Runtime Assurance from Signal Temporal Logic Safety Specifications

Luke Baird and Samuel Coogan

***Abstract*— In this paper, we propose a runtime assurance mechanism for online verification of a control system given a signal temporal logic (STL) specification that, at each time step, must hold for the remaining state trajectory. Given a nominal control input, we propose a mechanism that minimally adjusts the input at each time step in order to ensure existence of future inputs that maintain satisfaction of the STL specification. Because STL constraints generally impose requirements on future states, the runtime assurance mechanism also enforces continued satisfaction of the STL constraint evaluated at all past time steps. Lastly, to ensure a feasible input is always available, we provide a novel characterization of a persistently feasible set and require that the system state is always able to reach this set. We formulate this approach as a mixed integer convex program and demonstrate it on examples.**

I. Introduction

Modern control systems are highly complex and generally are intractable to fully verify offline prior to their deployment [1]. Despite extensive testing, unpredictable edge cases may arise due to time and cost constraints on testing. For instance, learning-based controllers often have no simple mathematical representation and are difficult to fully characterize.

Instead of performing offline verification of a control system, one path to guaranteed safe behavior is to verify a system online using a runtime assurance (RTA) mechanism [2]. RTA mechanisms monitor control systems to detect, alert, and act when an unsafe condition is detected. This differs from a simple monitoring scheme in that RTA automatically takes action when an unsafe condition is imminent. Many techniques aim to enforce safe or correct behavior online and thus fit in the broad category of RTA. For example, a simple but effective strategy, referred to as the simplex architecture [3], is to switch to a backup controller if needed. A large collection of recent literature uses control barrier functions to adjust control inputs near the boundary of a safe region [4]–[6]. See [2] for further discussion on various approaches to RTA.

With a few exceptions, RTA typically focuses on safety defined as avoiding a particular unsafe subset of the state-space, that is, invariance conditions. However, many practical specifications are not invariance conditions. For example, the requirement that "temperature can exceed a threshold for no more than 2 seconds in any 4 second window" cannot be posed as an invariance condition. To address a broader class of specifications, we utilize *temporal logics*. Temporal logics are formal languages for encoding logical propositions with respect to time and are well-suited for specifying behavior of control systems [1], [7]. In particular, in this paper, we focus on Signal Temporal Logic (STL), which allows for quantitative satisfaction semantics via a robustness metric [8] and is well-suited for specifying behavior of control systems [9]. Our RTA architecture is shown in Figure 1.

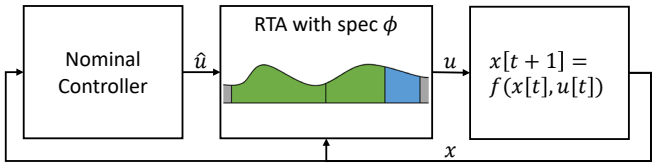

Fig. 1. Runtime assurance (RTA) of a signal temporal logic (STL) specification ϕ. A nominal controller provides inputs to a RTA which monitors and filters these inputs to ensure that there always exists a control strategy that satisfies ϕ for all time, i.e. guarantees safety. At each time t, the RTA uses knowledge of the discretized dynamical system to calculate future states to predict future safety violation dependent on the horizon of ϕ.

There are several tools and developing theory for evaluating and synthesizing from STL formula efficiently. Breach is a Matlab toolbox that evaluates STL formulas either offline or online [10]. The paper [11] uses Breach to monitor an airpath system in a diesel engine to ensure overshoot limits and transient decay requirements are met. `Stlcg` is a Python library for controller synthesis from STL constraints that uses autodifferentiation tools to efficiently compute gradients of STL robustness metrics [12]. Several works propose using smooth approximations to $\min$ and $\max$ operations in order to ensure differentiability of STL robustness metrics [12], [13]. Most relevant to this paper, [14] introduces a python library `stlpy` that efficiently encodes STL formulas for control synthesis as a Mixed-Integer Convex Program (MICP) using fewer binary variables than prior methods.

In this paper, we propose a RTA mechanism for a discrete-time control system subject to a STL safety constraint on the state that must hold for all times along the system execution. Given a nominal controller that may not always satisfy the STL constraint, we propose solving a MICP at each time step that computes a minimally invasive safe control input minimizing the distance to the desired nominal input. Since STL specifications generally impose constraints on future states, the MICP ensures safety by identifying a possible sequence of future inputs that guarantee satisfaction of the STL formula when evaluated at past and future states. When the STL constraint has a finite horizon, that is, its satisfaction is determined by a finite subset of the state trajectory, the MICP need only consider a finite number of past evaluations

This work was supported in part by the National Science Foundation under awards #1749357 and #2219755.

