TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Modularized Control Synthesis for Complex Signal Temporal Logic Specifications

Citation for published version (APA):
Zhang, Z., & Haesaert, S. (2023). Modularized Control Synthesis for Complex Signal Temporal Logic Specifications. *arXiv*, *2023*, Article 2303.17086. https://doi.org/10.48550/arXiv.2303.17086

DOI:
10.48550/arXiv.2303.17086

Document status and date:
Published: 30/03/2023

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 16. Nov. 2023

arXiv:2303.17086v1 [eess.SY] 30 Mar 2023

Modularized Control Synthesis for Complex Signal Temporal Logic Specifications*

Zengjie Zhang[1] and Sofie Haesaert[1]

***Abstract*— The control synthesis of a dynamic system subject to signal temporal logic (STL) specifications is commonly formulated as a mixed-integer linear programming (MILP) problem. Solving a MILP problem is computationally expensive when the STL formulas are long and complex. In this paper, we propose a framework to transform a long and complex STL formula into a syntactically separate form, i.e., the logical combination of a series of short and simple subformulas with non-overlapping timing intervals. Using this framework, one can easily modularize the synthesis of a complex formula using the synthesis solutions of the subformulas, which improves the efficiency of solving a MILP problem. Specifically, we propose a group of separation principles to guarantee the syntactic equivalence between the original formula and its syntactically separate counterpart. Then, we propose novel methods to solve the largest satisfaction region and the open-loop controller of the specification in a modularized manner. The efficacy of the methods is validated with a robot monitoring case study in simulation. Our work is promising to promote the efficiency of control synthesis for systems with complicated specifications.**

I. INTRODUCTION

Signal temporal logic (STL) has been widely used to describe the specifications of various practical systems, such as cyber-physical systems [1], [2] and robot systems [3], [4]. Compared to other temporal logic (TL) languages, STL can specify dense-time real-valued signals incorporating finite timing bounds [5]. It also has various metrics to quantify its satisfaction with real values instead of binary values, such as robustness degree [6] and the characteristic function [7]. Based on these metrics, the control synthesis of a system subject to STL specifications can be formulated as a mixed integer linear programming (MILP) problem which optimizes the real-valued specification satisfaction [5]. Such an optimization-based problem allows the synthesis of both open-loop and closed-loop controllers, where the latter is usually a model predictive controller (MPC) [8], [9], [10]. Nevertheless, solving a MILP problem is usually quite computationally expensive and time-consuming, especially for complex STL formulas that have long timing intervals or consist of many subformulas. A subformula refers to a simple STL formula that serves as a primitive unit of a complex formula. Such formulas are used to specify complex tasks with many sequential subtasks [11], [12], [13]. For example, a food delivery task for a delivery robot can be decomposed into three subtasks: (1). pick up the correct order at the restaurant; (2). navigate to the correct address of the customer; (3). find the customer and finish the delivery. The computational complexity of the optimization-based methods has become a bottleneck of the control synthesis of complex STL specifications.

Various efforts are devoted to reducing the complexity of an STL synthesis problem. One effective approach is the model-checking-based method which transforms an STL formula into an automaton with strict timing bounds [14]. Then, the control synthesis problem is solved as a game. Since the model-checking-based method is only concerned with a feasible solution instead of the optimal one, it is usually more efficient than solving a MILP problem. Control barrier functions (CBF) [12] and funnel functions [15] are also used to simplify the STL synthesis problems.They provide an elegant manner to encode STL specifications in spatiotemporal constraints and generate closed-form controllers. Another direction of simplifying an STL synthesis problem is to split an STL formula into shorter or simpler subformulas, such that a big optimization problem is split into several smaller ones. Usually, solving a MILP problem for a short and simple STL formula is far easier than a long and complex one. Furthermore, if the timing intervals of the subformulas are separated in time, it is possible to synthesize the individual subformulas one by one in the order of time, which forms the essential idea of modularized control synthesis. This idea is straightforward from a practical perspective: a complex task is usually composed of a series of smaller subtasks that have independent objectives and are ordered in time. Accomplishing the task means finishing all its subtasks.

However, STL specification separation and modularized synthesis are not trivial from a technical perspective. There exist several challenges. Firstly, the original formula and its separated counterpart must be syntactically equivalent. This is important to ensure consistency between the solution spaces of the two formulas [16]. Secondly, the subformulas may have overlapping timing intervals, which means that some subtasks require simultaneous execution. In this case, each subformula should not be synthesized independently, which is the major difficulty of modularized control synthesis of STL specifications. In existing work, the syntactic equivalence of a separated formula is ensured by *syntactic separation* which is originally discussed in [17]. In [18], a framework for separating an STL formula is presented based on a global time expression, although it is stated that separating a general STL formula in local time is still difficult. In general, the syntactic separation of STL formulas is still not well studied and remains an open question. Also, according

*This work was supported by the European project SymAware under the grant Nr. 101070802.

[1]Zengjie Zhang and Sofie Haesaert are with the Department of Electrical Engineering, Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, Netherlands {z.zhang3, s.haesaert}@tue.nl

to the best knowledge of the authors, there is currently no existing work discussing the modularized solution of control synthesis of complex STL formulas. However, we believe that syntactic separation is a critical technical point to realize modularized control synthesis which can greatly improve the efficiency of optimization-based STL synthesis methods.

This paper gives the first attempt to use the syntactic separation technology to realize modularized control synthesis for complex STL specifications. The synthesis problem is transformed into solving several small MILP problems, which achieves higher efficiency than directly solving an optimization problem for the original complex specification. The main contributions are summarized as follows. Firstly, we define a new type of STL formula with a sufficiently separated form that contains a group of subformulas with non-overlapping timing intervals. This type of formula is suitable to prescribe the specifications for most practical complicated tasks. Secondly, based on the existing work [17], [18], we provide a group of separation principles that can transform a certain class of STL formulas into a sufficiently separated formula with syntactic equivalence. Thirdly, we provide modularized methods for two important problems related to STL synthesis, namely the largest satisfying region (LSR) and the open-loop controller design for a complex STL formula, using the solutions of the sufficiently separated subformulas. The results of this paper are promising to provide efficient optimization-based approaches to the synthesis of specifications for complicated tasks in practice.

The rest of the paper is organized as follows. We introduce the preliminary knowledge of this paper in Sec. II. Sec. III presents the main results of this paper, including the separation principles to generate sufficiently separated STL formulas, the modularized method to compute the LSR of a complex STL formula, and the modularized approach for the synthesis of complex STL specifications. We validate the proposed methods using a simulation use case in Sec. IV. Finally, Sec. IV concludes this paper.

Notations: In this paper, we use $\mathbb{R}$, $\mathbb{R}^+$, and $\mathbb{R}_{\geq 0}$ to denote the sets of real numbers, positive real numbers, and non-negative real numbers. We also use $\mathbb{N}$ and $\mathbb{N}^+$ to represent non-negative integers and positive integers.

II. Preliminaries and Problem Statement

This section gives the preliminary knowledge of this paper, including the syntax, semantics, and syntactic separation of STL formulas. We also introduce the reachable set of a dynamic system and the largest satisfaction region (LSR) of an STL formula. Finally, we formulate the problem investigated in this paper.

A. Signal Temporal Logic (STL)

STL is a fragment of linear temporal logic (LTL) but with the capability of quantifying specifications for real-valued signals. For a real-valued signal defined in the continuous time domain $x:\mathbb{R}_{\geq 0}\to\mathbb{R}^n$, the syntax of STL is defined as

$$\varphi ::= \top \,|\, \mu \,|\, \neg\varphi \,|\, \varphi_1 \wedge \varphi_2 \,|\, \varphi_1 \mathsf{U}_{[a,b]} \varphi_2, \tag{1}$$

where φ_1, φ_2 are STL formulas, $\neg$, $\wedge$ are operators *negation* and *conjunction*, μ is a predicate that evaluates a predicate function $\eta:\mathbb{R}^n\to\{\top,\bot\}$ by $\mu = \begin{cases} \top & \text{if } \eta(x(t))\geq 0 \\ \bot & \text{if } \eta(x(t))<0 \end{cases}$, for a given time $t\in\mathbb{R}_{\geq 0}$, and $\mathsf{U}_{[a,b]}$ is the *until* operator bounded with time interval $[a,b]$, where $a,b\in\mathbb{N}^+$ and $a\leq b$. Note that the operator interval may also be semi-closed or open, like $(a,b]$, $[a,b)$, or (a,b), with $a<b$ to avoid being empty. For a given time $t\in\mathbb{R}_{\geq 0}$, $(x,t)\models\varphi$ defines the semantics of STL, i.e., x satisfies φ at time t, which is recursively assigned as follows. 1). $(x,t)\models\mu$, if $\eta(x(t))\geq 0$; 2). $(x,t)\models\neg\varphi$, if $\neg((x,t)\models\varphi)$; 3). $(x,t)\models\varphi_1\wedge\varphi_2$, if $(x,t)\models\varphi_1$ and $(x,t)\models\phi_2$; 4). $(x,t)\models\varphi_1\mathsf{U}_{[a,b]}\varphi_2$, if $\exists t'\in[t+a,t+b]$, such that $(x,t')\models\varphi_2$, and $(x,t'')\models\varphi_1$ holds for all $t''\in[t,t']$. Besides, additional operators *disjunction*, *eventually*, and *always* are respectively defined as $\varphi_1\vee\varphi_2=\neg(\neg\varphi_1\wedge\neg\varphi_2)$, $\mathsf{F}_{[a,b]}\varphi=\top\mathsf{U}_{[a,b]}\varphi$, and $\mathsf{G}_{[a,b]}\varphi=\neg\mathsf{F}_{[a,b]}\neg\varphi$. A formula φ is *feasible* if and only if $\exists x:\mathbb{R}_{\geq 0}:\to\mathbb{R}^n$ such that $(x,0)\models\varphi$. The *length* [19] or *future-reach* [17] of an STL, $\mathscr{L}(\varphi)$, is recursively defined as $\mathscr{L}(\mu)=0$, $\mathscr{L}(\neg\varphi)=\mathscr{L}(\varphi)$, $\mathscr{L}(\varphi_1\wedge\varphi_2)=\max\{\mathscr{L}(\varphi_1),\mathscr{L}(\varphi_2)\}$, $\mathscr{L}(\varphi_1\mathsf{U}_{[a,b]}\varphi_2)=b+\max\{\mathscr{L}(\varphi_1),\mathscr{L}(\varphi_2)\}$, which represents the maximum time it takes to determine the truth of the formula φ. The robustness semantics defined as follows are widely used to depict the extent of satisfaction with real values [6],

