Continuous-time control synthesis under nested signal temporal logic specifications

Pian Yu, Xiao Tan, and Dimos V. Dimarogonas

***Abstract*—In this work, we propose a novel approach for the continuous-time control synthesis of nonlinear systems under nested signal temporal logic (STL) specifications. While the majority of existing literature focuses on control synthesis for STL specifications without nested temporal operators, addressing nested temporal operators poses a notably more challenging scenario and requires new theoretical advancements. Our approach hinges on the concepts of signal temporal logic tree (sTLT) and control barrier function (CBF). Specifically, we detail the construction of an sTLT from a given STL formula and a continuous-time dynamical system, the sTLT semantics (i.e., satisfaction condition), and the equivalence or under-approximation relation between sTLT and STL. Leveraging the fact that the satisfaction condition of an sTLT is essentially keeping the state within certain sets during certain time intervals, it provides explicit guidelines for the CBF design. The resulting controller is obtained through the utilization of an online CBF-based program coupled with an event-triggered scheme for online updating the activation time interval of each CBF, with which the correctness of the system behavior can be established by construction. We demonstrate the efficacy of the proposed method for single-integrator and unicycle models under nested STL formulas.**

***Index Terms*—Signal temporal logic, control barrier function, control synthesis, continuous-time nonlinear systems**

I. Introduction

High level formal languages, originated from computer science for the specification and verification of computer programs [1], have attracted increasing attention to a wider audience over the last decades, ranging from biological networks [2], [3] to robotics [4], [5]. Temporal logics, such as Linear Temporal Logic (LTL) and Signal Temporal Logic (STL) [6], provide a rigorous, mathematical language characterizing the expected behaviors of the systems. LTL focuses on the Boolean satisfaction of events over a discrete-time state series. As a comparison, STL allows for characterizing system properties over dense time, and thus more favorable for continuous-time dynamical systems, e.g., robotic and cyber-physical system applications [7], [8].

Designing control strategies for systems to satisfy high level specifications is known as the control synthesis problem. For LTL specifications, the classic automaton-based control synthesis scheme has been well-studied [9], [10] for hybrid and discrete-time dynamical systems. In recent years, several different control synthesis schemes are proposed for STL specifications. One popular approach is to evaluate the satisfaction of the STL specification over the sampled time instants, encode it as a mixed-integer program (MIP), and then solve it in a model predictive control framework [11]–[13]. However, the exponential computational complexity with respect to the number of integer variables makes this approach difficult to be applied to STL formulas with long time horizons even for small dimensional dynamical systems. To address the exponential complexity of integer-based optimization, recent work proposes to smoothly approximate the robustness metric of STL, and then sequential quadratic programming [14] or convex-concave programming [15] is proposed to find a solution. In [16], STL formulas are interpreted over stochastic processes and the STL synthesis is reformulated as a probabilistic inference problem. Nevertheless, all these results are restricted to discrete-time systems.

There are some endeavours in recent years on the continuous-time control synthesis problem for STL specifications, including, to name a few, the control barrier function-based [17]–[19], automaton-based [20], [21], heuristic-based [22], sampling-based [23], and learning-based [24]–[26] methods. Different from the discrete-time control synthesis methods, most of the aforementioned approaches only can handle STL formulas with non-nested temporal operators (we will refer to these formulas as non-nested STL for simplicity in the following). To be more specific, the CBF-based method [17] deals with a fragment of non-nested STL formulas and linear predicates. The recent work in [19] considers a richer STL fragment and provides heuristics on the decomposition and the ordering of sub-tasks which are then used to construct CBFs. In [21], the sampling-based automaton-guided control synthesis approach allows the consideration of nonlinear predicates, yet it is still restricted to non-nested STL formulas. In [20], a fragment of signal interval temporal logic formulas is considered for the automaton-based control synthesis. Moreover, the timed abstraction of the dynamical system is needed, which is based on the assumption of existing feedback control laws. The case of STL formulas with nested temporal operators is substantially more challenging and requires new theoretical advancements. To the best of our knowledge, the continuous-time control synthesis for STL specifications with nested temporal operators is still an open problem.

In this work, we aim to develop an efficient control synthesis approach for continuous-time dynamical systems under STL specifications with nested temporal operators, e.g., $\mathsf{G}_{[a_1,b_1]}\mathsf{F}_{[a_2,b_2]}\mu$, $\mathsf{F}_{[a_1,b_1]}\mathsf{G}_{[a_2,b_2]}\mu$, $\mathsf{F}_{[a_1,b_1]}(\mu_1\mathsf{U}_{[a_2,b_2]}(\mathsf{F}_{[a_3,b_3]}\mu_2))$. Compared to previous CBF-based control synthesis works [17]–[19], we provide

This work was supported in part by the Swedish Research Council (VR), the Swedish Foundation for Strategic Research (SSF), the Knut and Alice Wallenberg Foundation (KAW), the ERC CoG LEAFHOUND project, and the EU CANOPIES project.

Pian Yu is currently at the Department of Computer Science, University of Oxford, United Kingdom. She was at the KTH Royal Institute of Technology when this work was conducted. Xiao Tan and Dimos V. Dimarogonas are with School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, 10044 Stockholm, Sweden. pian.yu@cs.ox.ac.uk, xiaotan,dimos@kth.se

a tangible tool, coined as the *signal temporal logic tree (sTLT)*, that explicitly transforms the satisfaction of an STL formula to a series of set invariance conditions, which naturally guides the design of corresponding CBFs. The main contributions of this work are summarized as follows. 1) We introduce a notion of sTLT, detail its construction from a given STL formula, its semantics (i.e., satisfaction condition), and establish equivalence or under-approximation relation between sTLT and STL. 2) We show how to design CBFs and online update their activation time intervals under the guidance of the sTLT. The control synthesis scheme is given by an online CBF-based program. 3) We deduce the correctness of the system behavior under certain assumptions.

The remainder of this paper is organized as follows. In Sec. II, we give some technical preliminaries and introduce the continuous-time control synthesis problem. In Sec. III, the notion of sTLT is introduced as well as its semantics. Then, we derive the equavilence or under-approximation relation between an STL formula and its constructed sTLT. Finally, we show how to design the CBFs, online update their activation time intervals, and the overall control synthesis scheme. Case studies with single integrator and unicycle dynamics are presented in Sec. IV. The work is then concluded in Sec. V.

II. Preliminaries and Problem Formulation

Notation. Let $\mathbb{R} := (-\infty, \infty)$, $\mathbb{R}_{\geq 0} := [0, \infty)$, and $\mathbb{N} := \{0, 1, 2, \dots\}$. Denote $\mathbb{R}^n$ as the n dimensional real vector space, $\mathbb{R}^{n\times m}$ as the $n \times m$ real matrix space. Throughout this paper, vectors are denoted in italics, $x \in \mathbb{R}^n$, and boldface $\boldsymbol{x}$ is used for continuous-time signals. Let $\|x\|$ and $\|A\|$ be the Euclidean norm of vector x and matrix A. Given a set $S \subset \mathbb{R}^n$, $\overline{S}$ denotes its complement and ∂S denotes its boundary. Given a point $x \in \mathbb{R}^n$ and a set $S \subset \mathbb{R}^n$, the distance function is defined as $\texttt{dist}(x, S) := \inf_{y\in S} \|x-y\|$. The signed distance function $\texttt{sdist}(x, S)$ is defined as

$$\texttt{sdist}(x, S) = \begin{cases} -\texttt{dist}(x, \overline{S}), & \text{if } x \in S, \\ \texttt{dist}(x, S), & \text{if } x \notin S. \end{cases}$$

Consider a continuous-time dynamical system of the form

$$\Sigma : \dot{x} = f(x, u), \tag{1}$$

where $x \in \mathbb{R}^n$ and $u \in U \subseteq \mathbb{R}^m$ are respectively the state and input of the system, the function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is locally Lipschitz continuous in x and u.

Let $\mathcal{U}$ be the set of all measurable functions that take their values in U and are defined on $\mathbb{R}_{\geq 0}$. A curve $\boldsymbol{x} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is said to be a trajectory of (1) if there exists an input signal $\boldsymbol{u} \in \mathcal{U}$ satisfying (1) for almost all $t \in \mathbb{R}_{\geq 0}$. We use $\boldsymbol{x}^{\boldsymbol{u}}_{x_0}(t)$ to denote the trajectory point reached at time t under the input signal $\boldsymbol{u}$ from the initial state x_0.

A. Signal temporal logic

Signal temporal logic (STL) [6] is a predicate logic based on continuous-time signals. When $\boldsymbol{x} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is considered, the predicate μ at time t is obtained after evaluation of a predicate function $g_\mu : \mathbb{R}^n \to \mathbb{R}$ as follows

$$\mu := \begin{cases} \top, & \text{if} \quad g_\mu(\boldsymbol{x}(t)) \geq 0 \\ \bot, & \text{if} \quad g_\mu(\boldsymbol{x}(t)) < 0. \end{cases}$$

In [13], it was shown that each STL formula has an equivalent STL formula in positive normal form (PNF), *i.e.,* negations only occur adjacent to predicates. The syntax of the PNF STL is given by

$$\begin{aligned} \varphi ::= \top \mid \mu \mid \neg\mu \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \\ \mid \varphi_1 \mathsf{U}_{[a,b]} \varphi_2 \mid \mathsf{F}_{[a,b]} \varphi \mid \mathsf{G}_{[a,b]} \varphi, \end{aligned} \tag{2}$$

where $\varphi, \varphi_1, \varphi_2$ are STL formulas and $[a, b], 0 \leq a \leq b < \infty$, denotes a time interval. Here, $\wedge$ and $\vee$ are logic operators "conjunction" and "disjunction", $\mathsf{U}_{[a,b]}$, $\mathsf{F}_{[a,b]}$, and $\mathsf{G}_{[a,b]}$ are temporal operators "until", "eventually", and "always", respectively.

Definition 1 (STL semantics [27])**.** *The validity of an STL formula φ with respect to a continuous-time signal $\boldsymbol{x}$ evaluated at time t, is defined inductively as follows:*

$$\begin{aligned} (\boldsymbol{x}, t) \vDash \mu &\Leftrightarrow g_\mu(\boldsymbol{x}(t)) \geq 0, \\ (\boldsymbol{x}, t) \vDash \neg\mu &\Leftrightarrow \neg((\boldsymbol{x}, t) \vDash \mu), \\ (\boldsymbol{x}, t) \vDash \varphi_1 \wedge \varphi_2 &\Leftrightarrow (\boldsymbol{x}, t) \vDash \varphi_1 \wedge (\boldsymbol{x}, t) \vDash \varphi_2, \\ (\boldsymbol{x}, t) \vDash \varphi_1 \vee \varphi_2 &\Leftrightarrow (\boldsymbol{x}, t) \vDash \varphi_1 \vee (\boldsymbol{x}, t) \vDash \varphi_2, \\ (\boldsymbol{x}, t) \vDash \varphi_1 \mathsf{U}_{[a,b]} \varphi_2 &\Leftrightarrow \exists t' \in [t + a, t + b] \text{ s.t.} \\ &\quad (\boldsymbol{x}, t') \vDash \varphi_2 \wedge \\ &\quad \forall t'' \in [t, t'], (\boldsymbol{x}, t'') \vDash \varphi_1, \\ (\boldsymbol{x}, t) \vDash \mathsf{F}_{[a,b]} \varphi &\Leftrightarrow \exists t' \in [t + a, t + b] \text{ s.t.} \\ &\quad (\boldsymbol{x}, t') \vDash \varphi, \\ (\boldsymbol{x}, t) \vDash \mathsf{G}_{[a,b]} \varphi &\Leftrightarrow \forall t' \in [t + a, t + b] \text{ s.t.} \\ &\quad (\boldsymbol{x}, t') \vDash \varphi. \end{aligned}$$

Definition 2. *Consider the dynamical system Σ in (1) and the STL formula φ in (2). We say φ* is satisfiable from the initial state x_0 *if there exists a control signal $\boldsymbol{u} \in \mathcal{U}$ such that $(\boldsymbol{x}^{\boldsymbol{u}}_{x_0}, 0) \vDash \varphi$.*

Given an STL formula φ, the set of initial states from which φ is satisfiable is denoted by

$$\mathbb{S}_\varphi := \{x_0 \in \mathbb{R}^n | \varphi \text{ is satisfiable from } x_0\}. \tag{3}$$

For simplicity, we will refer to $\mathbb{S}_\varphi$ as *the satisfying set* for φ in the following. Please be aware that the computation of the set $\mathbb{S}_\varphi$ is tailored to the dynamical system Σ under consideration. Here we omit it for notation simplicity.

B. Reachability operators

In this section, we define two reachability operators.

Definition 3. *Consider the system (1), a set $\mathcal{S} \subseteq \mathbb{R}^n$, and a time interval $[a, b]$. The* maximal reachable set $\mathcal{R}^M(\mathcal{S}, [a, b])$ *is defined as*

$$\begin{aligned} \mathcal{R}^M(\mathcal{S}, [a, b]) = \Big\{ x_0 \in \mathbb{R}^n : \exists \boldsymbol{u} \in \mathcal{U}, \exists t' \in [a, b], \\ \text{s.t. } \boldsymbol{x}^{\boldsymbol{u}}_{x_0}(t') \in \mathcal{S} \Big\}. \end{aligned}$$

Definition 4. *Consider the system (1), the set $\mathcal{S} \subseteq \mathbb{R}^n$, and a time interval $[a, b]$. The* minimal reachable set $\mathcal{R}^m(\mathcal{S}, [a, b])$ *is defined as*

$$\mathcal{R}^m(\mathcal{S}, [a, b]) = \Big\{x_0 \in \mathbb{R}^n : \forall \boldsymbol{u} \in \mathcal{U}, \exists t' \in [a, b], \text{ s.t. } \boldsymbol{x}^{\boldsymbol{u}}_{x_0}(t') \in \mathcal{S}\Big\}.$$

The set $\mathcal{R}^M(\mathcal{S}, [a, b])$ collects all states in $\mathbb{R}^n$ from which there exists an input signal $\boldsymbol{u} \in \mathcal{U}$ that drives the system to target set $\mathcal{S}$ at some time instant $t' \in [a, b]$. The set $\mathcal{R}^m(\mathcal{S}, [a, b])$ collects all states in $\mathbb{R}^n$ from which no matter what input signal $\boldsymbol{u} \in \mathcal{U}$ is applied, the system can reach the target set $\mathcal{S}$ at some time instant $t' \in [a, b]$.

