Signal Temporal Logic meets Hamilton-Jacobi Reachability: Connections and Applications

Mo Chen[1], Qizhan Tam[1], Scott C. Livingston[2], and Marco Pavone[1]★

[1] Dept. of Aeronautics and Astronautics, Stanford University
{mochen72, qtam, pavone}@stanford.edu
[2] rerobots, Inc.
scott@rerobots.net

Abstract. Signal temporal logic (STL) and Hamilton-Jacobi (HJ) reachability analysis are effective mathematical tools for formally analyzing the behavior of robotic systems. STL is a specification language that uses a combination of logic and temporal operators to precisely express real-valued and time-dependent requirements on system behaviors. While recursively defined STL specifications are extremely expressive and controller synthesis methods exist, so far there has not been work that quantifies the set of states from which STL formulas can be satisfied. HJ reachability, on the other hand, is a method for computing the reachable set, that is the set of states from which a system is able to reach a goal while satisfying state and control constraints. While reasoning about system requirements through sets of states is useful for predetermining whether it is possible to satisfy desired system properties as well as obtaining state feedback controllers, so far the applicability of HJ reachability has been limited to relatively simple reach-avoid specifications. In this paper, we merge STL and HJ reachability into a single framework that combines the key advantage of both methods – expressiveness of specifications and set quantification. To do this, we establish a correspondence between temporal and reachability operators, and utilize the idea of least-restrictive feasible controller sets (LRFCSs) to break down controller synthesis for complex STL formulas into a sequence of reachability and elementary set operations. LRFCSs are crucial for avoiding controller conflicts among the different reachability operations. In addition, the synthesized state feedback controllers are guaranteed to satisfy STL specifications if determined to be possible by our framework, and violate specifications minimally if not. We demonstrate our method through numerical simulations and robotic experiments.

1 Introduction

In recent years the number of safety-critical applications of robotics has grown quickly, for example with autonomous urban vehicles and surgical robots, where system failures can lead to loss of human life or large costs. In fact, safety analysis can be considered one of the key bottlenecks for the pace of development and more widespread deployment of autonomous systems. In safety-critical systems, formal guarantees are needed to ensure that the system behaves as expected, under the appropriate modelling assumptions such as actuation limits, external disturbances, and dynamic model. State-of-the-art

★ This work was supported by the Office of Naval Research YIP program (Grant N00014-17-1-2433) and by DARPA under the Assured Autonomy program. M. Chen and Q. Tam contributed equally to this work.

classes of methods making such guarantees include robust planning [16,24], reachability analysis [3,6], and temporal logic-based model checking and synthesis [1,4,14].

In particular, Signal Temporal Logic (STL) has gained popularity recently [8,10] due to a number of key advantages. For instance, it explicitly treats real-valued variables and dense-time requirements [18] both of which are essential in practical robotics. Also, in addition to the usual Boolean semantics, STL has quantitative semantics that provide robustness estimates of satisfaction by system trajectories [8,10]. The robustness estimate, a real-valued function, indicates satisfaction of an STL formula in terms of zero superlevel sets, which allows the specifications to represent subsets of the continuous state space. Checking the satisfaction of a specification amounts to evaluating a state on such a function [8]. This function representation of specifications allows modern optimization-based techniques to be used for controller synthesis [21,22]. Viewed from the perspective of reachability analysis and other verification techniques, STL provides a rich language for precisely describing complex system requirements through recursive application of logical and temporal operators. However, without reasoning about behaviors of sets of states as is done in reachability analysis, synthesizing state feedback controllers, as well as efficiently quantifying whether it is possible for any given state to satisfy a specification is challenging, especially for general nonlinear systems.

Reachability analysis provides a complementary perspective in system verification. Here, one is concerned with computing the reachable set (RS), defined as the *set of states* from which a system, with a given dynamics model, can reach some target set of states while satisfying state and control constraints. Reachability has also been extensively studied in the past couple of decades, with a large variety of methods for computing RSs [9,11,12,15,19] and many application domains [5,13,17]. In particular, Hamilton-Jacobi (HJ) reachability provides the globally optimal RS and feedback controller for general nonlinear systems through numerically solving an HJ variational inequality [20], as long as the system dimensionality is low [3,6,12]. Viewed from the perspective of STL, reachability methods, including HJ reachability, are concerned with just the reach, avoid, or reach-avoid operators. All of these operators can be expressed as simple STL formulas. Crucially, the notion of recursively constructing complex specifications from simpler ones and the associated nuances in controller synthesis have not been explored in the context of reachability.

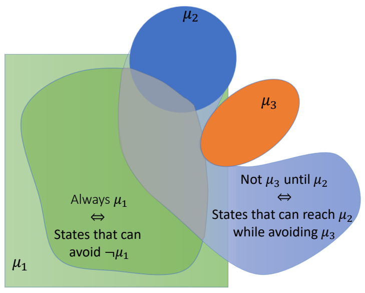

μ_2
μ_3
Not μ_3 until μ_2 $\Leftrightarrow$ States that can reach μ_2 while avoiding μ_3
Always μ_1 $\Leftrightarrow$ States that can avoid $\neg\mu_1$
μ_1

Fig. 1. STL meets HJ reachability and equivalence of temporal and reachability operators. The “Always” operator in STL corresponds to the avoidance operator in HJ reachability, and the “until” operator corresponds to the reach-avoid operator. The gray area in the middle of the illustration indicates that the logical conjunction corresponds to set intersection.

Contributions: In this paper, we connect STL and HJ reachability to take advantage of the features of both methods. From the perspective of STL, our proposed method moves beyond current controller synthesis methods, and provides the set of states, represented by value functions, from which any STL formula can be satisfied. From the perspective of HJ reachability, our method looks beyond single reachability problems, and takes advantage of the expressiveness of STL to define sequences of reachability problems; this is enabled by the least-restrictive feasible controller set (LRFCS), which is the set of controllers that guarantees the satisfaction of an STL formula, and the key concept that both provides a new controller synthesis method for STL formulas, and allows reachability to take advantage of the richness of specification description in STL. We also interpret value functions in the context of minimum violation, a particularly useful concept in many real-world scenarios such as autonomous driving [23]. We demonstrate our method in numerical simulations and robotic experiments in representative autonomous driving scenarios. A simple illustrative summary of the main features of our framework is shown in Fig. 1. Collectively, the results of this paper provide a single framework that quantifies the set of states from which STL formulas can be satisfied and breaks down controller synthesis of complex STL formulas into a sequence of reachability computations in using existing numerical tools.

Organization: In Sec. 2, we briefly introduce notation, background material on STL and HJ reachability, and our problem statement. In Sec. 3, we establish the equivalence among STL formulas, value functions, and the reachability operators. In particular, through reachability, we quantify set of states from which an STL formula can be satisfied. In Sec. 4, we discuss controller synthesis, with an emphasis on avoiding control conflicts in complex STL formulas. Sec. 5 provides a summary of the results of this paper, and allows the reader to compute the set of states satisfying any STL formula, as well as feedback controller synthesis. We present our numerical and robotic experiments on two representative autonomous driving scenarios in Sec. 6. Finally, we conclude and suggest future work in Sec. 7

2 Preliminaries

2.1 System Dynamics and Trajectories

Consider a dynamical system with an ordinary differential equation (ODE) model:

$$\frac{dx}{ds} := \dot{x} = f(s, x, u, d), \quad u \in \mathcal{U}, d \in \mathcal{D}, \tag{1}$$

where $x \in \mathcal{X}$ is the system state, u the control restricted to a compact set $\mathcal{U}$, and d the disturbance restricted to a compact set $\mathcal{D}$. We assume the control function $u(\cdot)$ is measurable, and if for all t, $u(t) \in \mathcal{U}$, we also write $u(\cdot) \in \mathbb{U}$, where $\mathbb{U}$ is the function space containing all admissible controls. In the analogous fashion, we also write $d(\cdot) \in \mathbb{D}$.

The system dynamics f are assumed to be uniformly continuous, bounded, and Lipschitz continuous in x for fixed u and d; given a measurable control and disturbance functions $u(\cdot)$ and $d(\cdot)$, there exists a unique trajectory solving (1) [7]. The system in (1) can model not only a single robot, but also the joint dynamics of multiple robots. Besides modeling unknown, bounded disturbances, d is also very useful for representing the control action of a different robot, as shown in our experiments in Sec. 6.

We denote trajectories satisfying (1) starting from state x at time t under control $u(\cdot)$ and disturbance $d(\cdot)$ as $\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot)$. The trajectory ξ satisfies (1) with an initial condition almost everywhere: $\dot{\xi}_{x,t}^{u(\cdot),d(\cdot)}(s) = f(s, \xi_{x,t}^{u(\cdot),d(\cdot)}(s), u(s), d(s)), \quad \xi_{x,t}^{u(\cdot),d(\cdot)}(t) = x$.

Remark 1. We make explicit the dependence of the system trajectory $\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot)$ on the control and disturbance so that the presentation of the material in the rest of the paper can be made clearer. Under this slightly more cumbersome notation, the state at any given time s is written as $\xi_{x,t}^{u(\cdot),d(\cdot)}(s)$, instead of the usual $x(s)$. In addition, we use t to denote the initial time, and $s \geq t$ to denote a time instant at or after t.

2.2 Signal Temporal Logic

We focus in this paper on STL, a specification language that expresses requirements directly for real-valued and dense-time signals [18]. STL admits a notion of robustness [8, 10], and recent work has used it for analyzing performance and safety properties of robotic systems. A brief overview of STL is given in this section, mostly following the development in [8]. An introduction to basic concepts can be found in [1].

The syntax of STL formulas is defined recursively by the grammar production

$$\varphi ::= \text{True} \mid \mu(\cdot) \geq 0 \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \mathbf{U}_I \psi, \tag{2}$$

where I is a closed interval of $\mathbb{R}$ of the form $[s_1, s_2]$ or $[s_1, \infty)$ with $0 \leq s_1 < s_2$, and where $\mu : \mathcal{X} \to \mathbb{R}$ is a function that maps states to real values. In this paper, we sometimes assume that the STL formula is in Negation Normal Form, i.e., negation ($\neg$) can only operate on predicates $\mu(\cdot) \geq 0$. Note that this is done without loss of generality because every STL formula has an equivalent formula in Negation Normal Form, which can be constructively obtained by the syntactic translation rules of Def. 11 in [10].

In temporal logics that only have Boolean semantics, e.g., LTL, the syntax has atomic propositions instead of real-valued functions (μ in the grammar above). Thus, to apply LTL formulas to the dynamical system (1), some relation between atomic propositions and states must be defined to decide whether a state trajectory satisfies a formula. In contrast, STL formulas can include predicates $\mu(\cdot) \geq 0$ that directly express the relation between states and Boolean semantics. For example, a convex polytope, defined by a set of linear inequalities of the form $Hx \leq K$, for $H \in \mathbb{R}^{m \times n}$, $K \in \mathbb{R}^m$, can be equivalently encoded by the STL formula $\mu_1(x) \geq 0 \wedge \cdots \wedge \mu_m(x) \geq 0$, where $\mu_i(x) = k_i - h_i x$ for $i = 1, \ldots, m$, and h_i is the i-th row of H.

