Control Barrier Navigation Functions for STL Motion Planning

Ashkan Zehfroosh and Herbert G. Tanner

***Abstract*— This paper describes an approach to constructing control barrier functions that realize planning and control objectives that are expressed in a fragment of signal temporal logic. The particular construction is based on the navigation function method for robot motion planning and is attractive because it offers a straightforward way to design the robot control law that implements the signal temporal logic specification. The efficacy of the reported method is illustrated through simulation examples.**

I. Introduction

As computational, sensing, and actuation capabilities improve, robots are called to complete increasingly more complex tasks that could involve several subtasks interleaved over time and space. One way that robot motion planning and control methodologies evolved to respond to this need is by encompassing temporal logic as a way of codifying complex specifications.

Signal temporal logic (STL) is a branch of temporal logic that is interpreted over continuous-time signals [1], and in contrast to linear temporal logic (LTL), STL also captures timing constraints associated with complex tasks. This feature makes STL an even more appropriate choice for defining robot tasks within time-limited applications (e.g., robot-assisted pediatric motor rehabilitation [2]).

Motion planning under STL tasks is known to be hard and usually leads to computationally demanding solutions [3]. The standard approach to STL-based motion planning involves a mixed-integer linear programming step [4], [5], [6], which is undoubtedly computationally demanding. The computational complexity of these methods makes real-time implementation challenging, especially in the presence of dynamic environments [7]. This is one of the reasons why control laws may be pre-computed [8], thus trading off some robustness against disturbance and uncertainty for the ability to implement control action in real-time.

The computational burden of STL motion-planning though can be somewhat reduced [3] by employing a control barrier function (CBF) based methodology (see [9]). That method accounts for a fragment of STL that includes conjunctions in the predicates or the temporal operators, and the associated control design still involves the solution of a quadratic programming problem at *each time step* of the motion planning and control loop. Subsequent extensions of this CBF-based STL motion planning method were made along the directions of multi-agent systems with conflicting local specifications [10], and dynamically coupled multi-agent systems [11]. A singular STL motion planning method currently available and able to circumvent the computational burden of solving a quadratic problem at every iteration is the funnel-based procedure that provides continuous-time control laws [12], [13]. The price paid for computational expediency is that only a much smaller fragment of STL can be covered.

This paper contributes to the area of STL-based motion planning by providing another computationally expedient alternative method, which can cover the original fragment of STL. The new methodology centers around the concept of *navigation functions* [14], which is a well-known motion planning technique that provides a direct path to the construction of feedback control laws with guaranteed properties of safety and convergence. Using navigation functions as the base for the construction of control barrier functions allows us to determine control laws that implement STL specifications in closed form through the computation of the navigation function gradient, thus circumventing the need to solve a quadratic problem in each iteration of the motion planning and control loop. What is more, the proposed method allows the incorporation of both robot motion planning destinations as well as regions for the robot to avoid within the same analytical expression, thus reducing the size of the STL specification for a desired task.

For clarity of exposition, the present paper describes the methodology as it applies to *sphere world* environments, with pathways to extensions to star world [15], as well as time-varying robot workspaces [16], [17] readily available.

The rest of this paper is organized as follows. Section II introduces some key technical concepts and necessary mathematical preliminaries. Section III formally states the problem addressed in this paper, and specifies the fragment of STL specifications that motion planning and control solutions are needed for. The control barrier navigation functions are introduced in Section IV as a solution to the stated problem, where both their construction steps as well as the control input computation procedure are outlined. Section V reports on simulation results that illustrate the efficacy of the method.

II. Technical Preliminaries

This section introduces necessary mathematical background needed for the subsequent technical discussion. The section briefly reviews STL, CBFs, and some known results that will be utilized in following sections.

A. Robot dynamics

The exposition in this section starts with some minimal technical terminology and a description of the particular

Zehfroosh and Tanner are with the Department of Mechanical Engineering at the University of Delaware; Email: {ashkanz,btanner}@udel.edu.

This work is made possible in part by support from NSF via award #2014264.

system dynamics that the proposed method applies at. To that end, let $x \in \mathbb{R}^n$ be the state of the robot's dynamics, $u \in \mathcal{U} \subset \mathbb{R}^m$ to be its control input, and let the robot's dynamics be expressed in the form

$$\dot{x} = f(x) + g(x)u \ , \tag{1}$$

assuming that, for now, functions $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are locally Lipschitz continuous.

When every solution to (1) which starts in a set stays in a set $\mathcal{C}$, then that set $\mathcal{C}$ is said to be *forward invariant* relative to (1). Here, we allow for a forward invariant set to be time varying [3] and say that a set $\mathcal{C}(t)$ is forward invariant with respect to (1) if $x(t_0) \in \mathcal{C}(t) \implies x(t) \in \mathcal{C}(t)$ $\forall\, t \in [t_0, t_1] \subset \mathbb{R}_+$.