Let $\mathcal{S}$ be represented by the zero superlevel set of a continuous function: $\mathcal{S} = \{x \in \mathbb{R}^n : h_{\mathcal{S}}(x) \geq 0\}$. Similarly, let $\mathcal{R}^M(\mathcal{S}, [a, b])$ and $\mathcal{R}^m(\mathcal{S}, [a, b])$ be represented by the zero superlevel set of some continuous functions, i.e.,

$$\mathcal{R}^M(\mathcal{S}, [a, b]) := \{x : h_{\mathcal{R}^M(\mathcal{S},[a,b])}(x) \geq 0\},$$
$$\mathcal{R}^m(\mathcal{S}, [a, b]) := \{x : h_{\mathcal{R}^m(\mathcal{S},[a,b])}(x) \geq 0\}.$$

As shown in [28], the calculation of maximal and minimal reachable sets can be casted as an optimal control problem, given below:

$$h_{\mathcal{R}^M(\mathcal{S},[a,b])}(x) = \max_{\boldsymbol{u}\in\mathcal{U}} \max_{s\in[a,b]} h_{\mathcal{S}}(\boldsymbol{x}^{\boldsymbol{u}}_x(s)),$$
$$h_{\mathcal{R}^m(\mathcal{S},[a,b])}(x) = \min_{\boldsymbol{u}\in\mathcal{U}} \max_{s\in[a,b]} h_{\mathcal{S}}(\boldsymbol{x}^{\boldsymbol{u}}_x(s)).$$

In the following, relations are established between the STL temporal operators $\mathsf{F}_{[a,b]}$ and $\mathsf{G}_{[a,b]}$ and the maximal/minimal reachable sets.

Lemma 1 ([28])**.** *Given the system (1) and the STL predicate μ_1, one has*

i) $\mathbb{S}_{\mathsf{F}_{[a,b]}\mu_1} = \overline{\mathcal{R}^M(\mathbb{S}_{\mu_1}, [a, b])}$, *and*

ii) $\mathbb{S}_{\mathsf{G}_{[a,b]}\mu_1} = \overline{\mathcal{R}^m(\overline{\mathbb{S}_{\mu_1}}, [a, b])}$,

where $\mathbb{S}_{\mathsf{F}_{[a,b]}\mu_1}$ and $\mathbb{S}_{\mathsf{G}_{[a,b]}\mu_1}$ are the satisfying sets for $\mathsf{F}_{[a,b]}\mu_1$ and $\mathsf{G}_{[a,b]}\mu_1$, respectively.

Definition 5. *Given any STL formula φ, let its satisfying set $\mathbb{S}_\varphi$ in* (3) *be represented by the zero superlevel set of a function $h_{\mathbb{S}_\varphi} : \mathbb{R}^n \to \mathbb{R}$, i.e., $\mathbb{S}_\varphi = \{x \in \mathbb{R}^n : h_{\mathbb{S}_\varphi}(x) \geq 0\}$. We refer to such function $h_{\mathbb{S}_\varphi}$ as a* value function *associated with the STL formula φ.*

Note that given a predicate μ, g_μ is a value function associated with μ. In the general case, we can use the signed distance function to denote a value function, i.e., $h_{\mathbb{S}_\varphi}(x) = -\texttt{sdist}(x, \mathbb{S}_\varphi)$.

C. Time-varying control barrier functions

Define a differentiable function $\mathfrak{b} : X \times [t_0, t_1] \to \mathbb{R}$ and the associated set

$$\mathcal{C}(t) := \{x \in X | \mathfrak{b}(x, t) \geq 0\}. \tag{4}$$

Then, we have the following definition.

Definition 6 (CBF [29])**.** *A function $\mathfrak{b} : X \times [t_0, t_1] \to \mathbb{R}$ is called a* valid control barrier function (vCBF) *for (1) if there exists a locally Lipschitz continuous class $\mathcal{K}$ function α such that, for all $(x, t) \in \mathcal{C}(t) \times [t_0, t_1]$,*

$$\sup_{u\in U} \left\{ \frac{\partial \mathfrak{b}(x,t)}{\partial x} f(x, u) + \frac{\partial \mathfrak{b}(x,t)}{\partial t} \right\} \geq -\alpha(\mathfrak{b}(x, t)). \tag{5}$$

If $x_0 \in \mathcal{C}(t_0)$ and $\mathfrak{b}(x, t)$ is a vCBF, then any locally Lipschitz control $\boldsymbol{u}$ satisfying (5) guarantees $\boldsymbol{x}^{\boldsymbol{u}}_{x_0}(t) \in \mathcal{C}(t)$ for all $t \in [t_0, t_1]$. This can be shown by, for example, invoking the Comparison Lemma [30].

D. Problem formulation

Before moving on, we first introduce the notion of nested STL formulas.

Definition 7 (Nested STL formula)**.** *We call an STL formula φ* nested *if it can be written in one of the following forms:*

$$\varphi = \mathsf{F}_{[a,b]}\varphi_1, \tag{6}$$
$$\varphi = \mathsf{G}_{[a,b]}\varphi_1, \tag{7}$$
$$\varphi = \varphi_1 \mathsf{U}_{[a,b]}\varphi_2, \tag{8}$$

where φ_1 in (6)-(7) and at least one of φ_1 and φ_2 in (8) include temporal operators. In addition, φ_1 in (6)-(7) and φ_1, φ_2 in (8) are called the argument(s) *of the STL formula φ.*

Examples of nested STL formulas include $\mathsf{F}_{[a_1,b_1]}\mathsf{G}_{[a_2,b_2]}\mu$, $\mathsf{G}_{[a_1,b_1]}\mathsf{F}_{[a_2,b_2]}\mu$, and $\mu_1 \mathsf{U}_{[a_1,b_1]}(\mathsf{G}_{[a_2,b_2]}\mu_2 \wedge \mathsf{F}_{[a_3,b_3]}\mu_3)$, etc.

In [17], [18], continuous-time control-affine system of the form

$$\dot{x} = f(x) + g(x)u \tag{9}$$

is considered, and appropriate CBFs are designed for a fragment of non-nested STL formulas, which we briefly recap here:

- For $\mathsf{G}_{[a,b]}\mu_1$, select $\mathfrak{b}(x, t)$ s.t. $\mathfrak{b}(x, t') \leq g_{\mu_1}(x)$ for all $t' \in [a, b]$,
- For $\mathsf{F}_{[a,b]}\mu_1$, select $\mathfrak{b}(x, t)$ s.t. $\mathfrak{b}(x, t') \leq g_{\mu_1}(x)$ for some $t' \in [a, b]$.
- For $\mu_1 \mathsf{U}_{[a,b]}\mu_2$, it is encoded as $\mathsf{G}_{[0,b]}\mu_1 \wedge \mathsf{F}_{[a,b]}\mu_2$,

where g_{μ_1} and g_{μ_2} are the predicate functions of μ_1 and μ_2, respectively. Once an vCBF is obtained (the explicit CBF construction is investigated in [18]) for the non-nested STL formula, then the control strategy for (9) is given by solving a quadratic program (QP)

$$\min_{u\in U} \quad u^T Q u$$
$$\text{s.t. } \frac{\partial b(x,t)}{\partial x}(f(x) + g(x)u) + \frac{\partial b(x,t)}{\partial t} \geq -\alpha(b(x, t)). \tag{10}$$

In this work, we consider the continuous-time control synthesis for nested STL formulas as per Definition 7. Formally, the problem is stated as follows.

Problem 1. *Consider the dynamical system in (1) and a nested STL formula φ. Derive a continuous-time control strategy $\boldsymbol{u}$ such that the resulting trajectory $\boldsymbol{x}$ of (1) with initial state x_0 satisfies φ, i.e., $(\boldsymbol{x}^{\boldsymbol{u}}_{x_0}, 0) \models \varphi$.*

III. Solving the Control Synthesis Problem

In this work, we aim to formulate the continuous-time control synthesis problem for a nested STL specification φ as a CBF-based program as in [17]. Here, the difficulty is: how to encode the task satisfaction constraint (i.e., $(\boldsymbol{x}_{x_0}^{\boldsymbol{u}}, 0) \models \varphi$) as a set of constraints on the system input u when φ is nested? Appropriate CBFs have been proposed in [17] for control-affine systems under non-nested STL formula φ, e.g., $\varphi = \mathsf{F}_{[a,b]}\mu$. However, when the STL formula φ is nested, extending the CBF design methodology in [17] to nested STL formulas is nontrivial.

To tackle this problem, in this work, we propose the notion of sTLT. This tree structure serves as a tool for guiding the design of CBFs for nested STL formulas.

This section is structured as follows. First, we introduce the notion of sTLT and its construction in Section III. A. Then we define sTLT semantics in Section III.B. The equivalence or under-approximation relation between STL and sTLT is discussed in Section III. C. Then, we explain the design of the CBFs based on the sTLT in Section III. D. In Section III. E, we show the overall algorithm. Finally in Section III. F, the computational complexity of the overall approach is discussed.

A. sTLT and its construction

An sTLT refers to a tree with linked set nodes and operator nodes. The formal definition is given as follows.

Definition 8 (sTLT)**.** *An* sTLT *is a tree for which the next holds:*

- *each node is either a* set *node that is a subset of* $\mathbb{R}^n$ *or an* operator *node that belongs to* $\{\wedge, \vee, \mathsf{U}_{[a,b]}, \mathsf{F}_{[a,b]}, \mathsf{G}_{[a,b]}\}$*;*
- *the root node and the leaf nodes are* set *nodes;*
- *if a* set *node is not a leaf node, its unique child is an* operator *node;*
- *the children of any* operator *node are* set *nodes.*

The sTLT is motivated by the notion of TLT defined in [31] for LTL formulas. Although graphically similar, the sTLT construction and its satisfaction condition are substantially different from TLT in [31]. We will provide additional clarification regarding the differences later in Remark 1.

Construct an sTLT from an STL formula φ: Before presenting the construction procedure of such an sTLT from a given STL formula φ and a continuous-time dynamical system Σ, we give the following definition.

Definition 9 (Desired form)**.** *Given an STL formula φ in Definition 2, we say φ* is in desired form *if i) it contains no "until" operators and ii) the argument of every "always" operator contains no "disjunction" operator.*

Next we detail the construction of sTLT from an STL formula φ using the reachability operators $\mathcal{R}^M$ and $\mathcal{R}^m$, which can be completed in 3 steps.

Step 1: Rewrite the STL formula φ into the desired form $\hat{\varphi}$ as per Definition 9. That is, i) if φ contains "until" operator, e.g., $\varphi = \varphi_1 \mathsf{U}_{[a,b]} \varphi_2$, it is encoded as $\hat{\varphi} = \mathsf{G}_{[0,b]}\varphi_1 \wedge \mathsf{F}_{[a,b]}\varphi_2$ and ii) if the argument of a temporal operator contains a "disjunction" operator, e.g., $\varphi = \Theta_{[a,b]}(\varphi_1 \vee \varphi_2)$, it is encoded as $\hat{\varphi} = \Theta_{[a,b]}\varphi_1 \vee \Theta_{[a,b]}\varphi_2$, where $\Theta \in \{\mathsf{G}, \mathsf{F}\}$. After this step, one has that $\hat{\varphi}$ contains no "until" operator and the "disjunction" operator, if it exists, appears in the form of $\hat{\varphi} = \varphi_1 \vee \varphi_2 \vee \ldots \vee \varphi_N$, and $\varphi_i, i = 1, 2, \ldots, N$, contain no "disjunction" operator. We call the fragment of STL formulas $\hat{\varphi}$, identified by Definition 9, *desired form*. This is because we will later observe that the constructed sTLT $\mathcal{T}_{\hat{\varphi}}$ is equivalent to $\hat{\varphi}$ in the sense that every trajectory that satisfies the sTLT $\mathcal{T}_{\hat{\varphi}}$ also satisfies the STL formula $\hat{\varphi}$, and conversely.

Step 2: For each predicate μ or its negation $\neg\mu$, construct the sTLT with only a single set node $\mathbb{X}_\mu = \mathbb{S}_\mu = \{x : g_\mu(x) \geq 0\}$ or $\mathbb{X}_{\neg\mu} = \mathbb{S}_{\neg\mu} = \{x : -g_\mu(x) \geq 0\}$. The sTLT of $\top$ or $\bot$ has only a single set node, which is $\mathbb{R}^n$ or $\emptyset$, respectively.

Step 3: Construct the sTLT $\mathcal{T}_{\hat{\varphi}}$ inductively. More specifically, for given STL formulas φ_1 and φ_2 and their corresponding constructed sTLTs $\mathcal{T}_{\varphi_1}, \mathcal{T}_{\varphi_2}$, the sTLT from a) $\varphi_1 \wedge \varphi_2$, b) $\varphi_1 \vee \varphi_2$, c) $\mathsf{F}_{[a,b]}\varphi_1$, and d) $\mathsf{G}_{[a,b]}\varphi_1$ can be constructed following the rules detailed below. Denote by $\mathbb{X}_{\varphi_1} := \{x : h_{\mathbb{X}_{\varphi_1}} \geq 0\}$ and $\mathbb{X}_{\varphi_2} := \{x : h_{\mathbb{X}_{\varphi_2}} \geq 0\}$ the root nodes of $\mathcal{T}_{\varphi_1}$ and $\mathcal{T}_{\varphi_2}$, respectively.

Case a): Boolean operator $\wedge$. The sTLT $\mathcal{T}_{\varphi_1 \wedge \varphi_2}$ can be constructed by connecting $\mathbb{X}_{\varphi_1}$ and $\mathbb{X}_{\varphi_2}$ through the operator node $\wedge$ and taking

$$\mathbb{X}_{\varphi_1 \wedge \varphi_2} := \{x : (h_{\mathbb{X}_{\varphi_1}} \geq 0) \wedge (h_{\mathbb{X}_{\varphi_2}} \geq 0)\}$$

to be the root node. An illustrative diagram for $\varphi_1 \wedge \varphi_2$ is given in Fig. 1(a).

Case b): Boolean operator $\vee$. The sTLT $\mathcal{T}_{\varphi_1 \vee \varphi_2}$ can be constructed by connecting $\mathbb{X}_{\varphi_1}$ and $\mathbb{X}_{\varphi_2}$ through the operator node $\vee$ and taking

$$\mathbb{X}_{\varphi_1 \vee \varphi_2} := \{x : (h_{\mathbb{X}_{\varphi_1}} \geq 0) \vee (h_{\mathbb{X}_{\varphi_2}} \geq 0)\}$$

to be the root node. An illustrative diagram for $\varphi_1 \vee \varphi_2$ is given in Fig. 1(b).

Case c): Eventually operator $\mathsf{F}_{[a,b]}$. The sTLT $\mathcal{T}_{\mathsf{F}_{[a,b]}\varphi_1}$ can be constructed by connecting $\mathbb{X}_{\varphi_1}$ through the operator $\mathsf{F}_{[a,b]}$ and making the set $\mathcal{R}^M(\mathbb{X}_{\varphi_1}, \mathbb{R}^n, [a,b])$ the root node. An illustrative diagram for $\mathsf{F}_{[a,b]}\varphi_1$ is given in Fig. 1(c).

Case d): Always operator $\mathsf{G}_{[a,b]}$. The sTLT $\mathcal{T}_{\mathsf{G}_{[a,b]}\varphi_1}$ can be constructed by connecting $\mathbb{X}_{\varphi_1}$ through the operator $\mathsf{G}_{[a,b]}$ and making the set $\overline{\mathcal{R}^m(\overline{\mathbb{X}_{\varphi_1}}, [a,b])}$ the root node. An illustrative diagram for $\mathsf{G}_{[a,b]}\varphi_1$ is given in Fig. 1(d).