$$\begin{aligned}
&\rho(x,t,\mu):=h(x(t)),\ \rho(x,t,\neg\varphi):=-\rho(x,t,\varphi),\\
&\rho(x,t,\varphi_1\wedge\varphi_2):=\min(\rho(x,t,\varphi_1),\rho(x,t,\varphi_2)),\\
&\rho(x,t,\varphi_1\vee\varphi_2):=\max(\rho(x,t,\varphi_1),\rho(x,t,\varphi_2)),\\
&\rho\left(x,t,\varphi_1\mathsf{U}_{[a,b]}\varphi_2\right):=\\
&\max_{t'\in[t+a,t+b]}\left(\min\left(\rho(x,t',\varphi_2),\min_{t''\in[t,t']}\rho(x,t'',\varphi_1)\right)\right),\\
&\rho\left(x,t,\mathsf{F}_{[a,b]}\varphi\right):=\max_{t'\in[t+a,t+b]}\rho(x,t',\varphi),\\
&\rho\left(x,t,\mathsf{G}_{[a,b]}\varphi\right):=\max_{t'\in[t+a,t+b]}\rho(x,t',\varphi).
\end{aligned}\tag{2}$$

For the case $t=0$, the symbol t can be omitted.

B. Syntactic Separation of STL

In general, syntactic separation refers to splitting an STL formula into a group of subformulas [17], most commonly first-order formulas [18]. Specifically, for an STL formula φ defined as (1), represent it in the following separated form,

$$\varphi ::= \varphi_1\wedge\varphi_2 \,|\, \varphi_1\vee\varphi_2, \tag{3}$$

where φ_1,φ_2 are STL formulas. An STL formula in (3) is a *subformula* if it can not be further separated into the logical combination of other STL formulas. For example, for a real-valued signal $x:\mathbb{R}_{\geq 0}\to\mathbb{R}^n$, $(x>-1)\wedge\mathsf{G}_{[0,2]}(x<0)\vee\mathsf{F}_{[1,3]}(x>2)$ is a syntactic separated formula, with $(x>-1)$, $\mathsf{G}_{[0,2]}(x<0)$, and $\mathsf{F}_{[1,3}(x>2)$ being subformulas connected using logical operators $\wedge$ and $\vee$. For an STL formula consisting of an until operator $\varphi=\varphi_1\mathsf{U}_{(a,b)}\varphi_2$ with an open timing interval (a,b), where φ_1, φ_2 are STL formulas, the following separation law for a separating point $\tau\in\mathbb{R}^+$, $a<\tau<b$

ensures syntactic equivalence [17], [18], i.e., both sides have the same set of satisfying signals,

$$\varphi = \varphi_1 \mathsf{U}_{(a,\tau)} \varphi_2 \vee \left(\mathsf{G}_{(a,\tau)} \varphi_1 \wedge \mathsf{F}_{\{\tau\}}\left(\varphi_1 \wedge \varphi_2 \vee \varphi_1 \mathsf{U}_{(0,b-\tau)} \varphi_2\right)\right) \quad (4)$$

Besides, the following properties hold [17],

$$\mathsf{F}_{\{\tau\}}(\neg\varphi) = \neg \mathsf{F}_{\{\tau\}} \varphi,\ \mathsf{F}_{\{\tau\}}(\varphi_1 \wedge \varphi_2) = \mathsf{F}_{\{\tau\}} \varphi_1 \wedge \mathsf{F}_{\{\tau\}} \varphi_2, \quad (5a)$$

$$\varphi_0 \mathsf{U}_{(a,b)}(\varphi_1 \vee \varphi_2) = \varphi_0 \mathsf{U}_{(a,b)} \varphi_1 \vee \varphi_0 \mathsf{U}_{(a,b)} \varphi_2, \quad (5b)$$

where φ_0 is an STL formula.

C. Optimization-Based Synthesis for STL Specifications

The STL formulas are used to model the specifications that prescribe the desired behavior of a system. In this paper, we consider a system that is represented as the following continuous-time dynamic model

$$\dot{x}(t) = f(x(t) + u(t)),\ t \in \mathbb{R}_{\geq 0}, \quad (6)$$

where $x(t) \in \mathbb{X}$ and $u(t) \in \mathbb{U}$ are respectively the states and the control input of the system at time t, where $\mathbb{X} \subseteq \mathbb{R}^n$ and $\mathbb{U} \subseteq \mathbb{R}^m$ are the admissible sets of the state and control input, and $f : \mathbb{X} \times \mathbb{U} \to \mathbb{R}^n$ is a smooth vector field. Suppose that the specification of system (6) is described by an STL formula φ. Then, the synthesis of the system can be formulated as the following optimization problem,

$$\max_{\mathbf{u}} \rho(\mathbf{x}, \varphi) \quad (7a)$$

$$\text{s.t. } x_{i+1} = x_i + h f(x_i + u_i), \quad (7b)$$

$$x_i \in \mathbb{X},\ u_i \in \mathbb{U},\ i = 0, 1, \cdots, L-1, \quad (7c)$$

where (7b) is the discrete-time form of the system in (6) with a sampling rate $h \in \mathbb{R}^+$, such that the discretization is performed in a zero-order manner, i.e., $x(\tau) = x_i$, for all $ih \leq \tau < (i+1)h$, $i \in \mathbb{N}_{\geq 0}$, $L \in \mathbb{N}^+$ in (7c) denotes the finite control horizon, and $\mathbf{u} = \{u_0,\ u_1,\ \cdots,\ u_{L-1}\}$ and $\mathbf{x} = \{x_0,\ x_1,\ \cdots,\ x_L\}$ in (7a) are respectively the control and state sequences of the system. The problem in (7) is commonly formulated as a MILP problem and solves an *open-loop* controller $\mathbf{u}$ that attempts to maximize the robust semantics of formula φ, such that the state signal of the system $\mathbf{x}$ satisfies the specification at the maximal extent. The closed-loop solution of the synthesis can be formulated as a model predictive control problem and is introduced in [20], [10].

D. Reachable Set and Largest Satisfaction Region

For a defined STL formula in (1) and a dynamic system in (6), an important concept is the largest satisfaction region (LSR) which prescribes the feasible initial conditions of the system for the STL specification [21]. Before this, we first introduce the reachable set of a dynamic system.

Definition 1: [22] The reachable set of system in (6) at any time $\tau \in \mathbb{R}^+$ from a set of initial states $\mathscr{X}_0 \subset \mathbb{X}$ for all admissible control inputs $\mathbf{u} \in \mathbb{U}^\tau$ is defined as

$$\mathscr{R}_\tau(\mathscr{X}_0) = \{x^{\mathbf{u}}(x_0, \tau) \,|\, x_0 \in \mathscr{X}_0, \mathbf{u} \in \mathbb{U}^\tau\}, \quad (8)$$

where $\mathbb{U}^\tau$ is the set of all admissible control signals from the time 0 to τ and $x^{\mathbf{u}}(x_0, \tau)$ is the system state at time τ with the initial condition x_0 and control signal $\mathbf{u} \in \mathbb{U}^\tau$. □

The reachable set $\mathscr{R}_\tau(\mathscr{X}_0)$ represents all possible states of the system at time τ with the initial conditions starting from $\mathscr{X}_0$. For a deterministic system (6) with a smooth vector field f, $x^{\mathbf{u}}(x_0, t)$ is unique for given x_0 and $\mathbf{u}$. Therefore, any initial state set $\mathscr{X}_0$ corresponds to a reachable set $\mathscr{R}_\tau(\mathscr{X}_0)$ at time τ, and vice versa. We refer to $\mathscr{X}_0 = \mathscr{R}_\tau^{-1}(\mathscr{X}_\tau)$ as the inverse reachable set of $\mathscr{X}_\tau = \mathscr{R}_\tau(\mathscr{X}_0)$ with backward time τ, which represents the largest set of initial states that can reach $\mathscr{X}_\tau$ at time τ, i.e.,

$$\mathscr{R}_\tau^{-1}(\mathscr{X}_\tau) = \{x_0 \in \mathbb{X} \,|\, \exists \mathbf{u} \in \mathbb{U}^\tau,\ \text{s.t. } x^{\mathbf{u}}(x_0, \tau) \in \mathscr{X}_\tau\}. \quad (9)$$

The specification φ is recognized as a constraint exerted on the system state signal $\mathbf{x} \in \mathbb{X}(\mathscr{X}_0, \mathbb{U}^\tau)$, where $\mathbb{X}(\mathscr{X}_0, \mathbb{U}^\tau)$ represents the set of system trajectories starting from the initial state set $\mathscr{X}_0$ and under the admissible control signals in $\mathbb{U}^\tau$. It affects the largest set of system initial states. We define the largest satisfying region (LSR) of system (6) for specification φ as follows.

