

Ro-To-Go!

Robust Reactive Control with Signal Temporal Logic

Roland Ilyes, Lara Bruder Müller, Nick Hawes, Bruno Lacerda

Abstract—Signal Temporal Logic robustness is a common objective for optimal robot control, but its dependence on history limits the robot’s decision-making capabilities when used in model predictive control approaches. In this work, we introduce Signal Temporal Logic robustness-to-go, a new quantitative semantics for the logic that isolates the contributions of suffix trajectories. We prove its relationship to formula progression for Metric Temporal Logic, and show that the robustness-to-go depends only on the suffix trajectory and progressed formula. We implement robustness-to-go as the objective in a model predictive control algorithm and use formula progression to efficiently evaluate it online. We test the algorithm in simulation and compare it to model predictive control using other robustness measures. Our experiments show that using robustness-to-go improves performance compared to using traditional robustness.

Index Terms—Formal Methods in Robotics and Automation, Hybrid Logical/Dynamical Planning and Verification, Optimization and Optimal Control

I. INTRODUCTION

Many robotic applications demand complex behaviors that are difficult to specify. Temporal Logics [1] provide a grounded mathematical approach to specify such behaviors. Linear Temporal Logic (LTL) [2] is a popular logic for discrete-time robotic applications. In recent years, there has been a growing interest in logics that describe more complex behaviors than LTL. Metric Temporal Logic (MTL) [3] and Metric Interval Temporal Logic (MITL) [4] extend LTL to reason about real-valued time. Signal Temporal Logic (STL) [5] is a subset of MITL defined over real-valued signals, making it particularly useful to robotic applications. In addition to providing a binary answer for whether a signal satisfies a specification, it also admits a quantitative score of how well the signal satisfies it, known as the *Robust Satisfaction Value*, or simply *robustness* [6]. This value characterizes how robust a planned trajectory is to unforeseen disturbances.

The state-of-the-art robot control algorithms for STL specifications are based on trajectory optimization techniques, and are split into two distinct approaches. One approach is to encode the STL specification as mixed integer constraints on an optimization problem. For convex objective functions, this results in a mixed integer convex program (MICP) [7]. Although this approach is complete, the constraints induced by the STL specification introduce new variables for each

This work received EPSRC funding via the “From Sensing to Collaboration” programme grant [EP/V000748/1]. R.I. and L.B. were supported by Amazon Web Services Lighthouse scholarships.

All authors are with the Oxford Robotics Institute, University of Oxford, United Kingdom. For correspondence: {rolandilyes, larab, nickh, bruno}@robots.ox.ac.uk

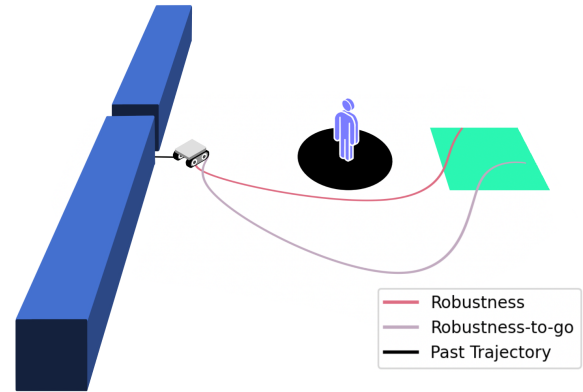


Fig. 1: This examples shows a robot planning a path to the goal while avoiding the human. With regular robustness, the proximity to the obstacles at the beginning limits the overall robustness, so there is no value in going further away from the human. With robustness-to-go, the robot has forgotten about its past proximity to the obstacles and can find trajectories that are more robust to human movement.

time step and this scales poorly. This approach also requires both the system dynamics [8] and the logical predicates [9] to be linear. The other approach directly optimizes the STL robustness [10], resulting in a nonlinear programming problem (NLP). This approach allows for nonlinear system dynamics and logical predicates. Although incomplete, this approach allows for nonlinear system dynamics and logical predicates, and scales better than the MICP approach due to the lack of integer constraints.

In addition to planning for complex behaviors, robots must also react to unforeseen changes in their environment. Consider a practical example, shown in Fig. 1, where a robot must reach a goal within a fixed time limit while avoiding obstacles and humans. For such a mission, the robot might be unaware of how the human will move and must react to the human’s motion.

One approach to achieve such reactive behavior is Model Predictive Control (MPC), which solves the aforementioned optimization problems periodically during mission execution. Given the current state, at each iteration, a new optimal trajectory is computed and executed until the next trajectory is found. This iterative replanning gives MPC approaches some level of resilience to dynamic environments.

While the STL robustness is a sound objective function for open-loop control synthesis, it has shortcomings in the

context of feedback control through MPC. This is because the robustness is a function of the entire trajectory, and so the entire history of the robot’s motion must be considered in the objective function. If the robot comes close to violating the specification, then all of its plans from that point onward are limited in how robust they can be. After such an event, the robot cannot distinguish between good and bad trajectories because the robustness does not provide any incentive for the robot to move further away from future possible violations. Fig. 1 captures this phenomenon. Both trajectories have the same robustness due to their proximity to obstacles at the beginning, even though the blue trajectory steers further from the human.

In this paper, we introduce *robustness-to-go* (Ro-To-Go), a new robustness measure designed to reason about the *suffix* of trajectories, the portion beyond a given time. This measure scores the trajectory from the given time onwards, instead of scoring the past trajectory of the robot. This measure is recursively defined on the syntax of the logic and generalizes to all of STL. At each MPC planning step, we use Ro-To-Go as our objective function, enabling the robot to remain robust to future disturbances, regardless of its past. We prove that Ro-To-Go can be efficiently calculated by using MITL formula progression [11], a technique that rewrites formulas iteratively after each observation.

The contributions of this paper are (i) introduction of robustness-to-go, a new STL robustness measure that focuses on reasoning about suffix trajectories, (ii) a technique to efficiently calculate the robustness-to-go using formula progression (iii) proofs of the soundness of the measure and its relationship to formula progression, and (iv) a series of case studies investigating the incorporation of this measure in a feedback control algorithm, and its impact on robot behavior.

II. RELATED WORK

Control synthesis for STL specifications was first investigated in [7], where they extend an MICP-approach to synthesis for LTL specifications from [12] to handle STL specifications. This approach automatically generates a set of mixed integer linear constraints for an optimal control problem. If the system is linear and the objective function is convex, then the optimal solution can be found using a MICP [9].