Thus we complete the construction of an sTLT from a STL formula φ. In what follows, if not stated otherwise, we will use $\hat{\varphi}$ as the desired form of φ obtained from Step 1 and $\mathcal{T}_{\hat{\varphi}}$ as the constructed sTLT for brevity.

Let us use the following example to show how to construct an sTLT from a nested STL formula.

Example 1. *Consider the nested STL formula $\varphi = \mathsf{F}_{[0,15]}(\mathsf{G}_{[2,10]}\mu_1 \vee \mu_2 \mathsf{U}_{[5,10]}\mu_3)$, where $\mu_i, i = \{1,2,3\}$ are predicates. Following Step 1, we can rewrite φ into the desired form $\hat{\varphi} = \mathsf{F}_{[0,15]}\mathsf{G}_{[2,10]}\mu_1 \vee \mathsf{F}_{[0,15]}(\mathsf{G}_{[0,10]}\mu_2 \wedge \mathsf{F}_{[5,10]}\mu_3)$. The constructed sTLT $\mathcal{T}_{\hat{\varphi}}$ is plotted in Fig. 2. Recall that the sTLT is constructed in a bottom-up manner, i.e., we first construct the leaf nodes corresponding to the three predicates, i.e., $\mathbb{X}_5 = \mathbb{S}_{\mu_1}$, $\mathbb{X}_8 = \mathbb{S}_{\mu_2}$, $\mathbb{X}_9 = \mathbb{S}_{\mu_3}$, and then build upon them*

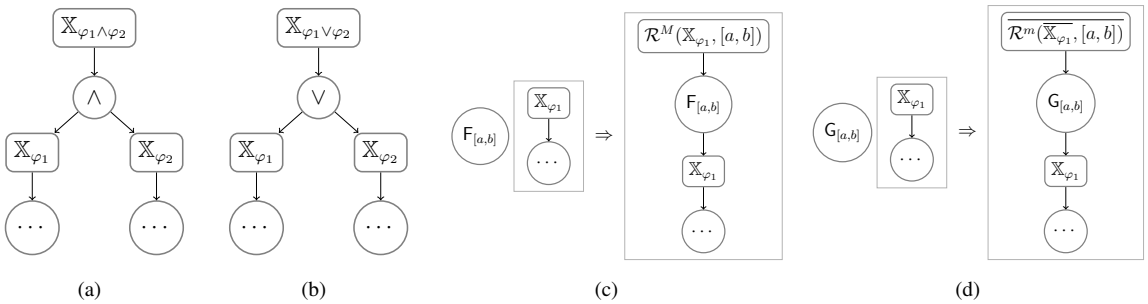

Fig. 1. The sTLT construction for: (a) $\varphi_1 \wedge \varphi_2$; (b) $\varphi_1 \vee \varphi_2$; (c) $\mathsf{F}_{[a,b]}\varphi_1$; (d) $\mathsf{G}_{[a,b]}\varphi_1$. The circles denote the operator nodes and the rectangles denote the set nodes.

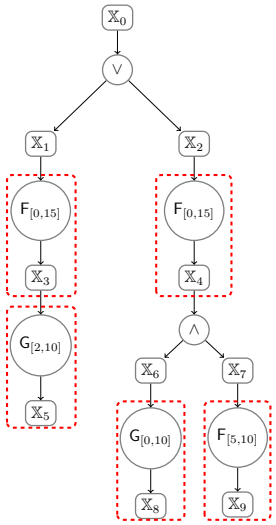

Fig. 2. The sTLT $\mathcal{T}_{\hat{\varphi}}$ for the nested STL formula $\varphi = \mathsf{F}_{[0,15]}(\mathsf{G}_{[2,10]}\mu_1 \vee \mu_2 \mathsf{U}_{[5,10]}\mu_3)$.

one can compute

$$\begin{aligned}
\mathbb{X}_3 &= \overline{\mathcal{R}^m(\overline{\mathbb{X}_5}, [2,10])},\\
\mathbb{X}_1 &= \mathcal{R}^M(\mathbb{X}_3, [0,15]),\\
\mathbb{X}_6 &= \overline{\mathcal{R}^m(\overline{\mathbb{X}_8}, [0,10])},\\
\mathbb{X}_7 &= \mathcal{R}^M(\mathbb{X}_9, [5,10]),\\
\mathbb{X}_4 &= \{x : h_{\mathbb{X}_6}(x) \geq 0 \wedge h_{\mathbb{X}_7}(x) \geq 0\},\\
\mathbb{X}_2 &= \mathcal{R}^M(\mathbb{X}_4, [0,15]),\\
\mathbb{X}_0 &= \{x : h_{\mathbb{X}_1}(x) \geq 0 \vee h_{\mathbb{X}_2}(x) \geq 0\}.
\end{aligned}$$

B. sTLT semantics

Before define the *sTLT semantics*, i.e., the satisfaction relation between a trajectory $\boldsymbol{x}$ and an sTLT $\mathcal{T}$, the definitions of complete path, temporal fragment, and time interval coding for an sTLT $\mathcal{T}$ are needed.

Definition 10 (Complete path)**.** *A* complete path $\boldsymbol{p}$ *of an sTLT is a path that starts from the root node and ends at a leaf node. It can be encoded in the form of* $\boldsymbol{p} = \mathbb{X}_0\Theta_1\mathbb{X}_1\Theta_2 \ldots \Theta_{N_f}\mathbb{X}_{N_f}$*, where* N_f *is the number of operator nodes contained in the complete path,* $\mathbb{X}_i, i \in \{0, 1, \ldots, N_f\}$ *represent set nodes, and* $\Theta_j, \forall j \in \{1, \ldots, N_f\}$ *represent operator nodes.*

Definition 11 (Temporal fragment)**.** *A* temporal fragment *of a complete path is a fragment that starts from one temporal operator node, i.e., the node* $\mathsf{U}_{[a,b]}, \mathsf{F}_{[a,b]}$ *or* $\mathsf{G}_{[a,b]}$*, and ends at its child set node.*

Definition 12 (Time interval coding)**.** *A* time interval coding *of a complete path involves assigning a time interval* $[\underline{t}_i, \bar{t}_i], 0 \leq \underline{t}_i \leq \bar{t}_i$ *to each set node* $\mathbb{X}_i$ *in the complete path.*

Given a time instant $\hat{t}$ and two time intervals $[a_1, b_1], [a_2, b_2]$, define

$$\begin{aligned}
\hat{t} + [a_1, b_1] &:= [\hat{t} + a_1, \hat{t} + b_1],\\
[a_1, b_1] + [a_2, b_2] &:= [a_1 + a_2, b_1 + b_2].
\end{aligned}$$

Now, we further define the satisfaction relation between a trajectory $\boldsymbol{x}$ and a complete path of the sTLT.

Definition 13. *Consider a trajectory* $\boldsymbol{x}$ *and a complete path* $\boldsymbol{p} = \mathbb{X}_0\Theta_1\mathbb{X}_1\Theta_2 \ldots \Theta_{N_f}\mathbb{X}_{N_f}$*. We say* $\boldsymbol{x}$ satisfies $\boldsymbol{p}$ from time t*, denoted by* $(\boldsymbol{x}, t) \cong \boldsymbol{p}$*, if there exists a time interval coding for* $\boldsymbol{p}$ *such that* $\underline{t}_0 = \bar{t}_0 = t$ *and, for* $i = 1, 2, \ldots, N_f$*,*

i) *if* $\Theta_i \in \{\wedge, \vee\}$*, then* $[\underline{t}_i, \bar{t}_i] = [\underline{t}_{i-1}, \bar{t}_{i-1}]$*;*
ii) *if* $\Theta_i \in \{\mathsf{U}_{[a,b]}, \mathsf{F}_{[a,b]}\}$*, then* $\exists t' \in [a, b]$ *s.t.* $[\underline{t}_i, \bar{t}_i] = t' + [\underline{t}_{i-1}, \bar{t}_{i-1}]$*;*
iii) *if* $\Theta_i = \mathsf{G}_{[a,b]}$*, then* $[\underline{t}_i, \bar{t}_i] = [a, b] + [\underline{t}_{i-1}, \bar{t}_{i-1}]$*;*

and, for $i = 0, 1, \ldots, N_f$*,*

iv) $\boldsymbol{x}(t) \in \mathbb{X}_i, \forall t \in [\underline{t}_i, \bar{t}_i]$.

With Definition 13, the sTLT semantics, i.e., the satisfaction relation between a trajectory $\boldsymbol{x}$ and an sTLT, can be defined as follows.

Definition 14 (sTLT semantics)**.** *Consider a trajectory* $\boldsymbol{x}$ *and an sTLT* $\mathcal{T}$*. We say* $\boldsymbol{x}$ satisfies $\mathcal{T}$ from time t*, denoted by* $(\boldsymbol{x}, t) \cong \mathcal{T}$*, if the output of Algorithm 1 is* true.

Algorithm 1 takes as inputs a trajectory $\boldsymbol{x}$ and an sTLT $\mathcal{T}$. The output is true or false. It works as follows. Given the sTLT $\mathcal{T}$, we first remove all its temporal fragments (line 1). When removing a temporal fragment, we reconnect the parent node of the corresponding temporal operator node and the child of the corresponding set node. In this way the resulting compressed tree $\mathcal{T}^c$ contains only Boolean operator nodes and set nodes. For the sTLT $\mathcal{T}_{\hat{\varphi}}$ shown in Fig. 2, the compressed tree $\mathcal{T}^c$ is depicted in Fig. 3. Then for each complete path $\boldsymbol{p}$ of $\mathcal{T}$, if $(\boldsymbol{x}, 0) \cong \boldsymbol{p}$, one sets the corresponding leaf node of $\boldsymbol{p}$ in $\mathcal{T}^c$ (note that $\mathcal{T}^c$ and $\mathcal{T}$ have the same number of leaf nodes)

Algorithm 1 *sTLT Satisfaction*

Input: a trajectory $\boldsymbol{x}$ and an sTLT $\mathcal{T}$.
Return: true or false.

1: $\mathcal{T}^c \leftarrow$ remove all temporal fragments in $\mathcal{T}$,
2: **for** each complete path $\boldsymbol{p}$ of $\mathcal{T}$, **do**
3: **if** $(\boldsymbol{x}, 0) \cong \boldsymbol{p}$ **then**
4: set the corresponding leaf node of $\boldsymbol{p}$ in $\mathcal{T}^c$ to true,
5: **else**
6: set the corresponding leaf node of $\boldsymbol{p}$ in $\mathcal{T}^c$ to false,
7: **end if**
8: **end for**
9: set all the non-leaf set nodes of $\mathcal{T}^c$ to false,
10: $\mathcal{T}^c \leftarrow$ *Backtracking*$(\mathcal{T}^c)$,
11: return the root node of $\mathcal{T}^c$.

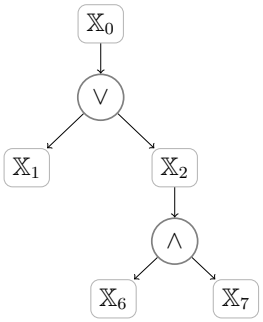

Fig. 3. The compressed tree $\mathcal{T}^c$ for the sTLT $\mathcal{T}_{\hat{\varphi}}$ plotted in Fig. 2.

.

to true. Otherwise, one sets the corresponding leaf node of $\boldsymbol{p}$ in $\mathcal{T}^c$ to false (lines 2-8). After that, we set all the non-leaf set nodes of $\mathcal{T}^c$ to false (line 9) and the resulting tree becomes a Boolean tree (a tree with Boolean operator and Boolean variable nodes). Finally, we backtrack the Boolean tree $\mathcal{T}^c$ using Algorithm *Backtracking*, given in Algorithm 2, and return the root node (lines 10-11).

Algorithm 2 *Backtracking*

Input: a tree $\mathcal{T}^c$ with Boolean operator and Boolean variable nodes.
Return: the updated $\mathcal{T}^c$.

1: **for** each operator node Θ of $\mathcal{T}^c$ through a bottom-up traversal, **do**
2: **if** $\Theta = \wedge$, **then**
3: $\text{PA}(\Theta) \leftarrow \text{PA}(\Theta) \vee (\text{CH}_1(\Theta) \wedge \text{CH}_2(\Theta))$,
4: **else**
5: $\text{PA}(\Theta) \leftarrow \text{PA}(\Theta) \vee (\text{CH}_1(\Theta) \vee \text{CH}_2(\Theta))$,
6: **end if**
7: **end for**

In Algorithm *Backtracking*, $\text{PA}(\Theta)$ and $\text{CH}_1(\Theta), \text{CH}_2(\Theta)$ represent the parent and two children nodes of the Boolean operator node $\Theta \in \{\wedge, \vee\}$, respectively.

Remark 1. *In [31], the TLT is introduced for the model checking and control synthesis of discrete-time systems under LTL tasks. In this work, the sTLT is designed to guide the design of CBFs for continuous-time dynamical systems under nested STL formulas. The much more complex time constraints encoded in STL formulas have naturally led to different construction procedures and semantics of sTLT when compared to TLT in [31], which we highlight as follows. First, the construction of sTLT is largely different from TLT. To incorporate the time constraints encoded in an STL formula, our construction of the sTLT relies on the finite time reachability analysis, i.e., the maximal and minimal reachablility operators $\mathcal{R}^M$ and $\mathcal{R}^m$ given respectively in Definitions 3 and 4. In [31], the TLT construction relies on the infinite time controlled invariant set and robust controlled invariant set. Second, the sTLT semantics is largely different from TLT semantics. In order to monitor the time constraint satisfaction in an STL formulas, we introduce in this work the definition of* time interval coding *(cf. Definition 12) for a complete path of an sTLT. On one hand, we show in Definitions 13 and 14 that the satisfaction of an sTLT can be characterized by the existence of a well-defined time interval coding. On the other hand, it will become clear later that the time interval coding is also vital in control synthesis. In [31], the TLT semantics is much simpler as it only requires an assignment of ascending integers as time indices for each complete path of TLT.*

To better understand the sTLT semantics, i.e., Definition 14, the following definitions are needed.

Definition 15. *We say* an sTLT $\mathcal{T}$ contains $\vee$ operator nodes only at its top layers *if for every complete path* $\boldsymbol{p} = \mathbb{X}_0\Theta_1\mathbb{X}_1\Theta_2\ldots\Theta_{N_f}\mathbb{X}_{N_f}$ *of* $\mathcal{T}$ *that contains* $\vee$ *operator nodes, there exists a* $1 \leq k \leq N_f$ *such that*

$$\Theta_j \in \begin{cases} \{\vee\} & j \in \{1, \ldots, k\}, \\ \{\wedge, F_{[a,b]}, G_{[a,b]}\}, & j \in \{k+1, \ldots, N_f\}. \end{cases} \tag{11}$$

Remark 2. *For any nested STL formula* φ*, the operator nodes* $\vee$*, if it exists, only appears in the top layers of the constructed sTLT* $\mathcal{T}_{\hat{\varphi}}$*. This can be seen from the fact that* $\hat{\varphi}$ *is in the form of* $\hat{\varphi} = \varphi_1 \vee \varphi_2 \vee \ldots \vee \varphi_N$*, and* $\varphi_i, i = 1, 2, \ldots, N$*, contain no* $\vee$ *operator as discussed in Step 1.*

Definition 16. *Let* $\boldsymbol{p}_l = \mathbb{X}_0\Theta_1^l\mathbb{X}_1^l\Theta_2^l\ldots\Theta_{N_f}^l\mathbb{X}_{N_f}^l$ *and* $\boldsymbol{p}_f = \mathbb{X}_0\Theta_1^f\mathbb{X}_1^f\Theta_2^f\ldots\Theta_{N_f'}^f\mathbb{X}_{N_f'}^f$ *be two complete paths of an sTLT* $\mathcal{T}$*. Denote by* $k_l = \arg\max_k\{\Theta_k^l = \vee\}$ *and* $k_f = \arg\max_k\{\Theta_k^f = \vee\}$*. We say* $\boldsymbol{p}_l$ and $\boldsymbol{p}_f$ belong to the same branch of $\mathcal{T}$ *if* $k_l = k_f$ *and* $\mathbb{X}_j^l = \mathbb{X}_j^f, \Theta_j^l = \Theta_j^f, \forall j = 1, \ldots k_l$.