Luke Baird and Samuel Coogan are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30318, USA {lbaird38,sam.coogan}@gatech.edu. S. Coogan is also with the School of Civil and Environmental Engineering.

of the STL formula. In turn, to ensure a finite number of future evaluations, we propose a novel characterization of persistently safe sets defined over multiple time steps of the trajectory and enforce in the MICP that the state trajectory is able to reach such a set. Provided that the system is initialized such that safety is possible, our proposed approach is guaranteed to maintain safety. We implement our approach by modifying the `stlpy` library to suit our problem domain. Although our approach is developed for future-time STL, by applying ideas from [15] we can incorporate past-time STL, including infinite past-time STL. Computationally, only a finite number of historical time steps are required to evaluate a past-time STL formula, even with an unbounded history. This is illustrated in the case study.

The RTA approach in this paper draws from and is related to several areas of research. First, monitoring a system trajectory against an STL formula to detect violation has been considered in [10], [11], [15]–[19]. These monitoring approaches generally do not consider the system generating the signal to be monitored and, in particular, do not provide a method for adjusting control actions to prevent violation. Next, while our approach poses a MICP that is reminiscent of optimization approaches to trajectory planning and model predictive control from STL constraints [9], [20], [21], our problem setting is different in that we aim to alter a nominal controller to ensure at all times satisfaction of a STL safety constraint. Mathematically, this requires a novel characterization of persistent feasibility and an MICP that appropriately considers STL evaluations at past states. Similar to this paper, [21] poses a MICP to minimize a cost at each time step, but does not guarantee persistent feasibility, instead synthesizing a minimally violating controller. The papers [9], [20] present a framework for enforcing satisfaction of a finite time horizon STL formula for all time by employing invariant set theory. These works propose treating a slice of the historical state up to the present as a higher-dimensional dynamical system and ensure persistent feasibility by finding robust control invariant sets for the higher dimensional system. Although our characterization of persistently feasible sets is similar, our work approaches the safety problem from an RTA framework while building on the monitoring literature, hence permitting extensions to past-time STL. Our selection of a terminal set condition for persistent feasibility is natural for an STL formula in an RTA framework. Furthermore, the theory in [20] requires positive monotone systems to apply the proposed solution techniques and account for disturbances.

The rest of the paper is laid out as follows. In Section II, we provide a brief overview of STL. We then formally pose the runtime assurance problem that is the focus of this paper in Section III. Then, we provide a necessary terminal set condition that leads to a theoretical guarantee of safety in Section IV followed by an algorithmic implementation in Section V. Section VI demonstrates our method while finally Section VII concludes the paper with planned future work.

II. Signal Temporal Logic Preliminaries

In this section, we provide a brief overview of Signal Temporal Logic (STL). Let x be a discrete time signal so that $x[t] \in \mathbb{R}^n$ is the value of x at time t. STL is defined using the syntax [9], [13]

$$\phi \triangleq \pi | \neg\phi | \phi \wedge \psi | \phi \mathcal{U}_{[t_1,t_2]} \psi \tag{1}$$

where π is a logical predicate of the form

$$\pi = (\mu(x[t]) - c \geq 0) \tag{2}$$

for a *predicate function* $\mu : \mathbb{R}^n \to \mathbb{R}$ and a constant c. The operators conjunction $\wedge$, until $\mathcal{U}$, and negation $\neg$ may be used to define disjunction $\vee$, eventually $\Diamond$, and always $\Box$.

The robustness ρ^ϕ of a specification ϕ is calculated at a time t recursively as [13]

$$\begin{aligned}
&\rho^\pi(x,t) = \mu^\pi(x[t]) - c \\
&\rho^{\neg\phi}(x,t) = -\rho^\phi(x,t) \\
&\rho^{\phi\wedge\psi}(x,t) = \min\left(\rho^\phi(x,t), \rho^\psi(x,t)\right) \\
&\rho^{\phi\vee\psi}(x,t) = \max\left(\rho^\phi(x,t), \rho^\psi(x,t)\right) \\
&\rho^{\Box_{t_1,t_2}\phi}(x,t) = \max_{t'\in[t+t_1,t+t_2]} \left(\rho^\phi(x,t')\right) \\
&\rho^{\Diamond_{t_1,t_2}\phi}(x,t) = \min_{t'\in[t+t_1,t+t_2]} \left(\rho^\phi(x,t')\right) \\
&\rho^{\phi\mathcal{U}_{[t_1,t_2]}\psi}(t,k) = \max_{t'\in[t+t_1,t+t_2]} \\
&\min\left(\rho^\phi(x,t'), \min_{t''\in[t+t_1,t']} \left(\rho^\psi(x,t'')\right)\right).
\end{aligned}$$

If a STL formula ϕ has non-negative robustness for signal x at time 0, then we write $x \models \phi$ [13].

