2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Sampling-based synthesis of maximally-satisfying controllers for temporal logic specifications

Cristian-Ioan Vasile[1] and Vasumathi Raman[2] and Sertac Karaman[1]

Abstract— **Sampling-based methods have advanced the state of the art in robotic motion planning and control across complex, high-dimensional domains. With few exceptions, such approaches only admit simple constraints and objectives, such as collision-avoidance and reaching a goal state. In this work we leverage the best of two worlds: the scalability of sampling-based motion planning and the precise formal guarantees of temporal logic. We present an incremental sampling-based algorithm that synthesizes a motion control policy satisfying a bounded Signal Temporal Logic formula over properties of a given environment. Our key insight is that we can bias the selection of samples using a quantitative measure of how well the best path in the current tree of samples satisfies the specification. This allows us both to converge to a path that satisfies the specification, and to improve upon an existing path, i.e. to satisfy the specification with maximum robustness. We illustrate the performance of our method in several case studies.**

I. INTRODUCTION

Sampling-based methods have advanced the state of the art in robotic motion planning across complex and high-dimensional domains. The goal is usually to find a controller that drives the robot from a start state to a goal state. However, as robots become more capable and versatile, the tasks we assign to them become more complex, and may be better specified using richer formalisms such as domain-specific languages, finite state machines, and temporal logic. Motivated by the burgeoning complexity of motion tasks assigned to robots, in this work, we apply sampling-based methods to produce controllers that solve temporal logic planning problems.

Temporal logic specifications are expressive and precise formalisms for describing desirable properties of a system that evolves over time.[1], [2], [3]. There exist rich theory and practical tools for *synthesis* from temporal logics, and recent research in robotics and control has brought these methods to bear on motion planning and control for a variety of systems across several domains [4], [5], [6], [7], [8], [9], [10]. These works leverage a variety of temporal logics, including Linear Temporal Logic (LTL) [3], Metric Temporal Logic (MTL) [11], Signal Temporal Logic (STL) [12], Time Window Temporal Logic (TWTL) [13], and syntactically co-safe LTL (scLTL) [14]. In this work, we focus on the problem of generating a control policy such that a system satisfies a Signal Temporal Logic (STL) specification.

A key advantage of STL as a specification language is that there is also rich body of work on *quantitatively* monitoring the satisfaction of STL formulae on system behavior in hybrid (discrete and continuous) domains, for both discrete and continuous time semantics [12], [15]. The idea is to be able to assess not just *whether* an execution satisfies the desired properties, but *how much* of the property is satisfied (or violated). For STL, such quantitative semantics have been successfully used for optimization-based synthesis [16], and methods developed for efficiently assessing satisfaction over system behavior in both offline [17] and online [18] settings.

Our contribution in this paper is the definition of a class of specification-based heuristics, and their use in sampling-based control synthesis for temporal logic specifications. These heuristics are not just helpful for finding an initial solution quickly, but can also be applied to improve upon an existing solution. More precisely, given a formula, we define the *Direction of Increasing Satisfaction (DIS)*, and propose a class of heuristic functions based on the DIS to capture the most promising direction of exploration to improve the quantitative satisfaction given a partial trajectory. The role of these heuristics is to provide a gradient-like information for exploration algorithms, which may be of independent interest beyond the use in this paper. For instance, these may be used in conjunction with optimization-based planning and learning algorithms to speed them up. We show that our algorithm is asymptotically optimal: in the limit of sampling it yields a solution that maximizes quantitative satisfaction of the temporal logic formula. In addition to the theoretical results, we demonstrate the effectiveness of our approach experimentally.

Sampling-based methods for planning have already proven useful in a variety of contexts, including temporal logic controller synthesis. Extensions of Rapidly Exploring Random Trees (RRT) [19] and their optimal version, RRT*[20], have been proposed to incorporate specifications in μ- calculus [21], [22] and LTL [23], [24]. Probabilistic Roadmaps (PRMs) [25] have also been exploited for temporal logic synthesis [26], [27]. However, ours is the first approach to explicitly use bounds on the quantitative satisfaction of a formal specification as a heuristic to guide sampling. Additionally, since we use STL, unlike previous approaches, we do not need an expensive discrete abstraction of the system in order to evaluate satisfaction of the specification.

Another related body of work is that of automata or language-guided synthesis [28], [29], where an automaton representing a temporal logic specification is used to guide a sequence of reachability problems, resulting in a hybrid solution to the motion planning problem. Our approach is similarly

*This work was partially supported by NSF grant no. 1350685.

[1] Cristian-Ioan Vasile and Sertac Karaman are with the Massachusetts Institute of Technology, Cambridge, MA, USA {cvasile, sertac}@mit.edu

[2] http://vraman.github.io, vasu@cds.caltech.edu

978-1-5386-2682-5/17/$31.00 ©2017 IEEE

guided by a temporal logic specification, but we avoid the construction of an explicit automaton, and instead rely on the quantitative semantics of the logic to guide sampling.

Our approach is for general dynamical systems and STL specifications, but inspired by robotic motion planning tasks. We therefore demonstrate its applicability with case studies on a double integrator and a rear wheel car.