Remark 3. *Definition 14 can be interpreted as follows:*

1) *Consider the case where the sTLT* $\mathcal{T}$ *contains no* $\vee$ *operator. Then Definition 14 dictates that* $(\boldsymbol{x}, t) \cong \mathcal{T}$ *if and only if* $(\boldsymbol{x}, t)$ *satisfies every complete path of* $\mathcal{T}$.
2) *Consider the case where the sTLT* $\mathcal{T}$ *contains* $\vee$ *operator nodes only at its top layers. Then Definition 14 dictates that* $(\boldsymbol{x}, t) \cong \mathcal{T}$ *if and only if* $(\boldsymbol{x}, t)$ *satisfies at least one branch of complete paths.*

Example (continued)**.** Let us continue with Example 1. According to Definition 10, the sTLT $\mathcal{T}_{\hat{\varphi}}$ (see Fig. 2) has in total

3 complete paths, i.e.,

$$\boldsymbol{p}_1 = \mathbb{X}_0 \vee \mathbb{X}_1 \mathsf{F}_{[0,15]} \mathbb{X}_3 \mathsf{G}_{[2,10]} \mathbb{X}_5,$$
$$\boldsymbol{p}_2 = \mathbb{X}_0 \vee \mathbb{X}_2 \mathsf{F}_{[0,15]} \mathbb{X}_4 \wedge \mathbb{X}_6 \mathsf{G}_{[0,10]} \mathbb{X}_8,$$
$$\boldsymbol{p}_3 = \mathbb{X}_0 \vee \mathbb{X}_2 \mathsf{F}_{[0,15]} \mathbb{X}_4 \wedge \mathbb{X}_7 \mathsf{F}_{[5,10]} \mathbb{X}_9,$$

and 5 temporal fragments, which are encircled by the red dashed rectangles in Fig. 2.

The sTLT $\mathcal{T}_{\hat{\varphi}}$ contains $\vee$ operator nodes only at its top layers since one has $k_1 = k_2 = k_3 = 1$ according to Definition 15. On one hand, one observes that $\Theta_1^2 = \Theta_1^3 = \vee$ and $\mathbb{X}_1^2 = \mathbb{X}_1^3 = \mathbb{X}_2$. Therefore, $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$ belong to the same branch. On the other hand, since $\mathbb{X}_1^1 = \mathbb{X}_1 \neq \mathbb{X}_2 = \mathbb{X}_1^2 = \mathbb{X}_1^3$, neither $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ nor $\boldsymbol{p}_1$ and $\boldsymbol{p}_3$ belong to the same branch. A trajectory $(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}}$ if and only if either of the following 2 conditions is satisfied: (1) $(\boldsymbol{x}, t) \cong \boldsymbol{p}_1$, (2) $(\boldsymbol{x}, t) \cong \boldsymbol{p}_2$ and $(\boldsymbol{x}, t) \cong \boldsymbol{p}_3$.

C. Relations between $\mathcal{T}_{\hat{\varphi}}$ and $\hat{\varphi}$ (φ)

In this section, we derive the relations between an STL formula $\hat{\varphi}$ (φ) and its constructed sTLT $\mathcal{T}_{\hat{\varphi}}$. First, we show the result for STL formulas in desired form, i.e., $\hat{\varphi}$.

Theorem 1. *Consider the system (1) and an STL task $\hat{\varphi}$ in desired form as per Definition 9. The sTLT $\mathcal{T}_{\hat{\varphi}}$ is equivalent to $\hat{\varphi}$ in the sense that*

$$(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}} \Leftrightarrow (\boldsymbol{x}, t) \models \hat{\varphi}. \tag{12}$$

Proof: For $\top$, predicate μ, its negation $\neg\mu$, $\mu_1 \wedge \mu_2$, and $\mu_1 \vee \mu_2$, it is trivial to verify that $(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}} \Leftrightarrow (\boldsymbol{x}, t) \models \hat{\varphi}$.

Next, we follow the induction rule to show that if $(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}_1} \Leftrightarrow (\boldsymbol{x}, t) \models \hat{\varphi}_1$ and $(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}_2} \Leftrightarrow (\boldsymbol{x}, t) \models \hat{\varphi}_2$, then the constructed sTLT $\mathcal{T}_{\hat{\varphi}}$ satisfies $(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}} \Leftrightarrow (\boldsymbol{x}, t) \models \hat{\varphi}$ for a) $\hat{\varphi} = \hat{\varphi}_1 \wedge \hat{\varphi}_2$, b) $\hat{\varphi}_1 \vee \hat{\varphi}_2$, c)$\mathsf{F}_{[a,b]}\hat{\varphi}_1$, and d) $\mathsf{G}_{[a,b]}\hat{\varphi}_1$.

Case a): $\hat{\varphi} = \hat{\varphi}_1 \wedge \hat{\varphi}_2$. Assume that a trajectory $(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}}$. According to Definition 14, we have $(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}_1}$ and $(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}_2}$. Under the assumption that $(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}_1} \Leftrightarrow (\boldsymbol{x}, t) \models \hat{\varphi}_1$ and $(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}_2} \Leftrightarrow (\boldsymbol{x}, t) \models \hat{\varphi}_2$, one can get that $(\boldsymbol{x}, t) \models \hat{\varphi}_1$ and $(\boldsymbol{x}, t) \models \hat{\varphi}_2$, which implies $(\boldsymbol{x}, t) \models \hat{\varphi}$. Thus, $(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}} \Rightarrow (\boldsymbol{x}, t) \models \hat{\varphi}$. The proof of the other direction is similar and hence omitted.

Case b): $\hat{\varphi} = \hat{\varphi}_1 \vee \hat{\varphi}_2$. The proof is similar to Case a) and hence omitted.

Case c): $\hat{\varphi} = \mathsf{F}_{[a,b]}\hat{\varphi}_1$. Assume that a trajectory $(\boldsymbol{x}, t) \cong \mathcal{T}_{\mathsf{F}_{[a,b]}\hat{\varphi}_1}$. As depicted in Fig.1(c), we know that each complete path of $\mathcal{T}_{\hat{\varphi}}$ can be written in the form of $\boldsymbol{p} = \mathbb{X}_0 \Theta_1 \boldsymbol{p}'$, where $\Theta_1 = F_{[a,b]}$ and $\boldsymbol{p}'$ is a complete path of $\mathcal{T}_{\hat{\varphi}_1}$. According to Definitions 13 and 14, we have $\exists t' \in [t + a, t + b], \boldsymbol{x}(t') \in \mathbb{S}_{\hat{\varphi}_1}$ and $(\boldsymbol{x}, t') \cong \mathcal{T}_{\hat{\varphi}_1}$. Under the assumption that $(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}_1} \Leftrightarrow (\boldsymbol{x}, t) \models \hat{\varphi}_1$, one can get that $\exists t' \in [a, b], (\boldsymbol{x}, t') \models \hat{\varphi}_1$, which implies $(\boldsymbol{x}, t) \models \mathsf{F}_{[a,b]}\hat{\varphi}_1$ by Definition 1. Thus, $(\boldsymbol{x}, t) \cong \mathcal{T}_{\mathsf{F}_{[a,b]}\hat{\varphi}_1} \Rightarrow (\boldsymbol{x}, t) \models \mathsf{F}_{[a,b]}\hat{\varphi}_1$. Assume now that $(\boldsymbol{x}, t) \models \mathsf{F}_{[a,b]}\hat{\varphi}_1$. Then one has from Definition 1 that $\exists t' \in [t + a, t + b], \boldsymbol{x}(t') \in \mathbb{S}_{\hat{\varphi}_1}$, which implies $\boldsymbol{x}(t) \in \mathcal{R}^M(\mathbb{S}_{\hat{\varphi}_1}, [a, b]) = \mathbb{X}_0$. According to Definitions 13 and 14, it means that $(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}}$. Therefore, $(\boldsymbol{x}, t) \models \mathsf{F}_{[a,b]}\hat{\varphi}_1 \Rightarrow (\boldsymbol{x}, t) \cong \mathcal{T}_{\mathsf{F}_{[a,b]}\hat{\varphi}_1}$.

Case d): $\hat{\varphi} = \mathsf{G}_{[a,b]}\hat{\varphi}_1$. Assume that a trajectory $(\boldsymbol{x}, t) \cong \mathcal{T}_{\mathsf{G}_{[a,b]}\hat{\varphi}_1}$. As depicted in Fig.1(d), we know that each complete path of $\mathcal{T}_{\hat{\varphi}}$ can be written in the form of $\boldsymbol{p} = \mathbb{X}_0 \Theta_1 \boldsymbol{p}'$, where $\Theta_1 = \mathsf{G}_{[a,b]}$ and $\boldsymbol{p}'$ is a complete path of $\mathcal{T}_{\hat{\varphi}_1}$. According to Definitions 13 and 14, we have $(\boldsymbol{x}, t') \cong \mathcal{T}_{\hat{\varphi}_1}, \forall t' \in [t+a, t+b]$. Under the assumption that $(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}_1} \Leftrightarrow (\boldsymbol{x}, t) \models \hat{\varphi}_1$, one can get that $(\boldsymbol{x}, t') \models \hat{\varphi}_1, \forall t' \in [t+a, t+b]$, which implies $(\boldsymbol{x}, t) \models \mathsf{G}_{[a,b]}\hat{\varphi}_1$ by Definition 1. Thus, $(\boldsymbol{x}, t) \cong \mathcal{T}_{\mathsf{G}_{[a,b]}\hat{\varphi}_1} \Rightarrow (\boldsymbol{x}, t) \models \mathsf{G}_{[a,b]}\hat{\varphi}_1$. Assume now that $(\boldsymbol{x}, t) \models \mathsf{G}_{[a,b]}\hat{\varphi}_1$. Then one has that $\boldsymbol{x}(t') \in \mathbb{S}_{\hat{\varphi}_1}, \forall t' \in [t+a, t+b]$. Since $\hat{\varphi}_1$ contains no "disjunction" operator according to Definition 9, one can further get that $\boldsymbol{x}(t) \in \overline{\mathcal{R}^m(\overline{\mathbb{S}_{\hat{\varphi}_1}}, [a, b])} = \mathbb{X}_0$. According to Definitions 13 and 14, it means that $(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}}$. Therefore, $(\boldsymbol{x}, t) \models \mathsf{G}_{[a,b]}\hat{\varphi}_1 \Rightarrow (\boldsymbol{x}, t) \cong \mathcal{T}_{\mathsf{G}_{[a,b]}\hat{\varphi}_1}$. ∎

For general STL tasks, we have the following result.

Theorem 2. *Consider the system (1) and an STL task φ in Definition 2. The sTLT $\mathcal{T}_{\hat{\varphi}}$ is an under-approximation of φ in the sense that*

$$(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}} \Rightarrow (\boldsymbol{x}, t) \models \varphi.$$

Proof: The proof can be completed by showing 1) $(\boldsymbol{x}, t) \cong \mathcal{T}_{\hat{\varphi}} \Leftrightarrow (\boldsymbol{x}, t) \models \hat{\varphi}$ and 2) $(\boldsymbol{x}, t) \models \hat{\varphi} \Rightarrow (\boldsymbol{x}, t) \models \varphi$. The proof for condition 1) is given in Theorem 1. Condition 2) is straightforward since $(\boldsymbol{x}, t) \models \mathsf{G}_{[0,b]}\varphi_1 \wedge \mathsf{F}_{[a,b]}\varphi_2 \Rightarrow (\boldsymbol{x}, t) \models \varphi_1 \mathsf{U}_{[a,b]}\varphi_2$, $(\boldsymbol{x}, t) \models \mathsf{G}_{[a,b]}\varphi_1 \vee \mathsf{G}_{[a,b]}\varphi_2 \Rightarrow (\boldsymbol{x}, t) \models \mathsf{G}_{[a,b]}(\varphi_1 \vee \varphi_2)$, and $(\boldsymbol{x}, t) \models \mathsf{F}_{[a,b]}\varphi_1 \vee \mathsf{F}_{[a,b]}\varphi_2 \Leftrightarrow (\boldsymbol{x}, t) \models \mathsf{F}_{[a,b]}(\varphi_1 \vee \varphi_2)$, one has from the construction of the sTLT, Step 1 that $(\boldsymbol{x}, t) \models \hat{\varphi} \Rightarrow (\boldsymbol{x}, t) \models \varphi$. The conclusion follows. ∎

Remark 4. *It becomes apparent from Theorems 1 and 2 that the under-approximation gap between general STL formula φ in Definition 2 and sTLT is a result of* Step 1 *of constructing the sTLT, i.e., rewrite the STL formula φ into the desired form $\hat{\varphi}$. In this step, we conduct two operations. First, we rewrite "until" operator $\varphi = \varphi_1 \mathsf{U}_{[a,b]} \varphi_2$ as $\hat{\varphi} = \mathsf{G}_{[0,b]}\varphi_1 \wedge \mathsf{F}_{[a,b]}\varphi_2$. This operation introduces conservatism because $\hat{\varphi}$ requires the pre-argument φ_1 of the "until" operator to be satisfied for the time interval $[0, b]$ while φ requires that $\exists t' \in [a, b]$ such that φ_1 is satisfied for the time interval $[0, t']$. Second, we rewrite $\varphi = \Theta_{[a,b]}(\varphi_1 \vee \varphi_2)$ as $\hat{\varphi} = \Theta_{[a,b]}\varphi_1 \vee \Theta_{[a,b]}\varphi_2$, where $\theta \in \{\mathsf{F}, \mathsf{G}\}$. In this operation, the conservatism comes from "always" operator because $(\boldsymbol{x}, t) \models \mathsf{G}_{[a,b]}\varphi_1 \vee \mathsf{G}_{[a,b]}\varphi_2 \Rightarrow (\boldsymbol{x}, t) \models \mathsf{G}_{[a,b]}(\varphi_1 \vee \varphi_2)$ while the other direction does not hold in general. For "eventually" operator, though, this operation introduces no conservatism. We further note that rewriting an "until" operator as a conjunction of an "always" operator and an "eventually" operator is also used in the CBF-based approaches for non-nested STL formulas [17], [18], whereas the "disjunction" operator is not addressed. Therefore, our approach is no more conservative compared to existing CBF-based approaches.*

D. Design of CBFs

In this subsection, we will use the sTLT $\mathcal{T}_{\hat{\varphi}}$ to guide the CBF design for a given STL formula φ. The intuition is that, we will design a time interval encoding and appropriate CBFs to enforce the synthesized system trajectory to satisfy the sTLT $\mathcal{T}_{\hat{\varphi}}$ as per the conditions given in Definitions 13 and 14.