STL is defined with two kinds of semantics. First, the *Boolean semantics* is defined as follows. Let $\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot)$ be the trajectory of (1) starting from state x at time t under control $u(\cdot)$ and disturbance $d(\cdot)$, as defined in the previous section. Then, $\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot)$ at time s satisfies an STL formula φ according to the inductive definition:

$$(\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \mu(\cdot) \geq 0 \quad \Leftrightarrow \quad \mu(\xi_{x,t}^{u(\cdot),d(\cdot)}(s)) \geq 0, \tag{3a}$$

$$(\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \varphi \wedge \psi \quad \Leftrightarrow \quad (\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \varphi \wedge (\xi_{x_t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \psi, \tag{3b}$$

$$(\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \neg\varphi \quad \Leftrightarrow \quad \neg(\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \varphi), \tag{3c}$$

$$\begin{aligned}(\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \varphi \mathbf{U}_{[s_1,s_2]} \psi \quad \Leftrightarrow \quad & \exists s' \in [s + s_1, s + s_2] \\ & \text{such that } (\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s') \models \psi \\ & \wedge \forall \tau \in [s, s'], (\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), \tau) \models \varphi.\end{aligned} \tag{3d}$$

From the syntax and Boolean semantics, we can derive other commonly used operators, e.g., $\neg\varphi \vee \neg\psi \equiv \neg(\varphi \wedge \psi)$ and, of particular interest:

– the operator "eventually" is denoted $\Diamond_{[s_1,s_2]}\varphi$ and defined by True$\mathbf{U}_{[s_1,s_2]}\varphi$; and
– the operator "always" is denoted $\Box_{[s_1,s_2]}\varphi$ and defined by $\neg\Diamond_{[s_1,s_2]}\neg\varphi$.

Perhaps the most practically useful aspect of STL is its *quantitative semantics*, represented by a function g that is defined inductively as follows:

$$g(\mu(\cdot) \geq 0, s, \xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot)) = \mu(\xi_{x,t}^{u(\cdot),d(\cdot)}(s)), \tag{4a}$$

$$g(\neg\varphi, s, \xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot)) = -g(\varphi, s, \xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot)), \tag{4b}$$

$$g(\varphi \wedge \psi, s, \xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot)) = \min\{g(\varphi, s, \xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot)), g(\psi, s, \xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot))\}, \tag{4c}$$

$$\begin{aligned} g(\varphi\mathbf{U}_{[s_1,s_2]}\psi, s, \xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot)) = \sup_{s' \in [s+s_1, s+s_2]} & \min\{g(\psi, s', \xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot)), \\ & \inf_{\tau \in [s,s']} g(\varphi, s', \xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot))\}. \end{aligned} \tag{4d}$$

As shown in [8], the function g has the property that its sign implies satisfaction, i.e., if $g(\varphi, s, \xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot))$ is positive, then the trajectory satisfies STL formula φ from time s; similarly, it does not satisfy φ if g is negative at time s. Because of its other fundamental property (correctness), the authors of [8] also refer to g as the *robustness estimate*.

As with the Boolean semantics, additional operators like $\Box$ ("always") can be derived for the quantitative semantics [8]. Finally, if $s = 0$, then the notation can be abbreviated by omitting s. In this paper, the notation for the quantitative semantics is made more concise by writing the STL formula (i.e., the first parameter of g) as a subscript; e.g., $g_{\neg\varphi}$ is the robustness estimate for the STL formula $\neg\varphi$. This way, we can concisely refer to multiple STL formulas and their corresponding robustness estimates of a trajectory.

In this paper, we will use $g(\cdot)$ (defined above) to denote functions whose zero *superlevel* sets represent satisfaction of an STL formula, and $h(\cdot)$ to denote functions whose zero *sublevel* sets represent sets used in HJ reachability as in (7) and (9). Also, whenever a function g_φ or its negation satisfies the HJ VI (8) or (11), introduced in Sec. 2.3, we will refer to g_φ as a "**value function associated with the STL formula** φ".

2.3 Time-Varying Reachability

To draw a connection between temporal logic and reachability, one must consider *time-varying* formulations of reachability, to capture the temporal aspects of logical specifications. In general, many (time-invariant) reachability methods can be used for solving time-varying reachability problems by augmenting the state space with time.

In this paper, we focus on HJ reachability [12], the most general reachability method in terms of system dynamics and set representation, and one which incorporates time without state space augmentation. We now provide a brief overview of [12] and adapt it to this paper. It is interesting to note the following parallel between STL and HJ reachability: reachable sets are solutions to a "game of kind" in which one is interested in determining *whether or not* the target set can be reached, and their computation is done by solving a "game of degree" in which one *minimizes a cost function* representing the target sets and constraints; this is analogous to the Boolean semantics and quantitative semantics, respectively, of STL described in Sec. 2.2.

Given target and constraint sets $\mathcal{T}(s), \mathcal{C}(s)$ as a function of time s, the maximal reachable set (RS) $\mathcal{RA}^{\mathrm{M}}(t, T; \mathcal{T}(\cdot), \mathcal{C}(\cdot))$ is defined as follows:

$$\begin{aligned} \mathcal{RA}^{\mathrm{M}}(t, T; \mathcal{T}(\cdot), \mathcal{C}(\cdot)) = \{x : \exists u(\cdot) \in \mathbb{U}, \forall d(\cdot) \in \mathbb{D}, \exists s \in [t, T], \xi_{x,t}^{u(\cdot),d(\cdot)}(s) \in \mathcal{T}(s), \\ \forall \tau \in [t, s], \xi_{x,t}^{u(\cdot),d(\cdot)}(\tau) \in \mathcal{C}(\tau)\} \end{aligned} \tag{5}$$

Informally, $\mathcal{RA}^{\mathrm{M}}(t,T;\mathcal{T}(\cdot),\mathcal{C}(\cdot))$ is the set of states from which there exists a control function $u(\cdot)$ that, despite all possible non-anticipative [19] disturbance functions[3] $d(\cdot)$, drives the system to the target set at some time $s \in [t,T]$ while satisfying constraints prior to reaching the target. In the HJ convention, a set $\mathcal{S}$ is represented by the zero *sublevel* set of some function: $\mathcal{S} = \{x : h_{\mathcal{S}}(x) < 0\}$. Such a function always exists, and can be defined as the signed distance function [12].

Given $h_{\mathcal{T}}(t,x)$ and $h_{\mathcal{C}}(x)$, define

$$h_{\mathcal{RA}^{\mathrm{M}}}(t,x) = \inf_{u(\cdot)\in\mathbb{U}} \sup_{d(\cdot)\in\mathbb{D}} \min_{s\in[t,T]} \max\Big\{h_{\mathcal{T}}(t,\xi_{x,t}^{u(\cdot),d(\cdot)}(s)), \max_{\tau\in[t,s]} h_{\mathcal{C}}(\xi_{x,t}^{u(\cdot),d(\cdot)}(\tau))\Big\}, \tag{6}$$

and we have that the zero sublevel set of $h_{\mathcal{RA}^{\mathrm{M}}}(t,x)$ represents the maximal RS:

$$\mathcal{RA}^{\mathrm{M}}(t,T;\mathcal{T}(\cdot),\mathcal{C}(\cdot)) = \{x : h_{\mathcal{RA}^{\mathrm{M}}}(t,x) < 0\}. \tag{7}$$

The value function $h_{\mathcal{RA}^{\mathrm{M}}}(t,x), t \in [0,T]$, can be computed by solving the following HJ VI, which is derived from dynamic programming:

$$\begin{aligned}\max\Big\{\min\big\{\frac{\partial}{\partial t} h_{\mathcal{RA}^{\mathrm{M}}}(t,x) + \min_{u\in\mathcal{U}}\max_{d\in\mathcal{D}} \nabla h_{\mathcal{RA}^{\mathrm{M}}}(t,x)\cdot f(t,x,u,d),\ h_{\mathcal{T}}(t,x) - h_{\mathcal{RA}^{\mathrm{M}}}(t,x)\big\},\\ h_{\mathcal{C}}(x) - h_{\mathcal{RA}^{\mathrm{M}}}(t,x)\Big\} = 0, \qquad h_{\mathcal{RA}^{\mathrm{M}}}(T,x) = \max\{h_{\mathcal{T}}(T,x), h_{\mathcal{C}}(x)\}.\end{aligned} \tag{8}$$

In addition, we also define the minimal RS as

$$\mathcal{R}^{\mathrm{m}}(t,T;\mathcal{T}(\cdot)) = \{x : \forall u(\cdot)\in\mathbb{U}, \exists d(\cdot)\in\mathbb{D}, \exists s\in[t,T], \xi_{x,t}^{u(\cdot),d(\cdot)}(s) \in \mathcal{T}(s)\}. \tag{9}$$

Intuitively, the minimal RS $\mathcal{R}^{\mathrm{m}}(t,T;\mathcal{T}(\cdot))$ is the set of states from which no matter what control function $u(\cdot)$ is applied, there exists a disturbance function $d(\cdot)$ that drives the system into the target set within some time horizon. Given $h_{\mathcal{T}}(t,x)$, define

$$h_{\mathcal{R}^{\mathrm{m}}}(t,x) = \sup_{u(\cdot)\in\mathbb{U}} \inf_{d(\cdot)\in\mathbb{D}} \min_{s\in[t,T]} h_{\mathcal{T}}(t,\xi_{x,t}^{u(\cdot),d(\cdot)}(s)), \tag{10}$$

and we have that the zero sublevel set of the solution $h_{\mathcal{R}^{\mathrm{m}}}(t,x)$ represents the minimal RS. The value function $h_{\mathcal{R}^{\mathrm{m}}}(t,x)$ can be computed by solving the following HJ VI:

$$\begin{aligned}\min\Big\{\frac{\partial}{\partial t} h_{\mathcal{R}^{\mathrm{m}}}(t,x) + \max_{u\in\mathcal{U}}\min_{d\in\mathcal{D}} \nabla h_{\mathcal{R}^{\mathrm{m}}}(t,x)\cdot f(x,u,d), h_{\mathcal{T}}(x) - h_{\mathcal{R}^{\mathrm{m}}}(t,x)\Big\},\ t\in[0,T],\\ h_{\mathcal{R}^{\mathrm{m}}}(T,x) = h_{\mathcal{T}}(T,x).\end{aligned} \tag{11}$$

The main differences between (8) and (11) are a lack of the outer maximum in (11) due to a lack of constraints in the minimal RS, and reversed optimization over u and d.

2.4 Problem Formulation

The goal of this paper is to achieve the following objectives within a *single* framework:

[3] With a slight abuse of notation, we will use $\mathbb{D}$ to denote non-anticipative disturbance functions. For the definition of non-anticipative strategies, we encourage the reader to refer to [19].