The time horizon of an STL formula ϕ, denoted $\|\phi\|$, is defined as the number of future time steps of a signal necessary to evaluate an STL formula and is characterized recursively as [18]

$$\begin{aligned}
&\|\pi\| = 0 \\
&\|\neg\phi\| = \|\phi\| \\
&\|\phi \wedge \psi\| = \max\left(\|\phi\|, \|\psi\|\right) \\
&\|\phi \vee \psi\| = \max\left(\|\phi\|, \|\psi\|\right) \\
&\|\Box_{t_1,t_2}\phi\| = \|\phi\| + t_2 \\
&\|\Diamond_{t_1,t_2}\phi\| = \|\phi\| + t_2 \\
&\|\phi\mathcal{U}_{[t_1,t_2]}\psi\| = \max\left(\|\phi\|, \|\psi\|\right) + t_2.
\end{aligned}$$

III. Problem Formulation

Consider the discrete dynamical system

$$x[t] = f(x[t], u[t]) \tag{3}$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$ is the state and $u \in \mathcal{U} \subset \mathbb{R}^m$ is the input. Suppose there is a safety constraint for the system given as an STL specification that must be satisfied at each time step, as formalized in the next definition.

Definition 1. *Given an STL formula ϕ, a state trajectory x of the system (3) is safe with respect to ϕ, or simply* safe, *if*

$$\rho^\phi(x,t) \geq 0 \quad \textit{for all } t \geq 0. \tag{4}$$

Now, suppose that there exists some unverified nominal controller that provides inputs $\hat{u}[t]$ to the system. The controller is unverified, meaning that this controller is not guaranteed to generate a safe state trajectory of the system. For example, the controller might be designed to achieve some ancillary performance objective with inadequate consideration for the safety constraints. The goal of this paper is to develop a runtime assurance (RTA) mechanism to modify the nominal controller to ensure safety.

Problem Statement. *Given the system in (3), an initial state $x[0] \in \mathcal{X}$ from which safety is possible, an STL formula ϕ with $\|\phi\| < \infty$, and a nominal controller providing inputs $\hat{u}[t]$, develop an RTA mechanism that modifies the nominal control inputs in order to provably guarantee that the resulting state trajectory is safe with respect to ϕ while minimizing the deviation from nominal inputs.*

Although the theory presented in this paper is developed for future time STL formulas with finite horizons, we demonstrate in the case study that by applying the methods proposed in [15], we can extend this work to past-time STL operators including unbounded past time operators.

IV. Sliding Window Control Solution

In this section, we propose a solution to the problem defined in Section III that consists of solving an optimization problem at each time step. The optimization problem considers a sliding window of future and past states and control actions to ensure safety. We start by defining a persistently safe set that is appropriate for a setting with a STL safety specification.

Definition 2 (persistently safe set, tail length, *and* persistently safe control strategy)**.** *Given an STL formula ϕ and the dynamical system (3), suppose there exists a set $\mathcal{S} \subset \mathcal{X}^b$ for some $b \geq 1$ along with a feedback control strategy $\mu : \mathcal{X}^b \to \mathcal{U}$ such that, at time t, if $(x[t-b+1], x[t-b+2], \ldots, x[t]) \in \mathcal{S}$, then applying the control strategy μ maintains satisfaction of the STL for all future time, that is, applying the inputs $u[\tau] = \mu(x[\tau - b + 1], x[\tau - b + 2], \ldots, x[\tau])$ for all $\tau \geq t$ ensures $\rho^{\phi}(x, \tau - b + 1) \geq 0$ for all $\tau \geq t$. Then $\mathcal{S}$ is called a* persistently safe set *for ϕ and (3), b is the* tail length *of S, and μ is a* persistently safe control strategy *for $\mathcal{S}$.*

At each time t, our proposed solution projects the future state of the system to N time steps and ensures that: 1) the final b steps of the state projection are in some persistently safe set $\mathcal{S}$ with tail length b, and 2) robustness is ensured for all states up to $\|\phi\|$ time steps in the past and N time steps in the future.