1) Time encoding for the sTLT: Before proceeding, the following notations are needed. Given an STL operator $\Theta \in \{\wedge, \vee, \mathsf{F}_{[a,b]}, \mathsf{G}_{[a,b]}\}$, define the possible start time (interval) of Θ (i.e., time to evaluate the satisfaction of $\varphi_1 \Theta \varphi_2$ or $\Theta\varphi$) as

$$[\underline{t}(\Theta), \bar{t}(\Theta)] := \begin{cases} [0,0], & \text{if } \Theta \in \{\wedge, \vee\}, \\ [a,b], & \text{if } \Theta \in \{\mathsf{F}_{[a,b]}\}, \\ [a,a], & \text{if } \Theta \in \{\mathsf{G}_{[a,b]}\}. \end{cases} \tag{13}$$

The start time for logic operators $\wedge$ and $\vee$ is 0. For the temporal operator $\mathsf{G}_{[a,b]}$, the start time is a. Note that for the temporal operator $\mathsf{F}_{[a,b]}$, any time instant in the interval $[a,b]$ fulfills item ii) of Definition 13. To accommodate this uncertainty, we set the start time for $\mathsf{F}_{[a,b]}$ to be the interval $[a,b]$.

In addition, we define the duration of Θ as

$$\mathcal{D}(\Theta) := \begin{cases} 0, & \text{if } \Theta \in \{\wedge, \vee, \mathsf{F}_{[a,b]}\}, \\ b-a, & \text{if } \Theta \in \{\mathsf{G}_{[a,b]}\}. \end{cases} \tag{14}$$

The root node of $\mathcal{T}_{\hat{\varphi}}$ is denoted by $\mathbb{X}_{\text{root}}$. Let $\mathbb{X}$ be the set which collects all the set nodes of the sTLT $\mathcal{T}_{\hat{\varphi}}$. For a set node $\mathbb{X}_i \in \mathbb{X}$, define $[\underline{t}_s(\mathbb{X}_i), \bar{t}_s(\mathbb{X}_i)]$ and $\mathcal{D}(\mathbb{X}_i)$ as the possible start time (interval) and the duration of $\mathbb{X}_i$, respectively. $\text{PA}(\mathbb{X}_i)$ denotes the parent of node $\mathbb{X}_i$. Therefore, one has that $\text{PA}(\mathbb{X}_i)$ is an operator node and $\text{PA}(\text{PA}(\mathbb{X}_i))$ is a set node.

Now, the calculation of the start time (interval) for each set node $\mathbb{X}_i$ (which is needed for ensuring the satisfaction of the sTLT $\mathcal{T}_{\hat{\varphi}}$ as shown in Theorem 1) is outlined in Algorithm 3.

Algorithm 3 *calculateStartTimeInterval*

Input: The sTLT $\mathcal{T}_{\hat{\varphi}}$.
Return: $\underline{t}_s(\mathbb{X}_i), \bar{t}_s(\mathbb{X}_i), \mathcal{D}(\mathbb{X}_i), \forall \mathbb{X}_i$.
1: $\underline{t}_s(\mathbb{X}_{\text{root}}) \leftarrow 0, \bar{t}_s(\mathbb{X}_{\text{root}}) \leftarrow 0, \mathcal{D}(\mathbb{X}_{\text{root}}) \leftarrow 0$
2: **for** each non-root node $\mathbb{X}_i$ of $\mathcal{T}_{\varphi}$ through a top-down traversal, **do**
3: $\underline{t}_s(\mathbb{X}_i) \leftarrow \underline{t}_s(\text{PA}(\text{PA}(\mathbb{X}_i))) + \underline{t}(\text{PA}(\mathbb{X}_i))$, $\bar{t}_s(\mathbb{X}_i) \leftarrow \bar{t}_s(\text{PA}(\text{PA}(\mathbb{X}_i))) + \bar{t}(\text{PA}(\mathbb{X}_i))$,
4: $\mathcal{D}(\mathbb{X}_i) \leftarrow \mathcal{D}(\text{PA}(\text{PA}(\mathbb{X}_i))) + \mathcal{D}(\text{PA}(\mathbb{X}_i))$,
5: **end for**

Due to the uncertainty of the start time for temporal operator $\mathsf{F}_{[a,b]}$, one can see that the start times of some set nodes $\mathbb{X}_i$ may be unknown and belong to an interval after running Algorithm 3. In the following, we show how to update the start times of such set nodes $\mathbb{X}_i$ online.

We develop an event-triggered scheme to update the start times. For each set node $\mathbb{X}_i$ such that $\underline{t}_s(\mathbb{X}_i) \neq \bar{t}_s(\mathbb{X}_i)$, an event is triggered at time t if:

$$t \in [\underline{t}_s(\mathbb{X}_i)), \bar{t}_s(\mathbb{X}_i))] \ \wedge \ \boldsymbol{x}(t) \in \mathbb{X}_i. \tag{15}$$

Once an event is triggered, we run Algorithm 4 to update the start times of the set nodes. Note that once an event is triggered for a set node, its start time is fixed.

Example (continued)**.** Let us continue with Example 1 to demonstrate the event-triggered online update scheme.

First, one can calculate the start time intervals for each set node $\mathbb{X}_i, i = \{0, 1, \cdots, 9\}$ in the sTLT $\mathcal{T}_{\hat{\varphi}}$ (see Fig. 2) according to Algorithm 3, which give $[\underline{t}_s(\mathbb{X}_0), \bar{t}_s(\mathbb{X}_0)] = [\underline{t}_s(\mathbb{X}_1), \bar{t}_s(\mathbb{X}_1)] = [\underline{t}_s(\mathbb{X}_2), \bar{t}_s(\mathbb{X}_2)] = [0,0]$, $[\underline{t}_s(\mathbb{X}_3), \bar{t}_s(\mathbb{X}_3)] = [\underline{t}_s(\mathbb{X}_4), \bar{t}_s(\mathbb{X}_4)] = [0,15]$, $[\underline{t}_s(\mathbb{X}_5), \bar{t}_s(\mathbb{X}_5)] = [2,17]$, $[\underline{t}_s(\mathbb{X}_6), \bar{t}_s(\mathbb{X}_6)] = [\underline{t}_s(\mathbb{X}_7), \bar{t}_s(\mathbb{X}_7)] = [0,15]$, $[\underline{t}_s(\mathbb{X}_8), \bar{t}_s(\mathbb{X}_8)] = [0,15]$, and $[\underline{t}_s(\mathbb{X}_9), \bar{t}_s(\mathbb{X}_9)] = [5,25]$. Note that due to the 'eventually' operator $\mathsf{F}_{[0,15]}$ which appears at the outermost layer of the nested STL formula $\varphi = \mathsf{F}_{[0,15]}(\mathsf{G}_{[2,10]}\mu_1 \vee \mu_2 \mathsf{U}_{[5,10]}\mu_3)$, the start times of all the set nodes that belong to temporal fragments (i.e., $\mathbb{X}_i, i \in \{3,4,5,8,9\}$) are uncertain (i.e., belong to an interval). To reduce conservatism, we update the start time intervals of these set nodes online using the event-triggered scheme (15).

Algorithm 4 *onlineUpdate*

Input: The sTLT $\mathcal{T}_{\hat{\varphi}}$ and the triggering time t.
Return: the updated $\underline{t}_s(\mathbb{X}_i), \bar{t}_s(\mathbb{X}_i), \forall \mathbb{X}_i$.
1: **for** each $\mathbb{X}_i$ such that the triggering condition (15) is satisfied, **do**
2: $\underline{t}_s(\mathbb{X}_i) \leftarrow t, \bar{t}_s(\mathbb{X}_i) \leftarrow t$,
3: **end for**
4: **for** each set node $\mathbb{X}_i$ such that $\underline{t}_s(\mathbb{X}_i) \neq \bar{t}_s(\mathbb{X}_i)$ through a top-down traversal, **do**
5: run line 3 of Algorithm 3,
6: **end for**

Assume that at time instant $t = 5s$, the event-triggered condition (15) is satisfied for set node $\mathbb{X}_4$, i.e., $5 \in [\underline{t}_s(\mathbb{X}_4), \bar{t}_s(\mathbb{X}_4)] = [0,15]$ and $\boldsymbol{x}(5) \in \mathbb{X}_4$, then Algorithm 4 is activated. Following lines 1-3 of Algorithm 4, one has that $\underline{t}_s(\mathbb{X}_4) = \bar{t}_s(\mathbb{X}_4) = 5$ (i.e., the start time of set node $\mathbb{X}_4$ is fixed). Then one can further fix the start times of the set nodes $\mathbb{X}_6 = \mathbb{X}_7 = \mathbb{X}_8$ (which are $5s$) and update the start time interval of the set node $\mathbb{X}_9$ as $[\underline{t}_s(\mathbb{X}_9), \bar{t}_s(\mathbb{X}_9)] = [10,15]$.

2) CBF design for each temporal fragment: First, we have the following definition.

Definition 17. *We call a temporal fragment f_j the* predecessor *of another temporal fragment f_i (or f_i the* successor *of f_j) if there exists a complete path $\boldsymbol{p}$ such that $\boldsymbol{p} = ...f_j\boldsymbol{p}'f_i...$ where $\boldsymbol{p}'$ does not contain any temporal fragments. We call f_i a* top-layer temporal fragment *if f_i has no predecessor temporal fragment.*

Given the sTLT $\mathcal{T}_{\hat{\varphi}}$ for a nested STL formula φ, we need to design one CBF for each temporal fragment f_i in view of the item iv) of Definition 13. Denote by $f_i = \Theta_{f_i}\mathbb{X}_{f_i}$, where Θ_{f_i} and $\mathbb{X}_{f_i}$ are the temporal operator node and the set node contained in f_i. Note that $\mathbb{X}_{f_i}$ is represented by its value function $\mathbb{X}_{f_i} = \{x : h_{\mathbb{X}_{f_i}}(x) \geq 0\}$. We require the corresponding CBF $\mathfrak{b}_i(x,t)$ to satisfy the following conditions:

1) $\mathfrak{b}_i(x,t)$ is continuously differentiable and is defined over $\mathcal{C}(t) \times [\min\{t_e(\text{PA}(\text{PA}(\mathbb{X}_{f_i}))), \underline{t}_s(\mathbb{X}_{f_i})\}, t_e(\mathbb{X}_{f_i})]$;
2) $\mathfrak{b}_i(x,t) \leq h_{\mathbb{X}_{f_i}}(x), \forall t \in [\bar{t}_s(\mathbb{X}_{f_i}), t_e(\mathbb{X}_{f_i})]$,

where $t_e(\mathbb{X}_i) = \bar{t}_s(\mathbb{X}_i) + \mathcal{D}(\mathbb{X}_i)$ (recall $\mathcal{D}(\mathbb{X}_i)$ is computed in Algorithm 3) can be interpreted as the end time of $\mathbb{X}_i$. Here $\underline{t}_s(\mathbb{X}_{f_i}), \bar{t}_s(\mathbb{X}_{f_i})$ and $t_e(\mathbb{X}_{f_i})$ are updated online according to Algorithm 4.

Define the *time domain* of the CBF $\mathfrak{b}_i(x,t)$ as

$$[\underline{t}_{\mathfrak{b}_i}, \bar{t}_{\mathfrak{b}_i}] := [\min\{t_e(\text{PA}(\text{PA}(\mathbb{X}_{f_i}))), \underline{t}_s(\mathbb{X}_{f_i})\}, t_e(\mathbb{X}_{f_i})]. \quad (16)$$

This is to guarantee that the CBF $\mathfrak{b}_i$, which corresponds to the temporal fragment f_i, is activated at $t_e(\text{PA}(\text{PA}(\mathbb{X}_{f_i})))$, for which the activation of the predecessor of f_i ends, or at $\underline{t}_s(\mathbb{X}_{f_i})$, for which f_i becomes active at its earliest, whichever comes earlier. A formal statement on this is given in Lemma 2.

Lemma 2. *Let f_i be a non top-layer temporal fragment, and f_j be the predecessor of f_i in the constructed sTLT. Denote their respective CBFs $\mathfrak{b}_j(x,t), \mathfrak{b}_i(x,t)$. Then $\underline{t}_{\mathfrak{b}_j} \leq \underline{t}_{\mathfrak{b}_i} \leq \bar{t}_{\mathfrak{b}_j} \leq \bar{t}_{\mathfrak{b}_i}$.*

Proof: It can be deduced from the tree structure that the predecessor of a non top-layer temporal fragment is unique. Denote the set nodes in the fragments f_j and f_i are $\mathbb{X}_{f_j}, \mathbb{X}_{f_i}$, respectively. The inequalities can be obtained as follows: 1) in view of (16) and Algorithm 3, $\underline{t}_{\mathfrak{b}_j} \leq t_e(\mathbb{X}_{f_j})$ and $\underline{t}_{\mathfrak{b}_j} \leq \underline{t}_s(\mathbb{X}_{f_j}) \leq \underline{t}_s(\mathbb{X}_{f_i})$, thus $\underline{t}_{\mathfrak{b}_j} \leq \underline{t}_{\mathfrak{b}_i} = \min(t_e(\mathbb{X}_{f_j}), \underline{t}_s(\mathbb{X}_{f_i}))$; 2) from (16), $\underline{t}_{\mathfrak{b}_i} \leq t_e(\mathbb{X}_{f_j}) = \bar{t}_{\mathfrak{b}_j}$; 3) from Algorithm 3 and the definition of $t_e(\cdot)$, $\bar{t}_{\mathfrak{b}_j} = t_e(\mathbb{X}_{f_j}) = \bar{t}_s(\mathbb{X}_j) + \mathcal{D}(\mathbb{X}_j) \leq \bar{t}_s(\mathbb{X}_i) + \mathcal{D}(\mathbb{X}_i) = t_e(\mathbb{X}_{f_i}) = \bar{t}_{\mathfrak{b}_i}$. ■

If f_i is not a top-layer temporal fragment, then the third condition on the corresponding CBF $\mathfrak{b}_i(x,t)$ is

3) $\mathfrak{b}_i(x, \underline{t}_{\mathfrak{b}_i}) \geq 0, \forall x \in \{x : \mathfrak{b}_j(x, \underline{t}_{\mathfrak{b}_i}) \geq 0\}$, where f_j is the unique predecessor of f_i.

Note that $\mathfrak{b}_j(x, \underline{t}_{\mathfrak{b}_i})$ is well-defined in view of Lemma 2.