Definition 2: For the system (6) with a set of initial states $\mathscr{X}_0$ and a specification φ with length $l = \mathscr{L}(\varphi)$, the *admissible control set* is defined as

$$\mathscr{U}(\mathscr{X}_0, \varphi) = \{\mathbf{u} \in \mathbb{U}^l \,|\, \mathbf{x}(x_0, \mathbf{u}) \models \varphi, \forall x_0 \in \mathscr{X}_0\} \quad (10)$$

which includes all admissible control signals such that all system trajectories $\mathbf{x}(x_0, \mathbf{u}) \in \mathbb{X}(\mathscr{X}_0, \mathbb{U}^\tau)$ starting from $x_0 \in \mathscr{X}_0$ and under $\mathbf{u} \in \mathscr{U}(\mathscr{X}_0, \varphi)$ satisfy specification φ. Based on this, the LSR of system (6) and formula φ is defined as

$$S_0(\varphi) = \{x_0 \in \mathbb{X} \,|\, \exists \mathbf{u} \in \mathbb{U}^l, \text{s.t.} \mathbf{x}(x_0, \mathbf{u}) \models \varphi\} \quad (11)$$

which is equivalent to $S_0(\varphi) = \{x_0 | \mathscr{U}(x_0, \varphi) \neq \varnothing\}$. □

Therefore, LSR denotes the largest set of initial conditions of the system subject to a certain STL specification. It is an important concept in the synthesis of LTL and STL specifications. A controller exists if and only if the initial state of the system lies in an LSR. Conventionally, LSR is solved by model-checking-based methods which are computationally expensive when the specification is long and complicated. It is noticed that LSR is very similar to the inverse reachable set defined in (9). In fact, they are the same if $x^{\mathbf{u}}(x_0, \tau) \in \mathscr{X}_\tau$ is formulated as a specification φ. In Sec. III-C, we will explain the connection of LSR and reachable set in detail.

E. Problem Statement

The main problem of this paper is formulated as follows.

Problem 1: Define a class of STL formulas φ such that it can be transformed into a syntactically separate form as (3) with the following objectives achieved.

1). The timing intervals of the subformulas $\varphi_1, \varphi_2, \cdots, \varphi_n$ be sufficiently short and have no overlapping, $n \in \mathbb{N}^+$.

2). Use the LSRs of the subformulas $S_0(\varphi_i)$, $i = 1, 2, \cdots, n$, to compute the LSR of the original formula $S_0(\varphi)$.

3). For system (6) with an initial state $x_0 \in S_0(\varphi)$, solve open-loop controller $\mathbf{u} \in \mathbb{U}^{\mathscr{L}(\varphi)}$, such that $\mathbf{x}(x_0, \mathbf{u}) \models \varphi$. □

The main purpose of Problem 1 is to realize modularized control synthesis for a long and complex STL formula. Objective 1) is necessary to provide the syntactically separated form for the formula. Objective 2) aims at solving the LSR

of the formula in a modularized way, such that the feasible initial states can be determined. Objective 3) describes a normal control synthesis problem. The main goal of this paper is to achieve it also in a modularized manner. The details of our solutions are given in Sec. III.

III. Main Results

In this section, we present the main results of this paper. We first define a class of STL formulas, namely S^2-formulas which ensure no overlappings for the timing intervals of the subformulas. Then, we give a group of separation principles to transform a complex STL formula into this form. Finally, we propose modularized methods to solve the LSR and the open-loop control law for the complex STL formula.

A. Sufficiently Separate Formulas

In this paper, we consider the complex tasks that are formulated as the following STL fragments,

$$\begin{aligned} &\psi := \gamma \,|\, \xi \,|\, \gamma \wedge \xi, \\ &\gamma ::= \gamma_1 \wedge \gamma_2 \,|\, \mathsf{G}_{[a,b]}\varphi,\ \xi ::= \xi_1 \vee \xi_2 \,|\, \mathsf{F}_{[a,b]}\varphi, \end{aligned} \tag{12}$$

where ψ, γ, ξ are respectively referred to as ψ-, γ-, and ξ-class formulas, γ_1, γ_2 are γ-class formulas, ξ_1, ξ_2 are ξ-class formulas, φ are arbitrary STL formulas defined in (1), and $a, b \in \mathbb{R}^+$, $a \leq b$. Therefore, a γ-class formula is the conjunction of several *always* formulas and a ξ-class formula is the disjunction of *eventually* formulas. Besides, a ψ-class formula can be either γ-class, or ξ-class, or the conjunction of the two. We represent a ψ-class formula as follows

$$\psi ::= \textstyle\bigwedge_{i=1}^{n} \mathsf{G}_{[a_i,b_i]}\varphi_i \wedge \bigvee_{j=1}^{\tilde{n}} \mathsf{F}_{[\tilde{a}_j,\tilde{b}_j]}\tilde{\varphi}_j, \tag{13}$$

where $n, \tilde{n} \in \mathbb{R}^+$, $a_i, b_i, \tilde{a}_j, \tilde{b}_j \in \mathbb{R}_{\geq 0}$, $a_i \leq b_i$, $\tilde{a}_j \leq \tilde{b}_j$, and $\varphi_i, \tilde{\varphi}_j$ are STL formulas, $i = 1, 2, \cdots, n$, $j = 1, 2, \cdots, \tilde{n}$. Note that a ξ-class formula can also be represented as (13) by setting $\varphi_i = \top$, for all $i = 1, 2, \cdots, n$. For a γ-class formula, we set any one $\tilde{\varphi}_j = \top$, $j = 1, 2, \cdots, \tilde{n}$, while all others as $\neg\top$.

In (13), we refer to $\mathsf{G}_{[a_i,b_i]}\varphi_i$, $i = 1, 2, \cdots, n$, as a γ-class subformula and $\mathsf{F}_{[\tilde{a}_j,\tilde{b}_j]}\tilde{\varphi}_i$, $j = 1, 2, \cdots, \tilde{n}$ as a ξ-class subformula. They represent the specifications of individual subtasks with individual objectives of a practical complex task. In this sense, a ψ-class formula is capable of prescribing a complex task that contains a sequence of subtasks that are assigned to different timing intervals. However, there may exist overlappings between the timing intervals of different subformulas, i.e., $[a_i, b_i]$ or $[\tilde{a}_i, \tilde{b}_i]$, $i = 1, 2, \cdots, n$, $j = 1, 2, \cdots, \tilde{n}$, which indicates that some of the subtasks have to be executed simultaneously. The overlappings of the timing intervals bring challenges to modularized control synthesis since the corresponding subformulas can not be solved independently. Now, we introduce an ideal syntactically separated formula that eases modularized synthesis. Later, we will discuss how to transfer a ψ-class formula into this form.

Definition 3: A ψ-class formula formulated as (13) is a *sufficiently separated formula (in time)*, or an *S^2-formula* if

1). $n = \tilde{n}$ and $a_i = \tilde{a}_i$, $b_i = \tilde{b}_i$ hold for all $i = 1, 2, \cdots, n$.

2). $(a_i, b_i) \cap (a_j, b_j) = \varnothing$, $\forall i, j = 1, 2, \cdots, n$, $i \neq j$. □

For γ and ξ-class formulas, sufficient separation means that the timing intervals of all subformulas are *sufficiently separated* and have no overlappings. For a ψ-class formula, it requires that its γ-and ξ-class components are all sufficiently separated and have the same number of subformulas. Also, the timing intervals of the corresponding γ-class and ξ-class subformulas $[a_i, b_i]$ are the same for $i = 1, 2, \cdots, n$. Consider the following examples of STL formulas,

$$\begin{aligned} \gamma_1 &= \mathsf{G}_{[0,4]}\varphi_0 \wedge \mathsf{G}_{[2,6]}\varphi_1,\ \xi_1 = \mathsf{F}_{[0,1]}\varphi_0 \vee \mathsf{F}_{[2,6]}\varphi_1, \\ \psi_1 &= \mathsf{G}_{[0,4]}\varphi_0 \wedge \mathsf{F}_{[0,4]}\varphi_1,\ \psi_2 = \mathsf{G}_{[0,1]}\varphi_0 \wedge \mathsf{F}_{[2,6]}\varphi_1, \end{aligned}$$

where φ_0, φ_1 are STL formulas. According to definition 3, ξ_1 and ψ_1 are S^2-formulas. However, γ_1 is not since the timing intervals $(0, 4)$ and $(2, 6)$ of its subformulas have an overlapping interval $(2, 4)$. Neither is ψ_2 since the timing interval of its γ-component $[0, 1]$ is different from its ξ-component $[2, 6]$. In the next subsection, we discuss how to transform a general ψ-class formula into an S^2-form.

B. Separation Principles of STL Formulas

This subsection proposes the syntactic separation principles to transform a ψ-class formula into an S^2-form. We start with the following lemmas.

Lemma 1: The following conditions hold for arbitrary STL formulas φ, φ_1, and φ_2.

1). $\mathsf{F}_{\{\tau\}}\varphi = \mathsf{G}_{\{\tau\}}\varphi$, for any $\tau \in \mathbb{R}_{\geq 0}$, where both sides are true for signal $x : \mathbb{R} \to \mathbb{R}^n$ if and only if $(x, \tau) \models \varphi$.

2). $\mathsf{G}_{\{\tau\}}(\varphi_1 \vee \varphi_2) = \mathsf{G}_{\{\tau\}}\varphi_1 \vee \mathsf{G}_{\{\tau\}}\varphi_2$ holds for any $\tau \in \mathbb{R}_{\geq 0}$.