The persistently safe set $\mathcal{S}$ is necessary for our receding horizon solution. At each iteration in our algorithm, we use this set to ensure that when we find a set of control inputs that will keep the system safe at time t, there will also exist a set of control inputs that will keep the system safe at time $t+1$, $t+2$, and so on.

Proposition 1. *Consider a nonlinear discrete dynamical system (3). Given STL formula ϕ, let $h = \|\phi\| < \infty$, and suppose that $\mathcal{S}$ is a persistently safe set for ϕ and (3) with tail length b and persistently safe control strategy μ. Let integer $N \geq b$. At some time t', if there exists a sequence of control inputs $u[t'], \ldots, u[t'+N-1]$ such that*

$$x[t'+N-b+1], \ldots, x[t'+N] \in \mathcal{S} \tag{5}$$

and

$$\rho^{\phi}(x, \tau) \geq 0, \forall \tau \in \{t'-h, \ldots, t'+N-b\}, \tag{6}$$

where, in order to evaluate $\rho^{\phi}(x, \tau)$, we append this sequence with μ, i.e., we take $u[t] = \mu(x[t-b+1], \ldots, x[t])$ for $t \geq t'+N-b+h$, then there exists a sequence of inputs such that $\rho^{\phi}(x, t) \geq 0$ for all $t \geq t'-h$.

Proof. At time t', apply the control strategy $u[t'], \ldots, u[t'+N-1]$. By (6), $\rho^{\phi}(x, t) \geq 0$ for all t satisfying $t'-h \leq t \leq t'+N-b$. Next, (5) implies that by substituting $t = t'+N$ into definition 2, there exists a control strategy such that $\rho^{\phi}(x, t) \geq 0$ for all $t \geq t'+N-b+1$. ■

Note that for the sequence of inputs $u[t'], \ldots, u[t'+N-1]$, there is flexibility in the inputs as the state need not remain in $\mathcal{S}$ until the last b steps. Thus, similar to MPC, selecting a larger value of N permits greater flexibility and less conservativeness in the verified input sequence.

Next, we show that a natural method for obtaining a persistently safe set is to compute a control invariant set.

Definition 3 (control invariant *and* invariance-enforcing control strategy)**.** *The set $\mathcal{C}$ is* control invariant *for the system (3) if, for all $x \in \mathcal{C}$, there exists $u \in \mathcal{U}$ such that $f(x, u) \in \mathcal{C}$. For a control invariant set, if $\mu : \mathcal{C} \to \mathcal{U}$ is a feedback control strategy such that $f(x, \mu(x)) \in \mathcal{C}$, then whenever $x[t'] \in \mathcal{C}$ for some t', applying μ thereafter results in a state trajectory satisfying $x[t] \in \mathcal{C}$ for all $t \geq t'$,* i.e., *$\mathcal{C}$ is forward invariant. Such a control strategy μ is called* invariance-enforcing *for $\mathcal{C}$.*

Proposition 2. *Given STL formula ϕ, suppose Ω_0 has the following property: for any signal x with $x[t] \in \Omega_0$ for all $t \geq 0$, $x \models \phi$. Then any control invariant set $\mathcal{C} \subseteq \Omega_0$ is a persistently safe set with tail length $b = 1$, and any invariance-enforcing control strategy for $\mathcal{C}$ is a persistently safe control strategy.*

Proof. Substituting $b = 1$ into Definition 2 implies that, given $x[t] \in \mathcal{S}$, there must exist a control strategy $u[\tau] = \mu(x[\tau])$ for all $\tau \geq t$ such that $\rho^{\phi}(x, \tau) \geq 0$. By hypothesis, any trajectory x with $x[t] \in \mathcal{C}$ for all $t \geq 0$ implies that $x \models \phi$. Therefore, it is enough to apply an invariance-enforcing control strategy for $\mathcal{C}$ to guarantee positive robustness for all $\tau \geq t$. ■

Given a set Ω_0 satisfying the conditions of Proposition 2, it is generally preferable to use the largest control invariant subset of Ω_0 as the persistently safe set; this set is called the *maximal control invariant set* for Ω_0. In practice, the maximal control invariant is found by iteratively calculating

$$\Omega_{k+1} = \text{Pre}(\Omega_k) \cap \Omega_k \tag{7}$$

until $\Omega_{k+1} = \Omega_k$ where $\text{Pre}(\cdot)$ is the precursor operator given by $\text{Pre}(A) = \{x \in \mathbb{R}^n : \exists u \in \mathcal{U} \text{ s.t. } f(x,u) \in A\}$ [22].