1. Establish a correspondence between STL operators and reachability operators. This is done by recognizing that logical operators in STL are equivalent to elementary set operations, and that STL temporal operators are equivalent to reachability operators.
2. Leverage HJ reachability in the context of STL to compute value functions that represent the set of states from which any STL formula can be satisfied.
3. Leverage the formalism of STL, along with our proposed notion of the LRFCS to perform controller synthesis through a sequence of reachability computations without introducing any controller conflicts.
4. Recognize the minimum violation interpretation of value functions.

We demonstrate our framework through simulations and experiments that are representative of autonomous driving scenarios.

3 STL Specifications in the HJI context

In this section, we go through the STL semantics and present how HJ reachability can be used to compute value functions that represent the set of states from which STL formulas can be satisfied. Sec. 3.1 describes the connection between logical operators and elementary set operations and provides a correspondence between function representations of specifications in the STL convention and of sets in HJ reachability. Sec. 3.2 describes the connection between temporal operators in STL – the until and always operators – and the maximal and minimal reachability operators in HJ reachability.

3.1 STL logical and elementary set operations, and functional representations

First given an atomic proposition μ, we define a set $\mathcal{S}_\mu := \{x : g_\mu(x) > 0\}$. In addition, given an STL formula, for example one denoted φ or ψ, we define $\mathcal{S}_\varphi$ and $\mathcal{S}_\psi$ to denote the set of states that satisfy φ or ψ, respectively.

$$(\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \mu \;\Leftrightarrow\; \xi_{x,t}^{u(\cdot),d(\cdot)}(s) \in \mathcal{S}_\mu \tag{12a}$$

$$(\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \varphi \;\Leftrightarrow\; \xi_{x,t}^{u(\cdot),d(\cdot)}(s) \in \mathcal{S}_\varphi \tag{12b}$$

$$(\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \psi \;\Leftrightarrow\; \xi_{x,t}^{u(\cdot),d(\cdot)}(s) \in \mathcal{S}_\psi \tag{12c}$$

Then, we have the following correspondence in terms of functions that represent satisfaction of STL formulas and functions that represent sets used in HJ reachability.

$$g_\mu(x) > 0 \Leftrightarrow\; h_{\mathcal{S}_\mu}(x) < 0,\; g_\varphi(x) > 0 \Leftrightarrow\; h_{\mathcal{S}_\varphi}(x) < 0,\; g_\psi(x) > 0 \Leftrightarrow\; h_{\mathcal{S}_\psi}(x) < 0, \tag{13}$$

Based on the set definitions above, the logical conjunction, disjunction, and negation are equivalent to set intersection, union, and complement, respectively:

$$(\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \varphi \wedge \psi \;\Leftrightarrow\; \xi_{x,t}^{u(\cdot),d(\cdot)}(s) \in \mathcal{S}_\varphi \cap \mathcal{S}_\psi \tag{14a}$$

$$(\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \varphi \vee \psi \;\Leftrightarrow\; \xi_{x,t}^{u(\cdot),d(\cdot)}(s) \in \mathcal{S}_\varphi \cup \mathcal{S}_\psi \tag{14b}$$

$$(\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \neg\varphi \;\Leftrightarrow\; \xi_{x,t}^{u(\cdot),d(\cdot)}(s) \in \mathcal{S}_\varphi^\complement \tag{14c}$$

In terms of the functional representation of formulas and sets, we have

$$g_{\varphi\wedge\psi}(x) = \min\left(g_\varphi(x), g_\psi(x)\right), \quad h_{\varphi\wedge\psi}(x) = \max\left(h_{\mathcal{S}_\varphi}(x), h_{\mathcal{S}_\psi}(x)\right) \tag{15a}$$

$$g_{\varphi\vee\psi}(x) = \max\left(g_\varphi(x), g_\psi(x)\right), \quad h_{\varphi\vee\psi}(x) = \min\left(h_{\mathcal{S}_\varphi}(x), h_{\mathcal{S}_\psi}(x)\right) \tag{15b}$$

$$g_{\neg\varphi}(x) = -g_\varphi(x), \quad h_{\mathcal{S}_\varphi^\complement}(x) = -h_{\mathcal{S}_\varphi}(x) \tag{15c}$$

3.2 "Until" and "always" as reachability operators

We now look at the important connections between the until and always operators and reachability. The until operator, defined in Equation (3d), can be interpreted as a constrained reachability operator. For example, in $\varphi \mathbf{U}_I \psi$, one is interested in reaching states that satisfy ψ within some time horizon, while satisfying the constraints φ. We now formally state this reachability interpretation.

Proposition 1 (The until operator and constrained reachability).
Define $T = t + s_2$ and the sets $\mathcal{T}_\psi(\cdot)$ and C_φ:

$$\mathcal{T}_\psi(s) = \begin{cases} \{x : (x(\cdot), s) \models \psi\}, & \text{if } s \in [t + s_1, t + s_2], \\ \emptyset, & \text{otherwise}, \end{cases} \qquad C_\varphi = \{x : (x(\cdot), s) \models \varphi\}. \tag{16}$$

Then, we have

$$\exists u(\cdot) \in \mathbb{U},\ \forall d(\cdot) \in \mathbb{D},\ (\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), t) \models \varphi \mathbf{U}_{[s_1,s_2]} \psi \ \Leftrightarrow\ x \in \mathcal{RA}^M(t, T; \mathcal{T}_\psi(\cdot), C_\varphi). \tag{17}$$

In addition, define $g_{\varphi \mathbf{U}_{[s_1,s_2]}\psi}(t,x) = -h_{\mathcal{RA}^M}(t,x)$, where $h_{\mathcal{RA}^M}(t,x)$ is such that $\mathcal{RA}^M(t, t+\tau_2; \mathcal{T}_\psi(\cdot), C_\varphi) = \{x : h_{\mathcal{RA}^M}(t,x) < 0\}$. Then, we have

$$\exists u(\cdot) \in \mathbb{U},\ \forall d(\cdot) \in \mathbb{D},\ (\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), t) \models \varphi \mathbf{U}_{[s_1,s_2]} \psi \ \Leftrightarrow\ g_{\varphi \mathbf{U}_{[s_1,s_2]}\psi}(t,x) > 0. \tag{18}$$

Note that $-g_{\varphi \mathbf{U}_{[s_1,s_2]}\psi}(t,x)$ satisfies the HJ VI (8), so $g_{\varphi \mathbf{U}_{[s_1,s_2]}\psi}$ is a value function associated with $\varphi \mathbf{U}_{[s_1,s_2]}\psi$.

Prop. 1 follows from the definition of the until operator and the maximal RS, and establishes an equivalence between the until operator and the maximal reachability operator. It also provides a single function $g_{\varphi \mathbf{U}_{[s_1,s_2]}\psi}(s,x)$ that captures the set of states from which *there exists* a controller to guarantee the satisfaction of $\varphi \mathbf{U}_{[s_1,s_2]}\psi$ regardless of disturbances. Controller synthesis will be discussed in Sec. 4 in Lem. 1.

Note that the eventually operator corresponds to an unconstrained reachability problem, which is the one presented in Prop. 1 with $\varphi = \text{True}$, or equivalently, $C_\varphi = \emptyset$.

The always operator can be indirectly interpreted as an unconstrained reachability operator. For example, in $\Box_{[s_1,s_2]}\varphi$, one is interested in staying in states that satisfy φ. In the language of reachability, one would equivalently stay out of states that may lead to a violation of φ. We now formally state this reachability interpretation.

Proposition 2 (Reachability interpretation of the always operator).
Define $T = s + s_2$, and

$$\mathcal{T}_\varphi(s) = \begin{cases} \{x : (x(\cdot), s) \not\models \varphi\}, & \text{if } s \in [s + s_1, s + s_2], \\ \emptyset, & \text{otherwise}. \end{cases} \tag{19}$$

Then, we have

$$\exists u(\cdot) \in \mathbb{U},\ \forall d(\cdot) \in \mathbb{D},\ (\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \Box_{[s_1,s_2]}\varphi \ \Leftrightarrow\ x \notin \mathcal{R}^m(s, T; \mathcal{T}_\varphi(\cdot)). \tag{20}$$

In addition, define $g_{\Box_{[s_1,s_2]}\varphi}(s,x) = h_{\mathcal{R}^m}(s,x)$, where $h_{\mathcal{R}^m}(s,x)$ is such that $\mathcal{R}^m(s, T; \mathcal{T}_\varphi(\cdot)) = \{x : h_{\mathcal{R}^m}(s,x) < 0\}$. Then, we have

$$\exists u(\cdot) \in \mathbb{U},\ \forall d(\cdot) \in \mathbb{D},\ (\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \Box_{[s_1,s_2]}\varphi \ \Leftrightarrow\ g_{\Box_{[s_1,s_2]}\varphi}(s,x) > 0. \tag{21}$$

Note that $g_{\Box_{[s_1,s_2]}\varphi}(s,x)$ satisfies the HJ VI (8), so $g_{\Box_{[s_1,s_2]}\varphi}$ is a value function associated with $\Box_{[s_1,s_2]}$.

Prop. 2 follows from the definition of the always operator and minimal RS. Controller synthesis will be discussed in Sec. 4 in Lem. 1.

4 Controller Synthesis

In this section, we provide a controller synthesis technique that is different from the optimal controller synthesis typically found in works related to HJ reachability. Instead of computing the *optimal* control, we propose to consider the set of controllers that satisfy a given STL formula by viewing the value function as a Lyapunov-like function. We call this set of controllers the **least-restrictive feasible controller set (LRFCS)**, which ensures the value function does not decrease along trajectories, an idea that is core to many reachability methods other than HJ [15, 16].

In the context of STL, the key benefit of considering the LRFCS is that a single controller synthesis procedure can be repeatedly used for satisfying recursively defined STL formulas. Thus, the LRFCS is what allows HJ reachability to leverage the complexity of specifications in the STL framework. The following lemma formalizes the LRFCS:

Lemma 1 (Least-restrictive feasible controller set (LRFCS) for satisfying an STL formula). *Let φ be any STL formula, and g_φ be a value function associated with φ. Suppose at time s, the system* (1) *is at a state x such that $g_\varphi(t,x) \geq c$. Define*

$$\mathbb{U}_\varphi := \{u(\cdot) : \forall s \geq t, u(s) \in \tilde{\mathcal{U}}_\varphi(s,x)\}, \text{ where} \tag{22a}$$

$$\tilde{\mathcal{U}}_\varphi(s,x) = \begin{cases} \mathcal{U}_\varphi(s,x), & \text{if } g_\varphi(s,x) \leq c, \\ \mathcal{U}, & \text{otherwise,} \end{cases} \tag{22b}$$

$$\mathcal{U}_\varphi(s,x) = \{u : \frac{\partial}{\partial s} g_\varphi(s,x) + \min_{d \in \mathcal{D}} \nabla g_\varphi(s,x) \cdot f(s,x,u,d) \geq 0\}. \tag{22c}$$

Then, $u(\cdot) \in \mathbb{U}_\varphi$ implies $\forall s \geq t, \forall d(\cdot) \in \mathbb{D}, g_\varphi(s, \xi_{x,t}^{u(\cdot),d(\cdot)}(s)) \geq c$.