B. Signal Temporal Logic

Signal temporal logic (STL) is a temporal logic formalism that involves logical *predicates* μ whose truth values are evaluated over continuous signals. In this particular case, the continuous signals are the system's state trajectories at time t, namely $x(t)$. The predicates assume their logical valuates based on a (continuous) *predicate function* $h : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}$ as in

$$\mu := \begin{cases} \text{True} & \text{if } h(x,t) \le 0 \\ \text{False} & \text{if } h(x,t) > 0 \ . \end{cases} \tag{2}$$

Based on such predicates, an STL *formula* φ can be recursively defined as

$$\varphi ::= \text{True} \mid \mu \mid \neg\, \varphi \mid \varphi_1 \wedge \varphi_2 \mid \Diamond_{[a,b]}\ \varphi \mid \Box_{[a,b]}\ \varphi \mid \varphi_1\ U_{[a,b]}\ \varphi_2 \ ,$$

where $a, b \in \mathbb{R}_+$ with $a \le b$ are timing bounds, $\neg$ represents negation, $\wedge$ represents conjunction, $\Diamond$ stands for the temporal operator *eventually*, $\Box$ stands for the temporal operator *always* and U denotes the temporal operator *until* (see [1]).

If a solution $x : \mathbb{R}_+ \to \mathbb{R}^n$ of (1) satisfies an STL specification φ at time t, then we write $(x,t) \models \varphi$. The STL semantics are recursively given by the following rules

$(x,t) \models \mu$	iff	$h(x,t) \le 0$
$(x,t) \models \neg\, \varphi$	iff	$\neg\ \big((x,t) \models \varphi\big)$
$(x,t) \models \varphi_1 \wedge \varphi_2$	iff	$(x,t) \models \varphi_1$ and $(x,t) \models \varphi_2$
$(x,t) \models \Diamond_{[a,b]}\ \varphi$	iff	$\exists t_1 \in [t+a, t+b]$ s.t. $(x,t_1) \models \varphi$
$(x,t) \models \Box_{[a,b]}\ \varphi$	iff	$\forall t_1 \in [t+a, t+b]$, $(x,t_1) \models \varphi$
$(x,t) \models \varphi_1\ U_{[a,b]}\ \varphi_2$	iff	$\exists t_2 \in [t+a, t+b]$ s.t. $(x,t_2) \models \varphi_2$ and $\forall t_1 \in [t+a, t_2]$, $(x,t_1) \models \varphi$.

C. Control Barrier Functions

A CBF enables controller synthesis for dynamic systems such that if the system starts safe set, it will never leave the safe set, rendering the set forward invariant with respect to the dynamics of the system. A CBF can characterize the set of allowable control inputs that guarantee forward invariance of certain regions for a dynamical system at hand. The required control input is picked from a set defined in terms of the CBF, for example by solving an optimization problem in a sampled-data fashion [18]. Such forward invariant sets can potentially be time-varying, as indicated in Section II-A.

For a domain $\mathcal{D} \subseteq \mathbb{R}^n$ and a (possibly time-varying) set of interest $\mathcal{C}(t) \subset \mathcal{D}$, a CBF appears in the form of a scalar differentiable function $\mathfrak{b} : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}$ for which

$$\mathcal{C}(t) = \{x \in \mathcal{D} \mid \mathfrak{b}(x,t) \ge 0\} \ .$$

Mirroring Lyapunov stability analysis, one considers *candidate* barrier functions, which satisfy some basic structural requirements, and *valid* barrier functions, which have been shown to work as intended relative to the dynamics of the system at hand. A function is called a candidate (control) barrier function if there exists (for some control law u) a solution to (1) such that a set $\mathcal{C}(t)$ is forward invariant for a given time interval starting at initial time t_0:

Definition 1 ([3]). *A differentiable scalar function $\mathfrak{b} : \mathcal{D} \times [t_0, t_1] \to \mathbb{R}$ where $\mathcal{D} \subseteq \mathbb{R}^n$ is a* candidate *control barrier function for (1) if for any $x_0 \in \mathcal{C}(t_0) \subset \mathcal{D}$ there exists an absolutely continuous function $x : [t_0, t_1] \to \mathbb{R}^n$ such that $x(t_0) = x_0$ and $x(t) \in \mathcal{C}(t)$ for all $t \in [t_0, t_1]$.*

Note that Definition 1 implies that $\mathcal{C}(t)$ is nonempty for all $t \in [t_0, t_1]$. A candidate (control) barrier function is called valid, if there exists a control law u for (1) that can keep its total time derivative above a negated class $\mathcal{K}$ function of the the barrier function itself:

Definition 2 ([3]). *A candidate control barrier function $\mathfrak{b} : \mathcal{D} \times [t_0, t_1] \to \mathbb{R}$ where $\mathcal{D} \subseteq \mathbb{R}^n$ is a* valid *control barrier function for (1) if there exists a locally Lipschitz extended class $\mathcal{K}$ function α such that for all $(x,t) \in \mathcal{C}(t) \times [t_0, t_1]$ it holds that*

$$\sup_u \left\{ \tfrac{\partial \mathfrak{b}}{\partial x}\big(f(x) + g(x)u\big) \right\} + \frac{\partial \mathfrak{b}}{\partial t} \ge -\alpha\big(\mathfrak{b}(x,t)\big) \ . \tag{3}$$

It is known that (3) is sufficient to ensure the nonnegativeness of $\mathfrak{b}(x,t)$ over the interval $[t_0, t_1]$:

Lemma 1 ([3]). *Let $\alpha : \mathbb{R}_+ \to \mathbb{R}_+$ be a locally Lipschitz (extended) class $\mathcal{K}$ function, and $\eta : [t_0, t_1] \to \mathbb{R}$ an absolutely continuous function. If $\dot{\eta}(t) \ge -\alpha\big(\eta(t)\big)$ for all $t \in [t_0, t_1]$, then $\eta(t) \ge 0$ $\forall\, t \in [t_0, t_1]$.*

III. Problem Statement

Following [3], this paper considers the following fragment of STL:

$$\psi ::= \text{True} \mid \mu \mid \psi_1 \wedge \psi_2 \tag{4a}$$

$$\varphi ::= \Diamond_{[a,b]}\ \psi \mid \Box_{[a,b]}\ \psi \mid \psi_1\ U_{[a,b]}\ \psi_2 \mid \varphi_1 \wedge \varphi_2 \ , \tag{4b}$$

where formulae ψ_1 and ψ_2 are of the type defined in (4a), and formulae φ_1, φ_2 are of the type defined in (4b).

Similarly [3], the common assumptions are made regarding the boundedness of solutions of (1) and the nature of the term $g(x)$ that appears in its right hand side:

Assumption 1 ([3]). *For an* STL *formula* φ *as defined in* (4b)*, there exists a constant* $C \geq 0$ *such that*

$$(x, t_0) \models \varphi \implies \|x(t)\| \leq C \ \ \forall\, t \geq t_0 \ \ .$$

In other words, satisfaction of formula φ guarantees a bounded trajectory. For the (control affine) dynamics of the robot in (1) we essentially assume that we can feedback linearize it:

Assumption 2 ([3]). *The vector function* $g(x)$ *in* (1) *is such that* $g(x)g(x)^\intercal$ *is positive definite for all* $x \in \mathcal{D}$.

Now the problem under consideration of this paper can be stated as follows:

Problem 1. *Find an input control law* $u(x, t)$ *that guarantees that the solution* $x : \mathbb{R}_+ \to \mathbb{R}^n$ *of* (1) *starting from* $x_0 = x(t_0)$ *be such that* $(x, t_0) \models \varphi$.

IV. Technical Approach

This section introduces a special time-varying CBF that is based on a navigation function [14]. This particular construction leverages the navigation function properties of a key component of the CBF to yield a direct method for obtaining the control law u in (1) that is guaranteed to satisfy the desired STL specification.

A. Navigation functions as control barrier functions

Navigation functions were originally introduced for robot navigation tasks in static environments [14]. In the context of *time-varying* forward invariant sets, the relatively recent extension of the navigation function methodology that allows for time-varying and set-valued destinations [16] seem particularly relevant. To this end, this section borrows from this extension and for simplicity restricts the discussion to *sphere world* robot workspace topologies, with the understanding that extensions to more complicated star world workspaces and time-varying destinations are available [15].

Interestingly, navigation functions can capture and codify predicates expressing regions of the robot workspace that need to be avoided (in the form of "obstacles"), and by doing so allow for a more compact representation of STL formulae and a shorter expression of the total STL system specification. Recent extensions of the navigation function methodology to time-varying obstacle cases [17] can offer additional avenues for future work, but for clarity of exposition are not leveraged in this particular paper.

To see how STL specifications can be codified in control barrier navigation functions (CBNFs), recall that an STL formula consists of logical predicates μ of the form (2), representing regions in the workspace of the robot that may be of interest at different time instants. Since the workspace of the robot has been assumed to be a sphere world, all such regions are naturally expressed as spheres. This assumption does not limit the generality of the approach given available extensions to star world environments [14], [15].