V. Implementation as a Mixed-Integer Convex Program

In this section we propose an algorithm to solve the problem formulation of Section III using the theoretical developments of Section IV. At each time step, we calculate a safe backup control strategy that satisfies the conditions of Proposition 1 while minimizing the deviation from the nominal input. Formally, at each time t we solve the optimization problem

$$\begin{aligned} \min_{\mu=\{\mu[t],\ldots,\mu[t+N-1]\}} \quad & \|\hat{u}[t] - \mu[t]\| \qquad (8) \\ \text{s.t. } x[\tau+1] &= f(x[\tau], \mu[\tau]), \qquad t-h \le \tau \le t+N \\ x[\tau] &\in \mathcal{S}, \qquad t+N-b+1 \le \tau \le t+N \\ \rho^{\phi}(x,\tau) &\ge 0, \qquad \max\{t-h, 0\} \le \tau \le t+N-b \end{aligned}$$

where $\mathcal{S}$ is a persistently safe set, $\hat{u}$ is a nominal control strategy, and μ is the optimization variable. The algorithm is given in Algorithm 1.

Algorithm 1 Runtime assurance for ϕ safety
Ensure $x[t+1] = f(x[t], u[t])$ satisfies ϕ

Input: ϕ, $\hat{u}$, $\mathcal{S}$, b, N
Output: u where $\rho^{\phi}(x,t) \ge 0$, $\forall t$
Initialization : $x[0] \in \mathcal{X}_{\text{safe}}$
1: $h \leftarrow \|\phi\|$
2: **while** $t \ge 0$ **do**
3: $u[t] \leftarrow \mu[t]$ where μ is the minimizer of (8)
4: $t \leftarrow t+1$
5: **end while**

To show that this algorithm guarantees safety for all time, we must first define a *safe initial condition*.

Definition 4 (safe initial condition)**.** *Given an STL formula ϕ, the dynamical system (3), and a persistently safe set $\mathcal{S}$, a* safe initial condition *is a state $x[0] \in \mathcal{X}$ such that there exists a control strategy $u[0], \ldots, u[h+N-1]$ where $\rho^{\phi}(x,0), \ldots, \rho^{\phi}(x, h+N-b) \ge 0$ and the conditions for Proposition 1 holds at time $t = h$. We denote the set of safe initial conditions by $\mathcal{X}_{safe}$.*

Finally, we make the following technical assumption on the predicate functions comprising the safety specification in order to apply theory from [9], [14].

Assumption 1. *STL formula ϕ has convex predicate functions μ. That is, $\mu(\theta x_1[t] + (1-\theta)x_2[t]) \le \theta\mu(x_1[t]) + (1-\theta)\mu(x_2[t])$ where $\theta \in [0,1]$ and $x_1, x_2 \in \mathcal{X}$.*

Theorem 1. *Consider a nonlinear discrete dynamical system (3) with STL formula ϕ, $h = \|\phi\| < \infty$ and persistently safe set $\mathcal{S}$. If $x[0]$ is a safe initial condition, then applying Algorithm 1 to filter any nominal controller $\hat{u}[t]$ will maintain safety for all time.*

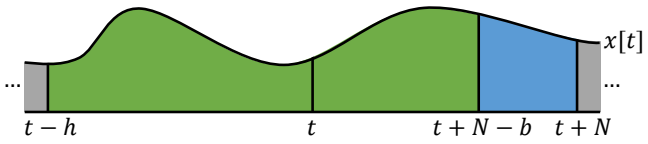

Fig. 2. Illustration of the constraints matching Proposition 1. Over the green region we must ensure positive robustness on x, while over the blue region we must ensure that x is in a persistently safe set.

Proof. By Proposition 1 and Definition 4, there exists a safe control strategy for all future time. At each iteration t of Algorithm 1, we construct an optimization problem with constraints corresponding to Proposition 1. By [9], a mixed-integer encoding of STL constraints is both sound and complete. Therefore, by the construction of Proposition 1 and Definition 4, if there is a solution at time t, then there is a solution for all future time including time $t+1$, thereby guaranteeing persistent feasibility of the algorithm. ■

Remark. *In general, we can provide a library of persistently safe sets with varying tail lengths instead of a static precomputed persistently safe set.*

This method is illustrated in Figure 2. At each time t, we must ensure that ϕ holds at each time τ where $\max(0, t-h) \le \tau \le t+N-b$ while the remaining steps up to $t+N$ must be in a persistently safe set.