Proposition 1. *Given a complete path $\boldsymbol{p}$ and an initial condition x_0, let $f_0, f_1, ..., f_N$ be the sequence of temporal fragments contained in $\boldsymbol{p}$ and $\mathfrak{b}_0, \mathfrak{b}_1, \ldots, \mathfrak{b}_N$ the corresponding CBFs. Assume that each $\mathfrak{b}_i, i \in 0, \ldots, N$ satisfies the conditions 1)-3). Furthermore, if $\mathfrak{b}_0(x_0, 0) \geq 0$ and each of the CBFs $\mathfrak{b}_i$ satisfies the condition* (5) *during the corresponding time domain, then the resulting trajectory satisfies this complete path $\boldsymbol{p}$.*

Proof: Without loss of generality, assume that f_i is the predecessor of f_{i+1}, $i = 0, 1, ..., N-1$. For the top-level temporal fragment f_0, since $\mathfrak{b}_0(x_0, 0) \geq 0$ and the CBF condition (5) holds in $[0, \bar{t}_{\mathfrak{b}_0}]$, we have $\mathfrak{b}_0(\boldsymbol{x}(t), t) \geq 0, \forall t \in [0, \bar{t}_{\mathfrak{b}_0}]$. Now assume $\mathfrak{b}_i(\boldsymbol{x}(t), t) \geq 0, \forall t \in [\underline{t}_{\mathfrak{b}_i}, \bar{t}_{\mathfrak{b}_i}]$. From condition 3), $\mathfrak{b}_{i+1}(\boldsymbol{x}(\underline{t}_{\mathfrak{b}_{i+1}}), \underline{t}_{\mathfrak{b}_{i+1}}) \geq 0$. In addition, the CBF condition (5) of $\mathfrak{b}_{i+1}$ is satisfied for $\forall t \in [\underline{t}_{\mathfrak{b}_{i+1}}, \bar{t}_{\mathfrak{b}_{i+1}}]$, and then $\mathfrak{b}_{i+1}(\boldsymbol{x}(t), t) \geq 0, \forall t \in [\underline{t}_{\mathfrak{b}_{i+1}}, \bar{t}_{\mathfrak{b}_{i+1}}]$. Inductively, we obtain $\mathfrak{b}_i(\boldsymbol{x}(t), t) \geq 0, \forall t \in [\underline{t}_{\mathfrak{b}_i}, \bar{t}_{\mathfrak{b}_i}]$ for $i = 0, 1, 2, ..., N$.

In addition, $\mathfrak{b}_i(\boldsymbol{x}(t), t) \geq 0, \forall t \in [\underline{t}_{\mathfrak{b}_i}, \bar{t}_{\mathfrak{b}_i}]$ implies that $\boldsymbol{x}(t) \in \mathbb{X}_i, \forall t \in [\bar{t}_s(\mathbb{X}_{f_i}), t_e(\mathbb{X}_{f_i})]$ from condition 2). One verifies that $[\bar{t}_s(\mathbb{X}_{f_i}), t_e(\mathbb{X}_{f_i})], \forall f_i$ is a valid time interval coding of the complete path from Definition 13 items i-iii). Thus, the resulting trajectory satisfies the complete path $\boldsymbol{p}$. ■

Up to now, we have shown the design of the CBFs and the online update of their time domains for each temporal fragment in the sTLT $\mathcal{T}_{\hat{\varphi}}$. In what follows, we will show how to incorporate them to conduct the online control synthesis.

E. The overall algorithm

In this subsection, we divide the nested STL formulas into 2 classes, i.e., nested STL formulas that contain no $\vee$ operator and nested STL formulas that contain $\vee$ operator. We differentiate these two cases because they have different sTLT satisfaction conditions as discussed in Remark 3.

1) Nested STL formulas that contain no $\vee$ operator: Let φ be a nested STL formula that contains no $\vee$ operator. Then, the corresponding sTLT $\mathcal{T}_{\hat{\varphi}}$ contains no operator nodes $\vee$. Let Π be the set which collects all the temporal fragments f_i. Denote by $\mathfrak{b}_i$ the CBF designed for the temporal fragment f_i. Note that when the start time interval is updated online (Algorithm 4), the time domain of the CBF $\mathfrak{b}_i$ will also be updated correspondingly. The continuous-time control synthesis problem (Problem 1) can be solved by the following program:

$$\begin{aligned} \min_{u \in U} \quad & u^T Q u \\ \text{s.t.} \quad & \theta_i(t)\Big(\frac{\partial \mathfrak{b}_i(x,t)}{\partial x} f(x,u) + \frac{\partial \mathfrak{b}_i(x,t)}{\partial t} \\ & \qquad + \alpha_i(\mathfrak{b}_i(x,t))\Big) \geq 0, \forall f_i \in \Pi, \end{aligned} \quad (17)$$

where $\theta_i(t) = \begin{cases} 1, & \text{if } t \in [\underline{t}_{\mathfrak{b}_i}, \bar{t}_{\mathfrak{b}_i}] \\ 0, & \text{otherwise} \end{cases}$ is an indicator function assigned to each CBF $\mathfrak{b}_i$. Note that since $\underline{t}_{\mathfrak{b}_i}, \bar{t}_{\mathfrak{b}_i}$ are updated online, $\theta_i(t)$ is also updated online.

2) Nested STL formulas that contain $\vee$ operator: Let φ be a nested STL formula that contains $\vee$ operators. Then, as discussed in Remark 2, the operator nodes $\vee$ only appear in the top layers of $\mathcal{T}_{\hat{\varphi}}$.

Recall from Remark 3 that to obtain $(\boldsymbol{x}, 0) \cong \mathcal{T}_{\hat{\varphi}}$, $(\boldsymbol{x}, 0)$ needs to satisfy at least one branch of complete paths. Deciding which group of complete paths to satisfy can be done offline or online. In the following we show the case where the branch is chosen offline.

Without loss of generality, let Π_l be the set which collects all the temporal fragments f_i that belongs to the chosen branch. Then the online control synthesis is given by

$$\begin{aligned} u = \operatorname*{argmin}_{u \in U} \quad & u^T Q u \\ \text{s.t.} \quad & \theta_i(t)\Big(\frac{\partial \mathfrak{b}_i(x,t)}{\partial x} f(x,u) + \frac{\partial \mathfrak{b}_i(x,t)}{\partial t} \\ & \qquad + \alpha_i(\mathfrak{b}_i(x,t))\Big) \geq 0, \forall f_i \in \Pi_l, \end{aligned} \quad (18)$$

where $\mathfrak{b}_i$ is the designed CBF according to f_i, $\theta_i(t) = \begin{cases} 1, & \text{if } t \in [\underline{t}_{\mathfrak{b}_i}, \bar{t}_{\mathfrak{b}_i}] \\ 0, & \text{otherwise} \end{cases}$ is an indicator function assigned to each CBF $\mathfrak{b}_i$. Similar to the previous case, $\theta_i(t)$ is updated online by Algorithm 4.

Remark 5 (Choice of branch)**.** *The constructed sTLT provides a general guideline on how to choose the branch to satisfy. For example, denote by $\boldsymbol{p}_l = \mathbb{X}_0 \Theta_1^l \mathbb{X}_1^l \Theta_2^l \ldots \Theta_{N_f}^l \mathbb{X}_{N_f}^l$ an arbitrary complete path in branch l. Let $k_l = \text{argmax}_k \{\Theta_k^l = \vee\}$. Then the branch l can be chosen only if the initial state $x_0 \in \mathbb{X}_{k_l}^l$. This condition is evident from the sTLT semantics. One numerical example is given in Case Studies where only one branch out of two can be chosen. Several other factors*

can be considered when selecting the branch. For example, one can use performance indexes like robustness metrics, optimal energy, shortest path or online re-plan in the presence of environmental uncertainties. This is however out of the scope of this work and will be pursued in the future.

Remark 6 (Online CBFs update)**.** *Even though the time domains of the offline designed CBFs change as the start time intervals update online, this does not impose a need to re-compute the barriers from scratch. Instead, a simple translation in time will suffice. To illustrate this point, assume that we have computed two barriers* $\mathfrak{b}_j(x,t), t \in [\underline{t}_{\mathfrak{b}_j}, \bar{t}_{\mathfrak{b}_j}]$ *and* $\mathfrak{b}_i(x,t), t \in [\underline{t}_{\mathfrak{b}_i}, \bar{t}_{\mathfrak{b}_i}]$ *for two consecutive temporal fragments* $f_j f_i = \Theta_{f_j}\mathbb{X}_{f_j}\Theta_{f_i}\mathbb{X}_{f_i}$*. Denote* $\bar{t}_s(\mathbb{X}_{f_j})$ *before the update by* t_1*. If, at time* $t' \in [\underline{t}_s(\mathbb{X}_{f_j}), \bar{t}_s(\mathbb{X}_{f_j})]$*,* $\underline{t}_s(\mathbb{X}_{f_j}) \neq \bar{t}_s(\mathbb{X}_{f_j})$ *and* $\boldsymbol{x}(t')$ *reaches* $\mathbb{X}_{f_j}$*, then Algorithm 4 updates* $\underline{t}_s(\mathbb{X}_{f_j}) = \bar{t}_s(\mathbb{X}_{f_j}) = t'$*, and, accordingly, the new time domains of the barriers become* $[\underline{t}'_{\mathfrak{b}_j}, \bar{t}'_{\mathfrak{b}_j}] := [t', t' + \mathcal{D}(\mathbb{X}_{f_j})]$ *and* $[\underline{t}'_{\mathfrak{b}_i}, \bar{t}'_{\mathfrak{b}_i}] := [\underline{t}_{\mathfrak{b}_i} + t' - t_1, \bar{t}_{\mathfrak{b}_i} + t' - t_1]$*. The updated barriers are* $\mathfrak{b}'_j(x,t) = \mathfrak{b}_j(x, t + t_1 - t'), t \in [\underline{t}'_{\mathfrak{b}_j}, \bar{t}'_{\mathfrak{b}_j}]$ *and* $\mathfrak{b}'_i(x,t) = \mathfrak{b}_i(x, t + t_1 - t'), t \in [\underline{t}'_{\mathfrak{b}_i}, \bar{t}'_{\mathfrak{b}_i}]$*, respectively.*

Remark 7. *Recall that the above analysis is done for nested STL formulas as per Definition 7. It is straightforward to extend the results to STL tasks that are given by conjunction and/or disjunction of nested STL formulas, for instance,* $\varphi = \mathsf{F}_{[0,15]}(\mathsf{G}_{[0,10]}\mu_1 \vee \mu_2 \mathsf{U}_{[5,10]}\mu_3) \wedge \mathsf{G}_{[a_5,b_5]}\mu_4$*. The sTLT thus is constructed for* $\hat{\varphi} = \mathsf{F}_{[0,15]}\mathsf{G}_{[2,10]}\mu_1 \vee \mathsf{F}_{[0,15]}(\mathsf{G}_{[0,10]}\mu_2 \wedge \mathsf{F}_{[5,10]}\mu_3) \wedge \mathsf{G}_{[a_5,b_5]}\mu_4$*. The implementation can be done by adding an extra barrier condition corresponding to* $\mathsf{G}_{[a_5,b_5]}\mu_4$ *into* (18)*.*

Now we summarize our proposed solution in the following theorem.

Theorem 3. *Consider a dynamical system* (1) *and a nested STL specification* φ*. Let the sTLT constructed according to Section III-A. If the initial condition* $x_0 \in \mathbb{X}_{root}$ *and the online program is feasible, then the resulting system trajectory* $(\boldsymbol{x}, 0) \models \varphi$*.*

Proof: The proof follows from Proposition 1 and Theorem 2. ∎

Remark 8 (Nominal control as heuristics)**.** *In literature dedicated to studying CBFs [29], it is common to incorporate nominal controls to improve overall performance. This is usually done by replacing the weighted quadratic cost in* (17) *and* (18) *to the form* $(u - u_{nom})^T Q (u - u_{nom})$*, where* u_{nom} *is usually designed based on heuristics. More details on designing* u_{nom} *will be detailed in Case Studies.*

Remark 9 (Online feasibility)**.** *Although we require each* $\mathfrak{b}_i$ *to be a valid CBF, in general there is no guarantee that they are compatible [32], i.e., the online programs in* (17) *and* (18) *are feasible for all* (x,t)*. When the system is control-affine, the feasibility of QPs is guaranteed when the time domains of individual CBFs do not overlap. In the general case, one can verify or falsify the compatibility of multiple CBFs a priori using the method from [32]. More detailed discussions are given in Case Studies with several empirical remedies.*

F. Computational complexity

The computational complexity of the overall approach involves offline and online computational complexities. The offline phase is composed of 1) the construction of the sTLT and 2) the design of a CBF for each temporal fragment of the sTLT.

Construction of sTLT: Given an STL formula φ in desired form with K operators, the constructed sTLT contains at most $3K+1$ nodes (K operator nodes and at most $2K+1$ set nodes). The bottleneck for constructing the sTLT, however, is the computation of set nodes, which involves computing maximal or minimal reachable sets (defined in Definitions 4 and 5) for the continuous-time dynamical systems under consideration. In the case of a linear continuous-time system, one can compute reachable sets efficiently for large-scale linear systems with several thousand state variables for bounded, but arbitrarily varying inputs [33]. In the case of a nonlinear continuous-time system, the computation of backward reachable sets is in general undecidable [34]. Fortunately, over the past decade, new approaches (e.g., decomposition approach [35] and deep learning approach [36] and software tools (e.g., Hamilton-Jacobi Toolbox [37] and CORA Toolbox [38]), have been developed for improving the efficiency of computing backward reachable sets. Once the sTLT is constructed, the design of a CBF further requires calculating the start time interval and duration of the set node in the corresponding temporal fragment (i.e., Algorithm 3). The complexity of Algorithm 3 is $\mathcal{O}(1)$.

Construction of CBFs: The construction of CBFs can be computationally expensive for general nonlinear systems. Luckily, there are several remedies to simplify the computations. In view of the satisfaction condition of an sTLT, we could always construct a CBF based on an under-approximation of the set node in the corresponding temporal fragment when the exact reachable sets are difficult to calculate. Moreover, if the system is fully actuated, the CBF can in general be constructed analytically. One such example is the single integrator dynamics shown in the Case Studies. Other approaches include sum-of-squares techniques [39], learning-based approaches [40], and HJB reachability-based approaches [41]. In particular, we highlight that the construction of CBFs through HJB reachability analysis is a byproduct of computing the maximal/minimal reachable sets, which are essential for building the sTLT. The HJB reachability approach is also demonstrated in the Case Studies with unicycle dynamics.

Online computations: The online phase is composed of 1) the online update of the CBFs and 2) solving the optimization program (17) or (18). As pointed out in Remark 6, a simple translation in time is sufficient for updating the CBFs. Therefore, the complexity of this step is determined by online updating the start time intervals of set nodes (i.e., Algorithm 4), which is $\mathcal{O}(1)$. The complexity of the optimization program (17) or (18) is determined by the system model. When the continuous-time dynamical system (1) is control-affine, i.e., (1) is of the form (9), the programs (17) and (18) are QPs.