One of the key features in the navigation function extension to time-varying destinations [16] is that it allowed for destination *manifolds* as opposed to isolated points. Such a destination manifold is still sphere-shaped and represents the zero level set of the function

$$h_i\big(x(t)\big) = \|x(t) - x_{c_i}\|^2 - r_i^2 \ \ , \tag{5}$$

which can be taken as the predicate function encoding logical predicate μ_i. In (5), one distinguishes the predicate function's (spherical) region having a center x_{c_i} and radius r_i. Consistent with the STL semantics (2), μ_i is true when $h_i(x) \leq 0$ and false otherwise. Regions of the robot's workspace that always need to be avoided can now be encoded as (static) obstacles and incorporated all together in a specific functional representation [14]. To that end, assume first that the implicit representation of each one of those isolated (obstacle) regions is defined as a function

$$\beta_j(x) = \|x(t) - x_{\mathrm{ob}_j}\|^2 - r_{\mathrm{ob}_j}^2, \qquad j = 1, \ldots, M$$

where x_{ob_j} and r_{ob_j} denote the center and radius of each undesirable spherical (obstacle) region. Our understanding is that obstacles are being avoided by the robot located at x as long as $\beta_i(x) > 0$. Similarly, the boundary of the workspace itself is captured by the function

$$\beta_0(x) = -\|x(t) - x_{\mathrm{ws}}\|^2 + r_{\mathrm{ws}}^2 \ \ ,$$

where x_{ws} and r_{ws} stand for the center and radius of the workspace boundary, respectively. Given these constructs, the combined obstacle representation can take the form

$$\beta(x) = \prod_{j=0}^{M} \beta_j(x) \ \ ,$$

and with that, a (time-varying) navigation function $\phi_i(x)$ can be explicitly constructed for predicate μ_i as

$$\phi_i(x, t) = \frac{h_i(x, t)}{\big[h_i(x, t)^\kappa + \beta(x)\big]^{1/\kappa}} \ \ , \tag{6}$$

with $\kappa = 2n$ for $n \in \mathbb{N}$ in the role of a positive tuning constant which be set sufficiently high to guarantee navigation function properties for (6). These navigation function properties are essentially that

- ϕ_i is continuously differentiable,
- the destination manifold globally minimizes ϕ_i,
- ϕ_i is uniformly maximal on the boundary of undesirable regions,
- the gradient of ϕ_i is nonzero everywhere but at a finite set of (possibly time-varying) isolated points, and that
- for the gradient system defined by ϕ_i the attraction region of the points where the gradient vanishes is a set of measure zero.

Note that as constructed in (6), $\phi_i(x, t) \in (0, 1]$ for all x that do not satisfy μ_i, and unless the initial condition $x(t_0)$ is on the zero-measure set of attraction of this finite set of isolated points, the flows of $\dot{x} = -\nabla_x \phi_i(x, t)$ eventually converge to the zero level set of $h_i(x, t)$. (For a proof of the convergence properties of such a time-varying gradient system, see [19].)

B. A Compositional Approach for CBNFs

1) STL specifications with no conjunctions: This subsection describes how to construct a CBNF for an STL specification that does not involve conjunctions of predicates and temporal operators.

If th STL specification at hand is of the form $\Diamond_{[a,b]}\mu_i$ then the CBNF can be constructed as

$$\mathfrak{b}_i(x,t) = 1 - \phi_i(x,t) - c_i(t) \ , \tag{7}$$

where $c_i : \mathbb{R}_+ \to [0,1]$ is a non-decreasing function satisfying $c_i(0) = 0$ and $c_i(t') = 1$ at *some* $a \leq t' \leq b$. Note that when $c_i(t') = 1$, to have $\mathfrak{b}_i(x,t) \geq 0$, there must be $\phi_i(x,t) \leq 0$ and thus $h_i(x,t) \leq 0$ which means μ_i is true at $t' \in [a,b]$.

If the specification has the form $\Box_{[a,b]}\mu_i$, the CBNF can have the same general form (7), only now the non-decreasing function $c_i : \mathbb{R}_+ \to [0,1]$ is such that $c_i(0) = 0$ and $c_i(t') = 1$ for *all* $t' \in [a,b]$.

The remaining case refers to specifications of the form $\mu_1 \ U_{[a,b]} \ \mu_2$, for which the CBNF is constructed as

$$\mathfrak{b}(x,t) = -\ln\left(\mathrm{e}^{-\mathfrak{b}_1(x,t)} + \mathrm{e}^{-\mathfrak{b}_2(x,t)}\right) \ , \tag{8}$$

where once again $\mathfrak{b}_i(x,t)$ is in the form (7), and the difference between $\mathfrak{b}_1$ and $\mathfrak{b}_2$ is that for $i = 2$ the function $c_2 : \mathbb{R}_+ \to [0,1]$ is a non-decreasing function satisfying $c_2(0) = 0$ and $c_2(t') = 1$ for some $t' \in [a,b]$, while for $i = 1$ the function $c_1 : \mathbb{R}_+ \to [0,1]$ is a non-decreasing function satisfying $c_1(0) = 0$ and $c_1(t'') = 1$ for all $t'' \in [a,t']$. It is noted here that such c_1 functions can actually constructed to be *smooth*, following an example given in [20].