3). For any $a, b \in \mathbb{R}_{\geq 0}$, $a \leq b$, $\mathsf{G}_{\{\tau\}}\big(\mathsf{G}_{[a,b]}\varphi\big) = \mathsf{G}_{[\tau+a,\tau+b]}\varphi$ and $\mathsf{F}_{\{\tau\}}\big(\mathsf{F}_{[a,b]}\varphi\big) = \mathsf{F}_{[\tau+a,\tau+b]}\varphi$ hold for $\tau \in \mathbb{R}_{\geq 0}$, $\tau < a$. □

Proof: See Appendix. ■

Lemma 2: Given $a, b \in \mathbb{R}_{\geq 0}$, $a \leq b$ and an arbitrary STL formula φ, $\mathsf{G}_{[a,b]}\varphi = \mathsf{G}_{\{a\}}\varphi \wedge \mathsf{G}_{(a,b)}\varphi \wedge \mathsf{G}_{\{b\}}\varphi$ and $\mathsf{F}_{[a,b]}\varphi = \mathsf{F}_{\{a\}}\varphi \vee \mathsf{F}_{(a,b)}\varphi \vee \mathsf{F}_{\{b\}}\varphi$ hold. □

Proof: See Appendix. ■

Lemmas 1 and 2 provide several properties as the complementaries of previous work on syntactic separation. These properties support the following theorem that renders an important principle for the separation of γ- and ξ-subformulas.

Theorem 1: Given $\tau, a, b \in \mathbb{R}_{\geq 0}$, $\tau \leq a \leq b$ and an arbitrary STL formula φ, the following conditions hold,

$$\mathsf{G}_{[a,b]}\varphi = \mathsf{G}_{[a,\tau]}\varphi \wedge \mathsf{G}_{[\tau,b]}\varphi, \tag{14a}$$

$$\mathsf{F}_{[a,b]}\varphi = \mathsf{F}_{[a,\tau]}\varphi \vee \mathsf{F}_{[\tau,b]}\varphi. \tag{14b}$$

Moreover, the following conditions hold,

$$\mathsf{G}_{[\tau_0,\tau_m]}\varphi = \textstyle\bigwedge_{i=1}^{m} \mathsf{G}_{[\tau_{i-1},\tau_i]}\varphi, \tag{15a}$$

$$\mathsf{F}_{[\tau_0,\tau_m]}\varphi = \textstyle\bigvee_{i=1}^{m} \mathsf{F}_{[\tau_{i-1},\tau_i]}\varphi, \tag{15b}$$

where $\tau_0, \tau_1, \cdots, \tau_m \in \mathbb{R}^+$, $\tau_0 < \tau_1 < \cdots < \tau_m$. □

Proof: See Appendix. ■

Theorem 1 provides two important separation principles for γ- and ξ-class subformula. It can be used to separate a subformula into an arbitrary number of subformulas. By properly selecting the separation time points, we can restrict the length of the timing intervals of the subformulas, which

solves Problem 1-1). The following proposition combines the subformulas with overlapping timing intervals.

Proposition 1: Given $a,b\in\mathbb{R}^+$, $a\le b$,

$$\mathsf{G}_{[a,b]}(\varphi_1\wedge\varphi_2)=\mathsf{G}_{[a,b]}\varphi_1\wedge\mathsf{G}_{[a,b]}\varphi_2. \tag{16a}$$

$$\mathsf{F}_{[a,b]}(\varphi_1\vee\varphi_2)=\mathsf{F}_{[a,b]}\varphi_1\vee\mathsf{F}_{[a,b]}\varphi_2, \tag{16b}$$

where φ_1 and φ_2 are arbitrary STL formulas. □

Proof: See Appendix. ■

Theorem 1 and Proposition 1 provide a method to separate subformulas and combine the ones with overlapping timing intervals. The method allows a ψ-class formula to be transformed into an S^2-formula. The transformed formula has the same semantics as the original formula since all the separation principles do not affect the soundness of the formulas. An example of such transformation will be shown in the case study in Sec. IV. From an intuitive perspective, these principles seem straightforward if the formulas are recognized as spatiotemporal constraints. Nevertheless, we provide formal proof for these separation principles, which is important for the formal inference of complex STL formulas.

C. Modularized Solution of Largest Satisfying Regions

Using the separation principles provided in Sec. III-B, we can transform any ψ-class formula into an S^2-form. This subsection discusses how to solve the LSR of a complex ψ-class formula using the LSRs of its S^2 subformulas. To ease the interpretation, we define a novel concept τ-LSR as follows.

Definition 4: For system (6) with an STL specification φ, the τ-LSR for any $\tau\in\mathbb{R}^+$, $0\le\tau<l$ is defined as $S_\tau(\varphi)=\{x^{\mathbf{u}}(x_0,\tau)\,|\,\exists\mathbf{u}\in\mathbb{U}^l, x_0\in\mathbb{X}, \text{s.t.}\,\mathbf{x}(x_0,\mathbf{u})\models\varphi\}$. □

Lemma 3: Given $a,b\in\mathbb{R}_{\ge0}$, $a\le b$, for the dynamic system in (6) and arbitrary STL formulas φ_1, φ_2, the following conditions hold, if $S_0(\varphi_1)\neq\varnothing$, and $S_0(\varphi_2)\neq\varnothing$.

$$S_0(\varphi_1\vee\varphi_2)=\check{S}_0,\ S_0(\varphi_1\wedge\varphi_2)=\tilde{S}_0(\varphi_1)\cap\tilde{S}_0(\varphi_2)\subset\hat{S}_0.$$

where $\check{S}_0=S_0(\varphi_1)\cup S_0(\varphi_2)$, $\hat{S}_0=S_0(\varphi_1)\cap S_0(\varphi_2)$, $\tilde{S}_0(\varphi_1)=\{x_0\in\hat{S}_0\,|\,\exists\mathbf{u}\in\mathbb{U}^l, \text{s.t.}\,\mathbf{x}(x_0,\mathbf{u})\models\varphi_1\}$, and $\tilde{S}_0(\varphi_2)=\{x_0\in\hat{S}_0\,|\,\exists\mathbf{u}\in\mathbb{U}^l, \text{s.t.}\,\mathbf{x}(x_0,\mathbf{u})\models\varphi_2\}$. □

Proof: See Appendix. ■

This lemma provides an important principle to obtain the LSR of the disjunction or conjunction of two STL formulas from their own LSRs. It is interesting to see that the LSR of the disjunction of two formulas equals the union of their individual LSRs, which, however, does not apply to the conjunction case. Instead, the LSR of conjunction is even smaller than the intersection of the LSRs of the individual formulas. The following lemma gives the relationship between LSR and τ-LSR and builds their connection with reachability.

Lemma 4: For system as (6) and an STL formula φ, $S_0(\varphi)=S_\tau\big(\mathsf{F}_{\{\tau\}}\varphi\big)=\mathscr{R}_\tau\big(S_0\big(\mathsf{F}_{\{\tau\}}\varphi\big)\big)$ holds for $\tau\in\mathbb{R}^+$. □

Proof: See Appendix. ■

This lemma reveals the role of τ-LSR as a bridge that connects the LSR and the reachable set of the system and the specification. It provides the foundation for the following theorem on modularized LSR solution for S^3-formulas.

Theorem 2: For an S^2-γ- or S^2-ξ-class formula in the form of (13) and prescribed in Definition 3, its LSR reads

$$S_0(\xi)=\bigcup_{i=1}^n S_0(\xi_i)=\bigcup_{i=1}^n\mathscr{R}_{a_i}^{-1}(S_0(\hat{\xi}_i)), \tag{17a}$$

$$S_0(\gamma)=\bigcap_{i=1}^n\tilde{S}_0(\gamma_i)=\bigcap_{i=1}^n\mathscr{R}_{a_i}^{-1}(\tilde{S}_0(\hat{\gamma}_i)), \tag{17b}$$

where $\xi_i=\mathsf{F}_{[a_i,b_i]}\varphi_i$, $\gamma_i=\mathsf{G}_{[a_i,b_i]}\varphi_i$, $\hat{\xi}_i=\mathsf{F}_{[0,b_i-a_i]}\varphi_i$, $\hat{\gamma}_i=\mathsf{G}_{[0,b_i-a_i]}\varphi_i$, φ_i are arbitrary STL formulas, and $\tilde{S}_0(\gamma_i)=\{x_0\in\cap_{j=1}^n S_0(\gamma_j)\,|\,\exists\mathbf{u}\in\mathbb{U}^l, \text{s.t.}, \mathbf{x}(x_0,\mathbf{u})\models\gamma_i,\}$, $i=1,2,\cdots,n$. □

Proof: See Appendix. ■

This theorem provides a solution to compute the LSR for a γ- or ξ-class formula using the LSRs of its subformulas. Suppose that ξ and γ are long and complex STL formulas that contain a finite number of short and simple subformulas ξ_i and γ_i, $i=1,2,\cdots,n$. Moreover, the subformulas can be transformed into the ones with timing intervals starting with 0, i.e., $\hat{\xi}_i$ and $\hat{\gamma}_i$ which are easier to be solved. Then, the LSR of the original STL formula can be solved using a finite number of set operations and inverse reachability computations, which greatly improves the efficiency compared to the conventional model-checking-based approaches. For a ψ-class formula $\psi=\xi\wedge\gamma$, its LSR can be calculated as

$$S_0(\psi)=\tilde{S}_0(\xi)\cap\tilde{S}_0(\gamma), \tag{18}$$

according to Lemma 3. An example showing how to use this method to calculate the LSR of a complex ψ-class formula will be shown in the case study in Sec. IV.

