cumulative robustness. However, we intend to minimize the cumulative cost $\sum_k J$ in Problem 1. We perform a third gradient ascent with the objective function $-\sum_k J$ and control policy initialized at u^{Δ_2}. In other words, we are updating the entire control policy gradually in order to decrease the cumulative cost at the expense of cumulative robustness. This stage must terminate before the specification is violated (i.e., while the cumulative robustness is still strictly positive). Hence, the termination condition at this stage is $\tilde{\rho}^+ < \xi$ with $\xi > 0$.

A small choice for ξ results in a control policy with an almost optimal cost that minimally satisfies φ, while a large choice for ξ results in a control policy with a higher level of cumulative robustness that is sub-optimal with respect to the cost. The user can tune ξ to achieve the desirable balance between the level of satisfaction and cost-optimality.

Remark 2: If stage 1 does not terminate, it means that φ is infeasible and we do not need to proceed to stage 2.

VII. Model Predictive Control

In this section, we describe our solution to Problem 2. Note that while Problem 1 requires specification φ to be satisfied at time 0 (i.e., $\langle \sigma, u^* \rangle^0 \vDash \varphi$), Problem 2 requires it to be satisfied at all times in a h_M horizon (i.e., $\langle \sigma, u^* \rangle^{t_k} \vDash \varphi \; \forall k \in \{0, \ldots, h_M\}$ or equivalently $\langle \sigma, u^* \rangle^0 \vDash \mathbf{G}_{[0,h_M]}\varphi$).

The following procedure is performed to solve this problem. For time step $k = 0$, we fix $\sigma[0]$ and use the algorithm described in the previous section to synthesize control policy $u^{h_0} = \{u^{h_0}[k'] | k' = 0, \ldots, h_\varphi - 1\}$. This ensures that ϕ is satisfied at time t_0. Now, we only execute $u^{h_0}[0]$. At time $k = 1$, we fix $\sigma[1]$, use the algorithm presented in the previous section to synthesize $u^{h_1} = \{u^{h_1}[k'] | k' = 0, \ldots, h_\varphi - 1\}$, and only execute $u^{h_1}[0]$, which ensure satisfaction of φ at time t_1. We continue this process until we reach $k = h_M$. Consequently, $\mathbf{G}_{[0,h_M]}\varphi$ is guaranteed to be satisfied for the following control policy, assuming that it is feasible.

$$u^*[k] = u^{h_k}[0] \quad k = 0, \ldots h_M. \tag{18}$$

The smooth optimization procedure becomes particularly advantageous when employed instead of mixed integer programming in a model predictive control setting, since a separate optimization problem needs to be solved at every time step and smooth optimization has a much greater potential for being fast enough to be applied online. Moreover, MILP solvers are very sensitive to the changes in the initial state γ, which is challenging since one has to solve a new MILP from scratch for any changes that occur in γ. This becomes a much more complex challenge when a MILP-based approach is used for multi-agent systems [19]. On the other hand, smooth gradient ascent is much less sensitive to small changes in individual variables such as initial state γ [18].

Similar to [13], we can stitch together trajectories of length h_M using a receding horizon approach to produce trajectories that satisfy $\mathbf{G}_{[0,\infty)}\varphi$. However, this does not guarantee recursive feasibility. In other words, we need to make sure that the resulting trajectory from the optimal controller consists of a loop. If this is the case, we can terminate the computation and keep repeating the control loop forever, which guarantees the satisfaction of the specification at all times, since we know that each loop satisfies the specification.

Formally, we add the following constraint to our optimization problem at every step.

$$\exists k \in \tau, \exists K \in \mathbb{N} \;\; s.t. \;\; \sigma[k + K] = \sigma[k], K > h_\varphi. \tag{19}$$

This ensures that the resulting system trajectory contains a loop. If we find a solution to the optimization problem with positive robustness ($\rho > 0$), this loop satisfies the given specification φ. Therefore, repeating the control strategy that produced this loop forever results in the satisfaction of the formula at all times, ensuring recursive feasibility.

We use a brute force approach to find k, K that satisfy (19). In other words, we start by guessing values for these parameters and keep changing them until we find values for which a solution to the optimization problem exists. Fixing the values of k and K eliminates the quantifiers in (19) and turns it into a linear constraint which is handled easily by any gradient descent solver through projection methods.

VIII. Case Study

In this section, we illustrate how our algorithms are able to solve Examples 1 and 2 in Section IV. All implementations were performed using MATLAB on a MacBook Pro with a 3.3 GHz Intel Core i7 processor and 16 GB of RAM.

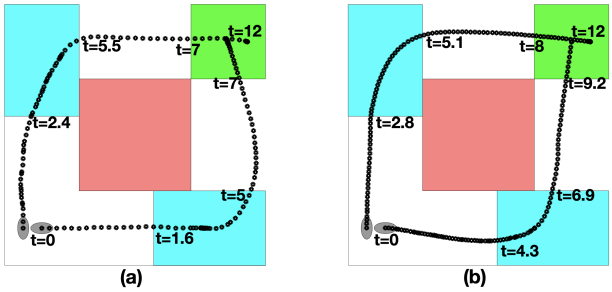

Fig. 3. The optimal path for two vehicles in Example 1 with: (a) maximal cumulative robustness $\tilde{\rho}^+$, (b) maximal traditional robustness $\tilde{\rho}$. (The numbers next to each path indicate time in seconds.)