These encodings only apply to STL formulas over linear predicates. Furthermore, nonlinear systems turn the optimization problem into a mixed-integer non-convex program, which is intractable. Therefore, instead of encoding the STL specification as mixed integer constraints, an alternate approach formulates an optimization problem where the specification is instead considered in the objective function, via the robustness measure. This approach still results in a nonconvex optimization problem, but finding local optima is faster because of the lack of integer constraints. The notion of optimizing temporal logic robustness was first introduced in [13], in the context of minimizing MTL robustness for verification purposes. Robustness is a highly discontinuous function, and so the authors in [10] propose a smooth

approximation of MTL robustness, and use gradient ascent tools to maximize it for control synthesis purposes. The works [14], [15], [16] extend this approach to optimize smooth approximations of STL robustness.

Robustness as an objective function, however, has its shortcomings. Called *locality* in [17], traditional robustness characterizes trajectories by their *most critical point*, that is, the point that requires the smallest alteration to change the satisfaction result. In the context of open-loop control synthesis, this is a reasonable objective to optimize for: making the most critical point more robust to disturbances makes the entire trajectory more robust to disturbances. In online feedback control, however, the most critical point could be a point in the past, that has already happened. The result of this is that all plans will be limited by this point, and the robot has trouble identifying trajectories that are more robust *from this point onwards*. Works [8] and [18] attempt to better characterize the entire trajectory by defining new “average” STL robustness measures to optimize for. Recently, the authors in [17] generalize these robustness measures, among others, by introducing *Generalized Mean Robustness*. These measures are not subject to the locality effect, meaning that the robot can distinguish between two trajectories whose most critical points are the same. In contrast, this work addresses the issue of locality specifically in the context of online control, where the most critical point happened in the robot’s past. We propose *robustness-to-go*, a measure that is defined on the *suffix* of trajectories. While still local, the most critical point for a formula’s robustness-to-go is within the planning horizon of the robot.

III. PRELIMINARIES

Let $\mathbf{x} \in X \subseteq \mathbb{R}^n$ be a *state*, and \mathcal{T} be a set of time instants t such that $\mathcal{T} \subseteq \mathbb{R}_{\geq 0}$. Then, a *signal* s is a mapping from \mathcal{T} to X , and *sample* from the signal is $(t, s(t))$.

A *signal predicate* is a formula of the form $g(\mathbf{x}) > 0$, where $g : X \rightarrow \mathbb{R}$. We express properties of signals with respect to signal predicates using *signal temporal logic* (STL) [5].

Definition 1: (STL Syntax). The STL Syntax is recursively defined by:

$$\varphi := \top \mid g(\mathbf{x}) > 0 \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi U_I \psi$$

where φ are STL formulas and $I = \langle a, b \rangle$ is a time interval that can be open or closed on both ends with $0 \leq a < b \leq \infty$. Notation \top , \neg , and \wedge are Boolean “true”, “negation” and “conjunction,” respectively, and U denotes the temporal “until” operator.

Boolean disjunction can be expressed as $\varphi \vee \psi = \neg(\neg\varphi \wedge \neg\psi)$, Boolean implication as $\varphi \rightarrow \psi = (\neg\varphi) \vee \psi$, and Boolean “false” as $\perp = \neg\top$. The temporal operator *eventually* ($\diamond_I \varphi$) can be expressed as $\top U_I \varphi$ and the temporal operator *globally* ($\square_I \varphi$) can be expressed as $\neg \diamond_I \neg \varphi$. The interval $I + c = \langle a + c, b + c \rangle$.

Definition 2: (STL Semantics). The semantics of STL is

defined over a signal s at time t as:

$$\begin{aligned}
(s, t) \models \top & \iff \top \\
(s, t) \models g(\mathbf{x}) > 0 & \iff g(s(t)) > 0 \\
(s, t) \models \neg\varphi & \iff (s, t) \not\models \varphi \\
(s, t) \models \varphi \wedge \psi & \iff (s, t) \models \varphi \wedge (s, t) \models \psi \\
(s, t) \models \varphi U_I \psi & \iff \exists t' \in I + t \text{ s.t. } (s, t') \models \psi \\
& \quad \wedge \forall t'' \in [t, t'), (s, t'') \models \varphi
\end{aligned}$$

where \models denotes satisfaction. A signal s satisfies an STL formula φ if $(s, 0) \models \varphi$.

In this work, we consider the set \mathcal{T} to be finite and strictly increasing, and employ discrete-time, or *pointwise*, semantics. Refer to [19] and [20] for a discussion on the differences between continuous and pointwise semantics.

The authors in [11] show that, under pointwise semantics, we can incrementally track the satisfaction of MTL formulas using *formula progression*. We adapt it here for STL.

Definition 3: (STL Formula Progression). Given an STL formula φ , a time step Δ , and an observation of a state \mathbf{x} , the progressed formula is given by $P(\varphi, \Delta, \mathbf{x})$:

$$\begin{aligned}
P(\top, \Delta, \mathbf{x}) & = \top \\
P(\perp, \Delta, \mathbf{x}) & = \perp \\
P(g(\mathbf{x}) > 0, \Delta, \mathbf{x}) & = \begin{cases} \top & \text{if } g(\mathbf{x}) > 0 \\ \perp & \text{otherwise} \end{cases} \\
P(\neg\varphi, \Delta, \mathbf{x}) & = \neg P(\varphi, \Delta, \mathbf{x}) \\
P(\varphi \wedge \psi, \Delta, \mathbf{x}) & = P(\varphi, \Delta, \mathbf{x}) \wedge P(\psi, \Delta, \mathbf{x}) \\
P(\varphi U_I \psi, \Delta, \mathbf{x}) & = \begin{cases} P(\varphi, \Delta, \mathbf{x}) \wedge \varphi U_{I_{\Delta}^{\leftarrow}} \psi & \text{if } 0 < I \\ P(\psi, \Delta, \mathbf{x}) \vee \\ (P(\varphi, \Delta, \mathbf{x}) \wedge \varphi U_{I_{\Delta}^{\leftarrow}} \psi) & \text{if } 0 \in I, \end{cases}
\end{aligned}$$

where $I_{\Delta}^{\leftarrow} = (I - \Delta) \cap \mathbb{R}_{\geq 0}$.

Note that we have added a truncation operation to the definition from [11] to enforce nonnegative interval endpoints. Here, I_{Δ}^{\leftarrow} is the subset of $I - \Delta$ that is greater than or equal to zero, which could be the empty set \emptyset . By Definition 2, $\varphi U_{\emptyset} \psi = \perp$ because there exists no $t' \in \emptyset$. Therefore, the truncation operation resolves the case $0 > I$ from [11], which is why it is absent from Definition 3.

The use of signal predicates allows interpretation of STL using *quantitative* semantics, called *robustness* [6], in addition to qualitative semantics.

