ANNUAL REVIEWS

Annual Review of Control, Robotics, and Autonomous Systems

Formal Methods for Control Synthesis: An Optimization Perspective

Calin Belta[1] and Sadra Sadraddini[2]

[1]Department of Mechanical Engineering, Boston University, Boston, Massachusetts 02215, USA; email: cbelta@bu.edu

[2]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA; email: sadra@mit.edu

Annu. Rev. Control Robot. Auton. Syst. 2019. 2:115–40

First published as a Review in Advance on December 10, 2018

The *Annual Review of Control, Robotics, and Autonomous Systems* is online at control.annualreviews.org

https://doi.org/10.1146/annurev-control-053018-023717

Keywords

formal methods, temporal logics, mathematical programming, model predictive control

Abstract

In control theory, complicated dynamics such as systems of (nonlinear) differential equations are controlled mostly to achieve stability. This fundamental property, which can be with respect to a desired operating point or a prescribed trajectory, is often linked with optimality, which requires minimizing a certain cost along the trajectories of a stable system. In formal verification (model checking), simple systems, such as finite-state transition graphs that model computer programs or digital circuits, are checked against rich specifications given as formulas of temporal logics. The formal synthesis problem, in which the goal is to synthesize or control a finite system from a temporal logic specification, has recently received increased interest. In this article, we review some recent results on the connection between optimal control and formal synthesis. Specifically, we focus on the following problem: Given a cost and a correctness temporal logic specification for a dynamical system, generate an optimal control strategy that satisfies the specification. We first provide a short overview of automata-based methods, in which the dynamics of the system are mapped to a finite abstraction that is then controlled using an automaton corresponding to the specification. We then provide a detailed overview of a class of methods that rely on mapping the specification and the dynamics to constraints of an optimization problem. We discuss advantages and limitations of these two types of approaches and suggest directions for future research.

1. INTRODUCTION

Temporal logics, such as computation tree logic and linear temporal logic (LTL), have customarily been used to specify the correctness of computer programs and digital circuits modeled as finite-state transition systems (1). The problem of analyzing such a model against a temporal logic formula, known as formal analysis or model checking, has received significant attention during the past 40 years, and several efficient algorithms and software tools are available (2, 3). The formal synthesis problem, in which the goal is to design or control a system from a temporal logic specification, was not studied extensively until a few years ago. Recent results include the use of model checking algorithms to control deterministic systems (4), automata games for controlling nondeterministic systems (5), and linear programming and value iteration for the synthesis of control policies for Markov decision processes (1, 6). Through the use of abstractions, such techniques have also been used for infinite systems, such as continuous- and discrete-time linear systems (7–12).

Optimal control is a mature research area and a well-used technique in applications. For a finite (weighted) deterministic transition system (i.e., a graph for which available transitions can be deterministically chosen at every node), the classical control problem is finding a shortest path between two nodes, for which there are efficient algorithms (13). For a finite purely nondeterministic system (i.e., one in which an action at a state enables several transitions, and their probabilities are not known), a controller optimizing a cost expressed using transition weights can be found through fix-point techniques (14). For finite Markov decision processes, the classical control problem is the stochastic shortest-path problem, where an optimal policy minimizing the expected value of a cumulative discounted cost is minimized by using value iteration or linear programming (15). For systems with infinite state and control sets, optimal control problems usually involve costs that penalize the deviation of the state from a reference trajectory and the control effort. For linear systems and quadratic costs, the problem involves solving a Riccati equation. For all the control problems enumerated above, there exist receding-horizon [also called model predictive control (MPC)] versions (16, 17). To account for time-varying objectives, in MPC, the optimal control problem is solved at every time step over a finite horizon, and the optimal action is applied only at the current time.

The connection between optimal and temporal logic control is an intriguing problem with a potentially high impact in several applications. The goal of combining these two seemingly unrelated areas is to optimize the behavior of a system subject to correctness constraints. Consider, for example, an autonomous vehicle involved in a persistent surveillance mission in a disaster relief application, where dynamic service requests can only be sensed locally in a neighborhood around the vehicle (see **Figure 1**). The goal is to accomplish the mission while maximizing the likelihood of servicing the local requests, and possibly minimizing the energy spent during the motion. The correctness requirement can be expressed as a temporal logic formula (see the caption of **Figure 1**), while the resource constraints translate to minimizing a cost over the feasible trajectories of the robot.

Current works on combining optimality and correctness can be roughly divided into two main classes: automata-based methods and optimization-based methods. Automata-based methods are based on the observation that an LTL formula can be mapped to an automaton in such a way that the language accepted by the automaton is exactly the language satisfying the formula. Depending on the desired expressivity of the specification language, these automata can be well-known finite-state automata (the acceptance condition is reaching a set of final states), Büchi automata (the acceptance condition is reaching a set of final states infinitely often), or, in the most general case, Rabin automata (the acceptance condition is to visit a set of good states infinitely often and a set of bad states finitely many times) (1). The control problem reduces to a game on the product

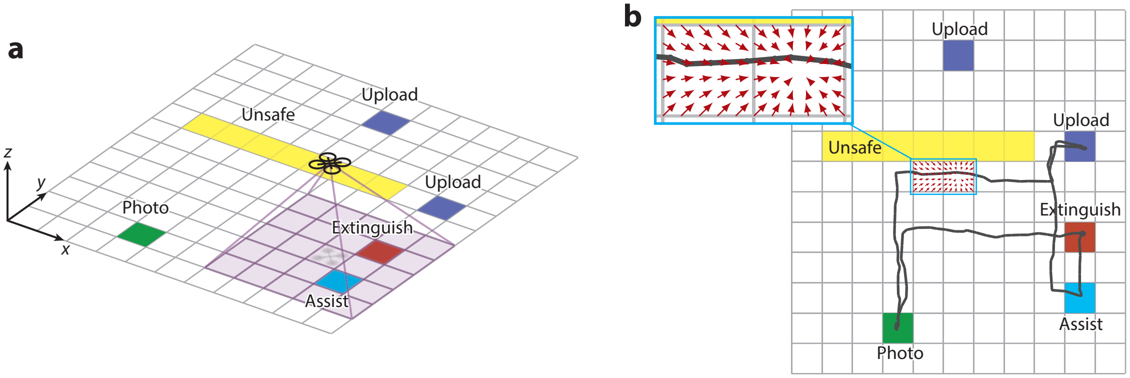

Figure 1

(*a*) An autonomous air vehicle is deployed from a high-level, temporal logic global specification over a set of static, known requests ("photo" and "upload") occurring at the regions of a known environment—e.g., "keep taking photos and upload the current photo before taking another photo." This specification translates to the following LTL formula: **GF** photo ∧ **G** (photo → (photo **U**(¬photo **U** upload))), where **G**, **F**, and **U** are the temporal operators "globally" (always), "future" (eventually), and "until," and ∧, →, and ¬ are Boolean operators for conjunction, implication, and negation, respectively. While moving in the environment, the vehicle can locally sense dynamically changing events such as survivors and fires, which generate (local) service requests, and unsafe areas, which need to be avoided. The goal is to accomplish the global mission while maximizing the likelihood of servicing the local requests and staying away from unsafe areas. (*b*) Through the use of an accurate quadrotor kinematic model, input–output linearizations or flat outputs, precise state information from a motion capture system, and control-to-facet results for linear and multiaffine systems, this problem can be (conservatively) mapped to a control problem for a finite transition system. This system can be deterministic or nondeterministic if single- or multiple-facet controllers in the output space are used, respectively. Figure adapted from Reference 18.

between a finite system, such as a transition system or Markov decision process, which can be the abstraction of an infinite deterministic or stochastic system, and the automaton obtained from the specification. The winning condition, which ensures correctness, is the Rabin (Büchi, finite-state automaton) acceptance condition of the automaton. For finite systems, the cost can be the average reward or cost per stage and adapted objectives that reflect the semantics of the temporal logic, such as the average reward or cost per cycle. For infinite systems, current works consider costs that penalize the deviation of the states and controls from desired reference trajectories. The main limitation of automata-based methods is the prohibitive computational complexity, which stems from the complexity of the game for the finite systems and the exponential blowup of the partition-based abstraction for infinite systems. In addition, due to the partition-based abstraction, such methods are, in general, conservative for infinite systems. In this article, we provide only a short review of these methods in Section 2.

Central to optimization-based methods are temporal logics with semantics over finite-time signals, such as signal temporal logic (STL) (19) and metric temporal logic (20). In addition to Boolean semantics, in which signals either satisfy or violate a formula, such logics have quantitative semantics, which allow one to assess the robustness of satisfaction. The starting point for this class of methods is the observation that Boolean satisfaction can be mapped to the feasibility part of an optimization problem. A cost that penalizes the deviation from a desired system trajectory of energy consumption can be combined with the robustness of satisfaction to obtain the overall objective of the optimization. Therefore, the main advantage of these types of methods is the seamless combination of correctness and optimality, with the added feature of robustness to satisfaction. The second advantage is scalability. We provide a detailed review of this class of methods in Section 5.

Throughout this article, we assume that the state of the system is fully observable. Even though most of the results in this article are valid for both discrete- and continuous-time dynamical systems, we restrict our attention to discrete-time systems.

2. AUTOMATA-BASED TEMPORAL LOGIC OPTIMAL CONTROL

2.1. Finite Systems

For weighted deterministic transition systems and specifications given as LTL formulas over a set of propositions labeling the states of the system, Smith et al. (4) considered a cost function quantifying the time between satisfying instances of a single proposition. This cost function is motivated by problems in monitoring and data gathering, such as the one described in **Figure 1**. The solution starts with converting the LTL specification to a Büchi automaton. The transition system is then synchronized with the Büchi automaton to create a product automaton. In this automaton, a satisfying run is any run that visits a set of accepting states infinitely often. It can be shown that there exists an optimal run that is in a prefix–suffix structure, implying that it is enough to search for runs with a finite transient state followed by a periodic steady state. A polynomial-time graph algorithm based on solutions of bottleneck shortest-path problems is used to find an optimal cycle containing an accepting state. Jing et al. (21) considered a related version of this problem, with particular application to robot motion planning in adversarial environments. Svorenova et al. (22) showed that this problem can also be solved for the case in which the deterministic transition system incurs time-varying penalties modeled as Markov chains. The cost was the expected average cumulative penalty incurred between consecutive satisfactions of a desired property, and the specification was a general LTL formula.

Ding et al. (6) considered a probabilistic version of this optimal synthesis problem. The specifications were given as LTL formulas over a set of propositions assigned to the states of a Markov decision process, and a control policy that minimized the expected cost between satisfying instances of an optimizing proposition over all policies that maximize the probability of satisfying the given LTL specification was derived. The Markov-decision-process optimization problem was formulated in terms of minimizing the average cost per cycle, where cycles are defined by successive satisfactions of the optimizing proposition. A connection was established between this problem and the well-known average-cost-per-stage problem, and it was shown that a dynamic programming algorithm can be used to produce provably correct, optimal solutions.

Ding et al. (23) established a connection between temporal logic control synthesis and receding-horizon optimal control for finite deterministic systems. The authors assumed that the specification was given as an LTL formula and that deterministic rewards could be sensed locally around the current state during the execution of the system. They then derived a control strategy in the form of an infinite iteration of a receding-horizon controller, which they computed based on local reward information. One limitation of this approach is that once an optimal sequence of controls over a finite horizon was chosen, it was applied at every time step to the end of the horizon. This issue was later addressed by Ding et al. (24) in a paper that developed a truly MPC strategy—i.e., only the first action in the sequence was applied. In addition to proving the correctness of the control strategy, this paper established some interesting automata-theoretic equivalents of basic concepts from MPC, such as recursive feasibility and terminal constraints.

2.2. Infinite Systems

Automata-based approaches for temporal logic optimal control of systems with infinite sets of states and/or controls are, in general, hierarchical, two-level methods. The bottom level is a continuous-to-continuous abstraction procedure, in which the possibly large state space and complex dynamics are mapped to a low-dimensional output space with simple dynamics. The most used techniques are input–output linearization and differential flatness. For a differentially flat system, its state and control variables can be expressed as a function of its outputs and its derivatives. The top level is a partition-based, continuous-to-discrete abstraction procedure, in which the output and control spaces are partitioned. The partition is driven by the specification (25) or by a prescribed accuracy of the approximation (26). The quotient of the partition is a finite system that is in some way equivalent to the original, infinite system. The most used notion of equivalence is bisimulation. **Figure 1** provides an example. The 12-dimensional quad-rotor dynamics of the quad-rotor shown in **Figure 1*a*** are differentially flat with four flat outputs (position and yaw), and up to four derivatives of the flat output are necessary to compute the original state and input. **Figure 1*b*** shows a two-dimensional section of the partition of the four-dimensional output space, together with the assignment of a vector field in two adjacent cells. Note that the dynamics corresponding to these vector fields treat all the states in a cell in the same way: In the cell in **Figure 1*a***, all the states will leave in finite time through the right facet, and in the cell in **Figure 1*b***, all states will stay inside for all times. Informally, these examples correspond to the bisimilarity equivalence mentioned above. The quotient of the partition is a finite transition system that is controlled from the temporal logic specification. The cost can penalize the execution time (as in 14) or traveled distance (as in 10, 18).