IV. Case studies

In this section, we explain the explicit procedures to construct CBFs and formulate the online QP for the nested STL specification given in Example 1. It is worth noting that the developed theory is dynamics agnostic. We will show this by designing control synthesis schemes for both single-integrator dynamics and unicycle dynamics, where analytical and numerical CBFs are constructed, respectively. In the end of this section, we demonstrate the efficacy of our proposed method under a more complex STL specification. All the implementation code can be found at https://github.com/xiaotan-git/sTLT.

A. Single integrator model

Consider a mobile robot with a single-integrator dynamics

$$\dot{x} = u, \tag{19}$$

where $x = (x_1, x_2) \in \mathbb{R}^2$ and $u = (u_1, u_2) \in U \subset \mathbb{R}^2$, and the control input set $U = \{u : |u_1| \leq 1, |u_2| \leq 1\}$. The STL task specification is given by $\varphi = \mathsf{F}_{[0,15]}(\mathsf{G}_{[2,10]}\mu_1 \vee \mu_2 \mathsf{U}_{[5,10]}\mu_3)$ (the same as in Example 1), where $\mathbb{S}_{\mu_1} = \{x \in \mathbb{R}^2 \mid (x_1+4)^2+(x_2+4)^2 \leq 1\}$, $\mathbb{S}_{\mu_2} = \{x \in \mathbb{R}^2 \mid (x_1-4)^2 + x_2^2 \leq 4^2\}$, and $\mathbb{S}_{\mu_3} = \{x \in \mathbb{R}^2 \mid (x_1-1)^2+(x_2+4)^2 \leq 2^2\}$. Recall from Example 1, the sTLT $\mathcal{T}_{\hat{\varphi}}$ is plotted in Fig. 2.

One observation is that, for single integrator dynamics and a given set node $\mathbb{X}_{\varphi_1}$, the sets $\mathcal{R}^M(\mathbb{X}_{\varphi_1}, [a,b])$ and $\overline{\mathcal{R}^m(\overline{\mathbb{X}_{\varphi_1}}, [a,b])}$, which are the set nodes obtained using the temporal operators $\mathsf{F}_{[a,b]}$ and $\mathsf{G}_{[a,b]}$ respectively, are monotonic increasing with respect to the input set U. Thus, to simplify the set calculation, we calculate subsets of the reachable sets by shrinking the input set U to $U' = \{u : \|u\| \leq 1\}$. Then one can get that

$$\begin{aligned}
&\mathbb{X}_5 = \mathbb{S}_{\mu_1}, \mathbb{X}_8 = \mathbb{S}_{\mu_2}, \mathbb{X}_9 = \mathbb{S}_{\mu_3},\\
&\mathbb{X}_3 = \{x \in \mathbb{R}^2 \mid (x_1+4)^2 + (x_2+4)^2 \leq 3^2\},\\
&\mathbb{X}_4 = \{x \in \mathbb{R}^2 \mid (x_1-4)^2 + x_2^2 \leq 4^2\},\\
&\mathbb{X}_1 = \{x \in \mathbb{R}^2 \mid (x_1+4)^2 + (x_2+4)^2 \leq 18^2\},\\
&\mathbb{X}_2 = \{x \in \mathbb{R}^2 \mid (x_1-4)^2 + x_2^2 \leq 19^2\},\\
&\mathbb{X}_0 = \{x \in \mathbb{R}^2 \mid (x_1+4)^2 + (x_2+4)^2 \leq 18^2 \text{ or }\\
&\qquad\qquad (x_1-4)^2 + x_2^2 \leq 19^2\}.
\end{aligned} \tag{20}$$

Here $\mathbb{X}_0, ..., \mathbb{X}_5$ are subsets of what one could obtain with the input set U. Yet the under-approximation relation still holds in view of the iv)th condition in Definition 13. Here we note that although the sets $\mathbb{X}_0, \mathbb{X}_1, \mathbb{X}_2$ are not needed for CBF design (since they do not correspond to any temporal fragments), they still play an important role that will become clear later. The sets $\mathbb{X}_3, \mathbb{X}_4, \mathbb{X}_5, \mathbb{X}_8, \mathbb{X}_9$ are depicted in Fig. 4.

Denote the temporal fragments $f_1 = \mathsf{F}_{[0,15]}\mathbb{X}_3, f_2 = \mathsf{G}_{[2,10]}\mathbb{X}_5, f_3 = \mathsf{F}_{[0,15]}\mathbb{X}_4, f_4 = \mathsf{G}_{[0,10]}\mathbb{X}_8, f_5 = \mathsf{F}_{[5,10]}\mathbb{X}_9$ and their corresponding control barrier functions $\mathfrak{b}_1, ..., \mathfrak{b}_5$. Using Algorithm 3 and (16), one obtainsthe initial starting time interval, the duration, and the time domain of the corresponding CBFs:

- $[\underline{t}_s(\mathbb{X}_3), \bar{t}_s(\mathbb{X}_3)] = [0,15], \mathcal{D}(\mathbb{X}_3) = 0, [\underline{t}_{\mathfrak{b}_1}, \bar{t}_{\mathfrak{b}_1}] = [0,15]$;
- $[\underline{t}_s(\mathbb{X}_5), \bar{t}_s(\mathbb{X}_5)] = [2,17], \mathcal{D}(\mathbb{X}_5) = 8, [\underline{t}_{\mathfrak{b}_2}, \bar{t}_{\mathfrak{b}_2}] = [2,25]$;
- $[\underline{t}_s(\mathbb{X}_4), \bar{t}_s(\mathbb{X}_4)] = [0,15], \mathcal{D}(\mathbb{X}_4) = 0, [\underline{t}_{\mathfrak{b}_3}, \bar{t}_{\mathfrak{b}_3}] = [0,15]$;
- $[\underline{t}_s(\mathbb{X}_8), \bar{t}_s(\mathbb{X}_8)] = [0,15], \mathcal{D}(\mathbb{X}_8) = 10, [\underline{t}_{\mathfrak{b}_4}, \bar{t}_{\mathfrak{b}_4}] = [0,25]$;
- $[\underline{t}_s(\mathbb{X}_9), \bar{t}_s(\mathbb{X}_9)] = [5,25], \mathcal{D}(\mathbb{X}_9) = 0, [\underline{t}_{\mathfrak{b}_5}, \bar{t}_{\mathfrak{b}_5}] = [5,25]$.

Taking into account the velocity limit, we design the initial CBFs as

$$\begin{aligned}
\mathfrak{b}_1(x,t) &= (18-t)^2 - (x_1+4)^2 - (x_2+4)^2, t \in [0,15];\\
\mathfrak{b}_2(x,t) &= \begin{cases} (18-t)^2 - (x_1+4)^2 - (x_2+4)^2, t \in [2,17];\\ 1^2 - (x_1+4)^2 - (x_2+4)^2, t \in [17,25]; \end{cases}\\
\mathfrak{b}_3(x,t) &= (19-t)^2 - (x_1-4)^2 - x_2^2, t \in [0,15];\\
\mathfrak{b}_4(x,t) &= \begin{cases} (19-t)^2 - (x_1-4)^2 - x_2^2, t \in [0,15];\\ 4^2 - (x_1-4)^2 - x_2^2, t \in [15,25]; \end{cases}\\
\mathfrak{b}_5(x,t) &= (27-t)^2 - (x_1-1)^2 - (x_2+4)^2, t \in [5,25].
\end{aligned} \tag{21}$$

It is evident that the zero super-level sets of the barriers are circular, which either remain static or shrink in radius at a velocity of 1. If the robot is about to leave the safe region, i.e., when $\mathfrak{b}_i(x,t) = 0$, the robot can always steer itself towards the center with unit velocity, and thus always stay safe. One could easily verify that, for $i = 1, 2, ..., 5$, 1) $\mathfrak{b}_i(x,t)$ is a valid CBF for the single integrator dynamics in (19); 2) $\mathfrak{b}_i(x,t) = h_{\mathbb{X}_{f_i}}(x), \forall t \in [\bar{t}_s(\mathbb{X}_{f_i}), t_e(\mathbb{X}_{f_i})]$, where $\mathbb{X}_{f_i}$ is the set node in the corresponding temporal fragment f_i; 3) $\mathfrak{b}_i(x, \underline{t}_{\mathfrak{b}_i}) \geq \mathfrak{b}_j(x, \underline{t}_{\mathfrak{b}_i}), \forall x$, where the corresponding temporal fragment f_j is the predecessor of f_i. Thus, CBFs in (21) fulfill the conditions in Sec. III.D. Note that here we calculate the initial CBFs, which will be updated online according to Algorithm 4 and Remark 6.

Since the nested STL formula contains $\vee$ operator, we need to determine which branch out of two branches $\{\boldsymbol{p}_1\}$ and $\{\boldsymbol{p}_2, \boldsymbol{p}_3\}$ (as in Example III-B) needs to be satisfied. The guideline to choose the branch is as follows: if the initial condition $x_0 \in \mathbb{X}_1$, we can choose $\Pi_l = \{f_1, f_2\}$; if $x_0 \in \mathbb{X}_2$, we can choose $\Pi_l = \{f_3, f_4, f_5\}$; if $x_0 \notin \mathbb{X}_0$, then the proposed scheme fails to generate a control signal with correctness guarantee and a larger input bound is expected.

It is worth highlighting that in the special case of $\Pi_l = \{f_1, f_2\}$, the feasibility of QP is guaranteed since for the time domains that $\mathfrak{b}_1$ and $\mathfrak{b}_2$ overlap, they pose the same CBF condition, so only one CBF is active at every time instant. We also observe that, empirically, the feasibility problem can be mitigated by further shrinking the input set U' or enlarging the class $\mathcal{K}$ functions in the QP, for example, by increasing the gain when it is linear. To incorporate heuristics in the control synthesis scheme, in this section, we choose $u_{nom}(t)$ in a way that guides the trajectory towards fulfilling the CBF $\mathfrak{b}_i$ with the smallest $\bar{t}_{\mathfrak{b}_i}$ among all active ones. Several other heuristics are also implemented in the code.

Now we demonstrate the numerical results with the proposed CBF-based QP control synthesis scheme. In Fig. 4, we illustrate trajectories with time snapshots starting from $(-6, 2)$ and $(-2, 3.5)$, both of which lie within $\mathbb{X}_1 \cap \mathbb{X}_2$. Here we set the α_i in (18) to be $\alpha_i(v) = v, v \in \mathbb{R}, \forall i$, and Q in (18) to be an identity matrix. For all the trajectories, the input bound U is respected. We observe that every trajectory satisfies the STL specification φ. If we take the initial condition $x_0 = (-20, -5)$, $x_0 \in \mathbb{X}_1$ and $x_0 \notin \mathbb{X}_2$, we observe that the

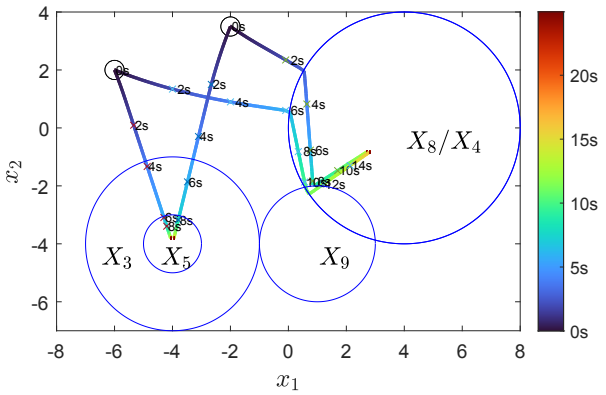

Fig. 4. Four trajectories of a mobile robot with single integrator dynamics are synthesized using the proposed method under the STL specification $\varphi = \mathsf{F}_{[0,15]}(\mathsf{G}_{[2,10]}\mu_1 \vee \mu_2 \mathsf{U}_{[5,10]}\mu_3)$. Two different starting positions (marked in circles) are tested and every trajectory satisfies the STL specification φ. It is observed that the robot starts its voyage from 0s and approaches the regions of interest without any stop. This was enabled by the design of the nominal controller and the online updates of the start time intervals of the set nodes. For the case that branch $\{\boldsymbol{p}_2, \boldsymbol{p}_3\}$ is chosen, we note that the trajectories leave $\mathbb{X}_9$ after reaching it. This behavior is due to the approximation gap as we require the trajectories to stay inside $\mathbb{X}_8$ after reaching $\mathbb{X}_9$ for the constructed sTLT.

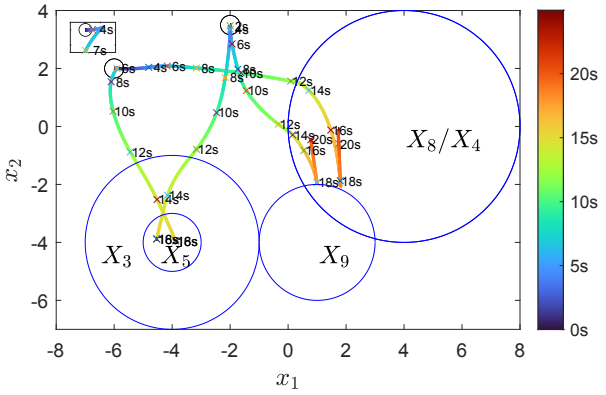

Fig. 5. Four trajectories of a mobile robot with unicycle dynamics are synthesised using the proposed method under the STL specification $\varphi = \mathsf{F}_{[0,15]}(\mathsf{G}_{[2,10]}\mu_1 \vee \mu_2 \mathsf{U}_{[5,10]}\mu_3)$. Two different starting configurations (marked in circles) are tested and every trajectory satisfies the STL specification φ. It is observed that the robot first adjusts its orientation and then starts to approach regions of interest without any stop. For the case that branch $\{\boldsymbol{p}_2, \boldsymbol{p}_3\}$ is chosen, we also see that the trajectories leave $\mathbb{X}_9$ after reaching it.

STL specification is fulfilled if we choose the branch $\{\boldsymbol{p}_1\}$; yet the online QP becomes infeasible if we choose the branch $\{\boldsymbol{p}_2, \boldsymbol{p}_3\}$. This is in line with the theoretical results.

When dealing with regions of irregular shapes or general nonlinear dynamics, the set nodes as well as the CBFs are difficult to calculate analytically. In the following we show a numerical construction scheme.

B. Unicycle model

Consider a mobile robot with a unicycle dynamics

$$\begin{aligned} \dot{x}_1 &= v\cos\theta, \\ \dot{x}_2 &= v\sin\theta, \\ \dot{\theta} &= \omega, \end{aligned} \tag{22}$$

where the state $x = (x_1, x_2, \theta)$, the control input $u = (v, \omega)$. Here (x_1, x_2) denotes the position, θ the heading angle, and v the velocity, ω the turning rate. We assume that the control input $u = (v, \omega) \in U = \{u \mid |v| \leq 1, |\omega| \leq 1\}$. The STL task specification is again given by $\varphi = \mathsf{F}_{[0,15]}(\mathsf{G}_{[2,10]}\mu_1 \vee \mu_2 \mathsf{U}_{[5,10]}\mu_3)$ (the same as in Example 1), where $\mathbb{S}_{\mu_1} = \{x \in \mathbb{R}^2 \times S^1 \mid (x_1+4)^2 + (x_2+4)^2 \leq 1\}$, $\mathbb{S}_{\mu_2} = \{x \in \mathbb{R}^2 \times S^1 \mid (x_1-4)^2 + x_2^2 \leq 4^2\}$, and $\mathbb{S}_{\mu_3} = \{x \in \mathbb{R}^2 \times S^1 \mid (x_1-1)^2 + (x_2+4)^2 \leq 2^2\}$. Recall from Example 1, the sTLT $\mathcal{T}_{\hat{\varphi}}$ is plotted in Fig. 2.