II. PRELIMINARIES

Let $\mathbb{R}$ be the set of real numbers and $t \in \mathbb{R}$. We denote the interval $[t, \infty)$ by $\mathbb{R}_{\geq t}$. Given interval $I = [a, b]$, the interval $[t + a, t + b]$ is denoted by $t + I$. The uniform and Bernoulli distributions are denoted by $Unif(S)$ and $Ber(p)$, respectively, where S is a set and p is a bias. Missing values in algorithms are denoted by $\bowtie$.

Let (M, d) be a compact metric space with $M \subset \mathbb{R}^n$, $n \geq 1$, and $\mathcal{S} = \{s : \mathbb{R}_{\geq 0} \to M\}$ the set of all infinite-time signals in M. The components of a signal $s \in \mathcal{S}$ are denoted by s_i, $i \in \{1, \ldots, n\}$. The set of all linear functions over $\mathbb{R}^n$ is denoted by $\mathcal{F} = \{\pi : \mathbb{R}^n \to \mathbb{R}\}$. Let $\delta_\tau : \mathcal{S} \to \mathcal{S}$ be the time-shift operator acting on signals in $\mathcal{S}$ with $\tau \geq 0$, i.e., $\delta_\tau s(t) = s(t + \tau)$ for all $t \geq 0$.

The syntax of STL is defined as follows [12]:

$$\phi ::= \top \mid p_{\pi(x)\sim\mu} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \mathcal{U}_{\langle a,b \rangle} \phi_2 \ ,$$

where $\top$ is the Boolean *true* constant; $p_{\pi(x)\sim\mu}$ is a predicate over $\mathbb{R}^n$ parameterized by $\pi \in \mathcal{F}$, $\mu \in \mathbb{R}$ and an order relation $\sim \in \{\geq, >, \leq, <\}$ of the form $p_{\pi(x)\sim\mu} = \pi(x) \sim \mu$; $\neg$ and $\wedge$ are the Boolean operators for negation and conjunction, respectively; and $\mathcal{U}_{\langle a,b \rangle}$ is the bounded temporal operator *until*, where $\langle \in \{ [, (\}$ and $\rangle \in \{],) \}$.

The semantics of STL is defined over signals in $\mathcal{S}$ recursively as follows [12]:

$$\begin{aligned}
&s \models \top && \Leftrightarrow \top \\
&s \models p_{\pi(x)\sim\mu} && \Leftrightarrow \pi(s(0)) \sim \mu \\
&s \models \neg\phi && \Leftrightarrow \neg(s \models \phi) \\
&s \models (\phi_1 \wedge \phi_2) && \Leftrightarrow (s \models \phi_1) \wedge (s \models \phi_2) \\
&s \models (\phi_1 \mathcal{U}_{\langle a,b \rangle} \phi_2) && \Leftrightarrow \exists t_u \in [a, b) \text{ s.t. } \big(\delta_{t_u} s \models \phi_2\big) \\
& && \quad \wedge \big(\forall t' \in [0, t_u)\ \delta_{t'} s \models \phi_1\big)
\end{aligned}$$

where $a, b \in \mathbb{R}$, $a < b$. A signal $s \in \mathcal{S}$ is said to satisfy an STL formula ϕ if and only if $s \models \phi$. The Boolean value *false* $\perp \equiv \neg\top$ and additional operations (i.e., disjunction, implication, and equivalence) are defined in the usual way. Also, the temporal operators *eventually* and *globally* are defined as $\diamondsuit_{\langle a,b \rangle}\phi \equiv \top\, \mathcal{U}_{\langle a,b \rangle}\phi$ and $\Box_{\langle a,b \rangle}\phi \equiv \neg\diamondsuit_{\langle a,b \rangle}\neg\phi$, respectively.

The *Abstract Syntax Tree* (AST) of an STL formula is a tree with predicates for leaves and operators for internal nodes, such that its inorder traversal yields the formula. Fig. 1 shows the AST for the formula $(\Box_{[t_1,t_2]} p_{\pi_1(x)\sim\mu_1} \wedge \Box_{[t_3,t_4]} p_{\pi_2(x)\sim\mu_2}) \Rightarrow \diamondsuit_{[0,t_5]} p_{\pi_3(x)\sim\mu_3}$

The *language* associated with an STL formula ϕ is the set of all signals in $\mathcal{S}$ that satisfy ϕ. This language is denoted by $\mathcal{L}(\phi) = \{s \in \mathcal{S} \mid s \models \phi\}$.