2) STL specifications with no temporal operator conjunctions: This subsection refers to STL specifications that may have conjunctions involving predicates but not temporal operators. Without loss of generality, let us work with formulae $\psi_1 = \mu_1 \wedge \mu_2$ and $\psi_2 = \mu_3 \wedge \mu_4$ as generic representatives of such predicates.

If the specification at hand has the form $\Diamond_{[a,b]}\psi_1$, the CBNF can once again take the form of (8) where $\mathfrak{b}_i(x,t)$ is built as in (7) for $i \in \{1,2\}$ and now *both* $c_i : \mathbb{R}_+ \to [0,1]$ functions are non-decreasing with $c_i(0) = 0$ and $c_i(t') = 1$ for some $t' \in [a,b]$.

For specifications of the form $\Box_{[a,b]}\psi_1$, the CBNF can be similarly constructed as in the form (8), where this time $\mathfrak{b}_i(x,t)$ following (7) for $i \in \{1,2\}$ both involve functions $c_i : \mathbb{R}_+ \to [0,1]$ that are non-decreasing and satisfy $c_i(0) = 0$ and $c_i(t') = 1$ for all $t' \in [a,b]$.

Finally, for specifications that involve the Until operator and are of the form $\psi_1 \ U_{[a,b]} \ \psi_2 = (\mu_1 \wedge \mu_2) U_{[a,b]} (\mu_3 \wedge \mu_4)$, the CBNF can be formed similarly to (8) in the form

$$\mathfrak{b}(x,t) = -\ln\left(\sum_{i=1}^{k} \mathrm{e}^{-\mathfrak{b}_i(x,t)}\right) \ , \tag{9}$$

where $k = 4$, and all $\mathfrak{b}_i(x,t)$ are based on (7), but for $i \in \{3,4\}$ the functions $c_i : \mathbb{R}_+ \to [0,1]$ are non-decreasing with $c_i(0) = 0$ and $c_i(t') = 1$ for some $t' \in [a,b]$, while $\mathfrak{b}_j(x,t) = 1 - \phi_j(x) - c_j(t)$ for $j \in \{1,2\}$, where $c_j : \mathbb{R}_+ \to [0,1]$ is a non-decreasing function with $c_j(0) = 0$ and $c_j(t'') = 1$ for all $t'' \in [a,t']$.

3) General case of STL specifications: Combining the constructions of Sections IV-B.1 and IV-B.2, one is now in position to form CBF for more general STL specifications in the fragment defined in (4).

Take for example the following STL formula:

$$(\Diamond_{[a_1,b_1]}\psi_1) \wedge (\Box_{[a_2,b_2]}\psi_2) \wedge (\psi_3 U_{[a_3,b_3]}\psi_4) \ .$$

Based on the aforementioned process, the CBNF that encodes this whole expression can take the form

$$\mathfrak{b}(x,t) = -\ln\left(\mathrm{e}^{-\mathfrak{b}_1(x,t)} + \mathrm{e}^{-\mathfrak{b}_2(x,t)} + \mathrm{e}^{-\mathfrak{b}_3(x,t)}\right) \ ,$$

where

- $\mathfrak{b}_1$ is constructed as in (8) with the c_i functions associated with the predicates in ψ_1 attaining the value of 1 for some $t \in [a_1,b_1]$;
- $\mathfrak{b}_2$ is constructed as in (8) with the c_i functions associated with the predicates in ψ_2 attaining the value of 1 for all $t \in [a_2,b_2]$; and
- $\mathfrak{b}_3$ formed as in (9), where the c_i functions associated with the predicates in ψ_4 achieving 1 for some $t' \in [a_3,b_3]$ and the c_i functions associated with the predicates in ψ_3 becoming 1 for all $t'' \in [a_3,t']$.

The exponential form (9) can sometimes lead to conservative control designs when an STL specification has large number of conjunctions. The conservativeness of the construction process outlined in Sections IV-B.1–IV-B.3 can be partially relaxed by using the deletion mechanism of [3], whereby a component CBF $\mathfrak{b}_i(x,t)$ drops from the composite construction $\mathfrak{b}(x,t)$ whenever time t exceeds the upper limit of the time interval of its corresponding temporal operator, say, $[a_i,b_i]$, i.e., $t > b_i$.

C. Efficient determination of the control input

Section IV-A primarily illustrated how the navigation functions can be used in the construction of CBNFs that encode STL specifications in the fragment defined by (4). This section focuses on demonstrating that control design can also be facilitated due to the navigation function properties afforded by the proposed component CBNFs.