Optimization problems in the form of (8) with a convex cost, STL constraints with convex predicates, and linear systems may be solved by posing the problem as a MICP. The advantage of MICPs is that mixed-integer encoding of robustness constraints yield sound and complete solutions for any STL formula [9]. However, due to the NP-hardness of these problems, the application of MICPs are limited to short STL time horizons and relatively simple STL formulae [9]. Note that in general, STL formula form non-convex sets and their evaluation is NP-hard even with convex predicates [23]. For convex STL formulae other methods like smooth STL approximations with gradient descent may be more appropriate [13].

The idea is to decompose an STL formula into its components (conjunctions, disjunctions, and predicates) forming a tree-like structure where the lowest level is composed of predicates. These predicate constraints are then encoded using a big-M formulation [9].

Often, the CPU time required to solve MICPs is directly related to the number of binary variables involved. Much research has been done in reducing the number of binary variables. The paper [14] builds on the work of [9], where the number of binary variables does not increase with conjunctions and increases logarithmically with disjunctions in exchange for more linear constraints. The code of [14] is written to evaluate an STL formula robustness at a single time t. In our implementation, we modify this open source library to accommodate multiple time steps.

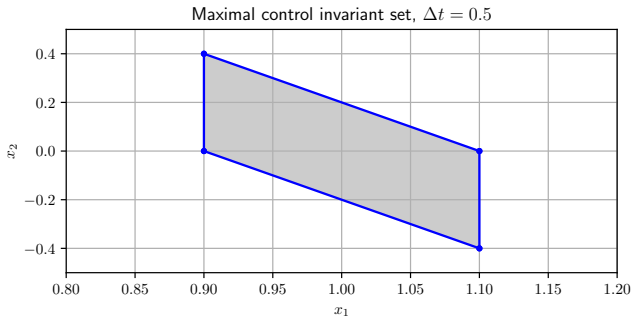

Fig. 3. Maximal control invariant set for a double integrator system with safe set $0.9 \leq x_1[t] \leq 1.1$ and discretization size $\Delta t = 0.5$.

VI. Case Study

Consider a discretized double integrator system where $x \in \mathbb{R}^2$, $u \in [-1, 1]$, $y \in \mathbb{R}$ given by

$$x[t+1] = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} x[t] + \begin{pmatrix} 0 \\ \Delta t \end{pmatrix} u[t] \tag{9}$$

$$y[t] = \begin{pmatrix} 1 & 0 \end{pmatrix} x[t]. \tag{10}$$

We consider the following specification: y must stay within 10% of unity, but may leave this range for up to two seconds if it resettles in this range for two seconds. This specification is a slight modification from [15], where the past-time version of this formula is used to monitor an engine air-to-fuel normalized stoichiometric ratio online. This is encoded as the STL formula

$$\phi = \neg(y \text{ in bounds}) \implies \Diamond_{[0,2]} \Box_{[0,2]} (y \text{ in bounds}) \tag{11}$$

where "y in bounds" is $(y \geq 0.9) \wedge (-y \geq -1.1)$. The implication operator may be written as a disjunction: $A \implies B$ is logically equivalent to $\neg A \vee B$.

The horizon of ϕ is $h = \|\phi\| = 4/\Delta t$. We obtain a persistently safe set $\mathcal{S}$ via Proposition 2 as follows. Let $\Omega_0 = \{x : 0.9 \leq y \leq 1.1\}$, and notice that Ω_0 satisfies the hypothesis of Proposition 2, namely, if $x[t] \in \Omega_0$ for all $t \geq 0$, then $x \models \phi$. To find a safe maximal control invariant set, we apply (7) recursively. For linear time-invariant systems, this calculation uses standard operations on polytopes [22]. For the discrete double integrator system, this routine terminates after $1/\Delta t$ iterations. An example set is plotted for $\Delta t = 0.5$ in Figure 3. Using the notation from Proposition 1, we then take $b = 1$ and choose $N = h$.

We implement Algorithm 1 in Python. We use a modified version of `stlpy` [14] to accommodate evaluation of an STL formula over past and future time relative to a nominal current time, and we use Gurobi to solve the resulting MICP. Because $u[t]$ is a scalar, we select a cost of $|\hat{u}[t] - u[t]|$ and introduce a slack variable to encode this as a linear cost with linear constraints. The relative run time results are tabulated in table I, executed on a 2022 Dell Precision 5570 with Windows 11.