Proof: We start with the expression in (22c):

$$0 \leq \frac{\partial}{\partial s} g_\varphi(s,x) + \min_{d \in \mathcal{D}} \nabla g_\varphi(s,x) \cdot f(s,x,u,d) \tag{23a}$$

$$= \min_{d \in \mathcal{D}} \frac{\partial}{\partial s} g_\varphi(s,x) + \nabla g_\varphi(s,x) \cdot f(s,x,u,d) = \min_{d \in \mathcal{D}} \dot{g}_\varphi(s,x) \tag{23b}$$

The minimization of d corresponds to the disturbance behaving adversarially to drive the system away from the satisfaction of φ. Therefore, for all d, we have $\dot{g}_\varphi(s,x) \geq 0$. In addition, since for all $s \geq t$, we have $u(s) \in \mathcal{U}_\varphi(s)$ whenever $g_\varphi(s,x) \leq c$ and in particular when $g_\varphi(s,x) = c$, we must have that $\forall s \geq t, \forall d(\cdot), g_\varphi(s, \xi_{x,t}^{u(\cdot),d(\cdot)}(s)) \geq c$. ■

Corollary 1. *If $c > 0$, then Lem. 1 is equivalent to*

$$u(\cdot) \in \mathbb{U}_\varphi \Rightarrow \forall d(\cdot), (\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \varphi. \tag{24}$$

Remark 2. Lem. 1 provides a set of controllers for satisfying $\varphi \mathbf{U}_{[s_1,s_2]} \psi$ using the value function $g_{\varphi \mathbf{U}_{[s_1,s_2]} \psi}$ in Prop. 1 and for $\Box_{[s_1,s_2]} \varphi$ using $g_{\Box_{[s_1,s_2]} \varphi}$ in Prop. 2.

Note that since g_φ is a value function associated with φ, the expression in (22c) is always non-empty. This is true by construction of the value function in either (8) or (11), in which the optimal controller is selected in a dynamic programming framework. The associated *optimal* controller, which we have omitted in this paper, guarantees the existence of a *feasible* controller; for a more thorough discussion of the optimal controller, please see one of [2, 6, 12].

We now continue our discussion of repeated controller synthesis via the LRFCS for recursively defined STL formulas. We first considering cases which do not involve control conflicts in Sec. 4.1; here, we provide a simple additional control logic to tie together multiple controllers from STL formulas that make up a more complex STL formula. Next, in 4.2, we address the cases involving control conflicts by describing why control conflicts arise, and how they can be resolved.

4.1 Negation, logical disjunction, and "always": no control conflicts

In this section we consider $\neg$, $\vee$, and $\Box_{[s_1,s_2]}$, for which the discussion of controller synthesis is relatively straight forward. We first note that to synthesize a controller that guarantees the satisfaction of $\neg\varphi$, one would simply first negate φ, and then perform controller synthesis recursively. This can be considered a "pre-processing step" when STL formulas are in Negation Normal Form (cf. Sec. 2.2).

Next, we provide a joint controller for satisfying $\varphi \vee \psi$, given two controllers that respectively guarantee the satisfaction of φ and ψ. Intuitively, Prop. 3 first states that applying a controller (in $\mathbb{U}_\varphi$) that satisfies φ also implies satisfaction of $\varphi \vee \psi$. On the other hand, if φ cannot be satisfied, then by assumption ψ can be satisfied using a controller drawn from $\mathbb{U}_\psi$, thereby also satisfying $\varphi \vee \psi$.

Proposition 3 (Joint controller for logical disjunction). *Suppose* (24) *holds, and* $u(\cdot) \in \mathbb{U}_\psi \Rightarrow \forall d(\cdot), (\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \psi$. *Then* $(\xi_{x,t}^{u_{\varphi\vee\psi}(\cdot),d(\cdot)}(\cdot), s) \models \varphi \vee \psi$, *where*

$$u_{\varphi\vee\psi}(\cdot) \in \begin{cases} \mathbb{U}_\varphi, & \textit{if } (24) \textit{ holds with } \varphi = \varphi, \\ \mathbb{U}_\psi, & \textit{otherwise}. \end{cases} \tag{25}$$

Finally, given a set of controllers $\mathbb{U}_\varphi$ that guarantees satisfaction of φ, and a set of controllers $\mathbb{U}_{\Box_{[s_1,s_2]}\varphi}$ that guarantees satisfaction of $\Box_{[s_1,s_2]}\varphi$, we provide the set of controllers that first drives the system into a position in which φ can be satisfied using a controller in $\mathbb{U}_{\Box_{[s_1,s_2]}\varphi}$, and then drives the system to satisfy φ using a controller in $\mathbb{U}_\varphi$.

Proposition 4 (Compound controller for the always operator). *Define some* $s' \in [s + s_1, s + s_2]$. *Suppose* (24) *holds, and* $u(\cdot) \in \mathbb{U}_{\Box_{[s_1,s_2]}\varphi} \Rightarrow \forall d(\cdot), (\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), s) \models \Box_{[s_1,s_2]}\varphi$. *Then,* $(\xi_{x,t}^{\bar{u}(\cdot),d(\cdot)}(\cdot), s) \models \varphi$, *where* $\bar{u}(\cdot) \in \begin{cases} \mathbb{U}_{\Box_{[s_1,s_2]}\varphi}, & \textit{if } s < s', \\ \mathbb{U}_\varphi, & \textit{if } s \geq s'. \end{cases}$

The reasoning behind Prop. 4 is as follows: If one applies the controller drawn from $\mathbb{U}_{\Box_{[s_1,s_2]}\varphi}$, until some time s' between $s + s_1$ and $s + s_2$, the system is then *in a position* to satisfy φ. The satisfaction of φ is *guaranteed* by applying a controller drawn from $\mathbb{U}_\varphi$ at and after time s'.

4.2 Logical conjunction and "until": avoiding control conflicts

In this section, we consider the operators $\wedge$ and $\mathbf{U}$, which require more careful treatment due to the possibility of control conflicts. For example, even if controllers drawn from $\mathbb{U}_\varphi$ and $\mathbb{U}_\psi$ allow the system to *independently* satisfy φ and ψ respectively, there may not exist a controller that leads to the satisfaction of $\varphi \wedge \psi$. This is because the system may only use a single control at any given time, and if $\mathbb{U}_\varphi$ and $\mathbb{U}_\psi$ do not intersect, then there would not be a controller that guarantees *simultaneous* satisfaction of φ and ψ. An analogous argument also applies for $\varphi\mathbf{U}_{[s_1,s_2]}\psi$.

Therefore, controller synthesis for satisfaction of ψ in the expressions $\varphi \wedge \psi$ or $\varphi\mathbf{U}_{[s_1,s_2]}\psi$ must use the LRFCS with respect to φ so the satisfaction of φ is guaranteed. This requirement can be formalized by restating Prop. 1 and 2 with the modified control constraint $u(\cdot) \in \mathbb{U}_\varphi$ instead of $u(\cdot) \in \mathbb{U}$. The full restatements can be found in the appendix. Here, we highlight that optimizing over a restricted function set $u(\cdot) \in \mathbb{U}_\varphi$, or equivalently, over a restricted control signal set $\tilde{\mathcal{U}}_\varphi$, does not significantly increase the computation burden for control and disturbance affine systems. To see this, let the dynamics (1) be $\dot{x} = f_x(t,x) + f_u(t,x)u + f_d(t,x)d$. Then, expression in (22c) becomes

$$\frac{\partial}{\partial s} g_\varphi(s,x) + \nabla g_\varphi(s,x) \cdot f_x(t,x) + \min_{d\in\mathcal{D}} \nabla g_\varphi(s,x) f_d(t,x) d + \nabla g_\varphi(s,x) f_u(t,x) u \geq 0, \quad (26)$$

which is a single affine control constraint. When solving the HJ VIs (8) and (11), the optimization $u \in \mathcal{U}$ becomes $u \in \tilde{\mathcal{U}}_\varphi$, which, according to (26), at most involves adding an affine constraint. In typical scenarios in which $u \in \mathcal{U}$ is a box constraint, adding (26) would result in a polytopic constraint, and therefore the optimization $u \in \tilde{\mathcal{U}}_\varphi$ does not involve significantly more computation than the optimization $u \in \mathcal{U}$.

5 Practical Summary of Theoretical Results

In this section, we compile all the theory in this paper into a list of corresponding STL and reachability operators. Given this list, one would be able to recursively compute the set of states that satisfies an arbitrary STL formula, and to synthesize the corresponding feedback controller. We also provide an illustrative example and brief discussion on minimum violation.

5.1 List of corresponding STL, set, and reachability operators

Using Table 1, a value function representing any STL formula φ can be computed through a sequence of reachability computations as well as point-wise negation, minimization, and maximization of functions. The number of reachability computations required is the number of until and always operators in the STL formula φ, and each individual reachability computation scales according to the reachability method. In the case of the HJ method, computational complexity of a single reachability computation is exponential with the number of system dimensions in (1). Practically speaking, this means that systems with up to 5D can be tractably analyzed.

In theory, any controller in the LRFCS can be used to satisfy g_φ, as long as the state $x(s)$ is such that $g(s,x) > 0$. In practice, additional criteria can be used for choosing the controller within the LRFCS. One simple choice, which is typically used in HJ reachability, is to choose the controller that maximizes $\dot{g}_\varphi$ in Eq. (22c).