The method described above is conservative. If a solution (of the automaton game) is not found at the top level, this does not mean that a controller does not exist for the original continuous system. Intuitively, the partition, and therefore the abstraction, might be too rough. Conservativeness can be reduced by refining the partition. Rungger & Reissig (27) showed that optimality can made arbitrarily precise by refining the state partition and proved optimality bounds. Gol et al. (25) took this approach for discrete-time linear systems, specifications given as syntactically cosafe LTL formulas over linear predicates in the state of the system, and quadratic costs penalizing the Euclidean distance from desired trajectories. **Figure 2** shows an example illustrating the compromise between correctness and optimality.

The expensive process of constructing the abstraction was avoided by Papusha et al. (28) for deterministic systems and by Horowitz et al. (29) for stochastic systems. Specifically, Papusha et al. (28) formulated a dynamic programming problem over the product of the continuous-time, continuous-state system and the specification automaton. They proposed an approximate-dynamic-programming approach for controller synthesis for both linear and nonlinear systems. Wolff et al. (30) used a similar approach for autonomous driving and robotic surveillance tasks. Li & Fu (31) considered a sampling-based policy iteration for optimal planning for a subclass of LTL specifications.

3. SPECIFICATIONS

STL was originally introduced by Maler & Nickovic (19) to reason about continuous-time real signals, and it can be viewed as a version of metric temporal logic (20) specifically tailored for real signals. An n-dimensional real signal is $\mathbf{s} : \mathbb{T} \to \mathbb{R}^n$, where $\mathbb{T}$ is the time domain: $\mathbb{R}_+$ for continuous-time signals or $\mathbb{N}$ for discrete-time signals. We use s_t to refer to the value at time $t \in \mathbb{T}$, and $(\mathbf{s}, [t_1 : t_2])$ to refer the portion of $\mathbf{s}$ in $[t_1, t_2], t_1, t_2 \in \mathbb{T}, t_2 \geq t_1$. We use the shorthand notation $(\mathbf{s}, t) := (\mathbf{s}, [t, \infty))$ for a signal suffix. The syntax of STL is defined as

$$\varphi ::= \pi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \,|\varphi|\, \varphi_1 \mathbf{U}_{[t_1,t_2]} \varphi_2, \qquad 1.$$

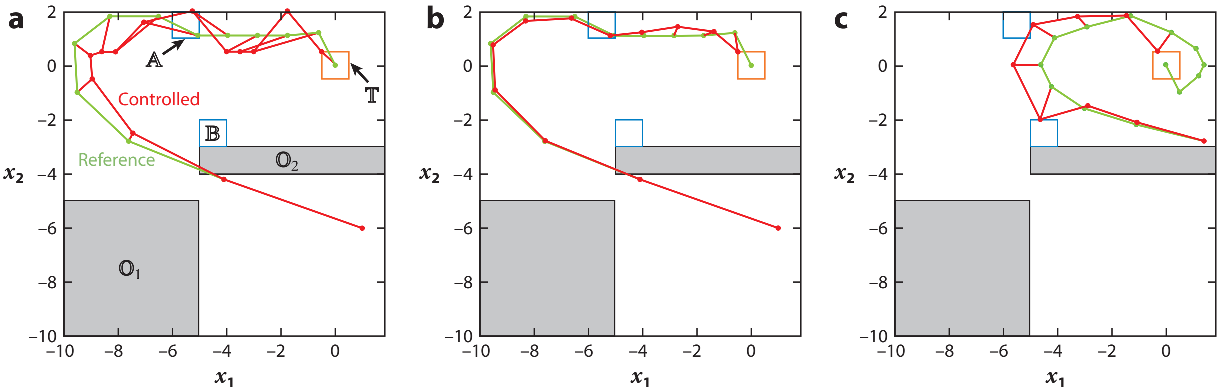

Figure 2

A planar discrete-time double-integrator system is required to satisfy the correctness specification "visit region $\mathbb{A}$ or region $\mathbb{B}$ and then the target region $\mathbb{T}$, while always avoiding obstacles $\mathbb{O}_1$ and $\mathbb{O}_2$ and staying inside the safe region $\mathbb{X}$ (the bounding box)," which translates to the syntactically cosafe linear temporal logic formula $((\neg\mathbb{O}_1 \wedge \neg\mathbb{O}_2 \wedge \mathbb{X})\, \mathbf{U}\, \mathbb{T}) \wedge (\neg\mathbb{T}\, \mathbf{U}\, (\mathbb{A} \vee \mathbb{B}))$. At the same time, the system should follow a desired reference-state trajectory that is available to the system over a short finite-time horizon N. The reference trajectory is shown in green (satisfying the correctness specification in panels *a* and *b*, and violating it in panel *c*), and the trajectory of the controlled system is shown in red. Pairs of points on the reference and controlled trajectories corresponding to the same time are connected. The cases shown in panels *a* and *b* correspond to increasing values of the horizon N for the same reference trajectory. Note that, in the situation shown in panel *c*, where the reference trajectory violates the correctness specification, the controller tries to compromise between correctness and optimality. Figure adapted from Reference 25.

where π is a predicate in the form $\pi = (p(s) \geq c)$ or $\pi = (p(s) \leq c)$, where $p : \mathbb{R}^n \to \mathbb{R}$ and $c \in \mathbb{R}$; $\neg$ and $\wedge$ denote Boolean negation and conjunction connectives, respectively; and $\mathbf{U}_{[t_1,t_2]}$ is a bounded temporal "until" operator with $t_1 \geq t_2$. Other Boolean operators are defined in the usual way. Additional temporal logic operators can be constructed: temporal "eventually" is $\mathbf{F}_{[t_1,t_2]}\varphi := \text{True}\mathbf{U}_{[t_1,t_2]}\varphi$, and temporal "always" is $\mathbf{G}_{[t_1,t_2]}\varphi := \neg(\mathbf{F}_{[t_1,t_2]}\neg\varphi)$. STL semantics is defined over suffixes of signals as follows.

Definition 1. The STL semantics is recursively defined as follows:

- $(\mathbf{s}, t) \models (p(s) \geq c) \Leftrightarrow p(s_t) \geq c$, $(\mathbf{s}, t) \models (p(s) \leq c) \Leftrightarrow p(s_t) \leq c$;
- $(\mathbf{s}, t) \models \neg\varphi \Leftrightarrow (\mathbf{s}, t) \not\models \varphi$;
- $(\mathbf{s}, t) \models \varphi_1 \wedge \varphi_2 \Leftrightarrow (\mathbf{s}, t) \models \varphi_1 \wedge (\mathbf{s}, t) \models \varphi_2$;
- $(\mathbf{s}, t) \models \varphi_1 \vee \varphi_2 \Leftrightarrow (\mathbf{s}, t) \models \varphi_1 \vee (\mathbf{s}, t) \models \varphi_2$;
- $(\mathbf{s}, t) \models \varphi_1 \mathbf{U}_{[t_1,t_2]} \varphi_2 \Leftrightarrow \exists t' \in [t_1, t_2], (\mathbf{s}, t') \models \varphi_2 \wedge \forall t'' \in [t_1, t'](\mathbf{s}, t'') \models \varphi_1$;
- $(\mathbf{s}, t) \models \mathbf{F}_{[t_1,t_2]}\varphi \Leftrightarrow \exists t' \in [t_1, t_2], (\mathbf{s}, t') \models \varphi$; and
- $(\mathbf{s}, t) \models \mathbf{G}_{[t_1,t_2]}\varphi \Leftrightarrow \forall t' \in [t_1, t_2], (\mathbf{s}, t') \models \varphi$,

where $(\mathbf{s}, t) \models \varphi$ is read as "signal suffix $(\mathbf{s}, t)$ satisfies STL formula φ."

Definition 2 (from Reference 32). The STL score, also known as the STL robustness degree (19), of a signal suffix $(\mathbf{s}, t)$ with respect to an STL formula φ is recursively defined as follows:

- $\rho(\mathbf{s}, (p(s) \geq c), t) = p(s_t) - c$, $\rho(\mathbf{s}, (p(s) \leq c), t) = c - p(s_t)$;
- $\rho(\mathbf{s}, \neg\varphi, t) = -\rho(\mathbf{s}, \varphi, t)$;
- $\rho(\mathbf{s}, \varphi_1 \wedge \varphi_2, t) = \min\left(\rho(\mathbf{s}, \varphi_1, t), \rho(\mathbf{s}, \varphi_2, t)\right)$;
- $\rho(\mathbf{s}, \varphi_1 \vee \varphi_2, t) = \max\left(\rho(\mathbf{s}, \varphi_1, t), \rho(\mathbf{s}, \varphi_2, t)\right)$;
- $\rho(\mathbf{s}, \mathbf{F}_{[t_1,t_2]}\varphi, t) = \max_{k \in [t_1,t_2]} \rho(\mathbf{s}, \varphi, t)$;

- $\rho(\mathbf{s}, \mathbf{G}_{[t_1,t_2]}\varphi, t) = \min_{k\in[t_1,t_2]} \rho(\mathbf{s}, \varphi, t)$; and
- $\rho(\mathbf{s}, \varphi_1 \mathbf{U}_{[t_1,t_2]} \varphi_2, t) = \max_{t'\in[t_1,t_2]} \min \left\{\rho(\mathbf{s}, \varphi, t'), \min_{t''\in[t_1,t)} \rho(\mathbf{s}, \varphi, t'')\right\}$.

The STL score is a measure of how strongly a formula is satisfied by a signal. Positive robustness indicates satisfaction, and negative robustness indicates violation. It is straightforward to show that any temporal logic formula can be brought into the negation normal form (1), where all negation connectives appear immediately before a predicate. We can further remove the negations of predicates by simply reversing the inequalities—e.g., $\neg(p(s) \geq c) = (p(s) \leq c)$. Note that we do not consider strict inequalities when performing computations for verification or synthesis. For the case that the formula is negation free, we simply consider a zero score to be satisfaction.

Definition 3 (from Reference 33). The bound (also known as the horizon) of an STL formula φ, denoted by b^{φ}, is the time length required to evaluate the satisfaction of φ and is recursively computed as follows:

- $b^{\pi} = 0$;
- $b^{\neg\varphi} = b^{\varphi}$;
- $b^{\varphi_1 \wedge \varphi_2} = b^{\varphi_1 \vee \varphi_2} = \max(b^{\varphi_1}, b^{\varphi_2})$;
- $b^{\mathbf{F}_{[t_1,t_2]}\varphi} = b^{\mathbf{G}_{[t_1,t_2]}\varphi} = t_2 + b^{\varphi}$; and
- $b^{\varphi_1 \mathbf{U}_{[t_1,t_2]} \varphi_2} = t_2 + \max(b^{\varphi_1}, b^{\varphi_2})$,

where π is a predicate and ϕ, ϕ_1, and ϕ_2 are STL formulas.

The satisfaction of φ by $(\mathbf{s}, t)$ is decided only by $(\mathbf{s}, [t : t + b^{\varphi}])$, and the rest of the signal is irrelevant. Thus, instead of $(\mathbf{s}, t) \models \varphi$, we can write $(\mathbf{s}, [t : t + b^{\varphi}]) \models \varphi$.

Example 1. Consider a discrete-time signal $\mathbf{s} : \mathbb{N} \rightarrow \mathbb{R}$, where $s_t = t, t \in \mathbb{N}$. Let $\varphi = \mathbf{G}_{[0,3]}\mathbf{F}_{[0,2]}(s^2 \leq 10)$. We have $b^{\varphi} = 3 + 2 = 5$. We have $\rho(\mathbf{s}, \varphi, 0) = \min(\max(10-0, 10-1, 10-4), \max(10-1, 10-4, 10-9), \max(10-4, 10-9, 10-16), \max(10-9, 10-16, 10-25)) = \min(10, 9, 6, 1) = 1$ (satisfaction), but we have $\rho(\mathbf{s}, \varphi, 1) = \min(9, 6, 1, -6) = -6$ (violation).

Definition 4. The language of an STL formula φ is defined as $L(\varphi) := \{\mathbf{s} | (\mathbf{s}, [0, b^{\varphi}]) \models \varphi\}$.

Definition 5. An STL formula φ is bounded if $b^{\varphi} < \infty$.

Definition 6 (from Reference 34). A safety STL formula is one in which, when it is written in the negation normal form, all of its "until" and "eventually" operators are bounded.