D. Modularized Synthesis of an ψ-class formula

Until now, we have introduced the separation and combination principles to transform an ψ-class formula into an S^3-form. We also proposed a method to compute the LSR of an S^2-formula using the LSRs of its subformulas. If a complex STL specification is represented in an S^2-form, its synthesis problem can be performed in an efficient and modularized manner. Specifically, the synthesis can be performed for individual subformulas one by one in the order of time, instead of directly for the entire formula. The synthesis for each subformula only generates a partition of the open-loop control sequence and the system trajectory. The sufficient separation of the complex STL ensures that, at any time, the system synthesis can incorporate as few subformulas as possible. In this subsection, we present how to use these principles to synthesize an open-loop control law for system (6) with an arbitrary ψ-class formula in three steps.

1) Step 1: Formula Separation: For any ψ-class formula, we perform the transformation for its γ- and ξ-component separately. For any γ- or ξ-class formula, we first use Theorem 1 to split its subformulas with overlapping intervals into several subformulas with shorter intervals. Then, we combine the subformulas that share the same overlapping intervals using Proposition 1. For ψ-class formulas, we may flexibly use $\mathsf{G}_I\top$ and $\mathsf{F}_I\neg\top$ to generate the paired form in (13), where $I\in 2^{\mathbb{R}_{\ge0}}$ is a proper interval. The resulting formula is in an S^2-form.

2) Step 2: Solving LSR: Before using the ψ-class formula to synthesize the system controller, we need to solve its LSR to find out the feasible initial conditions. The initial condition of the system must be placed within the LSR of the complete formula to guarantee the feasibility of the synthesis problem. We first compute the LSRs of all subformulas using model-checking methods. This process is efficient for subformulas with short timing horizons. Then, we use Theorem 2 to compute the LSR of a γ- or a ξ-class formula. Then, the LSR of a ψ-class formula can be solved using (18).

3) Step 3: Modularized Synthesis: For an S^2-formula in form of (13), at any time $\tau \in \mathbb{R}^+$, we define its head until its j-th subformula as

$$\bar{\psi}^j(\tau) ::= \bigwedge_{i=1}^{j} \mathsf{G}_{[a_i-\tau,b_i-\tau]}\varphi_i \wedge \bigvee_{i=1}^{j} \mathsf{F}_{[a_i-\tau,b_i-\tau]}\tilde{\varphi}_i,$$

and define its tail from its j-th subformula as

$$\vec{\psi}^j(\tau) ::= \bigwedge_{i=j}^{n} \mathsf{G}_{[a_i-\tau,b_i-\tau]}\varphi_i \wedge \bigvee_{i=j}^{n} \mathsf{F}_{[a_i-\tau,b_i-\tau]}\tilde{\varphi}_i.$$

Specially, we define $\bar{\psi}^j = \bar{\psi}^j(a_j)$ and $\vec{\psi}^j = \vec{\psi}^j(a_j)$. Then, for any time $\tau = a_j$, $j = 1,2,\cdots,n-1$, the synthesis of the system can be performed only for the head of the formula $\bar{\psi}^j$ while ensuring that the system state at $\tau = a_{j+1}$ falls into the LSR of the tail $\vec{\psi}^{j+1}$, i.e., $x(a_{j+1}) \in S_0(\vec{\psi}^{j+1})$, such that there exist control sequences such that the successive state trajectory satisfy the tail specification $\vec{\psi}^{j+1}$. This forms the most important principle for modularized synthesis. Note that the synthesis of $\bar{\psi}^j$ is usually quite efficient since the past-time temporal logic does not need to be incorporated.

There is an issue with the ξ-class component of formula ψ. If there exists any time $\tau = a_j$, such that $\bar{\psi}^j$ is satisfied by the existing system trajectory, we can state that the ξ-class component of ψ is satisfied and set all $\tilde{\varphi}_r = \neg\top$, $r = j+1, j+2, \cdots, n$, and remove all subformulas $\mathsf{F}_{[a_r,b_r]}\tilde{\varphi}_r$ from the tail $\vec{\psi}^{j+1}$. This can help reduce the computation for solving the τ-LSR $S_{a_{j+1}}(\vec{\psi}^{j+1})$. Based on these principles, we propose Algorithm 1 for modularized synthesis, where *int()* transforms any value to its closest integer, *sys_optimize()* is a function that solves problem (7), and *append()* attaches its second parameter to the end of its first parameter. Note that we prescribe $a_{n+1} = \mathscr{L}(\psi)$ and $\vec{\psi}^{n+1} = \top$.

Algorithm 1 Modularized Specification Synthesis

Input: Specification ψ, LSR $S_0(\psi)$, sampling time h, synthesis timing points a_1, a_2, $\cdots$, a_n, a_{n+1}, heads $\bar{\psi}^1$, $\bar{\psi}^2$, $\cdots$, $\bar{\psi}^n$, tails $\vec{\psi}^2$, $\vec{\psi}^3$, $\cdots$, $\vec{\psi}^{n+1}$

Output: Control signal $\mathbf{u}$

$x_0 \in S_0(\psi)$
for $j = 1$ **to** n **do**
  $L = \mathrm{int}((a_{j+1} - a_j)/h)$
  Compute $S_0(\vec{\psi}^{j+1})$ using Theorem 2
  $\varphi = \bar{\psi}^j \wedge x_{L-1} \in S_0(\vec{\psi}^{j+1})$
  $(\mathrm{FEASIBLE}, \tilde{\mathbf{u}}) = \mathrm{sys\_optimize}(x_0, L, \varphi)$
  if FEASIBLE **then**
    $x_0 = x^{\tilde{\mathbf{u}}}(x_0, a_{j+1} - a_j)$
    if $\tilde{\varphi}_j \neq \neg\top$ **then**
      $\tilde{\varphi}^j \leftarrow \top$
      for $r = j+1$ **to** n **do**
        $\tilde{\varphi}^r \leftarrow \neg\top$
      end for
    end if
    $\mathbf{u} \leftarrow \mathrm{append}(\mathbf{u}, \tilde{\mathbf{u}})$
  else
    $\tilde{\varphi}_j \leftarrow \neg\top$, $j = j-1$
  end if
end for

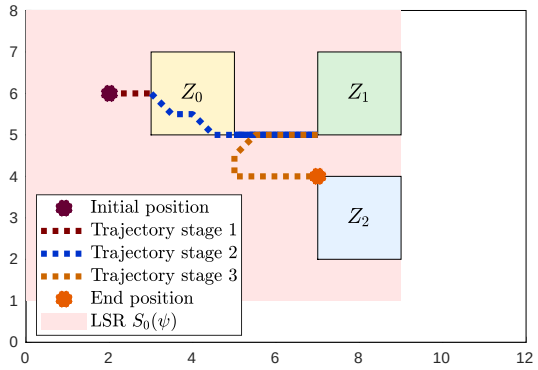

Fig. 1. The illustration of the robot monitoring scenario with the spatial information of the synthesized trajectory subject to specification ψ.

IV. Case Study in Simulation

In this section, we use a simple simulation to showcase how to use our proposed separation principles and modularized synthesis approaches to efficiently solve a synthesis problem for a complex specification. We consider a scenario where a robot is required to perform a monitoring task in a rectangular space $\mathscr{Z} \subset \mathbb{R}^2$ sized $12\,\mathrm{m} \times 8\,\mathrm{m}$, as shown in Fig. 1. Z_0, Z_1, and Z_2 are square regions in the space with a side length $2\,\mathrm{m}$, respectively representing TARGET, HOME, and CHANGER. The centers of the regions are $z_0 = (4,6)\,\mathrm{m}$, $z_1 = (8,6)\,\mathrm{m}$, and $z_2 = (8,3)\,\mathrm{m}$.

The motion of the robot is depicted as the following single-integrator dynamic model,

$$\dot{\zeta}(t) = u(t),\ t \in \mathbb{R}_{\geq 0}, \tag{19}$$

where $\zeta(t) = [\zeta_x, \zeta_y]^\top \in \mathbb{R}^2$ represents the planar coordinate of the robot position at time t, where $\zeta_x, \zeta_y \in \mathbb{R}$ are respectively the x- and y-axis coordinates, $u(t) = [u_x, u_y]^\top \in \mathbb{R}^2$ is the linear velocity of the robot as the control input of the system at time t, with $u_x, u_y \in \mathbb{R}$ being the x- and y-axis components. The control input of the system is subject to saturation constraints $|u_x| \leq 1\,\mathrm{m/s}$, $|u_y| \leq 1\,\mathrm{m/s}$. The monitoring task is described in natural language as follows.

1). Until 30 s, the robot should frequently visit TARGET Z_0. The duration between two visits should be less than 5 s.

2). From 15 s to 45 s, once the robot leaves HOME Z_1, it should get back to HOME Z_1 within 5 s.

3). Before 45 s, its should stay in CHANGER Z_2 continuously for at least 3 s to charge its battery.

These task specifications can be formulated as an overall ψ-class formula $\psi = \mathsf{G}_{[0,30]}\varphi_0 \wedge \mathsf{G}_{[15,45]}\varphi_1 \wedge \mathsf{F}_{[0,45]}\varphi_2$, where $\varphi_0 = \mathsf{F}_{[0,5]}(\zeta \in Z_0)$, $\varphi_1 = \neg(\zeta \in Z_1) \rightarrow \mathsf{F}_{[0,5]}(\zeta \in Z_1)$, $\varphi_2 = \mathsf{G}_{[0,3]}(\zeta \in Z_2)$ are STL formulas. Then, $\psi = \gamma \wedge \xi$ has a γ-class and a ξ-class components, $\gamma = \mathsf{G}_{[0,30]}\varphi_0 \wedge \mathsf{G}_{[15,45]}\varphi_1$, $\xi = \mathsf{F}_{[0,45]}(\zeta \in Z_1)$.