5.2 Illustrative Example

As a concrete example, consider the formula $\varphi = \Box_{I_1} \Diamond_{I_2} \mu_1 \wedge (\mu_2 \mathbf{U}_{I_3} \mu_3)$, for functions μ_1, μ_2, μ_3 representing atomic propositions, and time intervals I_1, I_2, I_3. Using Table 1, one could perform the following steps, which involve three reachability computations in total, to obtain a value function representing the set of states from which φ can be satisfied, and the corresponding LRFCS:

Formula	Value function computation	Controller	Details
$\varphi \vee \psi$	Set $g_{\varphi\wedge\psi} = \max\left(g_\varphi, g_\psi\right)$	$u(\cdot) \in \mathbb{U}_{\varphi\vee\psi}$ in (25)	Eq. (15b), Prop. 3
$\varphi \wedge \psi$	Given $\mathbb{U}_\varphi$, 1. compute g_ψ with constraint $u(\cdot) \in \mathbb{U}_\varphi$, 2. set $g_{\varphi\wedge\psi} = \min\left(g_\varphi, g_\psi\right)$.	$u(\cdot) \in \mathbb{U}_\varphi$ in (22) with $\varphi = \psi, g_\varphi = g_\psi, c \geq 0$	Eq. (15a), Sec.4.2
$\neg\varphi$	Set $g_{\neg\varphi}(x) = -g_\varphi(x)$	Synthesized recursively from negated formula	Eq. (15c), Sec. 4
$\varphi\mathbf{U}_{[s_1,s_2]}\psi$	Given $\mathbb{U}_\varphi, g_\psi$, 1. solve the HJ VI (8) with $\mathcal{T}_\psi, \mathcal{C}_\varphi$ given in (16) and constraint $u(\cdot) \in \mathbb{U}_\varphi$ to obtain the RS represented by $h_{\mathcal{R}\mathcal{A}\mathrm{M}}$, 2. set $g_{\varphi\mathbf{U}_{[s_1,s_2]}\psi} = -h_{\mathcal{R}\mathcal{A}\mathrm{M}}$	$u(\cdot) \in \mathbb{U}_\varphi$ in (22) with $\varphi = \varphi\mathbf{U}_{[s_1,s_2]}\psi, g_\varphi = g_{\varphi\mathbf{U}_{[s_1,s_2]}\psi}, c \geq 0$	Prop. 1, Sec. 4.2
$\square_{[s_1,s_2]}\varphi$	1. solve the HJ VI (11) with $\mathcal{T}_\varphi$ given in (19) and constraint $u(\cdot) \in \mathbb{U}_\varphi$ to obtain the RS represented by $h_{\mathcal{R}\mathrm{m}}$ 2. $g_{\varphi\mathbf{U}_{[s_1,s_2]}\psi} = h_{\mathcal{R}\mathrm{m}}$	– Use $u(\cdot) \in \mathbb{U}_\varphi$ in (22) with $\varphi = \square_{[s_1,s_2]}\varphi, g_\varphi = g_{\square_{[s_1,s_2]}\varphi}, c \geq 0$ to satisfy $\square_{[s_1,s_2]}\varphi$ – Use $u(\cdot) \in \mathbb{U}_\varphi$ when $s \in [s + s_1, s + s_2]$ to satisfy φ	Prop. 2, Sec. 4.1

Table 1. Summary of theoretical results. Given any STL formula φ, this table can be used to identify the sequence of reachability computations, and point-wise negation, minimization, and maximization of functions that produces a value function g_φ and the corresponding LRFCS $\mathbb{U}_\varphi$.

1. Compute $g_{\diamond_{I_2}\mu_1}$ with the control constraint $u(\cdot) \in \mathbb{U}$ (no control constraints other than that given by the dynamical system) according to the fifth row of Table 1.
2. Compute $g_{\square_{I_1}\diamond_{I_2}\mu_1}$ and $\mathbb{U}_{\square_{I_1}\diamond_{I_2}\mu_1}$ with the control constraint $u(\cdot) \in \mathbb{U}$ (no control constraints other than that given by the dynamical system) according to the sixth row of Table 1.
3. Compute $g_{\mu_2\mathbf{U}I_3\mu_3}$ and $\mathbb{U}_{\mu_2\mathbf{U}I_3\mu_3}$ with the control constraint $u(\cdot) \in \mathbb{U}_{\square_{I_1}\diamond_{I_2}\mu_1}$ (extra control constraint to ensure satisfaction of $\square_{I_1}\diamond_{I_2}\mu_1$) according to the fifth row of Table 1. Note that this is also step 1 in the third row of Table 1).
4. Set $g_\varphi = \min(g_{\square_{I_1}\diamond_{I_2}\mu_1}, g_{\mu_2\mathbf{U}I_3\mu_3})$. The set of states from which φ can be satisfied is then given by $\{x : g_\varphi > 0\}$ (step 2 in the third row of Table 1.
5. The LRFCS for satisfying φ is given by $\mathbb{U}_{\mu_2\mathbf{U}I_3\mu_3}$, (which already accounts for the satisfaction of $\square_{I_1}\diamond_{I_2}\mu_1$).

5.3 Minimum violation interpretation

Lem. 1 and Cor. 1 establish the condition to guarantee the continued satisfaction of an STL formula φ using the LRFCS in (22), as long as the formula is satisfied at some state x and time s. However, in some situations, the system may be in a situation in which a desired STL formula φ cannot be satisfied. Mathematically, this is characterized by $g_\varphi(s, x) \leq 0$. Such a situation could occur if the controller in (22) is not used, or if there is a sudden change in the STL formula the system needs to satisfy, perhaps as a result of a sudden change in the objective of the system.

Since Lem. 1 holds for any value of c, the controller in (22) is also the "minimum violation controller" in the situation in which $g_\varphi(s, x) \leq 0$. By using the controller in (22), one can guarantee that $g_\varphi(s, x)$ never decreases, which is interpreted as "the situation is guaranteed to not become worse, even under the worst-case disturbance". On the other hand, if the disturbance does not behave in worst-case fashion, $\dot{g}_\varphi(s, x)$ may be positive, and $g_\varphi(s, x)$ may evolve to become positive so that φ is satisfied at a later time. An example of a system "recovering" from a state that initially does not satisfy a desired STL formula is shown in the appendix.

6 Simulations and Experiments

In this section, we demonstrate our methodology on an example representative of a common autonomous driving scenario. We will consider a car overtaking maneuver in a 2-lane highway. As this example involves the logical conjunction of compound STL formulas, we will use the notion of LRFCS in Sec. 4.2 to avoid control conflicts.

In addition to this example, a toy numerical example, and a different numerical example and robotic experiment related to autonomous driving are presented in the appendix. The toy numerical example validates the notion of LRFCS. In the example involving autonomous driving in the appendix, we considered an autonomous car attempting to make a left turn in a four-way intersection just after the traffic light has transitioned to yellow. We performed the control synthesis described in Sec. 5. Here, the desired outcome is either making a successful left turn or stopping in initial lane, depending on the behavior of another car travelling in the opposite direction. We also demonstrated the notion of minimum violation introduced in Sec. 5.3 when the desired outcome is not initially guaranteed. For all examples, after obtaining the LRFCS that guarantees the satisfaction of the desired STL formula, we simply chose the controller that maximizes the g_φ expression in (22c).

6.1 Hardware Setup

We performed all computations[4] on a laptop with a Intel Core i5-6300HQ CPU and 8GB of RAM. Reachability computations which produced the value functions were done using the beacls toolbox[5]. Numerical simulations for both examples were performed in MATLAB using the helperOC[6] and Level Set Methods[7] [20] toolboxes. For the hardware experiments[8], we used TurtleBots in place of cars and tracked their positions using a Vicon motion capture system. With the exception of using MATLAB for the real-time evaluation of the optimal controller, all other processes including message-passing between devices and low-level controls were implemented in ROS using Python.

6.2 Highway example

We now consider a highway overtaking scenario. Initially, the autonomous car is behind a "slow car" which moves at a constant speed v_3. The autonomous car attempts to overtake the slow car while avoiding a collision with an "adjacent car" which travels in the left lane with a possible range of velocities. In addition to avoiding collisions with both cars, the autonomous car will also have to be able to re-enter the right lane within a short duration of time. This additional safety constraint is to plan for situations such as when emergency vehicles require passage through the left lane.

[4] The code used in this paper is available at https://github.com/StanfordASL/stlhj
[5] https://github.com/HJReachability/beacls
[6] https://github.com/HJReachability/helperOC
[7] http://www.cs.ubc.ca/ mitchell/ToolboxLS/
[8] The videos can be viewed at
https://www.youtube.com/playlist?list=PL8-2mtIlFIJoNkhcGI7slWX2W3kEW-9Pb

The adjacent car, which acts as the disturbance as described in Sec. 2.3, has its position represented by y_2. Since the slow car travels at a constant speed, we express the joint dynamics of the three vehicles in the reference frame of the slow car:

$$\dot{x}_R = v_R \cos\theta_R, \quad \dot{y}_R = v_R \sin\theta_R - v_3, \quad \dot{\theta}_R = \omega_R, \quad \dot{v}_R = a_R, \quad \dot{y}_2 = v_2 - v_3. \tag{27}$$

The autonomous car's primary objective is to overtake the slow car, which is captured by the proposition ψ^{pass}. It needs to adhere to the traffic laws and avoid colliding with either the slow car or the adjacent car. This constraint is represented by φ. If such a collision is possible or the overtaking maneuver cannot be achieved within the specified time frame, the autonomous car will then have the secondary objective of remaining behind the slow car, ψ^{stay}. We also include an additional recurrence requirement for the autonomous car to always be within 5 seconds of re-entering the right lane, represented by ψ^{lane}. The function representations of the STL propositions can be found in the appendix. Overall, the robot must satisfy the following STL formula:

$$\Box_{[0,25]}\Diamond_{[0,5]}\psi^{\text{lane}} \wedge \left(\varphi\mathbf{U}_{[0,25]}\psi^{\text{pass}} \vee \varphi\mathbf{U}_{[0,25]}\psi^{\text{stay}}\right), \; \varphi = \varphi^{\text{off-road}} \wedge \varphi^{\text{on-road}} \wedge \varphi^{\text{avoid}} \tag{28}$$

The results from numerical simulations and experiments with TurtleBots are shown in Fig. 2 and Fig. 3. After obtaining the value function and LRFCS for $\Box_{[0,25]}\Diamond_{[0,5]}\psi^{\text{lane}}$, the LRFCS is used as an additional constraint for the reachability computations associated with $\varphi\mathbf{U}_{[0,25]}\psi^{\text{pass}} \vee \varphi\mathbf{U}_{[0,25]}\psi^{\text{stay}}$ to avoid control conflicts as discussed in Sec. 4.2. This process is described in greater detail in the appendix. The white circle represents the autonomous car, while the red and green car represents the adjacent car and slow car respectively. The colors represent the values of $\max(g_{\varphi\mathbf{U}_{[0,25]}\psi^{\text{pass}}}, g_{\varphi\mathbf{U}_{[0,25]}\psi^{\text{stay}}})$. In Fig. 2, the autonomous car has sufficient time to pass the slow car as the adjacent car passes by quickly. It waits until the adjacent car crosses a threshold position, determined automatically from the reachability computations, before committing to the overtaking maneuver. This takes into account the possibility that the adjacent car may slow down drastically at any time. We can observe that the value at the autonomous car's position increases from $t = 0$s to 8s just as the car begins the overtaking maneuver; this is a result of the adjacent car not behaving in the worst-case manner.

In Fig. 3, the autonomous car is unable to pass the slow car within the time horizon and stops behind the slow car due to the adjacent car moving too slowly to reach the time-specific threshold position. The value at the autonomous car's position remains low. Computation of the value function took approximately 8 hours. The erratic movement seen in Fig. 3 is caused by the coarse state discretization coupled with the bang-bang controller; this numerical artifact can potentially be alleviated using a finer resolution grid along with GPU parallelization to offset additional computational burden.