We note that the temporal fragments, their time encodings, the time domains for the barrier functions, and the branch choosing guidelines are similar to those as in the case of single integrator dynamics and thus omitted here. We will instead explain how the set nodes as well as the barrier functions are constructed through the use of a level-set reachability analysis toolbox [37], [42].

Here the set nodes with the input set $U = \{u \mid |v| \leq 1, |\omega| \leq 1\}$ are computed via reachability analysis. We may also use a shrinked input set to mitigate the online QP infeasibility issue. In brevity, we numerically obtain the value function $h_{\mathbb{X}_i}$ for the sets $\mathbb{X}_i, i = 0, 1, .., 9$, following the reachability operations in Example 1. The projection of sets $\mathbb{X}_3, \mathbb{X}_4, \mathbb{X}_5, \mathbb{X}_8, \mathbb{X}_9$ to the first two dimensions are depicted in Fig. 5.

Now we show how the CBFs are constructed. Take the construction of $\mathfrak{b}_2$ as an example, which corresponds to $f_2 = \mathsf{G}_{[2,10]}\mathbb{X}_5$. Recall $\mathbb{X}_5 = \{x \mid h_{\mathbb{X}_5}(x) \geq 0\}$, $[\underline{t}_s(\mathbb{X}_5), \bar{t}_s(\mathbb{X}_5)] = [2, 17]$, $[\underline{t}_{\mathfrak{b}_2}, \bar{t}_{\mathfrak{b}_2}] = [2, 25]$. Here the function $\mathfrak{b}_2$ is expected to be a valid control barrier function for the unicycle dynamics in (22) which guides $\boldsymbol{x}(t)$ towards the set $\mathbb{X}_5$ for $t \in [2, 17]$ and keeps $\boldsymbol{x}(t)$ in the set $\mathbb{X}_5$ for $t \in [17, 25]$. We construct such a $\mathfrak{b}_2$ by solving the following optimal control problem:

$$\begin{aligned} V(x,t) = \max_{\boldsymbol{u}(s), s\in[t,17]} \; & h_{\mathbb{X}_5}(\boldsymbol{x}^{\boldsymbol{u}}_{x,t}(17)) \\ \text{s.t.} \quad & (22) \text{ and } \boldsymbol{u}(s) \in U, \end{aligned} \tag{23}$$

where $\boldsymbol{x}^{\boldsymbol{u}}_{x,t}$ denotes the continuous state signal starting from x at time t with the input signal $\boldsymbol{u}$. $V(x,t)$ can be computed numerically by solving the following Hamilton-Jacobi-Bellman (HJB) equation [1]

$$\begin{aligned} &\frac{\partial V}{\partial t} + \max_{u\in U} \langle \nabla_x V(x,t), f(x,u)\rangle = 0, \\ &V(x,17) = h_{\mathbb{X}_5}(x). \end{aligned}$$

Thus, we choose $\mathfrak{b}_2(x,t) = \begin{cases} V(x,t), & t \in [15,17]; \\ h_{\mathbb{X}_5}(x), & t \in [17,25]. \end{cases}$ One can verify that $\mathfrak{b}_2(x,t)$ is a valid CBF as per Definition 6. The remaining barrier functions $\mathfrak{b}_1, \mathfrak{b}_3, \mathfrak{b}_4, \mathfrak{b}_5$ are constructed in a similar manner.

The numerical results with the proposed scheme for unicycle dynamics are shown in Fig. 5. Here we illustrate the trajectories with time snapshots starting from $(-6, 2, 0)$ and

[1] In general, if $h_{\mathbb{X}_5}(x)$ is Lipschitz continuous but not smooth, then only the viscosity solution can be obtained from the HJB equation and in this case $V(x,t)$ is Lipschitz continuous, which is differentiable almost everywhere.

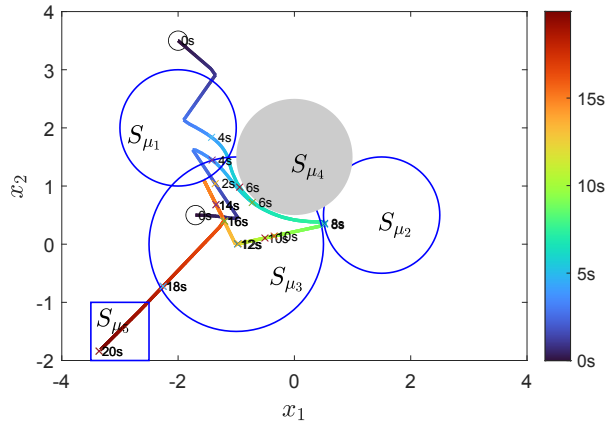

Fig. 6. Synthesised trajectories of a mobile robot with single integrator dynamics in (19) under the STL task in (24). Here $S_{\mu_1}, S_{\mu_2}, S_{\mu_3}, S_{\mu_4}$, and S_{μ_5} represent the regions in which the corresponding predicate functions are evaluated to be true. Two different starting positions (marked in circles) are tested and shown in the figure. For both trajectories, it is observed that the robot successfully follows the STL specifications and visits regions of interest in the specified time intervals, and always avoids the obstacle region. Thus, both trajectories satisfy the STL specification φ.

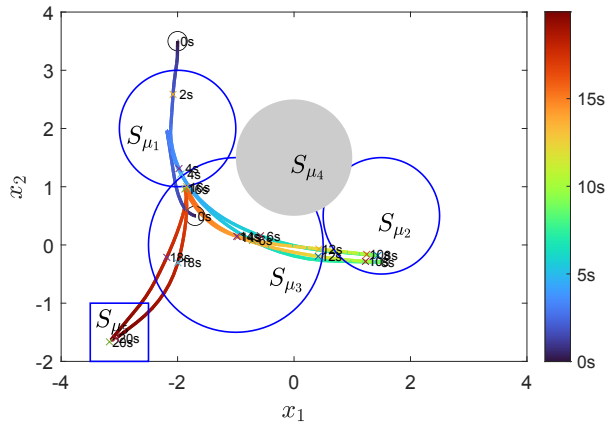

Fig. 7. Synthesised trajectories of a mobile robot with unicycle dynamics in (22) under the STL task in (24). Here $S_{\mu_1}, S_{\mu_2}, S_{\mu_3}$ and S_{μ_4} represent the projected regions in which the corresponding predicate functions are evaluated to be true. Two different starting configurations (marked with car-like symbols) are tested and shown in the figure. For both trajectories, it is observed that robot first adjusts its orientation and goes into regions of interest in the specified time intervals, and always avoids the region S_{μ_4}. Thus, both trajectories satisfy the STL specification φ.

$(-2, 3.5, \pi/2)$, both of which lie within $\mathbb{X}_1 \cap \mathbb{X}_2$. Here the α_i in (18) is set to be $\alpha_i(v) = v, v \in \mathbb{R}, \forall i$, and Q in (18) an identity matrix. An intuitive nominal controller similar to the single integrator case is also utilized in this example. For all the trajectories, the input bound U is respected. Again, we observe that every trajectory satisfies the STL specification φ.

C. Examples for more complex specifications

In this subsection, we consider the more complex STL formula below

$$\varphi = \mathsf{G}_{[0,1]}\mathsf{F}_{[2,3]}\mu_1 \wedge \mathsf{F}_{[6,7]}\mathsf{G}_{[1,2]}\mu_2 \wedge \mathsf{F}_{[13,14]}(\mu_3 \mathsf{U}_{[1,4]}\mu_1) \wedge \mathsf{G}_{[0,20]}\neg\mu_4 \wedge \mathsf{F}_{[15,20]}\mu_5. \quad (24)$$

The control synthesis process consists of constructing the corresponding sTLT, calculating the set nodes using reachability analysis, calculating the time encodings, and constructing the corresponding CBFs. This offline design process is similar to what we have detailed before, except that the region associated with the predicate μ_5 is a square. We take two different approaches: in the case of single integrator dynamics (Fig. 6), we use the largest inscribed circular region to under-approximate $\mathbb{S}_{\mu_5}$, and analytical CBFs are constructed; in the case of unicycle dynamics (Fig. 7), we use the signed distance function of the square as the superlevel set function and calculate the value function to the HJB equation as the CBF. Implementation details can be found in the online code repository. For the online synthesis, since the formula does not contain $\vee$, the QP in (17) will be used. Figure 6 and Figure 7 demonstrate the resulting system behaviors for a mobile robot with single integrator dynamics in (19) and with the unicycle dynamics in (22), respectively. We note that all trajectories satisfy the STL specification in (24), while respecting the dynamics and input bounds.

V. Conclusions

In this paper, we develop an efficient control synthesis approach for continuous-time dynamical systems under nested STL specifications. To this purpose, we introduce a notion of signal temporal logic tree (sTLT), detail on its construction from a given STL formula, its semantics (i.e., satisfaction condition), and the equivalence or under-approximation relation between the sTLT and the STL formula. Under the guidance of the sTLT, we show how to design CBFs and online update their activation time intervals. The control signal is thus given by an online CBF-based program. For future work, we will tackle the motion coordination problem of multi-agent systems under STL specifications leveraging task decomposition and distributed CBF techniques.

References

[1] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT press, 2008.

[2] R. Yan and A. A. Julius, "Interpretable seizure detection with signal temporal logic neural network," *Biomedical Signal Processing and Control*, vol. 78, p. 103998, 2022.

[3] U. Sanwal and U. Siddique, "Combining refinement and signal-temporal logic for biological systems," in *International Conference on Intelligent Computer Mathematics*. Springer, 2017, pp. 333–339.

[4] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE transactions on robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[5] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 61–70, 2007.

[6] 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.

[7] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Ničković, and S. Sankaranarayanan, "Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications," in *Lectures on Runtime Verification*. Springer, 2018, pp. 135–175.

[8] J. Eddeland, S. Miremadi, M. Fabian, and K. Åkesson, "Objective functions for falsification of signal temporal logic properties in cyber-physical systems," in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*. IEEE, 2017, pp. 1326–1331.

[9] P. Tabuada, *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media, 2009.

[10] C. Belta, B. Yordanov, and E. A. Gol, *Formal methods for discrete-time dynamical systems*. Springer, 2017, vol. 15.

[11] 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*. IEEE, 2014, pp. 81–87.

[12] 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*, 2015, pp. 239–248.

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

[14] Y. Gilpin, V. Kurtz, and H. Lin, "A smooth robustness measure of signal temporal logic for symbolic control," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 241–246, 2020.

[15] Y. Takayama, K. Hashimoto, and T. Ohtsuka, "Signal temporal logic meets convex-concave programming: A structure-exploiting sqp algorithm for stl specifications," *arXiv preprint arXiv:2304.01475*, 2023.

[16] K. M. B. Lee, C. Yoo, and R. Fitch, "Signal temporal logic synthesis as probabilistic inference," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 5483–5489.

[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] ——, "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.

[19] A. T. Buyukkocak, D. Aksaray, and Y. Yazıcıoğlu, "Control barrier functions with actuation constraints under signal temporal logic specifications," in *Proceedings of the 2022 European Control Conference*, 2022.

[20] L. Lindemann and D. V. Dimarogonas, "Efficient automata-based planning and control under spatio-temporal logic specifications," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 4707–4714.

[21] Q. H. Ho, R. B. Ilyes, Z. N. Sunberg, and M. Lahijanian, "Automaton-guided control synthesis for signal temporal logic specifications," *arXiv preprint arXiv:2207.03662*, 2022.

[22] 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.

[23] C.-I. Vasile, V. Raman, and S. Karaman, "Sampling-based synthesis of maximally-satisfying controllers for temporal logic specifications," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 3840–3847.

[24] R. Yan and A. Julius, "Neural network for weighted signal temporal logic," *arXiv preprint arXiv:2104.05435*, 2021.

[25] P. Kapoor, A. Balakrishnan, and J. V. Deshmukh, "Model-based reinforcement learning from signal temporal logic specifications," *arXiv preprint arXiv:2011.04950*, 2020.

[26] P. Varnai and D. V. Dimarogonas, "Prescribed performance control guided policy improvement for satisfying signal temporal logic tasks," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 286–291.

[27] 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*, 2015, pp. 239–248.

[28] M. Chen, Q. Tam, S. C. Livingston, and M. Pavone, "Signal temporal logic meets Hamilton-Jacobi reachability: connections and applications," in *Workshop on Algorithmic Foundations of Robotics*, 2018.

[29] 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.

[30] H. Khalil, "Nonlinear systems, printice-hall," *Upper Saddle River, NJ*, vol. 3, 1996.

[31] Y. Gao, A. Abate, F. J. Jiang, M. Giacobbe, L. Xie, and K. H. Johansson, "Temporal logic trees for model checking and control synthesis of uncertain discrete-time systems," *IEEE Transactions on Automatic Control*, 2021.

[32] X. Tan and D. V. Dimarogonas, "Compatibility checking of multiple control barrier functions for input constrained systems," in *IEEE Conference on Decision and Control*, 2022.

[33] M. Althoff, "Reachability analysis of large linear systems with uncertain inputs in the krylov subspace," *IEEE Transactions on Automatic Control*, vol. 65, no. 2, pp. 477–492, 2019.

[34] M. Althoff, G. Frehse, and A. Girard, "Set propagation techniques for reachability analysis," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 369–395, 2021.

[35] M. Chen, S. L. Herbert, M. S. Vashishtha, S. Bansal, and C. J. Tomlin, "Decomposition of reachable sets and tubes for a class of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 63, no. 11, pp. 3675–3688, 2018.

[36] S. Bansal and C. J. Tomlin, "Deepreach: A deep learning approach to high-dimensional reachability," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1817–1824.

[37] I. M. Mitchell and J. A. Templeton, "A toolbox of hamilton-jacobi solvers for analysis of nondeterministic continuous and hybrid systems," in *International workshop on hybrid systems: computation and control*. Springer, 2005, pp. 480–494.

[38] M. Althoff, "An introduction to cora 2015." *ARCH@ CPSWeek*, vol. 34, pp. 120–151, 2015.

[39] H. Wang, K. Margellos, and A. Papachristodoulou, "Safety verification and controller synthesis for systems with input constraints," *arXiv preprint arXiv:2204.09386*, 2022.

[40] A. Abate, D. Ahmed, A. Edwards, M. Giacobbe, and A. Peruffo, "Fossil: a software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks," in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, 2021, pp. 1–11.

[41] A. Wiltz, X. Tan, and D. V. Dimarogonas, "Construction of control barrier functions using predictions with finite horizon," in *IEEE Conference on Decision and Control*, 2023.

[42] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry, "Reach-avoid problems with time-varying dynamics, targets and constraints," in *Proceedings of the 18th international conference on hybrid systems: computation and control*, 2015, pp. 11–20.