1) Step 1: Formula Separation: Applying the separation principles in Sec. III-B, we rewrite γ and ξ as

$$\begin{aligned} \gamma &= \mathsf{G}_{[0,15]}\varphi_0 \wedge \mathsf{G}_{[15,30]}(\varphi_0 \wedge \varphi_1) \wedge \mathsf{G}_{[30,45]}\varphi_1, \\ \xi &= \mathsf{F}_{[0,15]}\varphi_2 \vee \mathsf{F}_{[15,30]}\varphi_2 \vee \mathsf{F}_{[30,45]}\varphi_2. \end{aligned} \tag{20}$$

Then, $\psi = \gamma \wedge \xi$ is in an S^2-form.

2) Step 2: Solving LSR: The form (20) can be formulated as $\gamma = \hat{\gamma}_0 \wedge \mathsf{G}_{\{15\}}(\hat{\gamma}_0 \wedge \hat{\gamma}_1) \wedge \mathsf{G}_{\{30\}}\hat{\gamma}_1$, $\xi = \hat{\xi}_0 \vee \mathsf{F}_{\{15\}}\hat{\xi}_0 \vee \mathsf{F}_{\{30\}}\hat{\xi}_0$, where $\hat{\gamma}_0 = \mathsf{G}_{[0,15]}\varphi_0$, $\hat{\gamma}_1 = \mathsf{G}_{[0,15]}\varphi_1$, $\hat{\xi}_0 = \mathsf{F}_{[0,15]}\varphi_2$. Therefore, the LSRs of γ and ξ are calculated as

$$\begin{aligned} S_0(\gamma) &= \tilde{S}_0(\hat{\gamma}_0) \cap \mathscr{R}_{15}^{-1}\big(\tilde{S}_0(\hat{\gamma}_0 \wedge \hat{\gamma}_1)\big) \cap \mathscr{R}_{30}^{-1}\big(\tilde{S}_0(\tilde{\gamma}_1)\big) \\ &= \{\zeta \in \mathscr{Z} \,|\, \|\zeta - z_0\| \le 5\}, \\ S_0(\xi) &= S_0(\hat{\xi}_0) \cup \mathscr{R}_{15}^{-1}(S_0(\hat{\xi}_0)) \cup \mathscr{R}_{30}^{-1}(S_0(\hat{\xi}_0)) = \mathscr{Z}. \end{aligned}$$

Therefore, considering the space size, the LSR of ψ reads

$$S_0(\psi) = \tilde{S}_0(\gamma) \cap \tilde{S}_0(\xi) = \big\{\zeta \,\big|\, 0 \le \zeta_x \le 9,\, 1 \le \zeta_y \le 8\big\}$$

which is marked as a red region in Fig. 1. Therefore, the robot must start from $S_0(\psi)$ to ensure the satisfaction of ψ.

3) Step 3: Modularized Synthesis: We use Algorithm 1 to synthesis an open-loop control signal for system (19) and specification ψ. The synthesis timing points are 0 s, 15 s, and 30 s, corresponding to three synthesis stages. The sampling time $h = 0.5$. The BluSTL toolbox [23] is used to implement the *sys_optimize()* method in Algorithm 1. The resulting robot trajectory $\zeta(t) = [\zeta_x(t), \zeta_y(t)]$ are shown in Fig. 1 and Fig. 2. The trajectories in different stages are marked with different colors. The synthesis procedure is introduced as follows.

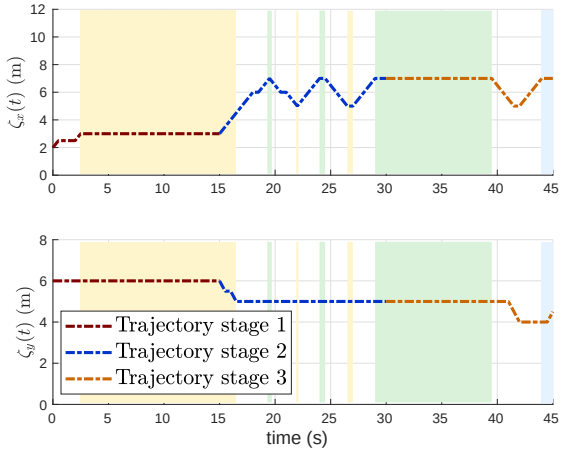

Fig. 2. The x- and y-axes of the robot trajectory $\zeta(t)$ as time changes.

• Stage 1: at $t = 0$ s, we set initial state as $x_0 = (2,2)$ m, $L = 30$, $\varphi = \bar{\psi}^1 \wedge x_{L-1} \in S_0(\bar{\psi}^2)$, $\bar{\psi}^1 = \hat{\gamma}_0 \wedge \hat{\xi}_0$, $\bar{\psi}^2 = \hat{\gamma}_0 \wedge \hat{\gamma}_1 \wedge \mathsf{G}_{[0,15]}\hat{\gamma}_1 \wedge (\hat{\xi}_0 \vee \mathsf{G}_{[0,15]}\hat{\xi}_0)$, and $S_0(\bar{\psi}^2) = \{\zeta \in \mathbb{R}^2 | 2 \le \zeta_x \le 9,\, 1 \le \zeta_y \le 8\}$. We use $\mathbf{u} = sys\_optimize(x_0, L, \varphi)$ to solve the optimal open-loop controller. Note that the problem is infeasible unless we set $\bar{\psi}^1 = \hat{\gamma}_0 \wedge \neg\top$. From Fig. 1 and Fig. 2, it is seen that the robot starts from the initial position x_0, moves towards TARGET Z_0, and stays on its boundary at $(3,6)$ m at 15 s. Such a behavior satisfies the specification ψ before 15 s, as prescribed by the language description 1).

• Stage 2: at $t = 15$ s, we know the robot trajectory in the last stage ends at $(3,6)$ m which is selected as the initial position in this stage. We determine $L = 30$, $\varphi = \bar{\psi}^2 \wedge x_{L-1} \in S_0(\bar{\psi}^3)$, where $\bar{\psi}^2 = \varphi_0 \wedge \hat{\gamma}_0 \wedge \hat{\gamma}_1 \wedge \hat{\xi}_0$, $\bar{\psi}^3 = \hat{\gamma}_1 \wedge \hat{\xi}_0$, and $S_0(\bar{\psi}^3) = \{\zeta \in \mathbb{R}^2 | 2 \le \zeta_x \le 12,\, 1 \le \zeta_y \le 7\}$. We use $\mathbf{u} = sys\_optimize(x_0, L, \varphi)$ to solve the optimal open-loop controller. The problem is also infeasible unless we set $\bar{\psi}^2 = \varphi_0 \wedge \hat{\gamma}_0 \wedge \hat{\gamma}_1 \wedge \neg\top$. From Fig. 1 and Fig. 2, it is seen that the robot starts from the initial position $x_0 = (3,6)$ m at 15 s, moves across TARGET Z_0 and towards HOME Z_1, and finally oscillates between Z_0 and Z_1. Such a behavior satisfies the specification ψ between 15 s and 30 s, as prescribed by both language descriptions 1) and 2).

• Stage 3: at $t = 30$ s, Fig. 2 shows that the robot ends at $(7,5)$ m, i.e., the left bottom corner of Z_1, as shown in Fig. 1. Thus, we set this point as the initial position of the robot in this stage. Also, $L = 30$, $\varphi = \bar{\psi}^3 = \varphi_0 \wedge \varphi_1 \wedge \hat{\gamma}_1 \wedge (\varphi_2 \vee \hat{\xi}_0)$. We use $\mathbf{u} = sys\_optimize(x_0, L, \varphi)$ to solve the optimal open-loop controller and the solution is feasible. From Fig. 1 and Fig. 2, it is seen that the robot starts from the initial position $x_0 = (7,5)$ m at 30 s and ends in CHARGER Z_2 ultimately, which satisfies the specification ψ after 30 s, as prescribed by both language descriptions 2) and 3).

Fig. 1 illustrates the spatio information of the robot trajectory $\zeta(t)$ from 0 s to 45 s, and Fig. 2 shows its temporal information during this period. Both figures indicate that the robot trajectory satisfies the overall complex specification ψ. Moreover, we notice that the computation time of our proposed modularized synthesis method is far shorter than directly solving the overall specification, which also verifies the efficiency of modularized synthesis.

V. CONCLUSIONS

In this paper, we discuss how to split a big optimization problem for the synthesis of a complex STL specification into several small optimization problems which are concerned with the separate subformulas. Our proposed separate principles ensure the syntactic equivalence between the original complex formula and its separate counterpart. We also use the LSR of the original specification to prescribe the soundness and feasibility of its separation form. Modularized methods are proposed to efficiently solve the LSR and the open-loop controller, which is the first step towards efficient optimization-based specification synthesis. There are still two limitations of our work. One is that we only investigate the separation of a certain class of STL formulas, i.e., the ones only with *always* and *eventually* temporal operators, although they are sufficient to prescribe most of the practical tasks. The other is we only synthesize open-loop controllers in this

paper. Our future work will extend our results in this paper to closed-loop controllers and more general STL formulas.

APPENDIX

This section presents the proof of all lemmas, propositions, and theorems of this paper.

A. Proof of Lemma 1

For 1), from the definition of STL syntax, we know, for an arbitrary STL formula φ, $\mathsf{F}_{\{\tau\}}\varphi = \top\mathsf{U}_{\{\tau\}}\varphi$ indicates that there must exist $k \in \{\tau\}$, such that $(x,k) \models \varphi$, which means $(x,\tau) \models \varphi$. Then, from $\mathsf{G}_{\{\tau\}}\varphi = \neg\mathsf{F}_{\{\tau\}}\neg\varphi$ and (5a), we know $\mathsf{G}_{\{\tau\}}\varphi \leftrightarrow \neg\left(\neg\mathsf{F}_{\{\tau\}}\varphi\right) = \mathsf{F}_{\{\tau\}}\varphi$.