Safety formulas can be satisfied by infinite-time signals and violated by finite-time signals (34). Safety formulas are ubiquitous in applications. Moreover, any nonsafety STL formula φ in the negation normal form that contains unbounded "eventually" or "until" operators can be approximated by a safety STL formula φ^{safe} by replacing the unbounded intervals with bounded intervals while maintaining $L(\varphi^{\text{safe}}) \subset L(\varphi)$. However, underapproximation of the unbounded "always" operator with a bounded one does not satisfy this property.

There are several methods for efficiently computing the STL score (35, 36). Computations are effectively performed for discrete-time signals. Given assumptions on signal continuity and Lipschitz constants, bounds for the difference between the score evaluated in discrete time and the actual one in continuous time can be provided (37). For verification and synthesis purposes, we generally need to discretize time to obtain a finite number of decision variables. In the remainder of this article, we focus on discrete-time signals and systems.

4. SYSTEMS

In this section, we introduce a broad class of discrete-time hybrid models and revisit the classical result of Heemels et al. (38) on the equivalence of them. We consider discrete-time systems as

$$x_{t+1} = F(x_t, u_t, w_t), \tag{2.}$$

where $t, t \in \mathbb{N}$ is time; $x_t \in X$ is the state, $X \subset \mathbb{R}^{n_\mathrm{r}} \times \{0,1\}^{n_\mathrm{b}}$; $u_t \in U$ is the control input, $U \subset \mathbb{R}^{m_\mathrm{r}} \times \{0,1\}^{m_\mathrm{b}}$; and $w_t \in W$ is the disturbance (environment input), $W \subset \mathbb{R}^{q_\mathrm{r}} \times \{0,1\}^{q_\mathrm{b}}$. Note that both X and U may include real and binary values. For example, the set of controls in a traffic signal is modeled as binary: 0 for a red light, and 1 for a green light. The system represented by Equation 2 is deterministic if W is a singleton (typically $W = \{0\}$).

4.1. Hybrid Models

In this section, we provide definitions for multiple families of hybrid systems.

Definition 7 (from Reference 39). The system represented by Equation 2 is in piecewise affine (PWA) form if

$$x_{t+1} = A_i x_t + B_i u_t + c_i + w_t, (x_t, u_t, w_t) \in \mathcal{H}_i, i = 1, \dots, \mathcal{H}_{n_\mathrm{M}}, \tag{3.}$$

where n_M is the number of modes; $\mathcal{H}_i$'s are interior-disjoint polyhedral sets, $\bigcup_{i=1}^{n_\mathrm{M}} \mathcal{H}_i = X \times U \times W$; and $A_i, B_i, c_i, i = 1, \dots, n_\mathrm{M}$ are constant matrices with appropriate dimensions.

PWA models with additive disturbances are able to capture all the behaviors of any nonlinear system. In order to reduce conservativeness, the number of modes is increased. If $n_\mathrm{M} = 1$, then the system represented by Equation 3 is reduced to a linear system with additive disturbances.

Definition 8 (from Reference 40). A max-min-plus-scaling (MMPS) system has the following syntax:

$$F ::= F_{\text{affine}} | \max(F_1, F_2) | \min(F_1, F_2) | F_1 + F_2 | \alpha F, \tag{4.}$$

where F_{affine} stands for linear systems of the form $x_{t+1} = Ax_t + B_u u_t + B_w w_t + c, \alpha \in \mathbb{R}$, and F_1 and F_2 are MMPS systems.

Using min and max operators, MMPS systems can handle discontinuities. MMPS models are common in systems with saturation constraints, such as traffic networks.

Definition 9 (from Reference 41). Linear complementarity (LC) systems are of the following form:

$$x_{t+1} = Ax_t + B_u u_t + B_w w_t + B_r r_t, \tag{5a.}$$

$$y_t = (E_x x_t + E_u u_t + E_w w_t + E_r r_t + e), y_t^T r_t = 0, y_t, r_t \geq 0, \tag{5b.}$$

where $y_t, r_t \in \mathbb{R}^{n_a}$ are auxiliary variables, and A, B_u, B_w, B_r, E_x, E_u, E_w, E_r, and e are appropriately defined constant matrices such that Equation 5 is well posed. All inequalities in this article are interpreted element-wise.

LC systems are useful in modeling systems in which phenomena of active/nonactive behaviors exist. For example, LC systems can model robots with mechanical contacts r in Equation 5 representing contact force, which is nonzero when only object penetrations exist, which corresponds to a $y = 0$ hyperplane (see, e.g., 42).

Definition 10 (from Reference 17). A mixed-logical dynamical (MLD) system is in the following form:

$$x_{t+1} = Ax_t + B_u u_t + B_w w_t + B_\delta \delta_t + B_r r_t, \tag{6a.}$$

$$E_\delta \delta_t + E_r r_t \leq E_x x_t + E_u u_t + E_w w_t + e, \tag{6b.}$$

where $\delta_t \in \{0, 1\}^{n_\delta}$ and $r_t \in \mathbb{R}^{n_r}$ are auxiliary variables and $A, B_u, B_w, B_\delta, B_r, E_\delta, E_r, E_x, E_u, E_w$, and e are appropriately defined constant matrices such that Equation 6 is well posed in the sense that, given x_t, u_t, and w_t, the feasible set for x_{t+1} is a single point equal to $F(x_t, u_t, w_t)$ in Equation 2. The inequality is interpreted element-wise.

Auxiliary variables δ and r, which are constrained by Equation 6b, allow x_{t+1} to behave in a very nonlinear manner in x, u, and w. The MLD form is very general and can be viewed simply as a set of mixed-integer constraints, making it amenable to mixed-integer programming.

4.2. The Equivalence of Hybrid Models

Heemels et al. (38) showed that PWA, MMPS, LC, and MLD systems are, under mild assumptions, equivalent. They also showed that extended LC systems (43)—a more general version of LC systems—also belong to this equivalency class, and the proofs and the underlying assumptions were well documented in their paper (38). The MLD form is the most preferred one, as its optimal control is easily cast as a mixed-integer programming problem. Below are two simple examples of converting a PWA system and an LC system into their MLD forms. MLD forms are often not unique. We remark on their efficiency in Section 5.2.2. Note that it is straightforward to write an MMPS system in PWA form.

Example 2 (PWA to MLD). Consider the following discrete-time switched PWA system with two modes. Let $X \subset \mathbb{R}$ be bounded, $U = \{0, 1\}$ ($H_1 = X \times \{0\}, H_2 = X \times \{1\}$):

$$x_{t+1} = \begin{cases} a_1 x_t + c, \; u_t = 0, \\ a_2 x_t + c, \; u_t = 1, \end{cases} \tag{7.}$$

where $a_1, a_2, c \in \mathbb{R}$ are constants. Using the big-M method, $M \geq \sup_{x \in X} |(a_2 - a_1)x|$, Equation 7 can be translated into the following MLD system:

$$\begin{gathered} x_{t+1} = a_1 x_t + c + r_t, \\ (a_2 - a_1)x_t - M(1 - u_t) \leq r_t \leq (a_2 - a_1)x_t + M(1 - u_t), -Mu_t \leq r_t \leq Mu_t. \end{gathered} \tag{8.}$$

It is straightforward to verify that Equations 7 and 8 are equivalent.

Example 3 (LC to MLD). Consider the following LC system with bounded $X, U \subset \mathbb{R}$:

$$x_{t+1} = ax_t + bu_t + r_t, r_t x_t = 0, x_t, r_t \geq 0, \tag{9.}$$

where $a, b \in \mathbb{R}$ are constants. Note that r_t is nonzero only when $ax_t + bu_t < 0$. Using $M \geq \max_{x \in X, u \in U} |ax + bu|$, one can translate the system represented by Equation 9 into the following MLD system:

$$\begin{gathered} x_{t+1} = ax_t + bu_t + r_t, \\ 0 \leq r_t \leq M\delta_t, ax_t + bu_t \geq -M\delta_t, 0 \leq ax_t + bu_t + r_t \leq M(1 - \delta_t), \delta_t \in \{0, 1\}. \end{gathered} \tag{10.}$$

5. OPEN-LOOP CONTROL: TEMPORAL LOGIC TRAJECTORY OPTIMIZATION

In this section, we review the techniques for trajectory optimization from STL specifications. The goal is to find a trajectory that satisfies an STL formula by solving an optimization problem. The core technique is translating the STL formula into an appropriate form of constraints that can be handled by standard optimization solvers. The cost is ad hoc, but a natural candidate is maximizing the STL score. Throughout this section, we assume the system is deterministic and therefore deal with a single trajectory. Trajectory optimization is an open-loop control algorithm, as it provides the schedule of control inputs for a certain initial condition. We discuss feedback strategies in Section 6.

5.1. Problem Statement

A trajectory is a real-valued signal $\mathbf{s} : \mathbb{N} \rightarrow X \times U \subset \mathbb{R}^{n+m}$, $n = n_{\mathrm{r}} + n_{\mathrm{b}}, m = m_{\mathrm{r}} + m_{\mathrm{b}}$, where s_t is the vector obtained by stacking the values in state and control inputs in a single vector. We slightly abuse the notation to write $s_t = (x_t, u_t)$, while noting that s_t is a real vector, not a tuple. While defining a signal, we treat existing binary values in state or controls in the same way as reals. We assume all predicates in the STL formula are linear in state and controls [a predicate $(p(s) \leq c)$ is linear if p is a linear function]. Piecewise linear predicates can be encoded using Boolean connectives. For example, $(|x| \leq c)$ is equivalent to $(x \leq c) \vee (-x \leq c)$. However, general nonlinear predicates cannot be handled within a mixed-integer linear/quadratic programming (MILP/MIQP) framework. Therefore, the assumption of linear predicates is necessary, unless one is willing to use nonconvex nonlinear optimization solvers. We are given a cost function $J : \mathbb{S} \rightarrow \mathbb{R}$, where $\mathbb{S}$ is the set of all signals. We assume that J is piecewise linear or quadratic. A natural candidate for J is $\rho(\mathbf{s}, \varphi, 0)$, which itself is a piecewise linear function of s since the predicates are linear and the pieces correspond to the min and max operators in Definition 2.

> **Problem 1.** Given Equation 2 as one of the forms in Section 4.1, the initial state x_{initial}, an STL formula φ over linear predicates in state or controls, and a cost $J : \mathbb{S} \rightarrow \mathbb{R}$, find the optimal trajectory $\mathbf{s}$ such that J is minimized, $x_0 = x_{\text{initial}}$, and $(\mathbf{s}, 0) \models \varphi$. If such a trajectory does not exist, find $\mathbf{s}$ such that $\rho(\mathbf{s}, \varphi, 0)$ is maximized (so φ is minimally violated).

5.2. Mixed-Integer Formulation

As mentioned above, the main challenge is capturing the STL formula in the optimization problem. This is achieved by translating STL formulas into mixed-integer linear constraints. In conjunction with transforming the system represented by Equation 2 into mixed-integer linear constraints (MLD), the trajectory optimization problem becomes an MILP/MIQP problem. Trajectory optimization using MILP/MIQP methods is a very powerful technique, as it is able to capture a broad range of constraints, and it has been extensively studied in the robotics literature (44–47). Similar methods were developed to find high-level plans for robotic swarms (48–51) subject to spatial-temporal logic specifications (52).

There are alternatives to MILP/MIQP for temporal logic trajectory optimization. Shoukry et al. (53, 54) used satisfiability module theories (SMTs) for temporal logic control. SMT solvers provide only a feasible solution, not necessarily the optimal one. One may perform an exhaustive binary search to find an overapproximate of the cost. More recently, Shoukry et al. (55) developed a method called satisfiability modulo convex (SMC) optimization to combine the benefits of

SMT solvers and convex optimization. It is also possible to completely circumvent mixed-integer optimization by employing gradient descent techniques. Abbas et al. (56) and Pant et al. (57) used the gradient of the STL score, which is approximated by smooth functions (58), for optimization. While this approach can handle smooth nonlinear systems and nonlinear predicates, gradient descent is not a complete algorithm for problems of a nonconvex nature. There is no guarantee that the gradient descent can find the global optimal trajectory or even a feasible one (i.e., a positive STL score) when one exists.

In this section, we review the main procedure for encoding STL requirements as mixed-integer constraints. The framework was developed by Karaman et al. (59) for bounded LTL formulas and was extended to STL specifications by Raman et al. (60), whose paper also included encoding STL score. There are many variations of encoding techniques. The key differentiating factor is the number of variables, particularly binary variables, and the number of constraints they require.

5.2.1. Predicate-based encoding.

Here, we explain the method we described in Reference 61, which is a more efficient version of the method of Raman et al. (60). Throughout this section, we assume that the STL formulas do not contain negation. As mentioned in Section 3, this assumption is not restrictive, as any STL formula can be written without negation.