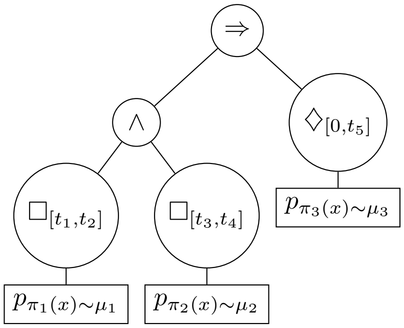

Fig. 1. Example of an Abstract Syntax Tree

The time horizon of an STL formula [30] is defined as

$$\|\phi\| = \begin{cases} 0 & \text{if } \phi = p_{\pi(x)\sim\mu} \\ \max\{\|\phi_1\|, \|\phi_2\|\} & \text{if } \phi \in \{\phi_1 \wedge \phi_2, \phi_1 \vee \phi_2\} \\ \|\phi_1\| & \text{if } \phi = \neg\phi_1 \\ b + \max\{\|\phi_1\|, \|\phi_2\|\} & \text{if } \phi = \phi_1 \mathcal{U}_{\langle a,b \rangle} \phi_2 \\ b + \|\phi_1\| & \text{if } \phi \in \{\diamondsuit_{\langle a,b \rangle}\phi_1, \Box_{\langle a,b \rangle}\phi_1\} \end{cases}$$

An STL formula ϕ is *bounded-time* if it has a finite time horizon.

Let ϕ be an STL formula. The set of all predicates in ϕ is denoted by $\mathfrak{P}(\phi)$, i.e., $\mathfrak{P}(\phi_1) = \{p_{u_1 \leq 2}, p_{u_2 > 3}\}$.

In addition to Boolean semantics, STL admits *quantitative semantics* [31], [15], which is formalized by the notion of *robustness degree*. The robustness degree of a signal $s \in \mathcal{S}$ with respect to an STL formula ϕ is a functional $\rho(s, \phi)$ recursively defines as

$$\begin{aligned}
&\rho(s, \top) && = \rho_\top \\
&\rho(s, \perp) && = -\rho_\top \\
&\rho(s, p_{\pi(x)\sim\mu}) && = (-1)^\iota(\pi(s(0)) - \mu) \\
&\rho(s, \neg\phi) && = -\rho(s, \phi) \\
&\rho(s, \phi_1 \wedge \phi_2) && = \min\{\rho(s, \phi_1), \rho(s, \phi_2)\} \\
&\rho(s, \phi_1 \vee \phi_2) && = \max\{\rho(s, \phi_1), \rho(s, \phi_2)\} \\
&\rho(s, \phi_1 \mathcal{U}_{\langle a,b \rangle} \phi_2) && = \sup_{t_u \in \langle a,b \rangle} \Big\{ \min \big\{ \rho(\delta_{t_u} s, \phi_2), \\
& && \qquad \inf_{t' \in [0,t_u)} \{\rho(\delta_{t'} s, \phi_1)\}\big\}\Big\} \\
&\rho(s, \diamondsuit_{\langle a,b \rangle}\phi) && = \sup_{t_u \in \langle a,b \rangle} \{\rho(\delta_{t_u} s, \phi)\} \\
&\rho(s, \Box_{\langle a,b \rangle}\phi) && = \inf_{t_u \in \langle a,b \rangle} \{\rho(\delta_{t_u} s, \phi)\}
\end{aligned}$$

where $\rho_\top \in \mathbb{R}_{\geq 0} \cup \{\infty\}$ is a large constant representing the maximum absolute value of robustness ($|\rho(s, \phi)| < \rho_\top$), and $\iota = 0$ if $\sim \in \{\geq, >\}$ and $\iota = 1$ otherwise. Note that a positive $\rho(s, \phi)$ implies that s satisfies ϕ. Moreover, the interpretation of the robustness degree as a quantitative measure of satisfaction is justified by the following proposition from [31], [17].

Proposition 2.1: Let $s \in \mathcal{S}$ be a signal and ϕ an STL formula such that $\rho(s, \phi) > 0$. All signals $s' \in \mathcal{S}$ with $\|s' - s\|_\infty < \rho(s, \phi)$ satisfy the formula ϕ, i.e., $s' \models \phi$ and $\rho(s', \rho) > 0$.

In an online setting, where the signal is being observed as it is generated, it is useful to assign a quantitative satisfaction value to partial signals. The authors in [18] introduced an interval-based semantics for bounding the quantitative satisfaction value given only a partial signal. This recursively-defined Robust Satisfaction Interval (RoSI) includes all possible robust satisfaction values corresponding to the suffixes of the partial signal.

Definition 2.1 (Prefix, Completions): Given a finite time horizon T_H, let $\{t_0, \cdots, t_i\}$ be a finite set of time instants such that $t_i \leq T_H$. Let $s_{[0,i]}$ denote a partial signal over the time domain $[t_0, t_i] \subset \mathbb{R}_{\geq 0}$, i.e. $s_{[0,i]} : [t_0, t_i] \rightarrow M$. Then $s_{[0,i]}$ is a *prefix* of a signal s if for all $t \leq t_i$, $s(t) = s_{[0,i]}(t)$. The set of *completions* of a partial signal $s_{[0,i]}$ (denoted by $C(s_{[0,i]})$) is defined as the set $\{s | s_{[0,i]} \text{ is a prefix of } s\}$.