An example is shown first for a constant nominal input of $\hat{u} \equiv 1$, $x[0] = \begin{pmatrix} 1.1 & 0 \end{pmatrix}^\top$, and $\Delta t = 1$ in Figure 4. The resulting output trace stays at 1.1 for two seconds, changes to 2.1 for two seconds, and then returns to 1.1 for two seconds.

TABLE I

Relative run time for time steps and inputs.

Input	Δt	Formula	Compute time per step [s]
$\hat{u} \equiv 1$	1.0	ϕ	0.0099
$\hat{u} \equiv 1$	0.25	ϕ	0.043
$\hat{u} =$(13)	1.0	ψ	0.011
$\hat{u} =$(13)	0.25	ψ	0.049

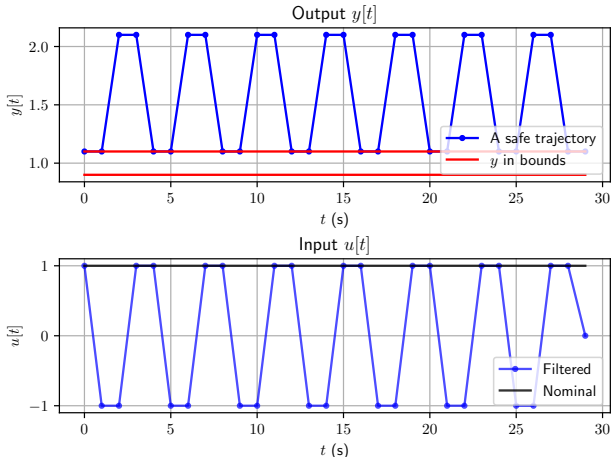

Fig. 4. Safe trajectory (top) and input (bottom) for a nominal input of $\hat{u} \equiv 1$ and discretization step of $\Delta t = 1$.

Next, we consider $\hat{u} \equiv 1$, $x[0] = \begin{pmatrix} 0.0 & 0 \end{pmatrix}^\top$, and $\Delta t = 0.25$. Here, $\|\phi\| = 8$, and thus the output trace spends at most eight time steps outside of the range $0.9 \leq y \leq 1.1$ before returning into that range for eight time steps. The results are plotted in Figure 5.

Now, consider adding an additional constraint where if y exceeds 1.3 for two seconds or more in the past, it must then always remain below 1.3. Now, the specification becomes

$$\begin{aligned} \psi = (\Diamond_{[2,\infty)}(y \geq 1.3) \implies y \leq 1.3) \wedge \\ \neg(y \text{ in bounds}) \implies \Diamond_{[0,2]} \Box_{[0,2]} (y \text{ in bounds}) \end{aligned} \tag{12}$$

where $\Diamond$ is eventually in the past. This specification demonstrates that by employing methods from the monitoring literature [15], we can further generalize the approach presented in this paper to infinite past time formulas. Consider a sinusoidal nominal input sequence

$$\hat{u}[t] = -0.3 \cos \frac{\pi}{30} t + 0.2 \tag{13}$$

Let $x[0] = \begin{pmatrix} 0.9 & 0 \end{pmatrix}^\top$ and $\Delta t = 0.25$. Figure 6 shows that most of the time the filtered input tracks the nominal input signal and remains safe. Whenever the system approaches an imminent unsafe future condition, the RTA mechanism overrides unsafe inputs and returns the system to a safe operating point. Note that the signal exceeds $y \geq 1.3$ only once and for less than two seconds. The code used to produce these plots may be found at `https://github.com/gtfactslab/Baird_ACC2023`.

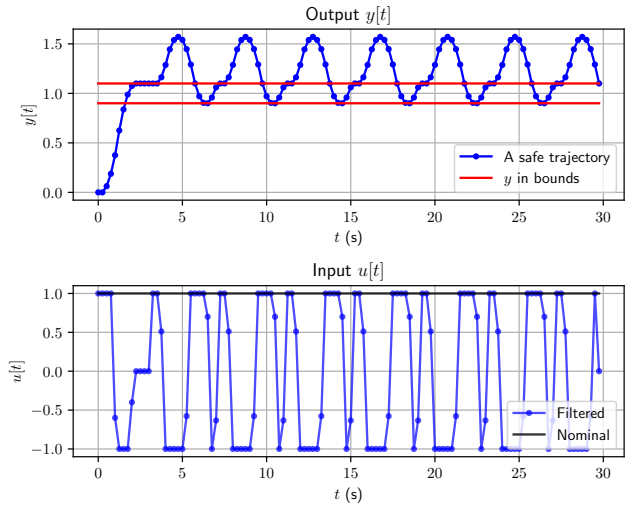

Fig. 5. Safe trajectory (top) and input (bottom) for a nominal input of $\hat{u} \equiv 1$ and discretization step of $\Delta t = 0.25$.