For each predicate $\pi = (y \geq 0)$, define a binary variable $z_t^\pi \in \{0, 1\}$ such that, at time t, 1 stands for true and 0 stands for false. The relation between z_t^π, the robustness margin ρ (which is closely connected to STL score, as will be shown shortly), and y is encoded as

$$y_t + M(1 - z_t^\pi) \geq \rho, y_t - M z_t^\pi \leq \rho. \qquad 11.$$

The constant M is a sufficiently large number such that for all times, $M \geq \max y_i, i = 1, \ldots, n_y$. Note that the largest ρ for which $z_t^\pi = 1$ is y, which is equal to the robustness score of π. Disjunctions and conjunctions are captured by the following constraints:

$$z = \bigwedge_{i=1}^{n_z} z_i \Rightarrow z \leq z_i, i = 1, \ldots, n_z, \; z = \bigvee_{i=1}^{n_z} z_i \Rightarrow z \leq \sum_{i=1}^{n_z} z_i, \qquad 12.$$

where $z \in [0, 1]$ is declared as a continuous variable. However, it can take only binary values, as is evident from Equation 12. Note the one-way (sufficiency) implications in Equation 13. The formulations of Karaman et al. (59) and Raman et al. (60) establish necessity and sufficiency by adding upper-bounding constraints as follows: $z = \bigwedge_{i=1}^{n_z} z_i \Leftrightarrow z \geq \sum_{i=1}^{n_z} z_i - n_z + 1, z \leq z_i, i = 1, \ldots, n_z$, and $z = \bigvee_{i=1}^{n_z} z_i \Leftrightarrow z \geq z_i, i = 1, \ldots, n_z, z \leq \sum_{i=1}^{n_z} z_i$. However, the upper-bound constraints are necessary only when the negation operator is present in the STL formula. Hence, they are safely removed in a negation-free setting, which reduces constraint redundancy and degeneracy in the optimization problem. Define $z_t^\varphi \in [0, 1]$ as the variable that indicates whether $(\mathbf{x}, t) \models \varphi$. An STL formula is recursively translated as

$$\begin{gathered} \varphi = \bigwedge\nolimits_{i=1}^{n_\varphi} \varphi_i \Rightarrow z_t^\varphi = \bigwedge\nolimits_{i=1}^{n_\varphi} z_k^{\varphi_i}; \; \varphi = \bigvee\nolimits_{i=1}^{n_\varphi} \varphi_i \Rightarrow z_t^\varphi = \bigvee\nolimits_{i=1}^{n_\varphi} z_t^{\varphi_i}; \\ \varphi = \mathbf{G}_I \psi \Rightarrow z_t^\varphi = \bigwedge\nolimits_{t' \in I} z_{t'}^\psi; \; \varphi = \mathbf{F}_I \psi \Rightarrow z_t^\varphi = \bigvee\nolimits_{t' \in I} z_{t'}^\psi; \\ \varphi = \psi_1 \mathbf{U}_I \psi_2 \Rightarrow z_t^\varphi = \bigvee\nolimits_{t' \in I} \left(z_{t'}^{\psi_2} \wedge \bigwedge\nolimits_{t'' \in [t, t']} z_{t''}^{\psi_1} \right). \end{gathered} \qquad 13.$$

Given φ, denote the set of constraints recursively constructed by Equations 11–13 as $\mathcal{C}_\varphi$.

Theorem 1. The following properties hold: (*a*) We have $(\mathbf{s}, t) \models \varphi$ if adding $z_t^\varphi = 1, \rho \geq 0$ makes $\mathcal{C}_\varphi$ feasible; (*b*) we have $(\mathbf{s}, t) \not\models \varphi$ if $z_t^\varphi = 1, \rho \geq 0$ makes $\mathcal{C}_\varphi$ infeasible; and (*c*) the largest ρ such that $z_t^\varphi = 1$ and $\mathcal{C}_\varphi$ is feasible is equal to $\rho(\mathbf{s}, \varphi, t)$.

Raman et al. (60) encoded min and max operators in Definition 2 using a separate set of binary variables. In almost all problems, the objective is either maximizing the STL score or declaring a constraint that sets a lower bound for it. The third property in Theorem 1 indicates that additional binaries are not required to capture the STL score. The advantage of the formulation of Raman et al. (60) is that it does not require preprocessing to remove negation operators; it also encodes the STL score directly so that one can, for example, minimize the STL score, which is of interest in falsification problems (62, 63).

5.2.2. Encoding with the tightest relaxations. The efficiency of MILP/MIQP problems depends greatly on the tightness of the constraints if binary variables are relaxed to continuous variables in [0, 1]. One drawback of big-M methods is their potentially very loose relaxations, particularly when very large big-M constants are chosen. One way to deal with this issue is precomputing minimum big-M constants for each constraint. An alternative that does not use the big-M method is encoding a mixed-integer linear constraint based on convex hull relaxations (64). Here, we briefly explain this method.

Let $\mathcal{P}_i = \{y | H_i y \leq h_i\}$, $i = 1, \ldots, n_p$, be a set of polytopes, where H_i and h_i are appropriately sized matrices. Let $z \in \{0, 1\}$ be such that $y \in \bigcup_{i=1}^{n_p} \mathcal{P}_i$ implies $z = 1$ and otherwise implies $z = 0$. Instead of using a big-M method and treating the set of polytopes as a disjunction over conjunctions of half-space predicates, the relation can be encoded as

$$H_i y_i \leq h_i z_i, z_i \in \{0, 1\}, i = 1, \ldots, n_p, z = \sum_{i=1}^{n_p} z_i, y = \sum_{i=1}^{n_p} y_i. \qquad 14.$$

It can be shown that relaxing $z_i \in \{0, 1\}$ to $z_i \in [0, 1]$, $z = 1$ is equivalent to $y \in \text{Convexhull}(\bigcup_{i=1}^{n_p} \mathcal{P}_i)$, which is the tightest convex relaxation of $\bigcup_{i=1}^{n_p} \mathcal{P}_i$. The formulation in Equation 14 introduces fewer binary variables than predicate-based encoding, but it also introduces additional continuous variables. Wolff & Murray (65) studied encoding based on Equation 14 for LTL optimal control and found that the computation times are sometimes greater than those of the big-M method, which is attributed to the additional continuous variables. The efficiency of encoding STL and MLD constraints based on Equation 14 has not been thoroughly studied yet.

5.3. Trajectory Optimization

The method to generate the trajectory is dependent on whether φ is bounded or not. The following cases are considered.

5.3.1. Bounded formulas. If φ is bounded, then we only need to consider $(\mathbf{s}, [0, h^{\varphi}])$ for the sake of STL correctness. It is also common to assume that the cost function is defined over $(\mathbf{s}, [0, h^{\varphi}])$, and the rest of the signal is irrelevant. We solve the following optimization problem:

$$\begin{aligned} u_0^*, u_1^*, \ldots, u_{h^{\varphi}}^* = \arg\min \quad & J((\mathbf{s}[0, h^{\varphi}])) + \tfrac{1}{2} M(|\rho| - \rho) \\ \text{subject to } & x_{\tau+1} = F(x_\tau, u_\tau), \tau = 0, 1, \ldots, h^{\varphi}, \\ & x_0 = x_{\text{initial}}, \mathcal{C}_\varphi, z_0^\varphi = 1, \end{aligned} \qquad 15.$$

where M is a sufficiently large number. Equation 15 is an MILP/MIQP problem, depending on the structure of J. The solution to Equation 15 has the following properties: (*a*) If $(\mathbf{s}, 0) \models \varphi$ is possible, then $J((\mathbf{s}, [0, h^{\varphi}]))$ is minimized (optimal control), and (*b*) if $(\mathbf{s}, 0) \models \varphi$ is impossible, then $\rho(\mathbf{s}, \varphi, 0)$ is maximized (minimal STL violation). Note that if $\rho \geq 0$ is feasible, then $\frac{1}{2} M(|\rho| - \rho)$

equals zero and the original cost J is minimized. If $\rho \geq 0$ is infeasible, then $\frac{1}{2}M(|\rho| - \rho)$ becomes $-M\rho$. Since M is a very large positive number, effectively ρ is maximized, which by virtue of Theorem 1 is equivalent to $\rho(\mathbf{s}, \varphi, 0)$.

Problem 15 is solved to global optimality. Since there is no conservativeness introduced, Equation 15 is a complete method: A solution is found if it exists. Completeness and global optimality are rarely achieved in automata-based approaches, as conservativeness is often introduced in finite-state abstractions, and optimality is as good as the resolution of partitions.

The complexity of MILP/MIQP problems grows exponentially (in the worst case) with respect to the number of binary variables and polynomially with the number of continuous variables and constraints. The number of binaries grows linearly with the number of predicates and the horizon of the formula. Therefore, the optimization runtime increases rapidly when the STL formula is more intricate. Exponential complexity also exists in automata-based approaches. However, there is an important difference in the complexity source. While the size of finite-state automata that capture temporal logic formulas also increases exponentially with the complexity of the formula, the main bottleneck often lies in the finite-state abstraction itself. For a fixed partitioning mesh size, the number of partitions is exponential in the state-control dimensions (66). However, there is no direct exponential growth in the number of states in optimization-based synthesis. Therefore, it is possible to solve very large formal synthesis problems (see Example 5 in Section 6.3)—optimization-based methods may be surprisingly fast. However, the runtimes are generally unreliable, and MILP/MIQP may be too slow for seemingly benign problems.

Example 4 (from Reference 67). We consider a two-dimensional PWA system with four modes:

$$A_1 = \begin{pmatrix} 1 & 1 \\ -0.7 & 1 \end{pmatrix}, A_2 = \begin{pmatrix} 1.3 & 1.3 \\ 0 & 1.3 \end{pmatrix}, A_3 = \begin{pmatrix} 0.7 & 0.7 \\ -0.3 & 0.7 \end{pmatrix}, A_4 = \begin{pmatrix} 1.3 & 1 \\ 0.3 & 0.7 \end{pmatrix}, \tag{16a}$$

$$B_1 = B_2 = B_4 = -B_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, c_1 = -c_3 = \begin{pmatrix} -5 \\ 0 \end{pmatrix}, c_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, c_4 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \tag{16b}$$

Let $X = [-20, 20]$ and $U = [-10, 10]$. We are interested in the following specification:

$$\varphi = \mathbf{F}_{[0,15]}\mathbf{G}_{[0,2]}\psi_1 \wedge \mathbf{F}_{[15,30]}\psi_2 \wedge \mathbf{G}_{[0,30]}\neg\psi_3, \tag{17}$$

where $\psi_1 = (x_{[1]} \geq -15) \wedge (x_{[2]} \geq -15) \wedge (x_{[1]} \leq -10) \wedge (x_{[2]} \leq -10)$, $\psi_2 = (x_{[1]} \geq 10) \wedge (x_{[2]} \geq 10) \wedge (x_{[1]} \leq 15) \wedge (x_{[2]} \leq 15)$, and $\psi_3 = (x_{[1]} \geq 0) \wedge (x_{[2]} \geq 2) \wedge (x_{[1]} \leq 15) \wedge (x_{[2]} \leq 8)$. Subspecifications ψ_1, ψ_2, and ψ_3 correspond to rectangular regions in X. **Figure 3** shows the workspace with the vector fields in each region. In plain English, Equation 17 states that ψ_1 is satisfied for three consecutive time steps between [0, 15], ψ_2 is satisfied at least once in [15, 30], and ψ_3 is never satisfied between [0, 30]. The cost function is considered the l_2 norm of controls $J = \sum_{t=0}^{30} u_t^2$. The initial condition is $x_0 = (-10, 15)^T$. **Figure 3** shows three possible example trajectories: a control-effort-optimal satisfying trajectory (**Figure 3*a***), a maximally satisfying trajectory (**Figure 3*b***), and a minimally violating trajectory with $U = [-2, 2]$ (**Figure 3*c***).