Definition 4: (STL Robustness). The quantitative semantics of STL is defined over a signal s at time t as $\rho(s, t, \varphi)$:

$$\begin{aligned}
\rho(s, t, \top) & = \infty \\
\rho(s, t, g(\mathbf{x}) > 0) & = g(s(t)) \\
\rho(s, t, \neg\varphi) & = -\rho(s, t, \varphi) \\
\rho(s, t, \varphi \wedge \psi) & = \min\{\rho(s, t, \varphi), \rho(s, t, \psi)\} \\
\rho(s, t, \varphi U_I \psi) & = \sup_{t' \in I + t} \min \left(\begin{array}{c} \rho(s, t', \psi), \\ \inf_{t'' \in [t, t']} \rho(s, t'', \varphi) \end{array} \right).
\end{aligned}$$

A signal s satisfies an STL formula φ if $\rho(s, 0, \varphi) > 0$.

Definition 5: (STL Time Horizon [21]). The Time Horizon of an STL formula $[\varphi]^{\uparrow}$ is the horizon within which a signal can impact the satisfaction of φ :

$$\begin{aligned}
[g(\mathbf{x}) > 0]^{\uparrow} & = 0 \\
[\neg\varphi]^{\uparrow} & = [\varphi]^{\uparrow} \\
[\varphi \wedge \psi]^{\uparrow} & = \max\{[\varphi]^{\uparrow}, [\psi]^{\uparrow}\} \\
[\varphi U_{(a,b)} \psi]^{\uparrow} & = b + \max\{[\varphi]^{\uparrow}, [\psi]^{\uparrow}\}.
\end{aligned}$$

A formula is bounded if $[\varphi]^{\uparrow} < \infty$, or unbounded otherwise.

IV. PROBLEM FORMULATION

Consider a discrete time robotic system of the form

$$\mathbf{x}_{t+\Delta}^r = f(\mathbf{x}_t^r, \mathbf{u}_t)$$

where $\mathbf{x}^r \in X^r \subset \mathbb{R}^{n_r}$ is the state of the robot, $\mathbf{u} \in U \subset \mathbb{R}^m$ is the control input, and $\Delta \in \mathbb{R}_{\geq 0}$. The environment state is $\mathbf{x}^e \in X^e \subset \mathbb{R}^{n_e}$, and its dynamics are unknown. The environment can capture dynamic yet not controllable aspects of the world that are relevant to the STL formula. This can include not only a dynamic physical environment, but also other agents acting in the same space as the robot.

We wish to specify tasks as *bounded* STL formula over the composed robot-environment system. For a specification φ , we consider time domain $\mathcal{T} = \{0, \Delta, 2\Delta, \dots, [\varphi]^{\uparrow}\}$. Our signal $s \in S$ is a mapping $s : \mathcal{T} \rightarrow X^r \times X^e \times U$. We denote the components of s using superscripts, i.e., the robot state component of signal s at time t is $s^r(t)$.

We first present the motion planning problem. Given that the robot finds motion plans iteratively for closed-loop feedback control, it will find motions *during* execution. Because STL satisfaction depends on the robot's future *and* past, the robot must consider its history. Let s_t be a signal defined over $[0, t]$ such that $t < \infty$. s_t is a *prefix* of s if, for all $t' \leq t$, $s(t') = s_t(t')$.

Problem 1: Given system f , bounded STL task φ , and signal prefix s_t , find s such that:

- (i) $s \models \varphi$,
- (ii) $s(t') = s_t(t') \forall t' \in [0, t]$,
- (iii) $s^e(t') = s_t^e(t') \forall t' \in [t, [\varphi]^{\uparrow}]$,
- (iv) $s^r(t' + \Delta) = f(s^r(t'), s^u(t')) \forall t' \in [0, [\varphi]^{\uparrow}]$.

Here, (i) ensures the solution satisfies the STL formula, (ii) enforces the solution to contain the observed history of the system trajectory, (iii) assumes the environment is static for planning purposes, and (iv) ensures that the solution obeys the robot dynamics f . Due to the static environment assumption, there are no guarantees on the safety of the closed-loop controller. However, by integrating solutions to Problem 1 in an MPC loop, subsequent solutions plan for the updated environment state, allowing the robot to react to the unknown environment disturbances.

V. RO-TO-GO!

Solutions to Problem 1 include both the NLP and MICP approaches discussed in section I. The MICP approach introduces extra constraints to ensure $s \models \varphi$, and optimize for some other convex objective function [7]. The NLP approach involves optimizing the robustness, or some other robustness measure [17]. They find $\max_{s \in S} \rho(s, 0, \varphi)$, and $s \models \varphi$ if the optimization converges on a local optima such that $\rho(s, 0, \varphi) > 0$.

We are interested in the NLP solution to Problem 1 in the context of MPC. The robustness depends on the entire trajectory, meaning that the robot must consider the past as a *prefix* s_t , and the robot only plans for the *suffix*. As the MPC algorithm iterates and the robot executes the mission, the portion of the trajectory that the robot can control shrinks. In many instances, the portion of the trajectory in the past, outside of the robot's control, dominates the robustness. The impact of this is that the portion of the trajectory that the robot *can* control does not have a big impact on the robustness, and the robot cannot easily distinguish between different behaviors.

Ideally, the robot should focus on what it *can* control, and find the most robust behaviors *within that window*. To this end, we propose *robustness-to-go* (Ro-To-Go).

Definition 6: (STL Robustness-To-Go). The robustness-to-go from time \hat{t} of STL formula φ is defined over a signal s at time t as $\rho^{\text{rtg}}(s, t, \hat{t}, \varphi)$:

$$\begin{aligned} \rho^{\text{rtg}}(s, t, \hat{t}, \top) &= \infty \\ \rho^{\text{rtg}}(s, t, \hat{t}, g(x) > 0) &= \begin{cases} g(s(t)) & \text{if } t > \hat{t} \\ \infty \cdot \text{sign}(g(s(t))) & \text{otherwise} \end{cases} \\ \rho^{\text{rtg}}(s, t, \hat{t}, \neg\varphi) &= -\rho^{\text{rtg}}(s, t, \hat{t}, \varphi) \\ \rho^{\text{rtg}}(s, t, \hat{t}, \varphi \wedge \psi) &= \min\{\rho^{\text{rtg}}(s, t, \hat{t}, \varphi), \rho^{\text{rtg}}(s, t, \hat{t}, \psi)\} \\ \rho^{\text{rtg}}(s, t, \hat{t}, \varphi U_I \psi) &= \sup_{t' \in I+t} \min \left(\begin{array}{c} \rho^{\text{rtg}}(s, t', \hat{t}, \psi), \\ \inf_{t'' \in [t, t']} \rho^{\text{rtg}}(s, t'', \hat{t}, \varphi) \end{array} \right). \end{aligned}$$

Ro-To-Go differs from traditional robustness in how it treats predicates. When the predicate robustness is evaluated after some user-specified \hat{t} , it is equivalent to traditional robustness. If, however, the predicate robustness is evaluated *before* \hat{t} , the robustness is instead ∞ or $-\infty$, depending on if the predicate is satisfied or not.