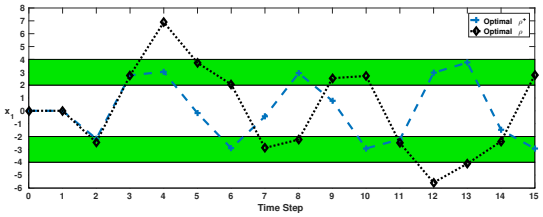

Fig. 4. Evolution of x_1 resulting from optimal control policies

A. Example 1: Path Planning for Autonomous Vehicles

Consider the system and specification of Example 1 with the dynamics of (7) and specification ϕ_1 (8). Assume that there are two vehicles (yielding a 6 dimensional state space). Each vehicle starts from a different initial position and orientation, but follows the same dynamics. Both are required to follow the specification ϕ_1 while avoiding collision. Recall that the objective is to visit one of the cyan regions in 6 seconds and go to the green destination in at most 4 seconds after that, while avoiding the red region (Fig. 1). The solution was computed in 18.3 seconds and the optimal path for each vehicle is shown in Fig. 3(a). The optimal path with respect to the traditional robustness score was also computed in 13.4 seconds and demonstrated in Fig. 3(b). It is obvious from these figures that optimizing the cumulative robustness results in paths in which the vehicles get to their respective destinations faster and remain there longer.

B. Example 2: MPC for a Linear System

Consider the system and specification of Example 2 with the dynamics shown in (10) and specification ϕ_2 presented in (11). We used the MPC framework as described in Section VII to solve Example 2 with optimal cumulative robustness. The computation took 6.73 seconds. Additionally, we computed the control policy derived from only optimizing the traditional robustness $\tilde{\rho}$.

The corresponding evolution of x_1 according to the system dynamics (10) for both cases is presented in Fig. 4. According to (11), the goal is for x_1 to periodically visit both of the green regions in this figure, always visiting each region within at most 4 time steps in the future. Fig. 4 shows that both control policies satisfy this requirement. However, the trajectory that results from optimizing $\tilde{\rho}^+$ (dashed blue) tends to stay in the green regions as long as possible. On the other hand, by optimizing the traditional STL robustness $\tilde{\rho}$, we are only ensuring that the green regions are visited once every 4 time steps, and do not have any control over the duration of satisfaction. As shown in Fig. 4, the dotted black trajectory visits the green regions, but does not stay in them and even goes beyond the green regions three times.

REFERENCES

[1] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-Time Dynamical Systems*. Springer, 2017.

[2] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*. IEEE Computer Society, 1977, pp. 46–57.

[3] 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.

[4] N. Mehdipour, D. Briers, I. Haghighi, C. M. Glen, M. L. Kemp, and C. Belta, "Spatial-temporal pattern synthesis in a network of locally interacting cells," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 3516–3521.

[5] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Trans. Automat. Contr.*, vol. 57, no. 11, pp. 2817–2830, 2012.

[6] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. ACM, 2015, pp. 239–248.

[7] S. Karaman, R. G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," in *Proc. of CDC 2008: the 47th IEEE Conference on Decision and Control*. IEEE, 2008, pp. 2117–2122.

[8] Y. V. Pant, H. Abbas, and R. Mangharam, "Smooth operator: Control using the smooth robustness of temporal logic," in *Control Technology and Applications (CCTA), 2017 IEEE Conference on*. IEEE, 2017, pp. 1235–1240.

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

[10] A. Rodionova, E. Bartocci, D. Nickovic, and R. Grosu, "Temporal logic as filtering," in *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*. ACM, 2016, pp. 11–20.

[11] T. Akazaki and I. Hasuo, "Time robustness in mtl and expressivity in hybrid system falsification," in *International Conference on Computer Aided Verification*. Springer, 2015, pp. 356–374.

[12] N. Mehdipour, C.-I. Vasile, and C. Belta, "Arithmetic-Geometric Mean Robustness for Control from Signal Temporal Logic Specifications," *2019 American Control Conference(ACC)*, arXiv preprint at https://arxiv.org/abs/1903.05186.

[13] 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 *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, 2014, pp. 81–87.

[14] E. S. Kim, S. Sadraddini, C. Belta, M. Arcak, and S. A. Seshia, "Dynamic contracts for distributed temporal logic control of traffic networks," in *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*. IEEE, 2017, pp. 3640–3645.

[15] X. C. Ding, M. Lazar, and C. Belta, "LTL receding horizon control for finite deterministic systems," *Automatica*, vol. 50, no. 2, pp. 399–408, 2014.

[16] L. Lindemann and D. V. Dimarogonas, "Robust control for signal temporal logic specifications using discrete average space robustness," *Automatica*, vol. 101, pp. 377–387, 2019.

[17] A. Dokhanchi, B. Hoxha, and G. Fainekos, "On-line monitoring for temporal logic robustness," in *International Conference on Runtime Verification*. Springer, 2014, pp. 231–246.

[18] D. P. Bertsekas, *Nonlinear programming*. Athena scientific Belmont, 1999.

[19] I. Haghighi, S. Sadraddini, and C. Belta, "Robotic swarm control from spatio-temporal specifications," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, 2016, pp. 5708–5713.