5.3.2. Unbounded safety formulas.

We consider formulas of the following form:

$$\phi = \varphi_{\mathrm{b}} \wedge \mathbf{G}_{[\Delta,\infty)}\varphi_{\mathrm{g}}, \tag{18}$$

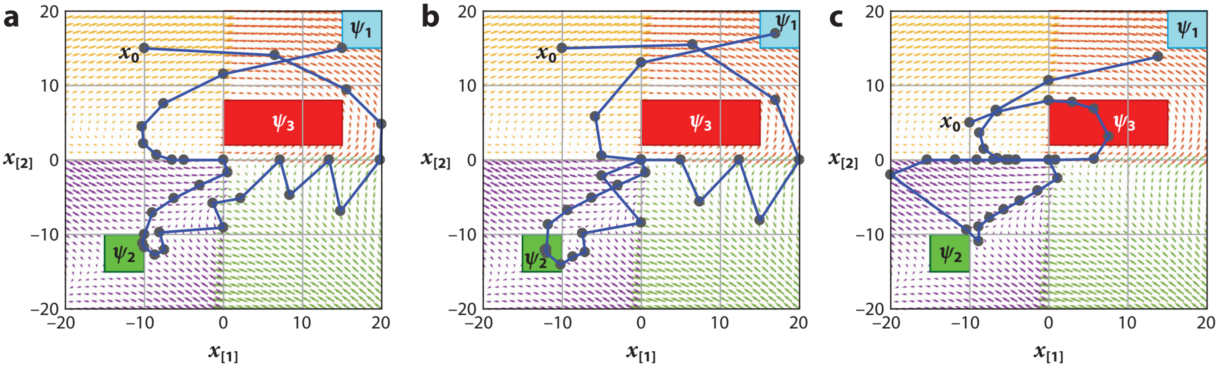

Figure 3

The three trajectories from Example 4. (*a*) A control-effort-optimal satisfying trajectory, with a signal temporal logic (STL) score of 0. This trajectory touches the boundaries of the rectangles, as getting further distance into ψ_1 and ψ_2 and getting away from ψ_3 increases control effort. (*b*) A maximally satisfying trajectory, with the maximum possible STL score of 2.5. This trajectory visits the centers of ψ_1 and ψ_2 and keeps a notable distance from ψ_3. (*c*) A minimally violating trajectory. To obtain a trajectory with violation, we first shrank U to $[-2, 2]$ to tighten the constraints. To obtain the trajectory shown here, we then multiplied the predicates of ψ_3 by 10 to further penalize violating trajectories entering ψ_3. The result is a maximally satisfying trajectory from $x_0 = (-10, 5)^T$, with $\rho(\zeta, \varphi, 0) = -3.9$. Note that all three trajectories are synthesized and verified in discrete time; the lines connecting the circles are included solely to illustrate the time progression. Figure adapted from Reference 67.

where $\Delta \in \mathbb{N}$, and φ_b and φ_g are bounded negation-free STL formulas. We also consider the case that φ_b is true, which reduces Equation 18 to $\mathbf{G}_{[\Delta,\infty)}\varphi_g$; we refer to these as global STL formulas. We call the formulas in the form of Equation 18 bounded-global STL formulas. In Reference 61, we showed that almost all common STL formulas can be written as disjunctions of a finite number of bounded-global STL formulas. The interpretation of the requirement in bounded-global formulas is straightforward: First, the signal must satisfy φ_b in finite time, and after Δ, all signal suffixes are required to satisfy φ_g.

We cannot use Equation 15 to synthesize an infinite-time trajectory, as it introduces an infinite number of variables. To find a trajectory that satisfies Equation 18, the idea is to impose a periodic suffix for **s**. This idea was implemented in optimization-based LTL synthesis of infinite-time trajectories by Wolff et al. (68) and was also used by Raman et al. (60) for STL control of global formulas. Prefix-periodic suffix trajectories are well known, and it can be shown that their existence is necessary for any temporal logic formula in the case of a finite system (1). However, for infinite systems, their existence is only sufficient. The following periodicity relation is imposed:

$$s_{T_0+kT} = s_{t_0}, k \in \mathbb{N}, \qquad 19.$$

where T_0 and T are the lengths of the prefix and the period of the suffix, respectively. The decision variables are in $s_0, s_1, \ldots, s_{T_0+T}$, which map to finding $T_0 + T + 1$ control inputs. Without loss of generality, let $T_0 \geq \Delta$. Then $(\mathbf{s}, 0) \models \phi$ is equivalent to

$$(\mathbf{s}, [0, T_0]) \models \varphi_b \wedge \mathbf{G}_{[\Delta,T_0]}\varphi_g, (\mathbf{s}, [T_0 + k, T_0 + k + h^{\varphi_g}]) \models \varphi_g, k = 0, \ldots, T - 1. \qquad 20.$$

It is easy to show that Equation 20 enforces $(\mathbf{s}, 0) \models \varphi_b \wedge \mathbf{G}_{[\Delta,\infty)}\varphi_g$. Thus, trajectory optimization for this class of specifications can also be cast as an optimization problem, similar to Equation 15, with additional constraints enforcing periodic suffixes. There is no straightforward way to choose T_0 and T. One must exhaustively try different numbers until feasibility or desirable optimality is achieved. The completeness property is also lost.

5.3.3. Other formulas. If φ is unbounded but does not belong to the safety class, we need to replace φ with a safety φ' such that $L(\varphi') \subset L(\varphi)$. Thus, by successful trajectory optimization for φ', we have obtained a trajectory satisfying φ. For example, let $\varphi = \mathbf{F}_{[0,\infty)}\psi$, where ψ is a bounded STL formula. We can replace φ with $\varphi' = \mathbf{F}_{[0,T]}\psi$, where T is a user-chosen positive integer. Larger T is less conservative but results in higher computational expense. The conservativeness introduced in replacing the formula removes the completeness property.

6. CLOSED-LOOP CONTROL: FEEDBACK STRATEGIES AND DISTURBANCE REJECTION

As mentioned above, trajectory optimization is an open-loop control strategy. Challenges rise when the system is nondeterministic, so it may not produce a single trajectory. Moreover, even if the system is deterministic, unmodeled phenomena may cause the system to deviate from the planned trajectory. Therefore, feedback mechanisms are necessary in practice. In this section, we review the current literature on optimization-based-feedback temporal logic control. Unfortunately, in our opinion, no satisfying solution has been provided in the literature so far. We remind the reader that automata-based synthesis provides feedback strategies and the set of (all) admissible initial conditions. A competitive technique is regrettably lacking here. The reason is the formidable complexity. Even finding the right structure for the control policy is challenging. We know from explicit MILP/MIQP solutions that the optimal controller has a PWA structure (69). However, obtaining the PWA solutions of Equation 15 requires multiparametric programming and enumerating a large number of binary combinations, which is intractable even for small problems.

6.1. Problem Statement

Recall the problem from Section 5. Here, we drop the assumption that the system is deterministic. The goal is to design a control policy valid for a set of states instead of a single trajectory. The control policy μ gives the input at time t by $u_t = \mu(x_0, \ldots, x_t, u_0, \ldots, u_{t-1})$. Temporal logic controllers are seldom memoryless. Given the system represented by Equation 2, control policy μ, and a set of initial conditions $X_{\text{initial}} \subseteq X$, the set of all closed-loop trajectories is

$$L(X_{\text{initial}}, \mu) := \Big\{ \mathbf{s} \in \mathbb{S} \Big| s_t = (x_t, u_t), x_0 \in X_{\text{initial}}, u_t = \mu(x_0, \ldots, x_t, u_0, \ldots, u_{t-1}), \\ x_{t+1} = F(x_t, u_t, w_t), w_t \in W, t \in \mathbb{N} \Big\}.$$

Problem 2. Design X_{initial} and μ such that $L(X_{\text{initial}}, \mu) \subseteq L(\varphi)$ (all the closed-loop trajectories satisfy φ). If $L(X_{\text{initial}}, \mu) \subseteq L(\varphi)$ is not possible, provide a lower-bound certificate for the STL score of all closed-loop trajectories by maximizing $\min_{\mathbf{s} \in L(X_{\text{initial}}, \mu)} \rho(\mathbf{s}, \varphi, 0)$.

We are also interested in having a large set for X_{initial}. Even for singleton X_{initial}, $L(X_{\text{initial}}, \mu)$ is usually not a singleton, as disturbances are present. It should be decided which trajectory in $L(X_{\text{initial}}, \mu)$ is optimized. In a probabilistic setting, the common approach is to optimize the expected value of the cost, given the knowledge of the underlying distributions. In the nondeterministic case, a common approach is minimizing the cost of the trajectory in $L(X_{\text{initial}}, \mu)$ that has the maximum cost. This approach is known as minimax robust optimal control (70), and it was adopted by Farahani et al. (71) for robust STL control. In Reference 72, we took a simpler approach in which $J(\bar{\mathbf{s}})$ is optimized, where $\bar{\mathbf{s}}$ is the nominal trajectory with zero disturbances. For generality, we let $\mathcal{J} : 2^{\mathbb{S}} \to \mathbb{R}$ denote the cost function that maps a set of trajectories into a real value.

6.2. Real-Time Robust Trajectory Optimization

The leading effort to obtain closed-loop optimization-based policies has been real-time trajectory optimization: At each time, given the history of the trajectory, a new trajectory suffix (beginning from the current time) is optimized. Only the control input corresponding to the current time is implemented, and the procedure is repeated at subsequent times. This scheme is similar to MPC, with considerations that we highlight below, and is effectively closed loop. This approach was adopted in References 71–73 for STL control of MLD or linear systems with additive disturbances. The same idea has been implemented in probabilistic scenarios where STL requirements were formulated as chance constraints (74–76).

6.2.1. Bounded formulas.

Let φ be a bounded formula. The control input $u_t = \mu(x_0, \dots, x_t, u_0, \dots, u_{t-1})$ is computed through the following robust optimization problem:

$$\begin{aligned} u_t = \arg\min\nolimits_{u'_t} \ & \mathcal{J}(\{\mathbf{s}'[0, h^\varphi]\}) + \tfrac{1}{2}M(|\rho_{\min}| - \rho_{\min}) \\ \text{subject to } & \mathcal{C}_\varphi, z_0^\varphi = 1, \rho_{\min} \leq \rho(\mathbf{s}', \varphi, 0), \\ & \forall \mathbf{s}' \text{ such that } s'_\tau = (x_\tau, u_\tau), \tau = 0, \dots, t-1, x'_t = x_t, \\ & x'_{\tau+1} = F(x'_\tau, u'_\tau, w'_\tau), w'_\tau \in W, \tau = t, \dots, h^\varphi - 1, s'_t = (x_t, u'_t), \end{aligned} \qquad 21.$$

where x', u', and w' are variables but x and u are constants; note that $x_t = x'_t$. Equation 21 does not provide a closed-form function for μ, but given values for its arguments, the optimization problem returns the function value. As mentioned above, finding the closed form for μ using parametric programming is often intractable. Also note that the history of the trajectory up to time t is taken into account in Equation 21, which is a shrinking-horizon problem, as the number of decision variables decreases as time proceeds. By construction, if $\rho_{\min}$ is positive at any time in Equation 21, then it is guaranteed that $(\mathbf{s}, 0) \models \varphi$. The reason is that the STL constraints are enforced for all possible future disturbances. Furthermore, by virtue of the same argument, $\rho_{\min}$ cannot decrease while time progresses. Despite its soundness, the method in Equation 21 is extremely conservative because it looks for a sequence of control inputs that provide robustness against all disturbances instead of a policy that is reactive to future disturbances.

6.2.2. Global formulas.

For a global formula $\varphi = \mathbf{G}_{[0,\infty)}\psi$, where ψ is a bounded STL formula, the method becomes similar to traditional MPC; we described this method in Reference 72. Let H be the MPC horizon. Recall that we need to enforce $(\mathbf{s}, \tau) \models \psi, \forall \tau \in \mathbb{N}$. At time t, the trajectory portion $(\mathbf{s}, [t, t+H])$ is optimized, and its values are required for deciding about $\rho(\mathbf{s}, \psi, \tau)$ for $\tau = t - h^\varphi, \dots, t + H - h^\varphi$. Therefore, at each time, only the recent h^φ step of the history is required: $u_t = \mu((\mathbf{s}, [t - h^\varphi, t-1]), x_t)$, where

$$\begin{aligned} u_t = \arg\min\nolimits_{u'_t} \ & \mathcal{J}(\{\mathbf{s}'[t, t+H]\}) + \tfrac{1}{2}M(|\rho_{\min}| - \rho_{\min}), \\ \text{subject to } & C_\psi, z_\tau^\psi = 1, \tau = t - h^\varphi, \dots, t + H - h^\varphi, \rho_{\min} \leq \rho(\mathbf{s}', \varphi, \tau), x'_t = x_t, \\ & \forall \mathbf{s}' \text{ such that } s'_\tau = (x_{\tau'}, u_{\tau'}), \tau' = t - h^\varphi, \dots, t-1, s'_t = (x_t, u'_t), \\ & x'_{\tau+1} = F(x'_\tau, u'_\tau, w'_\tau), w'_\tau \in W, \tau = t, \dots, h^\varphi - 1, s'_t = (x_t, u'_t). \end{aligned} \qquad 22.$$

The synthesis method in Equation 22 successfully establishes $(\mathbf{s}, 0) \models \varphi$ if and only if the solution for Equation 22 at all times has $\rho_{\min} \geq 0$. Besides excessive conservativeness, the paradigm used in References 71–76, reflected in Equation 21 and in Equation 22 and its variations, has multiple serious drawbacks. First, the characterization of all signals in Equations 21 and 22 is hard. Raman et al.