We propose solving Problem 1 by optimizing for the Ro-To-Go from the planning time onwards. This approach is sound because Ro-To-Go itself is a sound robustness measure.

Theorem 1: (Soundness). A signal s satisfies an STL formula if and only if it has a positive robustness-to-go, i.e. $(s, t) \models \varphi \iff \rho^{\text{rtg}}(s, t, \hat{t}, \varphi) > 0$.

Proof: The proof is in Appendix I.

Example 1: (Robustness-to-go.) Consider the formula $\varphi_{\text{reachAvoid}} = \square_{[0,20]}(\text{workspace} \wedge \neg(\text{human} \vee \text{obs}_1 \vee \text{obs}_2)) \wedge \diamond_{[15,20]} \text{goal}$, which specifies that the robot must avoid the human and obstacles for twenty seconds, and reach the goal between fifteen and twenty seconds from the start time. Here,

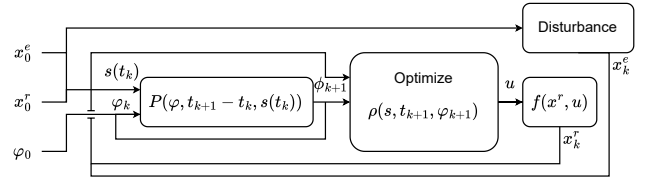


Fig. 2: Control Block Diagram for Ro-To-Go MPC with Formula Progression.

the subformulas are defined as

$$\begin{aligned} \text{workspace} &= (\mathbf{x}^r[0] > 0 \wedge \mathbf{x}^r[0] < 5 \wedge \mathbf{x}^r[1] > 0 \wedge \mathbf{x}^r[1] < 5) \\ \text{human} &= (\mathbf{x}^r[0] - \mathbf{x}^e[0])^2 + (\mathbf{x}^r[1] - \mathbf{x}^e[1])^2 < 0.25 \\ \text{obs}_1 &= (\mathbf{x}^r[0] > 0.5 \wedge \mathbf{x}^r[0] < 1 \wedge \mathbf{x}^r[1] > 0 \wedge \mathbf{x}^r[1] < 2.4) \\ \text{obs}_2 &= (\mathbf{x}^r[0] > 0.5 \wedge \mathbf{x}^r[0] < 1 \wedge \mathbf{x}^r[1] > 2.6 \wedge \mathbf{x}^r[1] < 5) \\ \text{goal} &= (\mathbf{x}^r[0] > 4 \wedge \mathbf{x}^r[0] < 5 \wedge \mathbf{x}^r[1] > 2 \wedge \mathbf{x}^r[1] < 3). \end{aligned}$$

Fig. 1 shows the robot planning for these specifications online. The robot has already executed a prefix trajectory, and began close to obs_1 and obs_2 , at $\mathbf{x}_0^r = [0.5, 2.5]^T$. Because of this, all trajectories beginning from this initial state will have a robustness of *at most* 0.1. Indeed, the red line in Fig. 1 shows the robot coming within 0.13 meters of the human. This is because, when using robustness, the robot could not distinguish between this trajectory and one that goes further from the human. In contrast, the Ro-To-Go from the planning point onwards does not consider the contribution of the initial point, but only the contribution from the points within the plan under consideration. As such, it can better distinguish between good and bad motion plans, finding a plan that only comes within 0.79 meters of the human. The plan found using Ro-To-Go is safer with respect to unforeseen human motion.

VI. FORMULA PROGRESSION FOR STL RO-TO-GO

Ro-To-Go is a function of the entire trajectory signal. As such, when planning trajectories during mission execution, the history of the robot's motion must be considered. During planning, these iterative Ro-To-Go evaluations must consider the same past. Because the robot cannot change the past, processing this data is redundant. In this section, we show how to use formula progression from Definition 3 to avoid these redundant operations. Informally, we show that **the Ro-To-Go from time \hat{t} is the same as the robustness of the progressed formula at time \hat{t}** .

Theorem 2: Let φ_0 be an STL formula, s be a signal, and consider sample $(t_i, s(t_i))$. Let $\Phi = \{\varphi_0, \varphi, \dots, \varphi_{i+1}\}$ be a set of STL formulas such that $\varphi_k = P(\varphi_{k-1}, t_k - t_{k-1}, s(t_{k-1}))$ for each $0 < k \leq i + 1$. Then,

$$t_i \geq t_0 \rightarrow \rho(s, t_{i+1}, \varphi_{i+1}) = \rho^{\text{rtg}}(s, t_0, t_i, \varphi_0).$$

Proof: The proof is in Appendix I.

Fig. 2 shows how to incorporate formula progression for MPC with Ro-To-Go. The approach begins by using the initial robot state, environment state, and STL formula φ_0 to find the progressed formula φ_{k+1} . In the NLP optimization problem, we optimize the robustness of trajectories with respect to φ_{k+1} , beginning from the next time step t_{k+1} . According to Theorem 2, the optimization objective $\rho(s, t_{k+1}, \varphi_{k+1})$ is the Ro-To-Go of the original formula *from the initial point onwards* $\rho^{\rightarrow}(s, t_0, t_0, \varphi_0)$. Note that, even though the first point evaluated is t_{k+1} , the state at t_k must still be considered to ensure that trajectories obey the robot dynamics f . Once the optimal solution is found, the robot applies control input u , and the environment evolves according to its unknown motion. The next iteration begins, with the progressed formula φ_{k+1} becoming φ , and the new robot and environment states become $s(t_k)$. For a given planning iteration at time t_i , the robot is optimizing the Ro-To-Go of the original formula *from the current time onwards* $\rho^{\rightarrow}(s, t_0, t_i, \varphi_0)$.

Using formula progression to find the Ro-To-Go has three benefits. The first is that it avoids reprocessing of past observations. This is because the robustness of the progressed formula depends only on points within the robot's planning horizon. The second benefit is that, by reducing the problem of calculating Ro-To-Go to the problem of calculating robustness, we can exploit readily available tools for calculating robustness such as Breach [22] or STLrom [23]. Finally, using formula progression paves the road for combining Ro-To-Go with other robustness measures, such as those discussed in [17].