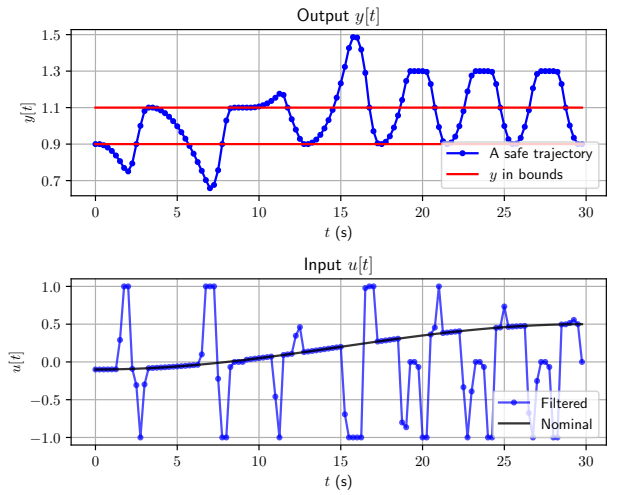

Fig. 6. Safe trajectory (top) and input (bottom) for a nominal input of $\hat{u}[t] = -0.3 \cos \frac{\pi}{30} t + 0.2$ and discretization step of $\Delta t = 0.25$.

VII. Conclusion and Future Work

In this paper, we presented a method for maintaining a STL safety specification at each time step of a discrete-time control system using a runtime assurance mechanism that filters nominal inputs by solving a MICP. The specification is assumed to have a finite horizon so that its satisfaction at a particular time step only depends on a finite number of future steps. The constraints in the MICP ensure satisfaction of the specification at past, present, and future steps and include necessary conditions for persistent safety in order to always guarantee feasibilty. Thus, our method is both sound and complete. Future work will explore allowing bounded uncertainty in the control system.

References

[1] C. Baier and J. Katoen, *Principles of model checking*. MIT press, 2008.

[2] K. Hobbs, M. Mote, M. Abate, S. Coogan, and E. Feron, "Run time assurance for safety-critical systems: An introduction to safety filtering approaches for complex control systems," *arXiv preprint arXiv:2110.03506*, 2021. [Online]. Available: https://arxiv.org/pdf/2110.03506.pdf

[3] J. G. Rivera and A. A. Danylyszyn, *Formalizing the uni-processor simplex architecture*. Citeseer, 1995.

[4] L. Wang, A. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

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

[6] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*. IEEE, 2019, pp. 3420–3431.

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

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

[9] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An optimization perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 115–140, 2019.

[10] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *International Conference on Computer Aided Verification*. Springer, 2010, pp. 167–170.

[11] J. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. Seshia, "Robust online monitoring of signal temporal logic," *Formal Methods in System Design*, vol. 51, no. 1, pp. 5–30, 2017.

[12] K. Leung, N. Aréchiga, and M. Pavone, "Back-propagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2020, pp. 432–449.

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

[14] V. Kurtz and H. Lin, "Mixed-integer programming for signal temporal logic with fewer binary variables," *IEEE Control Systems Letters*, vol. 6, pp. 2635–2640, 2022.

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

[16] D. Ničković and T. Yamaguchi, "Rtamt: Online robustness monitors from stl," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2020, pp. 564–571.

[17] E. A. Gol, "Efficient online monitoring and formula synthesis with past stl," in *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 2018, pp. 916–921.

[18] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

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

[20] S. Sadraddini and C. Belta, "Formal synthesis of control strategies for positive monotone systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 2, pp. 480–495, 2018.

[21] ——, "Robust temporal logic model predictive control," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2015, pp. 772–779.

[22] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[23] V. Raman, A. Donzé, M. Maasoumy, R. Murray, A. Sangiovanni-Vincentelli, and S. Seshia, "Model predictive control with signal temporal logic specifications," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 81–87.