(73) used an iterative counterexample-guided approach to generate the worst-case $\mathbf{s}'$. However, iterations may not terminate, and the size of Equation 22 becomes larger as the iterations proceed. Farahani et al. (71) exploited the duality of MILPs to solve the minimax problem, which is computationally expensive. The work in Reference 72 generates a bound for the worst-case scenario in advance and is computationally faster but conservative. Second, no means is provided to compute X_{initial}. One can only conservatively check $x_0 \in X_{\text{initial}}$ for a given x_0 and a bounded STL formula. Third, MILP/MIQP approaches are computationally expensive. In many applications with a small timescale, solving Equation 21 or Equation 22 is not possible in real time. There has been some work on improving the runtime of STL MPC using MILP/MIQP approaches (77, 78), which provide faster solutions for specific types of formulas. Finally, and theoretically most importantly, there is no guarantee for persistent feasibility for global optimal control. Therefore, it is necessary that Equation 22 treats the STL constraints in a soft manner: Satisfy the constraints whenever possible and minimally violate them otherwise. Ghosh et al. (79) took a similar approach, where instead of maximizing the STL score, the STL formula itself was minimally changed (including changing the temporal structure).

6.3. Global Formulas: Set-Invariance Control

In this section, we focus on global STL formulas, which are common in applications (71–73, 78). There is a seldom-exploited close connection between global STL control and set-invariance theories (80). Recall that

$$(\mathbf{s}, 0) \models \mathbf{G}_{[0,\infty)}\psi \Leftrightarrow (\mathbf{s}, \tau) \models \psi, \forall \tau \in \mathbb{N} \Leftrightarrow (\mathbf{s}, [\tau, \tau + b^{\psi}]) \in \mathcal{L}(\psi), \forall \tau \in \mathbb{N}, \qquad 23.$$

where $b^{\psi} < \infty$. Define $\zeta_t \in (X \times U)^{b^{\psi}+1}$, which is obtained by stacking values in $(\mathbf{s}, [t - b^{\varphi}, t])$ into a single vector. Then ζ_{t+1} corresponds to $(\mathbf{s}, [t - b^{\varphi} + 1, t + 1])$, which, in comparison with ζ_t, depends on w_t and u_{t+1}, as $s_{t+1} = (F(x_t, u_t), u_{t+1})$. Define $\xi_t := u_{t+1}$ as the new control input. The evolution of ζ is then given by the following discrete-time system:

$$\zeta_{t+1} = \mathcal{F}(\zeta_t, \xi_t, w_t), \qquad 24.$$

where $\zeta_t \in (X \times U)^{b^{\psi}+1}$ and $\xi_t \in U, w_t \in W$. The language $L(\psi)$ can also be written as a subset of $(X \times U)^{b^{\psi}+1}$, denoted by $\mathcal{L}(\psi)$, for which $\zeta_t \in \mathcal{L}(\psi)$ if and only if $(\mathbf{s}, [t - b^{\varphi}, t]) \models \psi$. Therefore, Equation 23 is equivalent to $\zeta_t \in \mathcal{L}(\psi), t = b^{\psi}, b^{\psi} + 1, \ldots,$ which is a forward set-invariance condition. The control problem of $(\mathbf{s}, 0) \models \mathbf{G}_{[0,\infty)}\psi$ then becomes equivalent to finding a robust control invariant (RCI) set $\Omega \subseteq \mathcal{L}(\psi)$ such that

$$\forall \zeta \in \Omega, \exists \xi \in U, \text{ such that } F(\zeta, \xi, w) \in \Omega, \forall w \in W. \qquad 25.$$

There is a substantial literature on the computation of RCI sets for hybrid systems (81, 82). Unlike traditional RCI sets, which are in the state space, Ω is constructed in the language space and is high dimensional and often nonconvex (unless ψ does not include any disjunctions or temporal "eventually" or "until" operators). Computing such RCI sets is computationally prohibitive. If one can compute the maximal RCI set, then a complete solution to global STL control is obtained: The largest set of initial conditions corresponds to the maximal RCI set, and any memoryless (in the language space) invariance-inducing controller, $u_t = \xi_{t-1} = \mu(\zeta_{t-1})$ (a function of the recent b^{ψ}-step history), is a valid control policy. One can use any RCI set Ω as the terminal constraint of Equation 22 to guarantee MPC recursive feasibility.

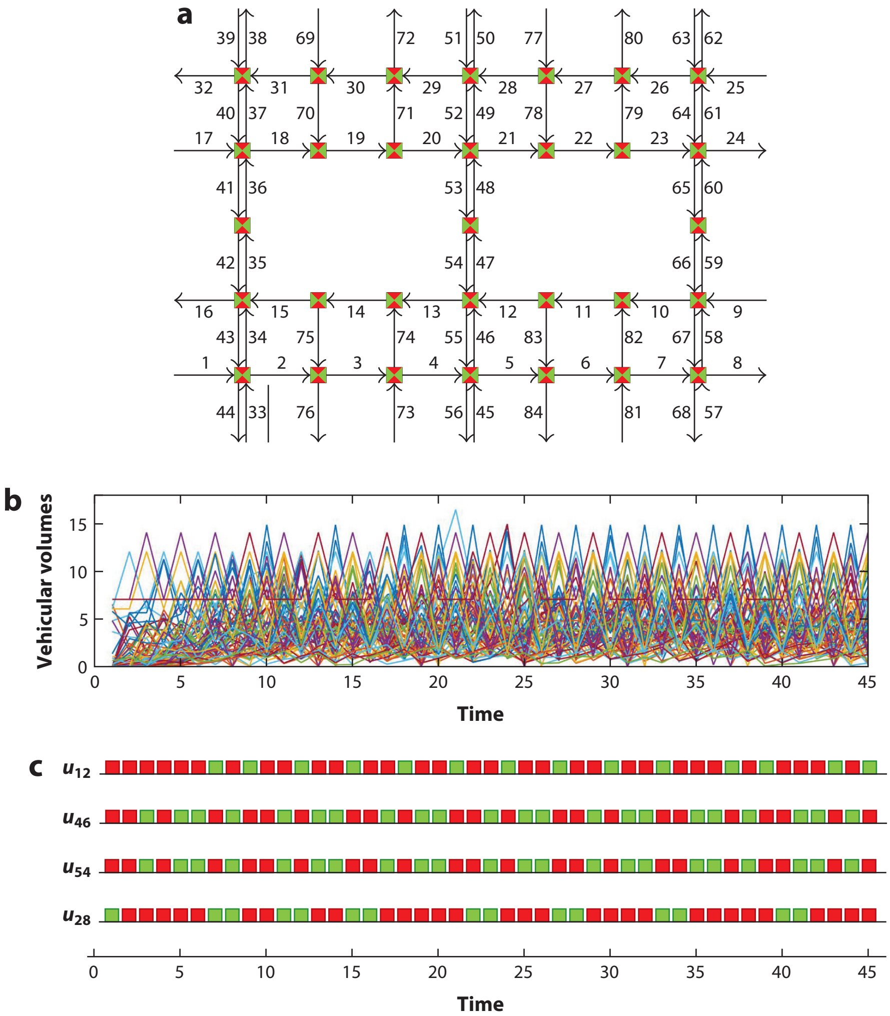

Figure 4

Traffic network from Example 5. (*a*) The traffic network. Each link is shown as a directed edge between nodes (intersections), which are shown as squares. (*b*) The number of vehicles on each link versus time. (*c*) Traffic lights versus time for links 12, 46, 54, and 28. Figure adapted from Reference 83.

For special classes of systems, RCI sets can be efficiently computed. RCI sets in the language space can be computed using a single mixed-integer program for positive monotone MLD systems and ordered STL requirements (61, 83; the formal definitions and technical details are omitted here). Example 5 is based on this result. For linear systems with additive disturbances, Rungger & Tabuada (84) introduced a method to compute arbitrarily precise under- or overapproximations of the maximal RCI set.

Example 5 (from Reference 83). Figure 4 shows a traffic network with 84 links; the details of the model were described by Sadraddini et al. (83). The number of vehicles on link $l \in \{1, \ldots, 84\}$ and the traffic light facing it are denoted $x_l \in \mathbb{R}_+$ and

$u_l \in \{0, 1\}$, where 1 stands for a green light and 0 stands for a red light. We consider the global STL formula $\mathbf{G}_{[0,\infty)}\psi$, $\psi = \bigwedge_{i=1,\ldots,5} \psi_i$, where $\psi_1 = (x, u) \in \Pi$, $\Pi \subset X \times U$ is a set that characterizes congestion-free flow in which the monotonicity holds (83); $\psi_2 = \mathbf{F}_{[0,6)}\left((u_{12} = 0) \wedge (u_{46} = 0) \wedge (u_{54} = 0)\right)$, which states that "within 6 time units, all the traffic lights of links heading toward the intersection in the middle southern area turn red (hence pedestrians can cross the intersection in diagonal directions)"; $\psi_3 = \neg\big((u_{28} = 0) \wedge \mathbf{F}_{[1,1]}(u_{28} = 1) \wedge \mathbf{F}_{[2,2]}(u_{28} = 0)\big)$, which means that "the traffic light of link 28 cannot be green for exactly 1 time step"; $\psi_4 = (x_{59} + x_{60} + x_{65} + x_{66} \leq 100)$, which states that "the total volume of the vehicles on the eastern bridge is less than 100"; and $\psi_5 = (x_{73} \leq 5) \vee \mathbf{F}_{[0,4)}(u_{73} = 1)$, which translates to "if the volume of vehicles on link 73 exceeds 5, its traffic light eventually turns green within 4 time units." Note that $b^{\psi} = 6$.

The method of Sadraddini et al. (83) solves an MILP problem with 5,981 variables (1,236 binary) and 2,902 constraints in 8.6 seconds to find an RCI set in $\mathcal{L}(\psi)$, which lies in $\mathbb{R}_+^{420} \times \{0, 1\}^{384}$. An MPC algorithm (which was shown to be distributable) synthesizes controls, while the RCI set acts as the MPC terminal constraint. Thus, MPC recursive feasibility and specification correctness are guaranteed. This fact is verified by simulations, where it was observed that all MPC problems with $\rho_{\min} \geq 0$ were feasible. Trajectories always remain in the congestion-free set, and all the subspecifications are always met. **Figure 4** shows the traffic lights corresponding to ψ_2 and ψ_3 and the vehicular volumes over time. The number of vehicles on the eastern bridge never exceeds 100 (ψ_4). Only once did x_{73} exceed 5, whereupon u_{73} turned green immediately (ψ_5).

6.4. Robust Model Predictive Control: Tubes and Disturbance Rejection

Here, we discuss the connection between feedback-optimization-based STL control and trajectory tracking. The idea is to compute a nominal trajectory in advance and use simple feedback policies to track the trajectory. The STL score can be exploited to characterize the tracking error. Lindemann and colleagues (85, 86) explored this idea for nonlinear continuous-time systems with unbounded control authority and a restricted class of STL specifications, and Jha et al. (87) did so for linear systems. Xu et al. (88) used a data-driven approach to design tubes robust against faulty behavior. In Reference 89, we developed a robust STL control method to deal with PWA systems with additive disturbances, which has provable guarantees.

In this section, we explain the method described in Reference 89. Assume $X \subset \mathbb{R}^n, U \subset \mathbb{R}^m$. The idea is to propose the following form for the control policy:

$$\mu(x_0, x_1 \ldots, x_t, u_0, u_1, \ldots u_{t-1}) = \mu^{\text{nom}}(x_0, t) + \mu^{\text{fb}}(x_t), \qquad 26.$$

where $\mu^{\text{nom}} : X \times \mathbb{N} \to U$ is an open-loop control policy and $\mu^{\text{fb}} : \mathbb{R}^n \to \mathbb{R}^m$ is a state-feedback control policy. Basically, $\mu^{\text{nom}}(x_0, t)$ results from trajectory optimization for the nominal system (a system without disturbances) and is computed offline, while μ^{fb} is the ancillary controller that corrects the deviations from the nominal trajectory in an online manner. Let $\mathbf{s}^{\text{nom}}$ be the nominal trajectory. The goal is to keep all the closed-loop trajectories close to $\mathbf{s}^{\text{nom}}$. Let $\Omega \in \mathbb{R}^n$ and $\Gamma \in \mathbb{R}^m$ be such that for all $x \in X, u \in U$, we have

$$\forall \delta x \in \Omega, \exists \delta u \in \Gamma, \text{such that}\, F(x + \delta x, u + \delta u, w) - F(x, u, 0) \in \Omega, \forall w \in W, \qquad 27.$$

which is stating that it is always possible to keep the disturbed state within the Ω neighborhood of the nominal state using controls in the Γ neighborhood of the nominal controls. This scheme is known as tube-based (or funnel-based) feedback design (90, 91); when implemented in an MPC framework, it is called tube MPC (92–95). The following is the core result in the connection between tube MPC and STL correctness.