VII. EXPERIMENTS

While the approach described in Fig. 2 is planner agnostic, we have implemented Ro-To-Go for use in solving Problem 1 using VP-STO [24], a state-of-the-art motion planning and MPC algorithm which uses CMA-ES [25] as the optimization technique. CMA-ES is zero-order optimization method and thus can handle discontinuous objective functions, making it a good fit for STL robustness. To calculate the STL robustness of progressed formulas while planning, we use STLrom [23]. The hyperparameters for our planner are provided in Appendix II.

For our robot dynamics, we consider a two-dimensional double integrator, i.e., $\mathbf{x}_t^r = [p_x, p_y, v_x, v_y]^T$, $\mathbf{u}_t = [a_x, a_y]^T$, and

$$\mathbf{x}_{t+\Delta}^r = f(\mathbf{x}_t^r, \mathbf{u}_t) = \begin{bmatrix} \mathbf{I}_{2 \times 2} & \Delta \mathbf{I}_{2 \times 2} \\ \mathbf{0}_{2 \times 2} & \mathbf{I}_{2 \times 2} \end{bmatrix} \mathbf{x}_t^r + \begin{bmatrix} 0.5 \Delta^2 \mathbf{I}_{2 \times 2} \\ \mathbf{I}_{2 \times 2} \end{bmatrix} \mathbf{u}_t,$$

where $\Delta = 0.1$ is the sampling time. The robot is subject to velocity constraints of 2 m/s and input constraints of 2 m/s² in both dimensions. The environment state $\mathbf{x}^e \in \mathbb{R}^2$ receives a random perturbation $w = [w_1, w_2]^T$, where $w_1, w_2 \sim \mathcal{N}(0, 1)$, at every planning step.

For our experiments, we examine the task described by $\varphi_{reachAvoid}$ from Example 1 and

$$\varphi_{stayIn} = \square_{[0,20] \text{ region}}, \text{ where} \\ \text{region} = (\mathbf{x}^r[0] - \mathbf{x}^e[0])^2 + (\mathbf{x}^r[1] - \mathbf{x}^e[1])^2 < 2.25,$$

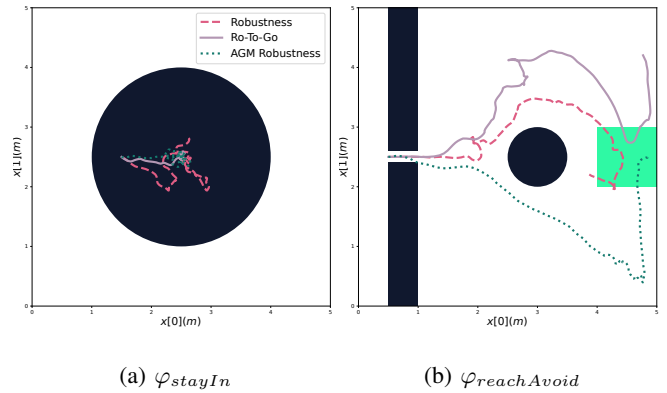


Fig. 3: Sample trajectories for two STL formulas, simulated with a static environment.

which specifies that the robot must remain within some region for twenty seconds. The robot begins at $\mathbf{x}_0^r = [0.5, 2.5]^T$ for $\varphi_{reachAvoid}$ and $\mathbf{x}_0^r = [1.5, 2.5]^T$ for φ_{stayIn} . The environment state begins at $\mathbf{x}_0^e = [3, 2.5]^T$ for $\varphi_{reachAvoid}$ and $\mathbf{x}_0^e = [2.5, 2.5]^T$ for φ_{stayIn} . Fig. 3 shows the environments for $\varphi_{reachAvoid}$ and φ_{stayIn} .

Our experiments investigate the impact of using Ro-To-Go as an objective function in an MPC algorithm. As baselines, we also use traditional robustness [6] and Arithmetic-Geometric Mean (AGM) Robustness [8], one of the ‘‘average’’ robustness measures designed to address the locality issues discussed in Section II [17].

Table I shows the averaged performance of all three approaches across 50 simulations, reporting the mean and standard deviation across a variety of measures. Along with robustness and AGM robustness, we report the success rate, trajectory length, and minimum distance from the dynamic environment. Our simulation stops upon formula violation, and because robustness, AGM robustness, and trajectory length are functions of complete trajectories, these values are representative of successful trials only.

A. Case Study 1: φ_{stayIn}

Fig. 3a shows closed loop trajectories from all three algorithms when simulated in a *static* environment. The circle represents the dynamic environment captured in `region` in φ_{stayIn} . Fig. 3a shows that the robot stays near the center of the region after steering to it when using Ro-To-Go or AGM robustness, but steers away from the center when using traditional robustness. This is due to the proximity to the edge of the circle at the beginning of the mission. When using traditional robustness, the robot cannot distinguish between trajectories that stay near the middle and those that do not, because this proximity at the beginning limits the achievable robustness of plans.

Table I shows the benchmarking results for this experiment, now simulated using a *dynamic* environment. The minimum distance measures are negative because the robot is inside of the region, and the lowest attainable value is -1.5. The robot performs very similarly when using Ro-To-Go and AGM

Planning Problem	Trajectory Length (m)	Min Distance (m)	Robustness	AGM Robustness	Success Rate (%)
φ_{stayIn} , Robustness	5.112 \pm 0.406	-1.381 \pm 0.211	0.238 \pm 0.132	0.126 \pm 0.014	80
φ_{stayIn} , Ro-To-Go	5.74 \pm 0.522	-1.444 \pm 0.134	0.354 \pm 0.119	0.152 \pm 0.01	96
φ_{stayIn} , AGM	6.867 \pm 0.604	-1.45 \pm 0.135	0.369 \pm 0.115	0.155 \pm 0.007	98
$\varphi_{reachAvoid}$, Robustness	6.046 \pm 0.375	0.236 \pm 0.252	0.077 \pm 0.022	0.162 \pm 0.012	68
$\varphi_{reachAvoid}$, Ro-To-Go	7.246 \pm 0.567	0.266 \pm 0.267	0.069 \pm 0.03	0.163 \pm 0.012	74
$\varphi_{reachAvoid}$, AGM	7.621 \pm 0.514	0.404 \pm 0.474	0.046 \pm 0.023	0.208 \pm 0.007	72

TABLE I: Results of 50 runs of each approach against two STL specifications, simulated with a dynamic environment. The results for trajectory length, robustness, and AGM robustness represent successful trials only.

robustness in all measures except for trajectory length, where AGM robustness results in longer trajectories. Both AGM robustness and Ro-To-Go outperform traditional robustness.