Without loss of generality, assume that the CBNF codifying the desired STL specification is constructed through the process outlined in Section IV-A and is presented in the form (9), where each component $\mathfrak{b}_i$ is continuously differentiable.

Similarly [3], the following assumption is made:

Assumption 3. *For any time $t_j \leq t \leq t_{j+1}$, the safe set $\mathcal{C}(t)$ is either non-decreasing or decreasing. For the latter case, it holds that $\frac{\partial \mathfrak{b}(x,t)}{\partial x} \neq 0$ for all $(x,t) \in \mathcal{C}(t) \setminus \mathcal{C}(t_{j+1}) \times [t_j, t_{j+1}]$.*

Now, following from Lemma 1 and the above assumption, the control input u for (1), that guarantees $\mathfrak{b}(x,t) \geq 0$ at

each time interval $[t_j, t_{j+1}]$ can be computed directly using the gradient of the CBNF:

$$u(x,t) := \begin{cases} \frac{-g(x)^\intercal f(x)}{g(x)g(x)^\intercal}, & \exists\, t' \in [t_j, t_{j+1}] : \frac{\partial \mathfrak{b}}{\partial x}\big|_{x(t')} = 0 \\ k\, g(x)^\intercal \frac{\partial \mathfrak{b}(x,t)}{\partial x} & \text{otherwise} \end{cases}, \quad (10)$$

and for the case $\frac{\partial \mathfrak{b}(x,t)}{\partial x} \neq 0$, k is selected such that the following condition, involving an extended class-$\mathcal{K}_\infty$ function α, holds:

$$\frac{\partial \mathfrak{b}(x,t)}{\partial x}^\intercal (f + g\, u) + \frac{\partial \mathfrak{b}(x,t)}{\partial t} \geq -\alpha\big(\mathfrak{b}(x,t)\big) \ . \quad (11)$$

Therefore, k can be selected as the solution of the equation

$$k \left[\frac{\partial \mathfrak{b}}{\partial x}^\intercal (gg^\intercal) \frac{\partial \mathfrak{b}}{\partial x}\right] = -\alpha\big(\mathfrak{b}\big) - \frac{\partial \mathfrak{b}}{\partial t} - \frac{\partial \mathfrak{b}}{\partial x}^\intercal f(x) \ . \quad (12)$$

Note that if Assumption 2 holds (i.e. $g(x)g(x)^\intercal$ is positive definite), a solution to (12) always exists for any $\frac{\partial \mathfrak{b}(x,t)}{\partial x} \neq 0$.

The following proposition states that the control law (10) in fact offers the minimum-norm input to keep $\mathfrak{b}(x,t) \geq 0$.

Proposition 1. *The second term in control law* (10) *coincides with the solution of the following quadratic programming:*

$$\begin{aligned} &\min \|\hat{u}\|^2 \qquad \text{such that} \\ &\tfrac{\partial \mathfrak{b}}{\partial x}^\intercal \big[f + g\, \hat{u}\big] + \tfrac{\partial \mathfrak{b}}{\partial t} \geq -\alpha\big(\mathfrak{b}\big) \quad . \end{aligned} \quad (13)$$

Proof: Let u^* be the solution to the quadratic programming (13). Toward contradiction argument, suppose $u \neq u^*$, so it must be the case that $\|u\| > \|u^*\|$. Then by (12) and (13), *both* the following conditions should be satisfied:

$$\left.\begin{aligned} \tfrac{\partial \mathfrak{b}}{\partial x}^\intercal\, g\, u &= -\alpha\big(\mathfrak{b}\big) - \tfrac{\partial \mathfrak{b}}{\partial t} - \tfrac{\partial \mathfrak{b}}{\partial x} f \\ \tfrac{\partial \mathfrak{b}}{\partial x}^\intercal\, g\, u^* &\geq -\alpha\big(\mathfrak{b}\big) - \tfrac{\partial \mathfrak{b}}{\partial t} - \tfrac{\partial \mathfrak{b}}{\partial x} f \end{aligned}\right\} \ . \quad (14)$$

Given now that u is by construction aligned to the vector $g^\intercal \frac{\partial \mathfrak{b}}{\partial x}$, and since $\|u\| > \|u^*\|$, the inner product $\big(\frac{\partial \mathfrak{b}}{\partial x}^\intercal g\big)\, u$ must always be bigger than $\big(\frac{\partial \mathfrak{b}}{\partial x}^\intercal g\big)\, u^*$, i.e. $\big(\frac{\partial \mathfrak{b}}{\partial x}^\intercal g\big)\, u > \big(\frac{\partial \mathfrak{b}}{\partial x}^\intercal g\big)\, u^*$. This contradicts (14).

Now comes the main theorem of this letter.