Theorem 2 (from Reference 89). Given the system represented by Equation 2; an STL formula φ with linear predicates over state or controls; sets Ω and Γ; $\mu^{\text{tube}} : \Omega \to \Gamma$ such that $\forall x \in X, \forall u \in U, \forall \delta x \in \Omega, F(x + \delta x, \mu^{\text{tube}}(u + \delta u, w)) - F(x, u, 0) \in \Omega$, and $\forall w \in W$; a nominal trajectory $\mathbf{s}^{\text{nom}}$ for the system with $W = \{0\}$ such that $\{x_t^{\text{nom}}\} \oplus \Omega \subseteq X$, $\{u_t^{\text{nom}}\} \oplus \Gamma \subseteq U$, $t \in [0, b^\varphi]$ (bounded φ), or $t \in \mathbb{N}$ (unbounded φ); and the control policy $\mu(x, t) = u_t^{\text{nom}} + \mu^{\text{tube}}(x - x_t^{\text{nom}})$, the following guarantee holds:

$$\min\left(\{\rho(\mathbf{s}, \varphi, 0) | \mathbf{s} \in L(x_0^{\text{nom}} \oplus \Omega, \mu)\}\right) \geq \rho(\mathbf{s}^{\text{nom}}, \varphi, 0) - \max_{\delta x \in \Omega, \delta u \in \Gamma} \|C_x \delta x + C_u \delta u\|_\infty, \qquad 28.$$

where $\oplus$ stands for the Minkowski sum and C_x and C_u are matrices obtained by vertically stacking the linear coefficients of x and u in the predicates of φ, respectively.

Theorem 2 encourages finding thin tubes, i.e., small Ω and Γ. In words, one first computes the RCI set and evaluates $\epsilon := \max_{\delta x \in \Omega, \delta u \in \Gamma} \|C_x \delta x + C_u \delta u\|_\infty$, which is the largest possible change in the left-hand side of a linear predicate. Next, a nominal trajectory with an STL score greater than ϵ is computed to obtain the certificate that all closed-loop trajectories satisfy the STL specification. If this is not possible, then the worst-case violation is minimized.

Computing Ω and Γ such that Equation 27 holds is difficult for most hybrid systems. For linear systems, $F(x + \delta x, u + \delta u, w) - F(x, u, 0)$ equals $A\delta x + B\delta u + w$, which means that Equation 27 maps to finding an RCI set. Various results exist for computing RCI sets for linear systems. The methods of Raković and colleagues (96, 97) in particular provide parameterized families of RCI sets in such a way that the best parameters are obtained by convex optimization. The invariance-inducing control policy also maps to a convex program. In Reference 89, we introduced a method of obtaining RCI sets for switching disturbed linear systems such that invariance holds regardless of how the switches occur. Similarly to RCI sets for tube MPC of linear systems, these switch-agnostic RCI sets were used for tube STL MPC of PWA systems (for the technical details, see Reference 89). Example 6 demonstrates the usefulness of this method. Implementing μ^{tube}, which is online, requires a convex program. The main disadvantage of this method is its conservativeness in the design of the tube policy. For many PWA systems, such RCI sets that are robust to all mode switches may not exist. A potential direction that has not yet been studied is the simultaneous design of nominal trajectories and time-varying tubes around them.

Example 6 (from Reference 67). Consider the system in Example 4 with additive disturbances in $W = [-0.2, 0.2]^2$ for all modes. The method described in Reference 89 is employed. The largest STL violation permitted by the tube is $\epsilon = 1.14$. Let $x_0 = (-15, 10)^T$. We design a nominal trajectory with an STL score of 1.14, and hence all disturbed trajectories are guaranteed to satisfy φ. The nominal trajectory has a maximum STL score of 2.5. For this trajectory, the tube policy for disturbed trajectories yields STL scores lower bounded by $2.5 - 1.14 = 1.36$. **Figure 5** shows sample trajectories.

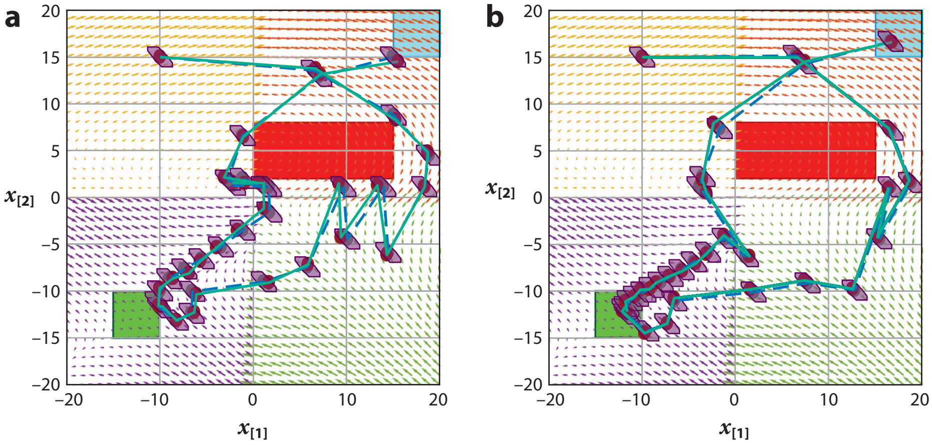

Figure 5

Sample trajectories from Example 6: tube-based design for a disturbed piecewise affine system. All trajectories are guaranteed to have a signal temporal logic score greater than (*a*) 0 or (*b*) 1.36. The robust-control-invariant set Ω is the purple polytope. As in **Figure 3**, the trajectories are synthesized and verified in discrete time, and the lines connecting the circles are included solely to illustrate the time progression. Figure adapted from Reference 67.

SUMMARY POINTS

1. Controllers for complex systems and specifications can be formally synthesized using mathematical optimization. In contrast to automata-based synthesis, optimization does not require finite-state abstractions and is applicable to higher dimensions.
2. For a broad range of problems, optimization-based synthesis is complete and achieves global optimality, whereas these problems are often approximate in automata-based synthesis depending on the resolution of the finite-state abstraction.

FUTURE ISSUES

1. Trajectory optimization using mixed-integer convex programming is a powerful tool for temporal logic control. However, it is too slow for online implementation. Improving the optimization formulation is a potential research direction.
2. Unlike automata-based approaches, the current optimization-based techniques are not as powerful in finding robust feedback policies and the set of admissible initial conditions. This is an important issue that must be addressed.
3. Robust trajectory tracking methods such as tube model predictive control are promising for robust signal temporal logic (STL) control, as there is a connection between STL quantitative semantics and the size of the tube. Efficient methods for computing the best tubes for various classes of STL control problems are open to further investigation.

DISCLOSURE STATEMENT

The authors are not aware of any affiliations, memberships, funding, or financial holdings that might be perceived as affecting the objectivity of this review.

LITERATURE CITED

1. Baier C, Katoen JP. 2008. *Principles of Model Checking*. Cambridge, MA: MIT Press
2. Hinton A, Kwiatkowska M, Norman G, Parker D. 2006. PRISM: a tool for automatic verification of probabilistic systems. In *Tools and Algorithms for the Construction and Analysis of Systems: 12th International Conference*, *TACAS 2006*, ed. H Hermanns, J Palsberg, pp. 441–44. Berlin: Springer
3. Cimatti A, Clarke E, Giunchiglia E, Giunchiglia F, Pistore M, et al. 2002. NuSMV version 2: an open source tool for symbolic model checking. In *Computer Aided Verification: 14th International Conference*, *CAV 2002*, ed. E Brinksma, K Guldstrand Larsen, pp. 359–64. Berlin: Springer
4. Smith S, Tumova J, Belta C, Rus D. 2011. Optimal path planning for surveillance with temporal logic constraints. *Int. J. Robot. Res.* 30:1695–708
5. Chatterjee K, Doyen L, Henzinger TA, Raskin JF. 2006. Algorithms for omega-regular games with imperfect information. In *Computer Science Logic: 20th International Workshop*, *CSL 2006*, ed. Z Ѐsik, pp. 287–302. Berlin: Springer
6. Ding X, Smith SL, Belta C, Rus D. 2014. Optimal control of Markov decision processes with linear temporal logic constraints. *IEEE Trans. Autom. Control* 59:1244–57
7. Tabuada P, Pappas G. 2006. Linear time logic control of discrete-time linear systems. *Trans. Autom. Control* 51:1862–77
8. Gol EA, Lazar M, Belta C. 2012. Language-guided controller synthesis for discrete-time linear systems. In *Proceedings of the 15th International Conference on Hybrid Systems: Computation and Control*, pp. 95–104. New York: ACM
9. Wongpiromsarn T, Topcu U, Murray RR. 2009. Receding horizon temporal logic planning for dynamical systems. In *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) Held Jointly with 2009 28th Chinese Control Conference*, pp. 5997–6004. New York: IEEE
10. Kloetzer M, Belta C. 2008. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Trans. Autom. Control* 53:287–97
11. Yordanov B, Tumova J, Cerna I, Barnat J, Belta C. 2012. Temporal logic control of discrete-time piecewise affine systems. *IEEE Trans. Autom. Control* 57:1491–504
12. Belta C, Yordanov B, Gol EA. 2017. *Formal Methods for Discrete-Time Dynamical Systems*. Cham, Switz.: Springer
13. Tarjan R. 1971. Depth-first search and linear graph algorithms. In *12th Annual Symposium on Switching and Automata Theory (SWAT 1971)*, pp. 114–21. New York: IEEE
14. Gol EA, Belta C. 2013. Time-constrained temporal logic control of multi-affine systems. *Nonlinear Anal. Hybrid Syst.* 10:21–23
15. Bertsimas D, Tsitsiklis JN. 1997. *Introduction to Linear Optimization*. Belmont, MA: Athena Sci.
16. Rawlings JB, Mayne DQ. 2009. *Model Predictive Control: Theory and Design*. Madison, WI: Nob Hill
17. Bemporad A, Morari M. 1999. Control of systems integrating logic, dynamics, and constraints. *Automatica* 35:407–27
18. Ulusoy A, Belta C. 2014. Receding horizon temporal logic control in dynamic environments. *Int. J. Robot. Res.* 33:1593–607
19. Maler O, Nickovic D. 2004. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, ed. Y Lakhnech, S Yovine, pp. 152–66. Berlin: Springer
20. Koymans R. 1990. Specifying real-time properties with metric temporal logic. *Real-Time Syst.* 2:255–99
21. Jing G, Ehlers R, Kress-Gazit H. 2013. Shortcut through an evil door: optimality of correct-by-construction controllers in adversarial environments. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4796–802. New York: IEEE

22. Svorenova M, Cerna I, Belta C. 2015. Optimal temporal logic control for deterministic transition systems with probabilistic penalties. *IEEE Trans. Autom. Control* 60:1528–41

23. Ding XC, Belta C, Cassandras CG. 2010. Receding horizon surveillance with temporal logic specifications. In *49th IEEE Conference on Decision and Control*, pp. 256–61. New York: IEEE

24. Ding XC, Lazar M, Belta C. 2012. Receding horizon temporal logic control for finite deterministic systems. In *12th American Control Conference*, pp. 715–20. New York: IEEE

25. Gol EA, Lazar M, Belta C. 2014. Language-guided controller design for linear systems. *IEEE Trans. Autom. Control* 59:1163–76

26. Pola G, Girard A, Tabuada P. 2008. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica* 44:2508–16

27. Rungger M, Reissig G. 2017. Arbitrarily precise abstractions for optimal controller synthesis. In *2017 IEEE 56th Annual Conference on Decision and Control*, pp. 1761–68. New York: IEEE

28. Papusha I, Fu J, Topcu U, Murray RM. 2016. Automata theory meets approximate dynamic programming: optimal control with temporal logic constraints. In *2016 IEEE 55th Conference on Decision and Control*, pp. 434–40. New York: IEEE

29. Horowitz MB, Wolff EM, Murray RM. 2014. A compositional approach to stochastic optimal control with co-safe temporal logic specifications. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1466–73. New York: IEEE

30. Wolff EM, Topcu U, Murray RM. 2012. Optimal control with weighted average costs and temporal logic specifications. In *Robotics: Science and Systems VIII*, ed. N Roy, P Newman, S Srinivasa, pp. 449–56. Cambridge, MA: MIT Press

31. Li L, Fu J. 2017. Sampling-based approximate optimal temporal logic planning. In *2017 IEEE International Conference on Robotics and Automation*, pp. 1328–35. New York: IEEE

32. Donzé A, Maler O. 2010. Robust satisfaction of temporal logic over real-valued signals. In *Formal Modeling and Analysis of Timed Systems: 8th International Conference, FORMATS 2010*, ed. K Chatterjee, TA Henzinger, pp. 92–106. Berlin: Springer