Definition 2.2 (Robust Satisfaction Interval (RoSI)): The *robust satisfaction interval* (also called simply the *robustness interval*) of an STL formula ϕ on a partial signal $s_{[0,i]}$ is an interval I such that:

$$\begin{array}{c} \inf(I) = \inf_{s' \in C(s_{[0,i]})} \rho(s', \phi) \\ \sup(I) = \sup_{s' \in C(s_{[0,i]})} \rho(s', \phi) \end{array}$$

We first define some interval operations as follows:

$$\begin{array}{lcl} -[a,b] & = & [-b, -a] \\ \min([a_1, b_1], [a_2, b_2]) & = & [\min(a_1, a_2), \min(b_1, b_2)] \\ \max([a_1, b_1], [a_2, b_2]) & = & [\max(a_1, a_2), \max(b_1, b_2)] \\ c + [a, b] & = & [c + a, c + b] \end{array}$$

We now define a recursive function $[\rho]$ that maps a given formula ϕ and a partial signal $s_{[0,i]}$ to an interval $[\rho](s_{[0,i]}, \phi)$. We first introduce a new symbol $\dagger \notin M$, and for every $s_{[0,i]} : [t_0, t_i] \rightarrow M$, define $s^{\omega}_{[0,i]} : \mathbb{R}_{\geq t_0} \rightarrow M \cup \{\dagger\}$ such that $s^{\omega}_{[0:i]} = s_{[0:i]}$ on the domain $[t_0, t_i]$ and $s^{\omega}_{[0:i]}(t) = \dagger$ for $t > t_i$. Then for any STL formula ϕ, $[\rho](s_{[0,i]}, \phi) = [\rho](s^{\omega}_{[0,i]}, \phi)$, where

$$\begin{array}{lcl} [\rho](s, p_{\pi(x)\sim\mu}) & = & \begin{cases} [-\rho_{\top}, \rho_{\top}] & \text{if } s(0) = \dagger \\ [\rho(s, p_{\pi(x)\sim\mu}), & \text{otherwise} \\ \qquad \rho(s, p_{\pi(x)\sim\mu})] & \end{cases} \\ [\rho](s, \neg\phi) & = & -[\rho](s, \phi) \\ [\rho](s, \varphi \wedge \psi) & = & \min([\rho](s, \varphi), [\rho](s, \psi)) \\ [\rho](s, \varphi \vee \psi) & = & \max([\rho](s, \varphi), [\rho](s, \psi)) \\ [\rho](s, \Diamond_{\langle a,b \rangle} \varphi) & = & \sup_{t \in \langle a,b \rangle} [\rho](\delta_t s, \varphi) \\ [\rho](s, \Box_{\langle a,b \rangle} \varphi) & = & \inf_{t \in \langle a,b \rangle} [\rho](\delta_t s, \varphi) \\ [\rho](s, \phi\, \mathcal{U}_{\langle a,b \rangle}\, \psi) & = & \sup_{t_u \in \langle a,b \rangle} (\min([\rho](\delta_{t_u} s, \psi), \\ & & \quad \inf_{t' \in [0, t_u)} [\rho](\delta_{t'} s, \phi) \end{array}$$

Lemma 2.2 ([18]): For any STL formula ϕ, the function $[\rho](s^{\omega}_{[0,i]}, \phi)$ defines the robust satisfaction interval for the formula ϕ over the signal $s_{[0,i]}$ at time t_0.

The proof is by induction over the structure of STL formulae.

The authors in [18] propose an efficient algorithm to compute and maintain $[\rho](s_{[0,i]}, \phi)$ for a large class of STL formulae.

Lemma 2.3 (Chain Inclusion of Intervals): Given a partial signal $s_{[0,i]}$, STL formula ϕ, and $j \leq i$, $[\rho](s^{\omega}_{[0,i]}, \phi) \subseteq [\rho](s^{\omega}_{[0,j]}, \phi)$.

Informally, as more of the signal becomes available, the bounds on the robustness of satisfaction shrink (i.e. the robustness of satisfaction becomes more certain).

III. PROBLEM FORMULATION

Let $\mathcal{R} = (f, X, U, x_{init})$ be a dynamical system, where $X \subseteq \mathbb{R}^n$ and $U \subseteq \mathbb{R}^m$ are the state and control spaces, $f : X \times U \rightarrow X$ is a Lipschitz continuous function, and x_{init} is the initial state of the system. The system behavior is given by:

$$\mathcal{R} : \dot{x} = f(x, u), \; x(0) = x_{init} \tag{1}$$

We denote by $\mathbf{x}[x_{init}, u]$ the state trajectory originating at x_{init} obtained by implementing control policy u. Let $\mathcal{U} = \{u : \mathbb{R}_{\geq 0} \rightarrow U\}$ be the set of all control policies.

The system $\mathcal{R}$ is said to satisfy an STL specification ϕ under a control policy $u \in \mathcal{U}$ if the state trajectory starting at x_0 satisfies ϕ, i.e., $\mathbf{x}[x_{init}, u] \models \phi$.

Problem 3.1: Given a dynamical system $\mathcal{R}$ and an STL specification ϕ, find a control policy u such that the system satisfies ϕ under policy u.

Following [32], Problem 3.1 can be restated in terms of robustness as an optimization problem, as follows

Problem 3.2: Given a dynamical system $\mathcal{R}$ and an STL specification ϕ, find a control policy u such that the robustness degree with respect to ϕ of the state trajectory originating in x_{init} under u is maximized, i.e.,

$$u^* = \operatorname*{argmax}_{u \in \mathcal{U}} \rho(\mathbf{x}[x_{init}, u], \phi). \tag{2}$$

Note that if $\rho(\mathbf{x}[x_{init}, u^*]) > 0$, then ϕ is satisfied.