For 2), according to (5a), we have $\mathsf{F}_{\{\tau\}}(\varphi_1 \vee \varphi_2) = \neg\mathsf{F}_{\{\tau\}}(\neg\varphi_1 \wedge \neg\varphi_2) = \neg(\mathsf{F}_{\{\tau\}}\neg\varphi_1 \wedge \mathsf{F}_{\{\tau\}}\neg\varphi_2)$. Then, using (5a), we obtain $\mathsf{F}_{\{\tau\}}(\varphi_1 \vee \varphi_2) = \neg(\neg\mathsf{F}_{\{\tau\}}\varphi_1 \wedge \neg\mathsf{F}_{\{\tau\}}\varphi_2) = \mathsf{F}_{\{\tau\}}\varphi_1 \vee \mathsf{F}_{\{\tau\}}\varphi_2$, which also holds for G, i.e., $\mathsf{G}_{\{\tau\}}(\varphi_1 \vee \varphi_2) = \mathsf{G}_{\{\tau\}}\varphi_1 \vee \mathsf{G}_{\{\tau\}}\varphi_2$, according to condition 1) of this lemma.

For 3), for $\tau < a$, we have

$$\begin{aligned}\mathsf{F}_{\{\tau\}}\left(\mathsf{F}_{[a,b]}\varphi\right) &= \top\mathsf{U}_{\{\tau\}}\left(\top\mathsf{U}_{[a,b]}\varphi\right)\\ &= \top\mathsf{U}_{[\tau+a,\tau+b]}\varphi = \mathsf{F}_{[\tau+a,\tau+b]}\varphi.\end{aligned} \tag{21}$$

Applying (5a) to $\mathsf{F}_{\{\tau\}}\mathsf{G}_{(a,b)}\varphi$, we have $\mathsf{F}_{\{\tau\}}\mathsf{G}_{(a,b)}\varphi = \mathsf{F}_{\{\tau\}}\left(\neg\mathsf{F}_{(a,b)}\neg\varphi\right) = \neg\mathsf{F}_{\{\tau\}}\left(\mathsf{F}_{(a,b)}\neg\varphi\right)$. Considering (21), we have $\mathsf{F}_{\{\tau\}}\mathsf{G}_{(a,b)}\varphi = \neg\mathsf{F}_{(\tau+a,\tau+b)}\neg\varphi = \mathsf{G}_{(\tau+a,\tau+b)}\varphi$. According to condition 1), we know,

$$\mathsf{G}_{\{\tau\}}\mathsf{G}_{(a,b)}\varphi = \mathsf{F}_{\{\tau\}}\mathsf{G}_{(a,b)}\varphi = \mathsf{G}_{(\tau+a,\tau+b)}\varphi. \tag{22}$$

Therefore, principle 3) is proved by (21) and (22).

B. Proof of Lemma 2

Applying the principle in (4) to formulate formula $\mathsf{F}_{[a,b]}\varphi$, we have

$$\begin{aligned}\mathsf{F}_{[a,b]}\varphi = \top\mathsf{U}_{[a,b]}\varphi &= \mathsf{F}_{\{a\}}\varphi \vee \top\mathsf{U}_{(a,b)}\varphi \vee \top\mathsf{U}_{\{b\}}\varphi\\ &= \mathsf{F}_{\{a\}}\varphi \vee \mathsf{F}_{(a,b)}\varphi \vee \mathsf{F}_{\{b\}}\varphi.\end{aligned} \tag{23}$$

Then, applying (23) to $\mathsf{G}_{[a,b]}\varphi$ we obtain

$$\begin{aligned}\mathsf{G}_{[a,b]}\varphi &= \neg\mathsf{F}_{[a,b]}\neg\varphi = \neg\left(\mathsf{F}_{\{a\}}\neg\varphi \vee \mathsf{F}_{(a,b)}\neg\varphi \vee \mathsf{F}_{\{b\}}\neg\varphi\right)\\ &= \neg\mathsf{F}_{\{a\}}\neg\varphi \wedge \neg\mathsf{F}_{(a,b)}\neg\varphi \wedge \neg\mathsf{F}_{\{b\}}\neg\varphi\\ &= \mathsf{G}_{\{a\}}\varphi \wedge \mathsf{G}_{(a,b)}\varphi \wedge \mathsf{G}_{\{b\}}\varphi.\end{aligned} \tag{24}$$

Thus, this lemma is proved by (23) and (24).

C. Proof of Theorem 1

Substituting $\varphi_1 = \top$ and $\varphi_2 = \varphi$ to (4), we have

$$\begin{aligned}\top\mathsf{U}_{(a,b)}\varphi &= \top\mathsf{U}_{(a,\tau)}\varphi \vee \left(\mathsf{G}_{(a,\tau)}\top \wedge \mathsf{F}_{\{\tau\}}\left(\varphi \vee \top\mathsf{U}_{(0,b-\tau)}\varphi\right)\right),\\ &= \mathsf{F}_{(a,\tau)}\varphi \vee \mathsf{F}_{\{\tau\}}\left(\varphi \vee \mathsf{F}_{(0,b-\tau)}\varphi\right).\end{aligned}$$

Applying properties 1) and 3) in Lemma 1, we obtain

$$\begin{aligned}\mathsf{F}_{(a,b)}\varphi &= \top\mathsf{U}_{(a,b)}\varphi = \mathsf{F}_{(a,\tau)}\varphi \vee \mathsf{F}_{\{\tau\}}\varphi \vee \mathsf{F}_{\{\tau\}}\mathsf{F}_{(0,b-\tau)}\varphi\\ &= \mathsf{F}_{(a,\tau)}\varphi \vee \mathsf{F}_{\{\tau\}}\varphi \vee \mathsf{F}_{(\tau,b)}\varphi.\end{aligned} \tag{25}$$

Substituting (25) to (23), we obtain

$$\begin{aligned}\mathsf{F}_{[a,b]}\varphi &= \mathsf{F}_{(a,\tau)}\varphi \vee \mathsf{F}_{\{\tau\}}\varphi \vee \mathsf{F}_{(\tau,b)}\varphi\\ &= \mathsf{F}_{\{a\}}\varphi \vee \mathsf{F}_{(a,\tau)}\varphi \vee \mathsf{F}_{\{\tau\}}\varphi \vee \mathsf{F}_{(\tau,b)}\varphi \vee \mathsf{F}_{\{b\}}\varphi\\ &= \mathsf{F}_{[a,\tau]}\varphi \vee \left(\mathsf{F}_{\{\tau\}}\varphi \vee \mathsf{F}_{(\tau,b)}\varphi \vee \mathsf{F}_{\{b\}}\right) = \mathsf{F}_{[a,\tau]}\varphi \vee \mathsf{F}_{[\tau,b]}\varphi.\end{aligned} \tag{26}$$

Then, applying it to $\mathsf{G}_{[a,b]}\varphi$, we obtain

$$\begin{aligned}\mathsf{G}_{[a,b]}\varphi &= \neg\mathsf{F}_{(a,b)}\neg\varphi = \neg\left(\mathsf{F}_{[a,\tau]}\neg\varphi \vee \mathsf{F}_{[\tau,b]}\neg\varphi\right)\\ &= \neg\mathsf{F}_{[a,\tau]}\neg\varphi \wedge \neg\mathsf{F}_{[\tau,b]}\neg\varphi = \mathsf{G}_{[a,\tau]}\varphi \wedge \mathsf{G}_{[\tau,b]}\varphi.\end{aligned} \tag{27}$$

Thus, (26) and (27) prove (14). Also, (15) can be proved by recursively applying (14) to the separating time points τ_0, τ_1, $\cdots$, τ_m.

D. Proof of Proposition 1

Substituting $\varphi_0 = \top$ to (5b), we have $\mathsf{F}_{[a,b]}(\varphi_1 \vee \varphi_2) = \top\mathsf{U}_{[a,b]}(\varphi_1 \vee \varphi_2) = \top\mathsf{U}_{[a,b]}\varphi_1 \vee \top\mathsf{U}_{[a,b]}\varphi_2 = \mathsf{F}_{[a,b]}\varphi_1 \vee \mathsf{F}_{[a,b]}\varphi_2$. Then, applying it to $\mathsf{G}_{[a,b]}(\varphi_1 \wedge \varphi_2)$, we have

$$\begin{aligned}&\mathsf{G}_{[a,b]}(\varphi_1 \wedge \varphi_2) = \neg\mathsf{F}_{[a,b]}\neg(\varphi_1 \wedge \varphi_2)\\ &= \neg\mathsf{F}_{[a,b]}(\neg\varphi_1 \vee \neg\varphi_2) = \neg\left(\mathsf{F}_{[a,b]}\neg\varphi_1 \vee \mathsf{F}_{[a,b]}\neg\varphi_2\right)\\ &= \neg\left(\mathsf{F}_{[a,b]}\neg\varphi_1\right) \wedge \neg\left(\mathsf{F}_{[a,b]}\neg\varphi_2\right) = \mathsf{G}_{[a,b]}\varphi_1 \wedge \mathsf{G}_{[a,b]}\varphi_2\end{aligned}$$

which proves this proposition.