7 Conclusions and Future Work

In this paper, we presented the combination of Signal Temporal Logic (STL) and Hamilton-Jacobi (HJ) reachability as a versatile verification technique that inherits the strengths of both. From the perspective of STL, our method moves beyond controller synthesis methods that generate single trajectories; instead, we provide guarantees in terms of sets of states, and produce state feedback controllers. From the perspective of HJ reachability, our method looks beyond single reachability problems, and takes advantage of the expressiveness of STL to define sequences of reachability problems. In terms of computational complexity, the number of reachability computations required

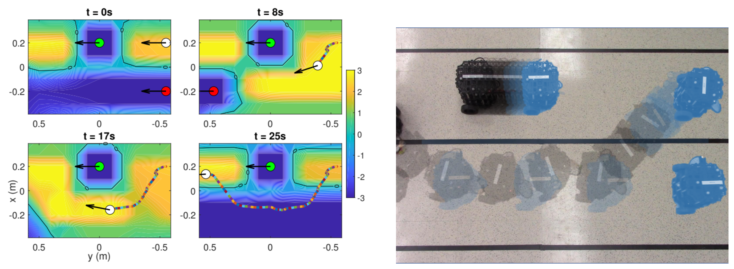


Fig. 2. The autonomous car (white) successfully overtakes the slow car (green) as the adjacent car (red) moves quickly past the threshold position that guarantees the autonomous car can complete the overtaking maneuver within the time horizon. **Left:** Contour plot of the value function. **Right:** Time-lapse of the experiment.

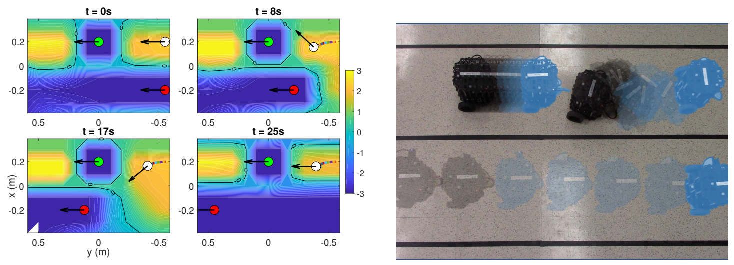


Fig. 3. The autonomous car (white) remains behind the slow car (green) as the adjacent car (red) moves too slowly to reach the threshold position. **Left:** Contour plot of the value function. **Right:** Time-lapse of the experiment.

for set computation and controller synthesis is the same as the number of temporal operators in the STL formula; for any individual reachability computation, our method inherits the same computational complexity as the reachability method. Our approach has been validated in three numerical examples and two robotic experiments.

Our work spawns a number of future research directions. For example, one could attempt to alleviate the effects of bang-bang control by using the LRFCS in conjunction with a different controller synthesis framework such as model predictive control. Also, in general, there may be many different orderings of reachability operators that produce LRFCS for an STL formula; thus, it is important to investigate the effect of different prioritization of STL formulas when using the LRFCS. Lastly, the ideas in this paper, although presented in the context of STL and HJ reachability, are transferrable to other temporal logics and reachability methods; exploring other formulations of temporal logic and reachability would help make trade-offs among computational scalability, conservatism, generality of system dynamics, and expressivness of logical formulas.

References

1. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
2. Bansal, S., Chen, M., Fisac, J.F., Tomlin, C.J.: Safe sequential path planning of multi-vehicle systems under presence of disturbances and imperfect information. In: American Control Conference (2017)
3. Bansal, S., Chen, M., Herbert, S., Tomlin, C.J.: Hamilton-Jacobi reachability: A brief overview and recent advances. In: Proc. IEEE Conf. on Decision and Control (2017)

4. Belta, C., Bicchi, A., Egerstedt, M., Frazzoli, E., Klavins, E., Pappas, G.J.: Symbolic planning and control of robot motion: Finding the missing pieces of current methods and ideas. IEEE Robotics & Automation Magazine pp. 61–70 (March 2007)
5. Chen, M., Hu, Q., Fisac, J., Akametalu, K., Mackin, C., Tomlin, C.: Reachability-based safety and goal satisfaction of unmanned aerial platoons on air highways. AIAA Journal of Guidance, Control, and Dynamics 40(6), 1360 – 1373 (2017)
6. Chen, M., Tomlin, C.J.: Hamilton-Jacobi Reachability: Some Recent Theoretical Advances and Applications in Unmanned Airspace Management. Annual Review of Control, Robotics, and Autonomous Systems 1(1), 333–358 (May 2018)
7. Coddington, E.A., Levinson, N.: Theory of Ordinary Differential Equations. McGraw-Hill (1955)
8. Donzé, A., Ferrère, T., Maler, O.: Efficient robust monitoring for STL. In: Proceedings of Computer Aided Verification (CAV) (2013)
9. Dreossi, T., Dang, T., Piazza, C.: Parallelotope bundles for polynomial reachability. In: Hybrid Systems: Computation and Control (2016)
10. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theoretical Computer Science 410, 4262–4291 (2009)
11. Feng, X., Villanueva, M.E., Chachuat, B., Houska, B.: Branch-and-lift algorithm for obstacle avoidance control. In: Proc. IEEE Conf. on Decision and Control (2017)
12. Fisac, J.F., Chen, M., Tomlin, C.J., Sastry, S.S.: Reach-avoid problems with time-varying dynamics, targets and constraints. In: Hybrid Systems: Computation and Control (2015)
13. Gattami, A., Al Alam, A., Johansson, K.H., Tomlin, C.J.: Establishing safety for heavy duty vehicle platooning: A game theoretical approach. IFAC World Congress 44(1), 3818–3823 (Jan 2011)
14. Kress-Gazit, H., Lahijanian, M., Raman, V.: Synthesis for robots: Guarantees and feedback for robot behavior. Annual Review of Control, Robotics, and Autonomous Systems 1, 211–236 (2018)
15. Landry, B., Chen, M., Hemley, S., Pavone, M.: Reach-avoid problems via sum-of-squares optimization and dynamic programming. In: IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (2018), submitted
16. Majumdar, A., Tedrake, R.: Funnel libraries for real-time robust feedback motion planning (2016), Available at https://arxiv.org/abs/1601.04037
17. Majumdar, A., Vasudevan, R., Tobenkin, M.M., Tedrake, R.: Convex optimization of nonlinear feedback controllers via occupation measures. Int. Journal of Robotics Research 33(9), 1209–1230 (Aug 2014)
18. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: FORMATS/FTRTFT. pp. 152–166 (2004)
19. Mitchell, I.M., Bayen, A.M., Tomlin, C.J.: A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. IEEE Transactions on Automatic Control 50(7), 947–957 (2005)
20. Mitchell, I.M.: The Flexible, Extensible and Efficient Toolbox of Level Set Methods. SIAM Journal on Scientific Computing 35(2-3), 300–329 (Jun 2008)
21. Raman, V., Donze, A., Maasoumy, M., Murray, R.M., Sangiovanni-Vincentelli, A., Seshia, S.A.: Model predictive control with signal temporal logic specifications. In: Proc. IEEE Conf. on Decision and Control (2014)
22. Raman, V., Donzé, A., Sadigh, D., Murray, R.M., Seshia, S.A.: Reactive synthesis from signal temporal logic specifications. In: Hybrid Systems: Computation and Control. pp. 239–248 (2015)
23. Reyes Castro, L.I., Chaudhari, P., Tumova, J., Karaman, S., Frazzoli, E., Rus, D.: Incremental sampling-based algorithm for minimum-violation motion planning. In: Proc. IEEE Conf. on Decision and Control (2013)
24. Singh, S., Majumdar, A., Slotine, J.J.E., Pavone, M.: Robust online motion planning via contraction theory and convex optimization. In: Proc. IEEE Conf. on Robotics and Automation (2017), Extended Version, Available at http://asl.stanford.edu/wp-content/papercite-data/pdf/Singh.Majumdar.Slotine.Pavone.ICRA17.pdf

A Formal presentation of Sec. 4.2

Proposition 5 (Reachability interpretation of the until operator with LRFCS). *Define $T = t + s_2$ and the sets*

$$\mathcal{T}_\psi(s) = \begin{cases} \{x : (x(\cdot), s) \vDash \psi\}, & \text{if } s \in [t+s_1, t+s_2], \\ \emptyset, & \text{otherwise}, \end{cases} \qquad C_\varphi = \{x : (x(\cdot), t) \vDash \varphi\}. \tag{A-1}$$

Furthermore, assume (24) *holds, and define* $\mathbb{U}_\varphi$ *and* $\tilde{\mathcal{U}}_\varphi$ *according to* (22). *Then,*

$$\exists u(\cdot) \in \mathbb{U}_\varphi,\ \forall d(\cdot) \in \mathbb{D},\ (\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), t) \vDash \varphi \mathbf{U}_{[s_1,s_2]} \psi \ \Leftrightarrow\ x \in \mathcal{RA}^M(t, T; \mathcal{T}_\psi(\cdot), C_\varphi). \tag{A-2}$$

In addition, define $g_{\varphi \mathbf{U}_{[s_1,s_2]}\psi}(t,x) = -h_{\mathcal{RA}^M}(t,x)$, *where* $h_{\mathcal{RA}^M}(t,x)$ *is such that* $\mathcal{RA}^M(t, t+s_2; \mathcal{T}_\psi(\cdot), C_\varphi) = \{x : h_{\mathcal{RA}^M}(t,x) < 0\}$. *Then, we have*

$$\exists u(\cdot) \in \mathbb{U}_\varphi,\ \forall d(\cdot) \in \mathbb{D},\ (\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), t) \vDash \varphi \mathbf{U}_{[s_1,s_2]} \psi \ \Leftrightarrow\ g_{\varphi \mathbf{U}_{[s_1,s_2]}\psi}(t,x) > 0. \tag{A-3}$$

Proposition 6 (Reachability interpretation of the always operator with LRFCS). *Define $T = t + s_2$, and*

$$\mathcal{T}_\varphi(s) = \begin{cases} \{x : (x(\cdot), s) \nvDash \varphi\}, & \text{if } s \in [t+s_1, t+s_2] \\ \emptyset, & \text{otherwise} \end{cases} \tag{A-4}$$

Furthermore, assume (24) *holds, and define* $\mathbb{U}_\varphi$ *and* $\tilde{\mathcal{U}}_\varphi$ *according to* (22). *Then,*

$$\exists u(\cdot) \in \mathbb{U}_\varphi,\ \forall d(\cdot) \in \mathbb{D},\ (\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), t) \vDash \Box_{[s_1,s_2]} \varphi \ \Leftrightarrow\ x \notin \mathcal{R}^m(t, T; \mathcal{T}_\varphi(\cdot)) \tag{A-5}$$

In addition, define $g_{\Box_{[s_1,s_2]}\varphi}(t,x) = h_{\mathcal{R}^m}(t,x)$, *where* $h_{\mathcal{R}^m}(t,x)$ *is such that* $\mathcal{R}^m(t, T; \mathcal{T}_\varphi(\cdot)) = \{x : h_{\mathcal{R}^m}(t,x) < 0\}$. *Then, we have*

$$\exists u(\cdot) \in \mathbb{U}_\varphi,\ \forall d(\cdot) \in \mathbb{D},\ (\xi_{x,t}^{u(\cdot),d(\cdot)}(\cdot), t) \vDash \Box_{[s_1,s_2]} \varphi \ \Leftrightarrow\ g_{\Box_{[s_1,s_2]}\varphi}(t,x) > 0. \tag{A-6}$$

B Additional numerical examples and experiments, and implementation details

B.1 Double integrator toy example

The double integrator has state components x representing position and v representing speed, and dynamics given by

$$\dot{x} = v, \quad \dot{v} = u, \qquad |u| \leq 0.5, \tag{A-7}$$

where the acceleration u is the control input, and there is no disturbance.