IV. APPROACH

In this section, we propose a sampling-based algorithm that computes a maximally-satisfying control policy. The RRT*-based algorithm generates a tree of state-formula pairs, where the states are randomly generated and the associated STL formulae capture the progress towards satisfaction of the overall STL specification. The algorithm is guided towards maximal satisfaction using two methods: *biased sampling* and *guided steering*. A space-time sampling procedure is defined that is biased towards a set of pairwise non-mutually exclusive *active* predicates at the randomly generated time. Informally, a predicate is active at a time t if its value may influence the robustness interval of trajectories at time t. For sampling, we consider a subset of the active predicates that together induce a non-empty region in the state space. To guide the steering of the system, we propose a class of heuristic functions that capture the *Direction of Increasing Satisfaction (DIS)*. The algorithm blends the stochastic search power of RRT* with targeted (greedy) heuristics, by steering the system towards a random convex combination between randomly generated states and states along the DIS. Finally, in order to maintain the incremental property of the sampling-based algorithm, we employ an online monitoring algorithm for STL formulae [18].

A. Algorithm

The proposed sampling-based algorithm is given in Alg. 1. It takes as input a dynamical system $\mathcal{R}$ and an STL specification ϕ, and returns an open-loop control policy that induces a satisfying trajectory, if one exists. Otherwise, the algorithm stops after the maximum number of iterations N^{max}. In the following, we assume that the specification ϕ is in positive normal form (i.e., without any negation operators), which is not restrictive because any STL formula can be put in this form [33]. *We also assume that all predicates are linear.* This admits a rich set of specifications in the motion planning domain, since many regions of the workspace, be they obstacles or goal regions, can be expressed as unions of polyhedra using conjunctions and disjunctions over linear halfspace predicates.

The tree $\mathcal{T} = (V, E)$ generated by Alg. 1 has tuples of states and formulae as vertices. We denote the (unique) parent and the set of children of a vertex $v \in V$ by $pa(v)$ and $ch(v)$, respectively. Each vertex of $v \in V$ is annotated with a time $time_{\mathcal{T}}(v)$, the control value $control_{\mathcal{T}}(v)$ used to steer the system from $pa(v)$ to v, the state trajectory $traj_{\mathcal{T}}(v)$ induced by the path from the root of $\mathcal{T}$ to v, and RoSI $rosi_{\mathcal{T}}(v)$ associated with $traj_{\mathcal{T}}(v)$. In the following, we assume tacitly that all these values are set when a vertex is added to $\mathcal{T}$.

Alg. 1 starts by initializing the RRT* tree with the pair of initial state x_{init} and specification ϕ (lines 1-2). If the top-level operation of ϕ is a disjunction, i.e., the root of the AST of ϕ, then procedure *initialize()* (line 2) generates multiple root vertices in $\mathcal{T}$ corresponding to each subformula of the disjunction. Note that if there are multiple initial states, then $\mathcal{T}$ might be a forest instead of a tree, but this situation does not pose problems since we can add a virtual root (either disjunctive or conjunctive depending on whether we can choose the initial state) to make $\mathcal{T}$ a tree.

In each iteration (lines 3-18), Alg. 1 randomly samples a time and state using the *sampling()* procedure (line 4); attempts to connect a *near* vertex of $\mathcal{T}$ to a new state x^* that minimizes a convex combination of random sampling and moving along the DIS (lines 5-14); updates $\mathcal{T}$ and associated RoSI of the new vertex v^* using *update()* (line 15); and, finally, performs a rewiring of the near vertices using v^* as a parent (lines 16-18). Connecting a new vertex to the tree $\mathcal{T}$ (lines 5-14) involves the following steps: 1) the set of near vertices $\mathcal{N}$ to the random state x^r and time t^r is computed (line 5), 2) a uniformly distributed convex coefficient is generated (line 6), 3) for each vertex $v' \in \mathcal{N}$ the optimal state x^s and cost J^s with respect to $J_\chi()$ are computed (line 10-12), and the vertex with minimum cost is selected as the parent of x^* (line 14) if the system can steer to it (line 13). We compute the duration a control value $u'' \in U$ is applied as the difference between the random time t^r and the time $time_{\mathcal{T}}(v')$ associated with vertex $v' \in \mathcal{N}$ (line 9). Thus, all vertices in $\mathcal{N}$ must respect the causality constraint, i.e., their associated times must be less than t^r, see Sec. IV-B for details.

Finally, after N^{max} iterations if there exists a solution (line 19), the algorithm return a piecewise-constant (*pwc*) control policy (line 21) obtained from the best path in the RRT* tree $\mathcal{T}$ (line 20), i.e., $best(\mathcal{T})$. A solution exists if there is a vertex $v \in V$ with $rosi_{\mathcal{T}}(v) = \{\rho\}$ and $\rho > 0$.