Theorem 1. *Consider an* STL *specification* φ *in* (4b) *is encoded in a* CBNF $\mathfrak{b}$ *as instructed by Sections IV-A and IV-B which is a valid control barrier function as well. If Assumptions 1, 2 and 3 hold, system* (1) *with the control law* (10) *for each time interval* $[t_j, t_{j+1}]$ *behaves in a way that leads to* $x \models \varphi$.

Proof: For the control law of the form (10), but with the second term being the solution of the quadratic programming (13), the proof mirrors that of [3, Theorem 2]. Proposition 1 shows that the second term of (10) in fact coincides with the solution to (13). Therefore, the proof argument in [3, Theorem 2] carries over.

V. Simulation Results

This section showcases how the CBNFs perform using a two-dimensional test case of a point robot navigating a sphere world and needing to satisfy a nontrivial STL specification.

Consider a robot in a 2D sphere world with an outer boundary having radius 1, in which the robot is initially positioned at coordinates $x_0 = (0.9, 0.2)^\intercal$. The workspace contains a static obstacle centered at $(0.5, 0)^\intercal$ with radius 0.2236. Let μ_i be the predicate that represents the i^{th} (spherical) region of interest, and consider an STL task specification constructed based on those μ_i predicates. Table I represents the center position and radius of all those regions of interest.

TABLE I: Geometrical features of the STL regions of interest

Predicate	Center	Radius	Predicate	Center	Radius
μ_1	$(-0.1, 0)^\intercal$	0.3	μ_2	$(-0.4, 0)^\intercal$	0.3
μ_3	$(-0.6, 0.2)^\intercal$	0.3	μ_4	$(-0.35, -0.3)^\intercal$	0.2
μ_5	$(-0.4, -0.6)^\intercal$	0.2			

The STL task specification given to the robot to satisfy is:

$$\varphi_1 = (\Box_{[3,7]}\mu_1 \wedge \mu_2) \wedge (\Diamond_{[4,5]}\mu_3) \wedge (\mu_4 U_{[6,10]}\mu_5) \ . \quad (15)$$

Figures 1, 2 and 3 depict snapshots from the robot's trajectory as it is steered based on (10) in an attempt to satisfy the STL specification φ_1 given in (15), and encoded using CBNFs. The trajectory snapshots capture key time instances as determined by the STL temporal operators involved in (15). In addition, figure 4 demonstrates the control input inserted for the robot to complete the STL task (15).

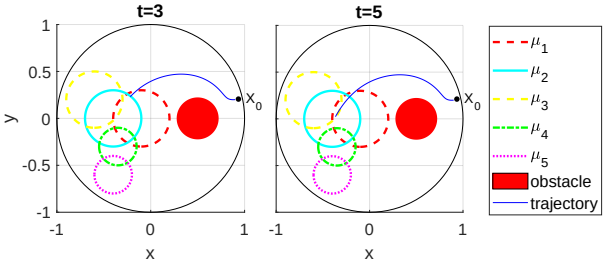


Fig. 1: The robot's path at time $t = 3$ is shown in the left; the dashed red circle marks the zero level set of the predicate function for μ_1, the cyan solid circle marks the zero level set of the predicate function for μ_2, and the solid disk represents the static obstacle. The robot is expected to satisfy the conjunction of μ_1 and μ_2 at $t = 3$. The robot's path at time $t = 5$ is shown in the right; at this time, in addition to satisfying the conjunction of μ_1 and μ_2, the robot is expected to satisfy μ_3 (eventually, for some $t \in [4, 5]$), which is associated with a predicate function that has a zero level set outlined with a yellow dashed line.

Figure 4 indicates the evolution of the x and y components of the input $u(x,t)$ over time, computed based on (10)–(12). The apparent jumps that can be noticed on the graphs of the two control input components are due to the application of the deletion mechanism [3] that simplifies the STL formulae when the temporal relevance of some predicates expires.

For the above simulation example, we discritized time interval $[0, 10]$ into 1000 time steps. Running the existing method [3] that requires quadratic programming solution at

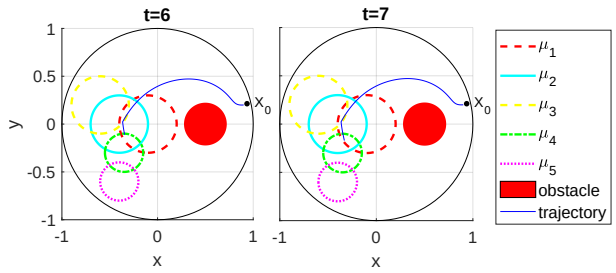


Fig. 2: The robot's path at time $t = 6$ (left) and $t = 7$ (right); at both these times, in addition to satisfying the conjunction of μ_1 and μ_2, the robot needs to satisfy the leftmost predicate in the Until operator, μ_4, the predicate function of which is zero where the dashed-dotted green circle appears. Predicate μ_5 on the right of the Until operator and marked on the workspace with a dotted purple circle does not need to be satisfied just yet.