This simple example is used to validate the use of the LRFCS for satisfying an STL formula, discussed in Lem. 1, to determine the set of states that satisfies $\varphi \wedge \psi$ according to the discussion in Section 4.2.

Here, we chose $\varphi = \Diamond_{[5,5]}\mu_1$, $\psi = \Diamond_{[5,5]}\mu_2$, where $\mu_1 = \{(x, v) : |x| \leq 0.5\}$ and $\mu_2 = \{(x, v) : |v| \leq 0.2\}$. The time horizon consisting of a single time instant is chosen so that the computation result can be compared with the ground-truth optimal solution, since $\Diamond_{[5,5]}\mu_1 \wedge \Diamond_{[5,5]}\mu_2$ and $\Diamond_{[5,5]}(\mu_1 \wedge \mu_2)$ are equivalent, *if the optimal control is used*. If the time horizon contains more than a single time instant, then one cannot factor out the eventually operator, since it is unclear at which exact time each of μ_1 and μ_2 is satisfied, and the conjunction is only satisfied when both μ_1 and μ_2 are satisfied at the same time.

Fig. 1 shows the computation results. On the left and middle plots, the region between the dashed lines represent μ_1 and μ_2 respectively, and the region between solid black curves represent the states that satisfy $\Diamond_{[5,5]}\mu_1$ and $\Diamond_{[5,5]}\mu_2$ respectively. For computing the set of states satisfying $\Diamond_{[5,5]}\mu_1$, the optimal controller, the one that maximizes $\dot{g}_{\mu_1}$, is used, and for $\Diamond_{[5,5]}\mu_2$, the controller that maximizes $\dot{g}_{\mu_1}$ but also satisfies (22) is used. In the right plot, the region inside the dashed blue curve represents $\mu_1 \wedge \mu_2$, and the region inside the solid blue curve represents the states that satisfy $\Diamond_{[5,5]}(\mu_1 \wedge \mu_2)$ under the (joint) optimal control. The region inside the black curve represents the states that satisfy $\Diamond_{[5,5]}\mu_1 \wedge \Diamond_{[5,5]}\mu_2$ when the computation of $\Diamond_{[5,5]}\mu_2$ involves the controller being restricted to feasible controllers for $\Diamond_{[5,5]}\mu_1$.

Here, one can see that the region inside the black curve is contained inside the blue curve, which must be true theoretically as well, since the blue curve is computed with the joint optimal control for satisfying $\Diamond_{[5,5]}(\mu_1 \wedge \mu_2)$.

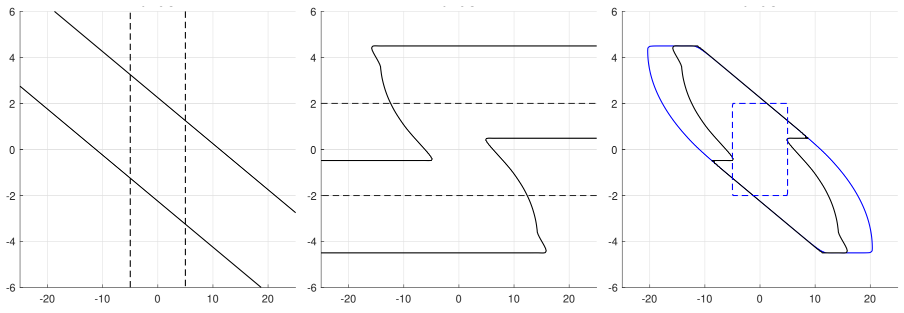


Fig. 4. Validation of least-restrictive controller set.

B.2 Traffic light example

For this example, we consider an autonomous car attempting a left turn at an intersection just as the traffic light turns yellow. At the same time, a second car approaches from the opposing direction. This scenario is depicted in Fig. 5. The temporal elements of STL are useful in encoding the transition of the traffic light to red after remaining in yellow for a fixed duration.

The joint dynamical system representing the two-car system is as follows:

$$\dot{x}_R = v_R \cos\theta_R, \quad \dot{y}_R = v_R \sin\theta_R, \quad \dot{\theta}_R = \omega_R, \quad \dot{v}_R = a_R, \quad \dot{y}_2 = v_2. \tag{A-8}$$

The first four state components $(x_R, y_R, \theta_R, v_R)$, represent the x and y positions, heading, and speed of the autonomous car. It is a dynamically-extended Dubins car that controls its turn rate, ω_R and its linear acceleration, a_R. The last state component, y_2 represents the y position of the second car, which changes according to its initial position and its speed, v_2.

The autonomous car's primary objective is to make a left turn before the light turns red without colliding with the second car. This specification is captured by the STL proposition ψ^{turn}. Alternatively, if it is not possible make the left turn, the autonomous car will simply stop before the intersection. This specification is captured by the STL proposition ψ^{stop}. The autonomous car must also follow traffic rules based on the layout of the road. These are captured by $\varphi^{\text{off-road}}$, which stipulates that the autonomous car must traveling in the proper direction in lane, $\varphi^{\text{on-road}}$, which ensures that the autonomous car stays on the road, and φ^{avoid}, which states that collisions must be avoided with the second car.

The STL proposition formulas are as follows:

$$\psi^{\text{turn}} = (\underline{x}^{\text{turn}} \le x \le \bar{x}^{\text{turn}}) \wedge (\underline{y}^{\text{turn}} \le y \le \bar{y}^{\text{turn}}) \wedge (\underline{\theta}^{\text{turn}} \le \theta \le \bar{\theta}^{\text{turn}}) \wedge (\underline{v}^{\text{turn}} \le v \le \bar{v}^{\text{turn}}) \tag{B-1a}$$

$$\psi^{\text{stop}} = (\underline{x}^{\text{stop}} \le x \le \bar{x}^{\text{stop}}) \wedge (\underline{y}^{\text{stop}} \le y \le \bar{y}^{\text{stop}}) \wedge (\underline{\theta}^{\text{stop}} \le \theta \le \bar{\theta}^{\text{stop}}) \wedge (\underline{v}^{\text{stop}} \le v \le \bar{v}^{\text{stop}}) \tag{B-1b}$$

$$\varphi^{\text{off-road}} = \neg\big((\underline{x}^{\text{lane}} \le x \le \bar{x}^{\text{lane}}) \wedge (\underline{y}^{\text{lane}} \le y \le \bar{y}^{\text{lane}})\big) \tag{B-1c}$$

$$\varphi^{\text{on-road}} = (\underline{x}^{\text{lane}} \le x \le \bar{x}^{\text{lane}}) \wedge (\underline{y}^{\text{lane}} \le y \le \bar{y}^{\text{lane}}) \wedge (\underline{\theta}^{\text{lane}} \le \theta \le \bar{\theta}^{\text{lane}}) \wedge (\underline{v}^{\text{lane}} \le v \le \bar{v}^{\text{lane}}) \tag{B-1d}$$

$$\varphi^{\text{avoid}} = \neg\big((\underline{y}_2 \le y \le \bar{y}_2) \wedge (\underline{x}_2 \le x \le \bar{x}_2)\big) \tag{B-1e}$$

From the propositions, we construct the STL formula that the autonomous car must satisfy

$$\varphi \mathbf{U}_{[0,12]} \psi^{\text{turn}} \vee \varphi \mathbf{U}_{[11,12]} \psi^{\text{stop}}, \quad \varphi = \varphi^{\text{off-road}} \wedge \varphi^{\text{on-road}} \wedge \varphi^{\text{avoid}} \tag{B-2}$$

where the detailed functional representations of the $\varphi, \psi^{\text{stop}}, \varphi^{\text{off-road}}, \varphi^{\text{on-road}}, \varphi^{\text{avoid}}$ for on-road states are

$$g_{\psi^{\text{turn}}} = b_1^{\text{turn}}\big(1 - b_2^{\text{turn}}(x - x_{\text{offset}})^2 - b_3^{\text{turn}}(y - y_{\text{turn}})^2 - b_4^{\text{turn}}(\theta - \theta_{\text{offset}})^2\big) \tag{B-3a}$$

$$g_{\psi^{\text{stop}}} = b_1^{\text{stop}}\big(1 - b_2^{\text{stop}}(x - x_{\text{stop}})^2 - b_3^{\text{stop}}(y - y_{\text{offset}})^2 - b_4^{\text{stop}}(\theta - \theta_{\text{offset}})^2\big) \tag{B-3b}$$

$$g_{\varphi} = \begin{cases} < 0 & \text{, if car 2's position} \\ a_1\big(1 - a_2(x - x_{\text{offset}})^2 - a_3(y - y_{\text{offset}})^2 - a_4(\theta - \theta_{\text{offset}})^2\big) & \text{, otherwise} \end{cases} \tag{B-3c}$$

The functional representations for off-road states are more straightforward:

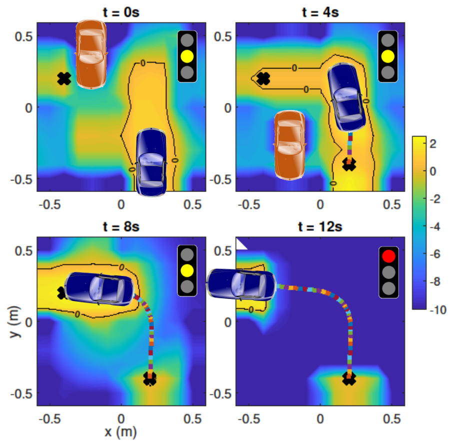
$$g_{\psi^{\text{turn}}} = g_{\psi^{\text{stop}}} = g_{\varphi} < 0 \tag{B-4}$$

Deviations from the desired states are penalized according to their weights specified as coefficients. In addition to off-road states, other lanes except for the top left and bottom right are set to have negative values in $g_{\psi^{\text{turn}}}$ and $g_{\psi^{\text{turn}}}$ respectively. The offset parameters x_{offset}, y_{offset} and θ_{offset} represent the desired states of the car. The goal parameters x_{turn} and y_{stop} represent the two desired final positions of the car. Deviations from the desired states are penalized through weights parameterized by the coefficients b_i^{turn}, b_i^{stop} and a_i for $i = 2, 3, 4$. The coefficients b_1^{turn}, b_1^{stop} determine the relative magnitude of the target propositions' formulas and hence their relative importance. For this example, we set $b_1^{\text{stop}} \leq b_1^{\text{turn}}$ to make the car take the turn whenever the constraints can be satisfied and only stop when it is unable to do so. We set a_1 to be larger than b_1^{turn} and b_1^{stop} such that the reachable target sets' values are the minimum in the set of feasible state trajectories. This is because as defined in (6), the maximal reachable set's value is the state trajectory's minimum value. Thus, when satisfied, a target set's values are required to be greater than the constraint set's values along the feasible state trajectory for the relative magnitudes of values in the target set to be observable in the maximal reachable set.