B. Case Study 2: $\varphi_{reachAvoid}$

Fig. 3b shows closed loop trajectories from all three algorithms when simulated in a static environment. The circle represents the human, described as subformula $human$ in Example 1. Fig. 3b shows that the robot steers further from the human when using Ro-To-Go as the objective function than it does when using traditional robustness. This is due to the proximity to the obstacles at the beginning of the mission. Online, when using traditional robustness, each plan is limited by this initial proximity, so trajectories that steer further from the human are no more robust than those that steer close to the human. Using Ro-To-Go, however, removes this dependence and allows the robot to steer further from the human.

Fig. 3b also shows a shortcoming of using AGM robustness as a cost function: because the function is not dominated by the most critical point, the robot can still achieve a high AGM robustness even when coming very close to collision. This is reflected in Fig. 3b when the robot steers much closer to the walls at the beginning of the mission.

Table I shows that the robot achieves similar performance with regards to both robustness and AGM robustness whether it uses robustness or Ro-To-Go as the cost function. Despite this, the robot achieves a higher success rate when using Ro-To-Go, performing comparably to AGM robustness. The reason for this performance difference is more apparent when examining the trajectory lengths and minimum distances to the environment. The robot stays further from the dynamic region using Ro-To-Go when compared to traditional robustness, resulting in longer trajectories.

C. Discussion

One advantage to using AGM robustness over traditional robustness and Ro-To-Go is that it results in trajectories that are temporally robust as well, e.g., the robot will spend more time in the $goal$ region for $\varphi_{reachAvoid}$. This is in contrast to the behavior seen when using Ro-To-Go in Fig. 3b: The robot does not go deeper into the goal nor stay in it, but instead leaves. This is because, once the robot has entered the goal region, the subformula $goal$ in $\varphi_{reachAvoid}$ progresses to \top and is not considered in the Ro-To-Go afterwards. One interpretation of this behavior is that the robot displays *more* robust behavior under “globally” (\square)

type formulas, and *less* robust behavior under “eventually” (\diamond) type formulas. In the context of MPC, however, there is a more pragmatic interpretation. STL robustness represents a trajectory’s robustness to change. Because the past cannot change, satisfying points in the past are in some sense *infinitely* robust (reflected in Definition 6). There is no disturbance that would cause this past satisfaction to become a violation, and so planning to satisfy it more robustly can be wasteful.

Ro-To-Go also has advantages over AGM robustness. Table I shows that the robot achieves a lower traditional robustness when using AGM robustness for $\varphi_{reachAvoid}$ when compared to Ro-To-Go. This is a result of the aforementioned behavior seen in Fig. 3b: the robot can still have a high AGM robustness while coming close to collision at the beginning. Ro-To-Go is also less demanding to calculate online. This is because, when leveraging formula progression for Ro-To-Go, the number of samples that must be considered decrements at each planning step while AGM robustness still depends on past states.

We reiterate that Ro-To-Go is an orthogonal concept to AGM, and a future direction for research could investigate the impact of using AGM robustness alongside formula progression.

VIII. CONCLUSION AND FUTURE WORK

The main contribution of this paper is STL robustness-to-go (Ro-To-Go), a new quantitative semantics for the logic, along with proofs of its relationship to MTL formula progression. This robustness measure addresses the tendency of events in the past limiting decision making capabilities of robotic systems when optimizing for traditional robustness. We implement robustness, robustness-to-go, and AGM robustness as the objective functions in an MPC algorithm, and find that robustness-to-go outperforms traditional robustness.

For future work, we will investigate the relationship between robustness-to-go, which focuses on the contribution of *suffix* trajectories, and the Robust Satisfaction Interval (RoSI) [27], which focuses on the contribution of *prefix* trajectories. The work [28] uses the RoSI for MPC of unbounded STL specifications by considering the robustness of signals *up to* a given point in time. RoSI as a cost function has similar issues to robustness, and combining it with robustness to go would enable the robot to consider only the robustness of a user-specified window of time. We plan to investigate how the RoSI-To-Go could address the locality issues and impact robot performance.

REFERENCES

- [1] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.
- [2] A. Pnueli, “The temporal logic of programs,” in *18th annual symposium on foundations of computer science (sfcs 1977)*. IEEE, 1977, pp. 46–57.
- [3] R. Koymans, “Specifying real-time properties with metric temporal logic,” *Real-time systems*, vol. 2, no. 4, pp. 255–299, 1990.
- [4] R. Alur, T. Feder, and T. A. Henzinger, “The benefits of relaxing punctuality,” *Journal of the ACM (JACM)*, vol. 43, no. 1, pp. 116–146, 1996.
- [5] O. Maler and D. Nickovic, “Monitoring temporal properties of continuous signals,” in *International symposium on formal techniques in real-time and fault-tolerant systems*. Springer, 2004, pp. 152–166.
- [6] 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.
- [7] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, “Model predictive control with signal temporal logic specifications,” in *53rd IEEE Conference on Decision and Control*, 2014, pp. 81–87.
- [8] N. Mehdipour, C.-I. Vasile, and C. Belta, “Arithmetic-geometric mean robustness for control from signal temporal logic specifications,” in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 1690–1695.
- [9] C. Belta and S. Sadraddini, “Formal methods for control synthesis: An optimization perspective,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, no. 1, pp. 115–140, 2019.
- [10] Y. V. Pant, H. Abbas, and R. Mangharam, “Smooth operator: Control using the smooth robustness of temporal logic,” in *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2017, pp. 1235–1240.
- [11] F. Bacchus and F. Kabanza, “Planning for temporally extended goals,” *Annals of Mathematics and Artificial Intelligence*, vol. 22, pp. 5–27, 1998.
- [12] E. M. Wolff, U. Topcu, and R. M. Murray, “Optimization-based trajectory generation with linear temporal logic specifications,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 5319–5325.
- [13] H. Abbas and G. Fainekos, “Computing descent direction of mtl robustness for non-linear systems,” in *2013 American Control Conference*. IEEE, 2013, pp. 4405–4410.
- [14] Y. V. Pant, H. Abbas, R. A. Quaye, and R. Mangharam, “Fly-by-logic: Control of multi-drone fleets with temporal logic objectives,” in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 186–197.
- [15] 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.
- [16] R. Takano, H. Oyama, and M. Yamakita, “Continuous optimization-based task and motion planning with signal temporal logic specifications for sequential manipulation,” in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 8409–8415.
- [17] N. Mehdipour, C.-I. Vasile, and C. Belta, “Generalized mean robustness for signal temporal logic,” *IEEE Transactions on Automatic Control*, 2024.
- [18] L. Lindemann and D. V. Dimarogonas, “Robust control for signal temporal logic specifications using discrete average space robustness,” *Automatica*, vol. 101, pp. 377–387, 2019.
- [19] P. Bouyer, U. Fahrenberg, K. G. Larsen, N. Markey, J. Ouaknine, and J. Worrell, “Model checking real-time systems,” *Handbook of model checking*, pp. 1001–1046, 2018.
- [20] H. Abbas, Y. V. Pant, and R. Mangharam, “Temporal logic robustness for general signal classes,” in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, 2019, pp. 45–56.
- [21] C.-I. Vasile, V. Raman, and S. Karaman, “Sampling-based synthesis of maximally-satisfying controllers for temporal logic specifications,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3840–3847.
- [22] A. Donzé, “Breach, a toolbox for verification and parameter synthesis of hybrid systems,” in *Computer Aided Verification*, T. Touili, B. Cook, and P. Jackson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 167–170.
- [23] A. Donzé, Decyphir et. al. Stlom. [Online]. Available: <https://github.com/decyphir/STLRob>
- [24] J. Jankowski, L. Bruder Müller, N. Hawes, and S. Calinon, “Vp-sto: Via-point-based stochastic trajectory optimization for reactive robot behavior,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 10 125–10 131.
- [25] N. Hansen, “The cma evolution strategy: A tutorial.”
- [26] A. Donzé, T. Ferrere, and O. Maler, “Efficient robust monitoring for stl,” in *International conference on computer aided verification*. Springer, 2013, pp. 264–279.
- [27] J. V. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. A. Seshia, “Robust online monitoring of signal temporal logic,” *Formal Methods in System Design*, vol. 51, pp. 5–30, 2017.
- [28] R. Ilyes, L. Bruder Müller, N. Hawes, and B. Lacerda, “Receding horizon control for signal temporal logic using robustness-conserving partial formula evaluation,” *IEEE Robotics and Automation Letters*, vol. 10, no. 10, pp. 10 775–10 782, 2025.