B. Biased space-time sampling

In this section, we describe a space-time sampling procedure biased based on an STL specification, and a modified query function for near vertices in $\mathcal{T}$ that ensures causality.

Algorithm 1: Algorithm

```
   Input: $\mathcal{R} = (f, X, U, x_{init})$ – dynamical system
   Input: $\phi$ – STL formula in positive normal form
   Output: $\mathbf{u}$ – a satisfying control policy w.r.t. $\phi$
1  $\mathcal{T} = (V = \emptyset, E = \emptyset)$
2  $V \leftarrow initialize(x_{init}, \phi)$
3  for $k = 1 : N^{max}$ do
4      $t^r, x^r \leftarrow sample(X, \mathcal{T}, \phi)$
5      $\mathcal{N} \leftarrow near(\mathcal{T}, x^r, t^r)$
6      $\lambda \leftarrow Unif([0, 1])$
7      $v^{pa} \leftarrow \bowtie$, $J^* \leftarrow \infty$, $x^* \leftarrow \bowtie$
8      foreach $v' = (x', \phi') \in \mathcal{N}$ do
9          $\Delta t^r = t^r - time_{\mathcal{T}}(v')$
10         $u^s \leftarrow \mathrm{argmin}_u\, J_\chi(u, \mathbf{x}[x', u](\Delta t^r), x', \phi', x^r; \lambda)$
11         $x^s \leftarrow \mathbf{x}[x', u^s](\Delta t^r)$
12         $J^s \leftarrow J_\chi(u^s, x^s, x', \phi', x^r; \lambda)$
13         if $J^s < J^* \wedge steer(x', x^s)$ then
14             $J^* \leftarrow J^s$, $v^{pa} \leftarrow v'$, $x^* \leftarrow x^s$
15     $update(v^{pa}, (x^*, \bowtie))$
16     for $v'' = (x'', \phi'') \in Near(\mathcal{T}, x^*, t^r)$ do
17         if $steer(v^*, x'')$ then
18             $update(v^*, v'')$
19 if $existsSolution()$ then
20     $[v^0 = v^{init}, \ldots, v^s] = best(\mathcal{T})$
21     return $\mathbf{u} = pwc\big[(control_{\mathcal{T}}(v^i), time_{\mathcal{T}}(v^i))_{i=1}^{s}\big]$
22 else  return $\bowtie$
```

The need for generating both a random time and state stems from the desire to bias the sampling towards promising regions of the state space. Since satisfaction is history-dependent, we need a procedure to resolve conflicting constraints. We choose to generate a random time, because this enables the computation of active predicates.

Let $s_{[0,i]}$ be a partial signal over the time domain $[t_0, t_i] \subset \mathbb{R}_{\geq 0}$. A predicate p is called *active* at time $t_j > t_i$ if there exist two signals $s^1, s^2 \in C(s_{[0,i]})$ such that $\rho(\delta_{t_j} s^1, p) \neq \rho(\delta_{t_j} s^2, p)$ implies $[\rho](s^1_{[0,j]}, \phi) \neq [\rho](s^2_{[0,j]}, \phi)$, otherwise it is called *inactive*. Informally, this means that the robustness of an active predicate p at time t_j determines the RoSI of some signals in $C(s_{[0,i]})$, while inactive predicates do not influence the RoSI of any signal in the completion. Note that activity status of predicates depend on the time bounds associated with temporal operators, hence the need for sampling over time.

Lemma 4.1: Procedure Alg. 3 correctly computes the set of active predicates of a formula ϕ at time $t \geq 0$.

Proof: Follows directly from the semantics of STL. ■

The sampling procedure is described in Alg. 2. First, a time t^r is generated uniformly distributed in $[0, T^{max}_{\mathcal{T}} + T_H]$ (line 1), where $T^{max}_{\mathcal{T}} = \max_{v \in V} time_{\mathcal{T}}(v)$ is the maximum time of any vertex of $\mathcal{T}$ and T_H is the maximum horizon between vertices. Next, the active predicates are computed (line 2), and the conflicts between mutually-exclusive pairs of predicates (i.e., their conjunction is identically false) are resolved by randomly removing one of them (lines 3-5).