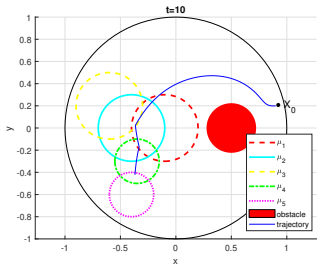
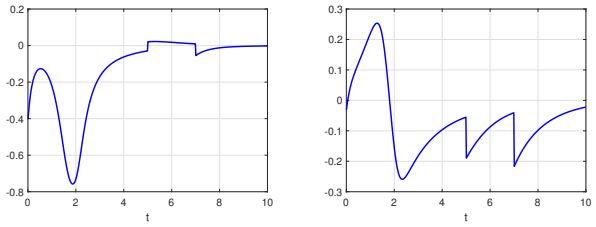


Fig. 3: The robot's path at time $t = 10$; at this time, no other predicate needs to be satisfied but μ_5. Indeed, the robot has reached the boundary of the zero level set of the predicate function for μ_5, marked with the dotted purple circle.

every steps for the above example takes 4.6562 seconds on a system with Intel Core i5 (3.30GHz) processor and 16GB of memory using Wolfram Mathematica. With the presented CBNF method, running the above simulation example on the same system takes only 2.6250 seconds, which shows an improvement of 43.6%.



(a) u_x (b) u_y

Fig. 4: Control input $u = (u_x, u_y)^\intercal$ as computed by (10) for the STL specification φ_1 expressed in (15).

VI. Conclusion

Navigation functions, a key tool for feedback-based robot motion planning and control, can play a key role in the construction of barrier functions that are designed to enforce STL specifications. Their introduction into this framework for the first time with this paper allows for significant computational benefits in the implementation of STL-based control designs, as it obviates the need for solving a quadratic programming problem *in every iteration* of the control loop for the determination of the robot control input. Due to the way that STL predicates are composed in this paper, the navigation function properties of individual component barrier functions are unfortunately not preserved; however, this limitation seems to be circumvented through nonsmooth compositional constructions, which will be reported in a forthcoming paper.

References

[1] 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.

[2] A. Zehfroosh and H. G. Tanner, "Reactive motion planning for temporal logic tasks without workspace discretization," in *IEEE American Control Conference*, 2019, pp. 4872–4877.

[3] 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.

[4] 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 *53rd IEEE Conference on Decision and Control*, 2014, pp. 81–87.

[5] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *53rd Annual Allerton Conference on Communication, Control, and Computing*, 2015, pp. 772–779.

[6] Z. Liu, J. Dai, B. Wu, and H. Lin, "Communication-aware motion planning for multi-agent systems from signal temporal logic specifications," in *IEEE American Control Conference*, 2017, pp. 2516–2521.

[7] 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.

[8] A. M. Jones, K. Leahy, C. Vasile, S. Sadraddini, Z. Serlin, R. Tron, and C. Belta, "ScRATCHS: Scalable and robust algorithms for task-based coordination from high-level specifications," in *International Symposium on Robotics Research*, 2019, pp. 1–16.

[9] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *18th European Control Conference*, 2019, pp. 3420–3431.

[10] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for multi-agent systems under conflicting local signal temporal logic tasks," *IEEE Control Systems Letters*, vol. 3, no. 3, pp. 757–762, 2019.

[11] ——, "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.

[12] ——, "Feedback control strategies for multi-agent systems under a fragment of signal temporal logic tasks," *Automatica*, vol. 106, pp. 284–293, 2019.

[13] ——, "Funnel control for fully actuated systems under a fragment of signal temporal logic specifications," *Nonlinear Analysis: Hybrid Systems*, vol. 39, p. 100973, 2021.

[14] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[15] C. Li and H. G. Tanner, "Navigation functions with time-varying destination manifolds in star worlds," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 35–48, 2018.

[16] J. Sun and H. G. Tanner, "Constrained decision-making for low-count radiation detection by mobile sensors," *Autonomous Robots*, vol. 39, no. 4, pp. 519–536, 2015.

[17] C. Chen, C. Li, and H. G. Tanner, "Navigation functions with non-point destinations and moving obstacles," in *Proceedings of the IEEE American Control Conference*, 2020, pp. 2532–2537.

[18] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[19] I. Yadav and H. G. Tanner, "Exact decentralized receding horizon planning for multiple aerial vehicles," in *Proceedings of the IEEE Conference on Decision and Control*, 2021, (to appear).

[20] W. M. Boothby, *An Introduction to Differentiable Manifolds and Riemannian Geometry*, 2nd ed. Academic Press Inc., 1986.