APPENDIX I PROOFS

All proofs assume (i) STL is interpreted using pointwise semantics, (ii) time intervals I are nonnegative, and (iii) sampling time $\Delta = t_{k+1} - t_k > 0$.

A. Theorem 1

A signal s satisfies an STL formula if and only if it has a positive robustness to go, i.e.

$$(s, t) \models \varphi \iff \rho^{\text{rt}}(s, t, \hat{t}, \varphi) > 0.$$

Proof: We provide a proof sketch. The result holds by induction on the structure of STL. The base case of $(s, t) \models \top \iff \rho^{\text{rt}}(s, t, \hat{t}, \top)$ is trivially true. The base case of $(s, t) \models g(\mathbf{x}) > 0 \iff \rho^{\text{rt}}(s, t, \hat{t}, g(\mathbf{x})) > 0$ is true if and only if $g(s(t)) > 0$, regardless of whether $t > \hat{t}$. The proof for the inductive cases of negation, conjunction, and until follow from the analogous proof for traditional robustness, i.e., $(s, t) \models \varphi \iff \rho(s, t, \varphi) > 0$. This is because the Ro-To-Go of these operators according to Definition 6 are structurally identical to the robustness of these operators according to Definition 4. ■

B. Theorem 2

Lemma 1: When $\hat{t} < t_k$, robustness-to-go is equal to robustness, i.e.,

$$\hat{t} < t_k \rightarrow \rho^{\text{rt}}(s, t_k, \hat{t}, \varphi) = \rho(s, t_k, \varphi).$$

Proof: We provide a proof sketch. The result holds by induction on the structure of STL. The base case of $\hat{t} < t_k \rightarrow \rho^{\text{rt}}(s, t_k, \hat{t}, \top) = \rho(s, t_k, \top)$ is trivially true. The base case of $\hat{t} < t_k \rightarrow \rho^{\text{rt}}(s, t_k, \hat{t}, g(\mathbf{x})) > 0 = \rho(s, t_k, g(\mathbf{x})) > 0$ is true according to Definitions 4 and 6 because $\hat{t} < t_k$. The inductive cases for negation, conjunction, and until are straightforward because of the identical structure according to Definitions 4 and 6. For negation and conjunction, the inductive hypothesis holds because $\hat{t} < t_k$ for the child operators. The inductive hypothesis for the until operator also holds because $t', t'' > t_k$ in Definition 6 for nonnegative intervals I , so $\hat{t} < t_k \rightarrow \hat{t} < t', t''$. ■

Lemma 2: When $\hat{t} = t_k$, robustness-to-go is equal to robustness of the progressed formula, i.e.,

$$\hat{t} = t_k \rightarrow \rho^{\text{rt}}(s, t_k, \hat{t}, \varphi) = \rho(s, t_{k+1}, P(\varphi, t_{k+1} - t_k, s(t_k))).$$

Proof: We provide a proof sketch. The result holds by induction on the structure of STL. The base case of $\hat{t} = t_k \rightarrow \rho^{\text{qtr}}(s, t_k, \hat{t}, \top) = \rho(s, t_{k+1}, P(\top, t_{k+1} - t_k, s(t_k)))$ is trivially true. According to Definitions 3, 4, and 6, the base case of $\hat{t} = t_k \rightarrow \rho^{\text{qtr}}(s, t_k, \hat{t}, g(\mathbf{x}) > 0) = \rho(s, t_{k+1}, P(g(\mathbf{x}) > 0, t_{k+1} - t_k, s(t_k)))$ is true because both resolve to $\infty \cdot \text{sign}(g(s(t)))$ when $\hat{t} = t_k$. The inductive cases for negation and conjunction follow from the shared structure of these operations according to Definitions 3, 4, and 6, and the inductive hypothesis holds because $\hat{t} < t_k$ for the child operators. The inductive case for the until operation relies on leveraging the pointwise semantics to separate the atomic contribution of the point t_k from the Ro-To-Go, that is,

$$\rho^{\text{qtr}}(s, t_k, \hat{t}, \varphi U_I \psi) = \begin{cases} \max(A, \min(B, C)) & \text{if } 0 \in I \\ \min(B, C) & \text{otherwise,} \end{cases} \quad (1)$$

where

$$A = \rho^{\text{qtr}}(s, t_k, \hat{t}, \psi) \quad (2)$$

$$B = \rho^{\text{qtr}}(s, t_k, \hat{t}, \varphi) \quad (3)$$

$$C = \rho^{\text{qtr}}(s, t_{k+1}, \hat{t}, \varphi U_{I_{(t_{k+1}-t_k)}}^{\leftarrow} \psi). \quad (4)$$