Finally, a random state is generated uniformly distributed in the region spanned by the predicates (line 6).

```
Algorithm 2: $sample(X, \mathcal{T}, \phi)$
1 $t^r \leftarrow Unif([0, T_{\mathcal{T}}^{max} + T_H])$
2 $\mathfrak{P}^r \leftarrow active(\phi, t^r)$
3 while $\exists\ p_{\pi_1(x) \leq \mu_1}, p_{\pi_2(x) > \mu_2} \in \mathfrak{P}^r$ s.t. $\mu_1 \leq \mu_2$ do
4     if $Ber(0.5) = 0$ then $\mathfrak{P}^r \leftarrow \mathfrak{P}^r \setminus \{p_{\pi_1(x) \leq \mu_1}\}$
5     else $\mathfrak{P}^r \leftarrow \mathfrak{P}^r \setminus \{p_{\pi_2(x) > \mu_2}\}$
6 $x_r \leftarrow Unif(\{x \in X \mid \bigwedge_{p \in \mathfrak{P}^r} p(x)\})$
7 return $t^r, x^r$
```

```
Algorithm 3: $active(\phi, t)$
if $\phi \in \{\top, \bot\}$ then
    return $\emptyset$
else if $\phi = p_{\pi(x) \sim \mu}$ then
    if $t = 0$ then return $\{p_{\pi(x) \sim \mu}\}$
    else return $\emptyset$
else if $\phi \in \{\phi_1 \wedge \phi_2, \phi_1 \vee \phi_2\}$ then
    return $active(\phi_1, t) \cup active(\phi_2, t)$
else if $\phi \in \{\Diamond_{\langle a,b \rangle} \phi_1, \Box_{\langle a,b \rangle} \phi_2\}$ then
    return $\bigcup_{t' \in \langle a,b \rangle} active(\phi_1, t - t')$
else if $\phi = \phi_1 \mathcal{U}_{\langle a,b \rangle} \phi_2$ then
    return $\left(\bigcup_{t' \in [0,b\rangle} active(\phi_1, t - t')\right) \cup$
    $\left(\bigcup_{t' \in \langle a,b \rangle} active(\phi_2, t - t')\right)$
```

A consequence of sampling space and time is that we need to redefine the *near()* primitive function to take into account causality, i.e., vertices come before t^r. Thus, we have $near(\mathcal{T}, x^r, t^r) = \{v = (x, \phi^x) \in V \mid \|x - x^r\|_2 \leq \gamma(\log(N)/N)^{\frac{1}{n}} \wedge time_{\mathcal{T}}(v) \in [-T_H + t^r, t^r]\}$, where n is the dimension of the state space, $N = |\mathcal{T}| = |V|$, and $\gamma \in \mathbb{R}_{\geq 0}$.

C. Direction of Increasing Satisfaction

We propose a class of heuristic functions that capture the most promising direction to improve the robustness bounds given a partial trajectory. Their role is to provide a gradient-like information for exploration algorithms, which may be of independent interest beyond the use in this paper. For instance, these may be used in conjunction with optimization-based planning and learning algorithms to speed up convergence.

The class of heuristic functions is parameterized by two procedures: *choose()* and *blend()*. The choice function decides which of the two input formulae yields the largest robustness gain for their conjunction. The blending function combines the two directions computed for the two sub-formulae of the conjunction, and should give priority to the one returned by the choice function. Additionally, we require that the blending function satisfies the following *orthogonality* condition:

$$\chi(s, \phi_1, t) \perp \chi(s, \phi_2, t) \Rightarrow blend(\chi(s, \phi_c, t), \chi(s, \phi_{\neg c}, t)) = \sum_{i \in \{c, \neg c\}} \chi(s, \phi_i, t) \quad (3)$$

The *Direction of Increasing Satisfaction (DIS)* is defined as

$$\begin{aligned}
\chi(s, \top, t) &= 0_n \\
\chi(s, p_{\pi(x) \sim \mu}) &= \begin{cases} \dot{x}_i(t) e_i & (-1)^\iota \dot{x}_i(t) > 0 \\ 0_n & \text{otherwise} \end{cases} \\
\chi(s, \neg\phi, t) &= -\chi(s, \phi, t) \\
\chi(s, \phi_1 \wedge \phi_2, t) &= \begin{cases} blend(\chi(s, \phi_c, t), \chi(s, \phi_{\neg c}, t)) \\ c, \neg c = choose(s, t, \phi_1, \phi_2) \end{cases} \\
\chi(s, \phi_1 \mathcal{U}_{[a,b)} \phi_2, t) &= \begin{cases} \chi(s, \phi_1, t) & t < a \\ \chi(s, \phi_1 \wedge \phi_2, t) & t \in [a, b) \\ 0_n & t \geq b \end{cases}
\end{aligned}$$

where $\dot{x}_i(t) e_i = f_i(x(t), u) e_i$ is a vector dependent on the control input u. Thus, the DIS $\chi(s, \phi, t)$ is (as expected) dependent on the applied control input. In the case of the $\mathcal{U}$ operator, we define the DIS for the interval $[a, b)$ above for illustrative purposes: it is defined similarly for any $\langle a, b \rangle$.

In this paper, we consider a stochastic choice function that depends only on robustness intervals associated with the partial trajectory and the two formulae:

$$\begin{aligned}
choose(s, t, \phi_1, \phi_2) &= choose([\rho](\delta_t s, \phi_1), [\rho](\delta_t s, \phi_2)) = \\
&\begin{cases} 1 & \text{if } a_1 < a_2 \wedge b_1 < b_2 \\ 2 & \text{if } a_1 > a_2 \wedge b_1 > b_2 \\ 1 + Ber(p) & \text{otherwise} \end{cases} \\
p &= 0.5 + \frac{(a_1 + b_1) - (a_2 + b_2)}{8\rho_\top}
\end{aligned} \quad (4)$$

where $[a_1, b_1] = [\rho](\delta_t s, \phi_1)$, $[a_2, b_2] = [\rho](\delta_t s, \phi_2)$.