E. Proof of Lemma 3

According to the definition of LSR, we know

$$\begin{aligned}S_0(\varphi_1 \vee \varphi_2) &= \{x_0 \in S_0(\varphi_1) \cup S_0(\varphi_2) \,|\, \mathscr{U}(x_0, \varphi_1 \vee \varphi_2) \neq \varnothing\}\\ &= \mathscr{U}(S_0(\varphi_1), \varphi_1 \vee \varphi_2) \cup \mathscr{U}(S_0(\varphi_2), \varphi_1 \vee \varphi_2),\\ S_0(\varphi_1 \wedge \varphi_2) &= \{x_0 \in S_0(\varphi_1) \cap S_0(\varphi_2) \,|\, \mathscr{U}(x_0, \varphi_1 \wedge \varphi_2) \neq \varnothing\}\\ &= \mathscr{U}(S_0(\varphi_1), \varphi_1 \wedge \varphi_2) \cap \mathscr{U}(S_0(\varphi_2), \varphi_1 \wedge \varphi_2),\end{aligned}$$

where $\mathscr{U}(S_0(\varphi_1), \varphi_1 \vee \varphi_2) = S_0(\varphi_1) \cup \tilde{S}_0(\varphi_1)$, $\mathscr{U}(S_0(\varphi_2), \varphi_1 \wedge \varphi_2) = S_0(\varphi_2) \cap \tilde{S}_0(\varphi_2)$, where $\tilde{S}_0(\varphi_1)$ and $\tilde{S}_0(\varphi_2)$ are defined in Lemma 3. Note that $\tilde{S}_0(\varphi_1) \subset S_0(\varphi_1)$ and $\tilde{S}_0(\varphi_2) \subset S_0(\varphi_2)$. Thus, we obtain $S_0(\varphi_1 \vee \varphi_2) = S_0(\varphi_1) \cup S_0(\varphi_2)$ and $S_0(\varphi_1 \wedge \varphi_2) = \tilde{S}_0(\varphi_1) \cap \tilde{S}_0(\varphi_2)$, which proves the lemma.

F. Proof of Lemma 4

According to Definition 4, $S_\tau(\mathsf{F}_{\{\tau\}}\varphi)$ reads,

$$S_\tau(\mathsf{F}_{\{\tau\}}\varphi) = \{x_\tau \in \mathbb{X} \,|\, \exists \mathbf{u} \in \mathbb{U}^{\tau+l}, \text{s.t.}\, \mathbf{x}(x_0, \mathbf{u}) \models \mathsf{F}_{\{\tau\}}\varphi\}, \tag{28}$$

where $\tau + l$ denotes the length of formula $\mathsf{F}_{\{\tau\}}\varphi$. According to condition 1) of Lemma 1, we know the equivalence,

$$\mathbf{x}(x_0, \mathbf{u}) \models \mathsf{F}_{\{\tau\}}\varphi \leftrightarrow (\mathbf{x}(x_0, \mathbf{u}), \tau) \models \varphi \leftrightarrow \mathbf{x}(x_\tau, \mathbf{u}^\tau) \models \mathsf{F}_{\{\tau\}}\varphi, \tag{29}$$

where x_τ is the system state at τ with initial condition x_0 and control signal $\mathbf{u}$, $\mathbf{u}^\tau$ is the partition of control signal $\mathbf{u}$ after time τ, and $\mathbf{x}(x_\tau, \mathbf{u}^\tau)$ is the system trajectory with initial state x_τ and control signal $\mathbf{u}^\tau$. Thus, (28) can be rewritten as

$$S_\tau\left(\mathsf{F}_{\{\tau\}}\varphi\right) = \{x_\tau \in \mathbb{X} \,|\, \exists \mathbf{u}^\tau \in \mathbb{U}^l, \text{s.t.}\, \mathbf{x}(x_\tau, \mathbf{u}^\tau) \models \varphi\}. \tag{30}$$

In fact, (11) and (30) refer to the same set, only with the difference of symbols, which renders $S_0(\varphi) = S_\tau(\mathsf{F}_{\{\tau\}}\varphi)$.

Let us now inspect $S_0(\mathsf{F}_{\{\tau\}}\varphi)$ which reads

$$S_0\left(\mathsf{F}_{\{\tau\}}\varphi\right) = \{x_0 \in \mathbb{X} \,|\, \exists \mathbf{u} \in \mathbb{U}^{\tau+l}, \text{s.t.}\, \mathbf{x}(x_0, \mathbf{u}) \models \mathsf{F}_{\{\tau\}}\varphi\}.$$

Substituting (29) to it, we have

$$\begin{aligned}S_0\left(\mathsf{F}_{\{\tau\}}\varphi\right) = \{&x_0 \in \mathbb{X} \,|\, \exists \mathbf{u}^0 \in \mathbb{U}^\tau, \mathbf{u}^\tau \in \mathbb{U}^l,\\ &\text{s.t.}\, \mathbf{x}(x_\tau, \mathbf{u}^\tau) \models \mathsf{F}_{\{\tau\}}\varphi, x_\tau = x^{\mathbf{u}^0}(x_0, \tau)\}.\end{aligned} \tag{31}$$

According to the definition of the inverse reachable set in (9), we know that (31) implies $S_0\big(\mathsf{F}_{\{\tau\}}\varphi\big) = \mathscr{R}_\tau^{-1}\big(S_\tau\big(\mathsf{F}_{\{\tau\}}\varphi\big)\big)$. Therefore, this lemma is proved.

G. Proof of Theorem 2

Let us first inspect the case of a ξ-class formula. Then, Lemma 3 implies

$$S_0(\xi) = \bigcup_{i=1}^{n} S_0(\xi_i)\,. \tag{32}$$

Note that $\xi_i = \mathsf{F}_{\{a_i\}}\hat{\xi}_i$. According to Lemma 4, we have

$$S_0(\xi_i) = \mathscr{R}_{a_i}^{-1}\Big(S_0(\hat{\xi}_i)\Big)\,,\ i = 1,2,\cdots,n. \tag{33}$$

Substituting (33) to (32), we prove (17a).

Then, we inspect the case of a γ-class formula. Lemma 3 implies

$$S_0(\gamma) = \bigcup_{i=1}^{n} \tilde{S}_0(\gamma_i)\,. \tag{34}$$

Similar to the ξ-class formula, due to $\gamma_i = \mathsf{G}_{\{a_i\}}\hat{\gamma}_i$, we have

$$\tilde{S}_0\big(\mathsf{F}_{[a_i,b_i]}\varphi_i\big) = \mathscr{R}_{a_i}^{-1}(\gamma_i)\,,\ i = 1,2,\cdots,n. \tag{35}$$

Substituting (35) to (34), we prove (17b).

REFERENCES

[1] H. Abbas, G. Fainekos, S. Sankaranarayanan, F. Ivančić, and A. Gupta, "Probabilistic temporal logic falsification of cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 2s, pp. 1–30, 2013.

[2] O. A. Beg, L. V. Nguyen, T. T. Johnson, and A. Davoudi, "Signal temporal logic-based attack detection in dc microgrids," *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 3585–3595, 2018.

[3] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges," *AI communications*, vol. 29, no. 1, pp. 151–162, 2016.

[4] X. Li, G. Rosman, I. Gilitschenski, C.-I. Vasile, J. A. DeCastro, S. Karaman, and D. Rus, "Vehicle trajectory prediction using generative adversarial network with temporal logic syntax tree features," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3459–3466, 2021.

[5] 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.

[6] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[7] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Formal Modeling and Analysis of Timed Systems: 8th International Conference, FORMATS 2010, Klosterneuburg, Austria, September 8-10, 2010. Proceedings 8*. Springer, 2010, pp. 92–106.

[8] S. S. Farahani, V. Raman, and R. M. Murray, "Robust model predictive control for signal temporal logic synthesis," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 323–328, 2015.

[9] L. Lindemann, G. J. Pappas, and D. V. Dimarogonas, "Reactive and risk-aware control for signal temporal logic," *IEEE Transactions on Automatic Control*, vol. 67, no. 10, pp. 5262–5277, 2021.

[10] A. Salamati, S. Soudjani, and M. Zamani, "Data-driven verification of stochastic linear systems with signal temporal logic constraints," *Automatica*, vol. 131, p. 109781, 2021.

[11] S. Alatartsev, S. Stellmacher, and F. Ortmeier, "Robotic task sequencing problem: A survey," *Journal of intelligent & robotic systems*, vol. 80, pp. 279–298, 2015.

[12] 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.

[13] M. Srinivasan and S. Coogan, "Control of mobile robots using barrier functions under temporal logic specifications," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 363–374, 2020.

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

[15] S. Liu, A. Saoud, P. Jagtap, D. V. Dimarogonas, and M. Zamani, "Compositional synthesis of signal temporal logic tasks via assume-guarantee contracts," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 2184–2189.

[16] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2817–2830, 2012.

[17] P. Hunter, J. Ouaknine, and J. Worrell, "Expressive completeness for metric temporal logic," in *2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*. IEEE, 2013, pp. 349–357.

[18] K. Bae and J. Lee, "Bounded model checking of signal temporal logic properties using syntactic separation," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–30, 2019.

[19] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems: Joint International Conferences on Formal Modeling and Analysis of Timed Systmes, FORMATS 2004, and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004, Grenoble, France, September 22-24, 2004. Proceedings*. Springer, 2004, pp. 152–166.

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

[21] C. Belta, B. Yordanov, E. Aydin Gol, C. Belta, B. Yordanov, and E. Aydin Gol, "Largest satisfying region," *Formal Methods for Discrete-Time Dynamical Systems*, pp. 119–139, 2017.

[22] 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.

[23] V. Raman and A. Donze, "Blustl: A model predictive control toolbox with singal temporal logic constraints," 2015. [Online]. Available: https://github.com/BluSTL/BluSTL