By the induction hypothesis, Equations 2 and 3 are equivalent to $\rho(s, t_{k+1}, P(\psi, t_{k+1} - t_k, s(t_k)))$ and $\rho(s, t_{k+1}, P(\varphi, t_{k+1} - t_k, s(t_k)))$, respectively, because $\hat{t} = t_k$, and Equation 4 becomes $\rho(s, t_{k+1}, \varphi U_{I_{(t_{k+1}-t_k)}}^{\leftarrow} \psi)$ due to Lemma 1 because $t_{k+1} > t_k$ and $t_k = \hat{t}$ imply $t_{k+1} > \hat{t}$. According to Definitions 4 and 3, Equation 1 is therefore equivalent to $\rho(s, t_{k+1}, P(\varphi U_I \psi, t_{k+1} - t_k, s(t_k)))$. ■

Lemma 3: When $\hat{t} > t_k$, robustness-to-go is equal to robustness-to-go of the progressed formula, i.e.,

$$\hat{t} > t_k \rightarrow \rho^{\text{qtr}}(s, t_k, \hat{t}, \varphi) = \rho^{\text{qtr}}(s, t_{k+1}, \hat{t}, P(\varphi, t_{k+1} - t_k, s(t_k))).$$

Proof: We provide a proof sketch. The result holds by induction on the structure of STL. The base case of $\hat{t} > t_k \rightarrow \rho^{\text{qtr}}(s, t_k, \hat{t}, \top) = \rho^{\text{qtr}}(s, t_{k+1}, \hat{t}, P(\top, t_{k+1} - t_k, s(t_k)))$ is trivially true. The base case of $\hat{t} > t_k \rightarrow \rho^{\text{qtr}}(s, t_k, \hat{t}, \top) = \rho^{\text{qtr}}(s, t_{k+1}, \hat{t}, P(\top, t_{k+1} - t_k, s(t_k)))$ follows from Definitions 3 and 6 for the case that $\hat{t} > t_k$. The inductive cases for negation and conjunction follow from the shared structure of these operations according to Definitions 3 and 6, and the inductive hypothesis holds because $\hat{t} > t_k$ for the child operators. Similar to the proof of Lemma 2, the inductive case for the until operation relies on separating the atomic contribution of the point t_k from the Ro-To-Go, resulting in Equation 1. By inductive hypothesis, Equations 2 and 3 are equivalent to $\rho^{\text{qtr}}(s, t_{k+1}, \hat{t}, P(\psi, t_{k+1} - t_k, s(t_k)))$ and $\rho^{\text{qtr}}(s, t_{k+1}, \hat{t}, P(\varphi, t_{k+1} - t_k, s(t_k)))$, respectively, because $\hat{t} > t_k$. According to Definitions 3 and 6, Equation 1 is therefore equivalent to $\rho^{\text{qtr}}(s, t_{k+1}, \hat{t}, P(\varphi U_I \psi, t_{k+1} - t_k, s(t_k)))$. ■

Lemma 4: Let φ_0 be an STL formula, s be a signal, and consider sample $(t_i, s(t_i))$. Let $\Phi = \{\varphi_0, \varphi, \dots, \varphi_{i+1}\}$ be a set of STL formulas such that $\varphi_k = P(\varphi_{k-1}, t_k - t_{k-1}, s(t_{k-1}))$ for each $0 < k \leq i + 1$. Then,

$$t_0 < t_k \leq t_i \rightarrow \rho^{\text{qtr}}(s, t_k, t_i, \varphi_k) = \rho^{\text{qtr}}(s, t_0, t_i, \varphi_0).$$

Proof: We provide a proof sketch. The result holds by induction of Lemma 3 through time. Because $t_k \leq t_i$ and $t_{k-1} < t_k$, we know $t_{k-1} < t_i$. From Lemma 3, this means that $\rho^{\text{qtr}}(s, t_k, t_i, \varphi_k) = \rho^{\text{qtr}}(s, t_{k-1}, t_i, \varphi_{k-1})$. The base case is $t_1 = t_k \leq t_i$, meaning $\rho^{\text{qtr}}(s, t_k, t_i, \varphi_k) = \rho^{\text{qtr}}(s, t_0, t_i, \varphi_0)$. In the inductive case, $t_1 < t_k \leq t_i$, implying that $t_0 < t_{k-1} < t_i$. This allows for application of the inductive hypothesis, meaning $\rho^{\text{qtr}}(s, t_{k-1}, t_i, \varphi_{k-1}) = \rho^{\text{qtr}}(s, t_0, t_i, \varphi_0)$. Accordingly, $\rho^{\text{qtr}}(s, t_k, t_i, \varphi_k) = \rho^{\text{qtr}}(s, t_{k-1}, t_i, \varphi_{k-1}) = \rho^{\text{qtr}}(s, t_0, t_i, \varphi_0)$ for all $0 < k \leq i$. ■

We now turn our attention to Theorem 2:

Let φ_0 be an STL formula, s be a signal, and consider sample $(t_i, s(t_i))$. Let $\Phi = \{\varphi_0, \varphi, \dots, \varphi_{i+1}\}$ be a set of STL formulas such that $\varphi_k = P(\varphi_{k-1}, t_k - t_{k-1}, s(t_{k-1}))$ for each $0 < k \leq i + 1$. Then,

$$t_i \geq t_0 \rightarrow \rho(s, t_{i+1}, \varphi_{i+1}) = \rho^{\text{qtr}}(s, t_0, t_i, \varphi_0).$$

Proof: From Lemma 2, $\rho(s, t_{i+1}, \varphi_{i+1}) = \rho^{\text{qtr}}(s, t_0, t_i, \varphi_0)$ when $t_i = t_0$. From Lemma 4, $\rho^{\text{qtr}}(s, t_0, t_i, \varphi_0) = \rho^{\text{qtr}}(s, t_k, t_i, \varphi_k)$ for any $t_0 < t_k \leq t_i$ when $t_i > t_0$. If we chose $t_k = t_i$, $\rho^{\text{qtr}}(s, t_0, t_i, \varphi_0) = \rho^{\text{qtr}}(s, t_i, t_i, \varphi_i)$, and, from Lemma 2, $\rho^{\text{qtr}}(s, t_i, t_i, \varphi_i) = \rho(s, t_{i+1}, \varphi_{i+1})$. ■

APPENDIX II HYPERPARAMETERS

Parameter	Value
VP-STO degrees of freedom	2
VP-STO velocity limits	2 m/s
VP-STO acceleration limits	2 m/s ²
VP-STO number of via points	4
VP-STO population size	10
VP-STO CMA-ES initial variance	10 m ²
VP-STO maximum CMA-ES iterations	50
X domain lower bound	0 m
X domain upper bound	5 m
Y domain lower bound	0 m
Y domain upper bound	5 m
Cost penalty for violating domain bounds	1e8
Planning Horizon	20 s

TABLE II: Experiment hyperparameters.