33. Dokhanchi A, Hoxha B, Fainekos G. 2014. On-line monitoring for temporal logic robustness. In *Runtime Verification, 5th International Conference, RV 2014*, ed. B Bonakdarpour, SA Smolka, pp. 1–20. Cham, Switz.: Springer

34. Ouaknine J, Worrell J. 2006. Safety metric temporal logic is fully decidable. In *Tools and Algorithms for the Construction and Analysis of Systems: 12th International Conference, TACAS 2006*, ed. H Hermanns, J Palsberg, pp. 411–25. Berlin: Springer

35. Donzé A. 2010. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification: 22nd International Conference, CAV 2010*, ed. T Touili, B Cook, P Jackson, pp. 167–70. Berlin: Springer

36. Donzé A, Ferrère T, Maler O. 2013. Efficient robust monitoring for STL. In *Computer Aided Verification: 25th International Conference, CAV 2013*, ed. N Sharygina, H Veith, pp. 264–79. Berlin: Springer

37. Fainekos GE, Pappas GJ. 2009. Robustness of temporal logic specifications for continuous-time signals. *Theor. Comput. Sci.* 410:4262–91

38. Heemels W, Schutter BD, Bemporad A. 2001. Equivalence of hybrid dynamical models. *Automatica* 37:1085–91

39. Bemporad A, Ferrari-Trecate G, Morari M. 2000. Observability and controllability of piecewise affine and hybrid systems. *IEEE Trans. Autom. Control* 45:1864–76

40. De Schutter B, Van den Boom T. 2001. Model predictive control for max-min-plus-scaling systems. In *Proceedings of the 2001 American Control Conference*, Vol. 1, pp. 319–24. New York: IEEE

41. Heemels W, Schumacher JM, Weiland S. 2000. Linear complementarity systems. *SIAM J. Appl. Math.* 60:1234–69

42. Anitescu M, Potra FA. 1997. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dyn.* 14:231–47

43. De Schutter B, De Moor B. 1997. The extended linear complementarity problem and the modeling and analysis of hybrid systems. In *Hybrid Systems V*, ed. P Antsaklis, M Lemmon, W Kohn, A Nerode, S Sastry, pp. 70–85. Berlin: Springer

44. Richards A, How JP. 2002. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the 2002 American Control Conference*, Vol. 3, pp. 1936–41. New York: IEEE
45. Karaman S, Frazzoli E. 2011. Linear temporal logic vehicle routing with applications to multi-UAV mission planning. *Int. J. Robust Nonlinear Control* 21:1372–95
46. Mellinger D, Kushleyev A, Kumar V. 2012. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *2012 IEEE International Conference on Robotics and Automation*, pp. 477–83. New York: IEEE
47. Wang Y, De Schutter B, van den Boom TJ, Ning B. 2013. Optimal trajectory planning for trains—a pseudospectral method and a mixed integer linear programming approach. *Transp. Res. C* 29:97–114
48. Haghighi I, Sadraddini S, Belta C. 2016. Robotic swarm control from spatio-temporal specifications. In *2016 IEEE 55th Conference on Decision and Control*, pp. 5708–13. New York: IEEE
49. Liu Z, Dai J, Wu B, Lin H. 2017. Communication-aware motion planning for multi-agent systems from signal temporal logic specifications. In *2017 American Control Conference*, pp. 2516–21. New York: IEEE
50. Sahin YE, Nilsson P, Ozay N. 2017. Provably-correct coordination of large collections of agents with counting temporal logic constraints. In *Proceedings of the 8th International Conference on Cyber-Physical Systems*, pp. 249–58. New York: ACM
51. Liu Z, Wu B, Dai J, Lin H. 2017. Distributed communication-aware motion planning for multi-agent systems from STL and SpaTeL specifications. In *2017 IEEE 56th Annual Conference on Decision and Control*, pp. 4452–57. New York: IEEE
52. Haghighi I, Jones A, Kong Z, Bartocci E, Gros R, Belta C. 2015. SpaTeL: a novel spatial-temporal logic and its applications to networked systems. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pp. 189–98. New York: ACM
53. Shoukry Y, Nuzzo P, Balkan A, Saha I, Sangiovanni-Vincentelli AL, et al. 2017. Linear temporal logic motion planning for teams of underactuated robots using satisfiability modulo convex programming. In *2017 IEEE 56th Annual Conference on Decision and Control*, pp. 1132–37. New York: IEEE
54. Shoukry Y, Nuzzo P, Saha I, Sangiovanni-Vincentelli AL, Seshia SA, et al. 2016. Scalable lazy SMT-based motion planning. In *2016 IEEE 55th Conference on Decision and Control*, pp. 6683–88. New York: IEEE
55. Shoukry Y, Nuzzo P, Sangiovanni-Vincentelli AL, Seshia SA, Pappas GJ, Tabuada P. 2017. SMC: satisfiability modulo convex optimization. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, pp. 19–28. New York: ACM
56. Abbas H, Winn A, Fainekos G, Julius AA. 2014. Functional gradient descent method for metric temporal logic specifications. In *2014 American Control Conference*, pp. 2312–17. New York: IEEE
57. Pant YV, Abbas H, Quaye RA, Mangharam R. 2018. Fly-by-logic: control of multi-drone fleets with temporal logic objectives. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems*, pp. 186–97. New York: IEEE
58. Pant YV, Abbas H, Mangharam R. 2017. Smooth operator: control using the smooth robustness of temporal logic. In *2017 IEEE Conference on Control Technology and Applications*, pp. 1235–40. New York: IEEE
59. Karaman S, Sanfelice RG, Frazzoli E. 2008. Optimal control of mixed logical dynamical systems with linear temporal logic specifications. In *2008 47th IEEE Conference on Decision and Control*, pp. 2117–22. New York: IEEE
60. Raman V, Donzé A, Maasoumy M, Murray RM, Sangiovanni-Vincentelli A, Seshia SA. 2014. Model predictive control with signal temporal logic specifications. In *53rd IEEE Conference on Decision and Control*, pp. 81–87. New York: IEEE
61. Sadraddini S, Belta C. 2019. Formal synthesis of control strategies for positive monotone systems. *IEEE Trans. Autom. Control* 64:480–95
62. Abbas H, Fainekos G, Sankaranarayanan S, Ivančić F, Gupta A. 2013. Probabilistic temporal logic falsification of cyber-physical systems. *ACM Trans. Embed. Comput. Syst.* 12:95
63. Abbas H, O'Kelly M, Mangharam R. 2017. Relaxed decidability and the robust semantics of metric temporal logic. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, pp. 217–25. New York: ACM
64. Vielma JP. 2015. Mixed integer linear programming formulation techniques. *SIAM Rev.* 57:3–57

65. Wolff EM, Murray RM. 2016. Optimal control of nonlinear systems with temporal logic specifications. In *Robotics Research: The 16th International Symposium ISRR*, ed. M Inaba, P Corke, pp. 21–37. Cham, Switz.: Springer

66. Mazo M, Davitian A, Tabuada P. 2010. PESSOA: a tool for embedded controller synthesis. In *Computer Aided Verification: 22nd International Conference, CAV 2010*, ed. T Touili, B Cook, P Jackson, pp. 566–69. Berlin: Springer

67. Sadraddini S. 2018. *Formal methods for resilient control*. PhD Thesis, Boston Univ., Boston

68. Wolff EM, Topcu U, Murray RM. 2014. Optimization-based trajectory generation with linear temporal logic specifications. In *2014 IEEE International Conference on Robotics and Automation*, pp. 5319–25. New York: IEEE

69. Bemporad A, Borrelli F, Morari M. 2000. Piecewise linear optimal controllers for hybrid systems. In *Proceedings of the 2000 American Control Conference*, Vol. 2, pp. 1190–94. New York: IEEE

70. Löfberg J. 2003. *Minimax approaches to robust model predictive control*. PhD Thesis, Linköping Univ., Linköping, Swed.

71. Farahani SS, Raman V, Murray RM. 2015. Robust model predictive control for signal temporal logic synthesis. *IFAC-PapersOnLine* 48:323–28

72. Sadraddini S, Belta C. 2015. Robust temporal logic model predictive control. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing*, pp. 772–79. New York: IEEE

73. Raman V, Donzé A, Sadigh D, Murray RM, Seshia SA. 2015. Reactive synthesis from signal temporal logic specifications. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pp. 239–48. New York: ACM

74. Sadigh D, Kapoor A. 2015. Safe control under uncertainty. arXiv:1510.07313 [cs.SY]

75. Mehr N, Sadigh D, Horowitz R, Sastry SS, Seshia SA. 2017. Stochastic predictive freeway ramp metering from signal temporal logic specifications. In *2017 American Control Conference*, pp. 4884–89. New York: IEEE

76. Farahani SS, Majumdar R, Prabhu VS, Soudjani SEZ. 2017. Shrinking horizon model predictive control with chance-constrained signal temporal logic specifications. In *2017 American Control Conference*, pp. 1740–46. New York: IEEE

77. Saha S, Julius AA. 2016. An MILP approach for real-time optimal controller synthesis with metric temporal logic specifications. In *2016 American Control Conference*, pp. 1105–10. New York: IEEE

78. Lindemann L, Dimarogonas DV. 2017. Robust motion planning employing signal temporal logic. In *2017 American Control Conference*, pp. 2950–55. New York: IEEE

79. Ghosh S, Sadigh D, Nuzzo P, Raman V, Donzé A, et al. 2016. Diagnosis and repair for synthesis from signal temporal logic specifications. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pp. 31–40. New York: ACM

80. Blanchini F. 1999. Set invariance in control—a survey. *Automatica* 35:1747–67

81. Kerrigan EC, Kouramas KI, Mayne DQ, Raković SV. 2005. Invariant approximations of the minimal robust positively invariant set. *IEEE Trans. Autom. Control* 50:406–10

82. Kerrigan EC. 2000. *Robust constraint satisfaction: invariant sets and predictive control*. PhD Thesis, Univ. Cambridge, Cambridge, UK

83. Sadraddini S, Rudan J, Belta C. 2017. Formal synthesis of distributed optimal traffic control policies. In *Proceedings of the 8th International Conference on Cyber-Physical Systems*, pp. 15–24. New York: ACM

84. Rungger M, Tabuada P. 2017. Computing robust controlled invariant sets of linear systems. *IEEE Trans. Autom. Control* 62:3665–70

85. Lindemann L, Verginis CK, Dimarogonas DV. 2017. Prescribed performance control for signal temporal logic specifications. In *2017 IEEE 56th Annual Conference on Decision and Control*, pp. 2997–3002. New York: IEEE

86. Lindemann L, Dimarogonas DV. 2018. Control barrier functions for signal temporal logic tasks. *IEEE Control Syst. Lett.* 3:96–101

87. Jha S, Raj S, Jha SK, Shankar N. 2018. Duality-based nested controller synthesis from STL specifications for stochastic linear systems. In *Formal Modeling and Analysis of Timed Systems: 16th International Conference, FORMATS 2018*, ed. D Jansen, P Prabhakar, pp. 235–51. Cham, Switz.: Springer

88. Xu Z, Julius A, Chow JH. 2019. Energy storage controller synthesis for power systems with temporal logic specifications. *IEEE Syst. J.* 13:748–59
89. Sadraddini S, Belta C. 2018. Formal guarantees in data-driven model identification and control synthesis. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control*, pp. 147–56. New York: ACM
90. Majumdar A, Ahmadi AA, Tedrake R. 2013. Control design along trajectories with sums of squares programming. In *2013 IEEE International Conference on Robotics and Automation*, pp. 4054–61. New York: IEEE
91. Tedrake R, Manchester IR, Tobenkin M, Roberts JW. 2010. LQR-trees: feedback motion planning via sums-of-squares verification. *Int. J. Robot. Res.* 29:1038–52
92. Langson W, Chryssochoos I, Raković S, Mayne DQ. 2004. Robust model predictive control using tubes. *Automatica* 40:125–33
93. Mayne DQ, Seron MM, Raković S. 2005. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica* 41:219–24
94. Limon D, Alvarado I, Alamo T, Camacho EF. 2010. Robust tube-based MPC for tracking of constrained linear systems with additive disturbances. *J. Process Control* 20:248–60
95. Ghasemi MS, Afzalian AA. 2017. Robust tube-based MPC of constrained piecewise affine systems with bounded additive disturbances. *Nonlinear Anal. Hybrid Syst.* 26:86–100
96. Raković SV, Kerrigan EC, Mayne DQ, Kouramas KI. 2007. Optimized robust control invariance for linear discrete-time systems: theoretical foundations. *Automatica* 43:831–41
97. Raković SV, Baric M. 2010. Parameterized robust control invariant sets for linear systems: theoretical advances and computational remarks. *IEEE Trans. Autom. Control* 55:1599–614