The numerical simulations and experiments involving TurtleBots are shown in Fig. 5, 6, and 7. The left plots of Fig. 5 show four time snapshots of the numerical simulation. The image of the blue car represents the autonomous car, and the image of the orange car represents the second car. The colors represent the values of $\max(g_{\varphi \mathbf{U}_{[0,12]} \psi^{\text{turn}}}, g_{\varphi \mathbf{U}_{[11,12]} \psi^{\text{stop}}})$, which can be used to synthesize the optimal controller without considering any controller restrictions, as discussed in Sec. 4.1 and summarized in Table 1. From the contours in the left top subplot, one can see that the maximum value of the value functions at the given state is positive; therefore, there exists a controller for the autonomous car to either complete the left turn, or stop before the intersection. The rest of the subplots show how the value functions change with time, and the trajectory the autonomous car takes to complete the turn. The right plot of Fig. 5 shows the robot experiment, with the final position of the TurtleBots shown in solid black, initial position shown in solid blue, and intermediate positions shown in translucent colors. The blue portions of the trajectory represent the first half of the trajectory, and the black portions represent the second half; this is done for added clarity, so that it is easier to see the joint positions of the TurtleBots at any given time snapshot. Fig. 6 shows the same scenario, but with the TurtleBot representing the autonomous car stop at the intersection because its initial position is further from the intersection and the other TurtleBot is moving very slowly across the intersection, blocking the way for a left turn. There is a small drift velocity caused by the coarse discretization of the state space and the bang-bang controller. This resulted in the autonomous car crossing slightly into the intersection.

In Fig. 7, we explore the notion of minimum violation by starting the autonomous car in a state that has negative value, as shown by the initial state of the autonomous car being outside the zero superlevel set. In fact, the autonomous car is even in the wrong lane initially on a collision course with the second car. Under the optimal disturbance, which in this case is the action of the second car, satisfaction of the STL formula is not possible; this may happen, for example, if the other travels slowly to block the autonomous car's way for making a left turn. However, in practical situations, it is unlikely for the second car to behave adversarially. Thus, by using a controller that maximizes the value function, the autonomous car soon moves into states with positive value, and completes the left turn on time.

Computation of the value function took approximately 3 hours.



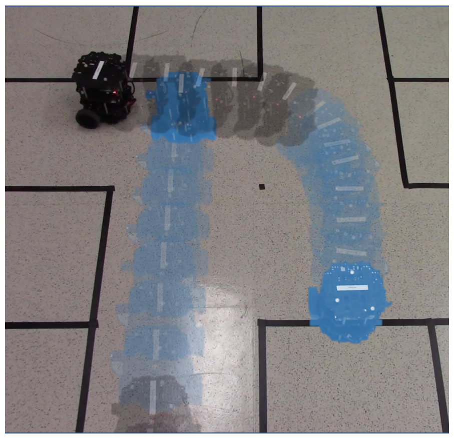
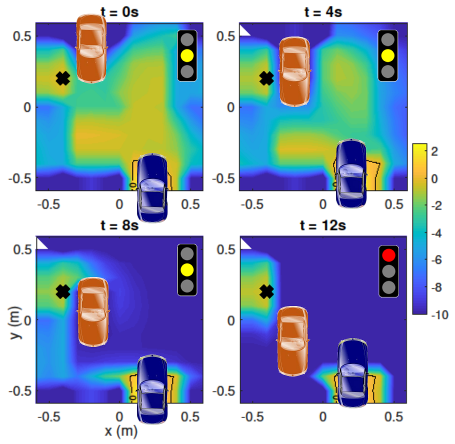


Fig. 5. The autonomous car (blue) successfully makes the left turn as the other car (red) moves quickly past the lane's entrance. **Left:** Contour plot of value function. **Right:** Time-lapse of the experiment.



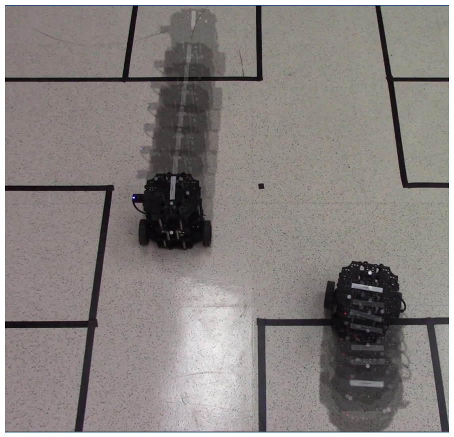


Fig. 6. The autonomous car (blue) stops in its initial lane due to a combination of its initial position being further back in the lane and the other car (red) moving slowly, blocking the lane's entrance. **Left:** Contour plot of the value function. **Right:** Time-lapse of the experiment.

B.3 Implementation Details of The Highway Example

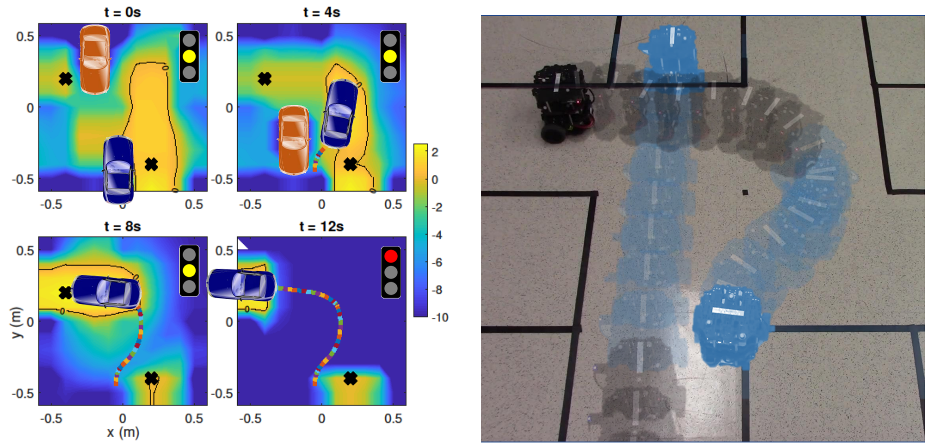


Fig. 7. The autonomous car (blue) minimizes violation of the STL formula. It successfully makes the left turn as the other car (red) did not behave adversarially. **Left:** Contour plot of the value function. **Right:** Time-lapse of the experiment.

STL Formulas

$$\psi^{\text{lane}} = (\underline{x}^{\text{lane}} \le x \le \bar{x}^{\text{lane}}) \wedge (\underline{y}^{\text{lane}} \le y \le \bar{y}^{\text{lane}}) \wedge (\underline{\theta}^{\text{lane}} \le \theta \le \bar{\theta}^{\text{lane}}) \wedge (\underline{v}^{\text{lane}} \le v \le \bar{v}^{\text{lane}}) \tag{B-5a}$$

$$\psi^{\text{pass}} = (\underline{x}^{\text{pass}} \le x \le \bar{x}^{\text{pass}}) \wedge (\underline{y}^{\text{pass}} \le y \le \bar{y}^{\text{pass}}) \wedge (\underline{\theta}^{\text{pass}} \le \theta \le \bar{\theta}^{\text{pass}}) \wedge (\underline{v}^{\text{pass}} \le v \le \bar{v}^{\text{pass}}) \tag{B-5b}$$

$$\psi^{\text{stay}} = (\underline{x}^{\text{stay}} \le x \le \bar{x}^{\text{stay}}) \wedge (\underline{y}^{\text{stay}} \le y \le \bar{y}^{\text{stay}}) \wedge (\underline{\theta}^{\text{stay}} \le \theta \le \bar{\theta}^{\text{stay}}) \wedge (\underline{v}^{\text{stay}} \le v \le \bar{v}^{\text{stay}}) \tag{B-5c}$$

$$\varphi^{\text{off-road}} = \neg\big((\underline{x}^{\text{lane}} \le x \le \bar{x}^{\text{lane}}) \wedge (\underline{y}^{\text{lane}} \le y \le \bar{y}^{\text{lane}})\big) \tag{B-5d}$$

$$\varphi^{\text{on-road}} = (\underline{x}^{\text{lane}} \le x \le \bar{x}^{\text{lane}}) \wedge (\underline{y}^{\text{lane}} \le y \le \bar{y}^{\text{lane}}) \wedge (\underline{\theta}^{\text{lane}} \le \theta \le \bar{\theta}^{\text{lane}}) \wedge (\underline{v}^{\text{lane}} \le v \le \bar{v}^{\text{lane}}) \tag{B-5e}$$

$$\varphi^{\text{avoid}} = \neg\big((\underline{y}_2 \le y \le \bar{y}_2) \wedge (\underline{x}_2 \le x \le \bar{x}_2) \wedge (\underline{y}_3 \le y \le \bar{y}_3) \wedge (\underline{x}_3 \le x \le \bar{x}_3)\big) \tag{B-5f}$$

Function Representations

For on-road states:

$$g_{\psi^{\text{lane}}} = b_1^{\text{lane}}\big(1 - b_2^{\text{lane}}(x - x_{\text{offset}})^2 - b_3^{\text{lane}}(\theta - \theta_{\text{offset}})^2\big) \tag{B-6a}$$

$$g_{\psi^{\text{pass}}} = b_1^{\text{pass}}\big(1 - b_2^{\text{pass}}(x - x_{\text{offset}})^2 - b_3^{\text{pass}}(y - y_{\text{pass}})^2 - b_4^{\text{pass}}(\theta - \theta_{\text{offset}})^2\big) \tag{B-6b}$$

$$g_{\psi^{\text{stay}}} = b_1^{\text{stay}}\big(1 - b_2^{\text{stay}}(x - x_{\text{offset}})^2 - b_3^{\text{stay}}(y - y_{\text{stay}})^2 - b_4^{\text{stay}}(\theta - \theta_{\text{offset}})^2\big) \tag{B-6c}$$

$$g_{\varphi} = \begin{cases} < 0 & \text{, if car 2's or 3's position} \\ a_1\big(1 - a_2(x - x_{\text{offset}})^2 - a_3(\theta - \theta_{\text{offset}})^2\big) & \text{, otherwise} \end{cases} \tag{B-6d}$$

For off-road states:

$$g_{\psi^{\text{lane}}} = g_{\psi^{\text{pass}}} = g_{\psi^{\text{stop}}} = g_{\varphi} < 0 \tag{B-7}$$

Similar to the traffic light example, the offset variables are the desired states of the car. Deviations are penalized according to the coefficients which function as weights. The goal positions are represented by y_{pass} and y_{stay} for the top and bottom halves of the right lane. For all g_{ψ} functions, only the right lane has non-negative values. For the same reason given in the traffic light example, $b_1^{\text{pass}} < b_1^{\text{lane}} < a$.