To minimize the computational overhead we chose a simple blending function that returns the direction of the sub-formula given by the choice function in case the sub-formulae' directions are not orthogonal, otherwise (3) is enforced, i.e.,

$$blend(\chi(s, \phi_c, t), \chi(s, \phi_{\neg c}, t)) = \begin{cases} \sum_{i \in \{c, \neg c\}} \chi(s, \phi_i, t) & \chi(s, \phi_c, t) \perp \chi(s, \phi_{\neg c}, t) \\ \chi(s, \phi_c, t) & \text{otherwise} \end{cases} \quad (5)$$

D. Guided steering

In this section we focus on guided steering used in line 10 of Alg. 1 posed as an optimization problem over control values with the cost function:

$$\begin{aligned}
J_\chi(u, x'', x', \phi', x^r; \lambda) = &\lambda \left\|x'' - (x' + d_\chi(u) * \Delta t^r)\right\|_2^2 \\
&+ (1 - \lambda) \left\|x'' - x^r\right\|_2^2
\end{aligned} \quad (6)$$

where $d_\chi(u) = \chi(traj_{\mathcal{T}}(v'), \phi', time_{\mathcal{T}}(v'))$ is the DIS, $v' = (x', \phi')$, and $x'' = \mathbf{x}[x', u](\Delta t^r)$ is the terminal state after applying the constant control input u for duration Δt^r. Informally, the cost function $J_\chi()$ balances the greedy heuristic of moving in the direction that improves robustness, captured by the first term of (6), and the stochastic search power of RRT*of steering towards random samples, imposed by the second term of (6). However, optimizing over this cost function simultaneously with respect to both u and x'' is hard. Therefore, we propose a relaxation that divides it into two optimization problems:

$$d_\chi = \max_u \|\chi(traj_\mathcal{T}(v'), \phi', time_\mathcal{T}(v'))\|_2^2 \tag{7}$$

$$\begin{aligned}\min_{x''} J_\chi(x'', x', x^r) = \min_{x''} \{ \lambda \|x'' - (x' + d_\chi \cdot \Delta t^r)\|_2^2 \\ + (1-\lambda)\|x'' - x^r\|_2^2 \}\end{aligned} \tag{8}$$

where d_χ is now the *direction of maximum increasing satisfaction (DMIS)* and independent of u, and the second optimization is in terms of x''.

The following results highlight the advantages of the relaxed formulation.

Lemma 4.2: Let $x', x^r \in X$. The minimizer of $J_\chi(\cdot, x', x^r)$ is $x^* = \lambda x^\chi + (1-\lambda)x^r$ and $J_\chi(x^*) = \lambda(1-\lambda)\|x^\chi - x^r\|_2^2$, where $x^\chi = x' + d_\chi \cdot \Delta t^r$.

Proof: The proof is immediate, and is omitted. ■

Lemma 4.3: Let ϕ be an STL formula, $s \in \mathcal{S}$, and $t \in \mathbb{R}_{\geq 0}$. If the blending function (5) is used, then there exists $I \subseteq \{1, \ldots, n\}$ such that $\chi(s, \phi, t) = \sum_{i \in I} f_i(x(t), u(t))e_i$.

Proof: The form of the DIS follows from the orthogonality condition (3), where $\dot{x}_i(t) = f_i(x(t), u(t))$. ■

Theorem 4.4: Let $\mathcal{R} = (f, X, U, x_{init})$ be a dynamical system, ϕ an STL formula, $s \in \mathcal{S}$ and $t \in \mathbb{R}_{\geq 0}$. If U is a convex polytope, $f(x,u) = h(x) + G(x)u$ is input-affine, and χ is parameterized by (4) and (5), then (7) is a maximization problem of a quadratic function with linear constraints

$$\max_{u \in U} \left\| \sum_{i \in I} e_i^T (h + Gu) e_i \right\|_2^2 \quad \text{s.t. } (-1)^{\iota_i}(h + Gu)^T e_i > 0.$$

Proof: The form of the cost function follows from the Lemma 4.3, while the linear constraints correspond to the predicates selected in $\chi()$, and the polytope U. ■

Note that for non-linear predicates, the form of the constraints will vary accordingly.

E. Rewiring

We assume available an oracle $om()$ to compute the RoSI of a formula on a partial trace. An example is an implementation of the algorithm in [18], which leverages Lemire's running maximum filter algorithm. More efficient variants can be constructed for subsets of STL.

Adding (line 15 in Alg. 1) and rewiring (line 18) vertices is done using the *update()* procedure. First, the monitoring algorithm $om()$ is used to compute the RoSI for the child v_1 by considering v_2 its parent. If v_2 is a new vertex (line 2 in Alg. 4), then a simplified formula ϕ_2 is computed from the formula of v_2 (line 5). The vertex v_1 is added to the tree if the new connection induces a RoSI with positive upper bound. The latter constraint ensures that satisfaction of the specification is still feasible, otherwise violation is certain. The *simplify()* function prunes formulae based on partial trajectories. Temporal operators that do not influence the outcome, such as those referring to previous times, are discharged. Another important role is to keep track of disjunctions and decide which alternative to choose to satisfy. The decision was implemented as a stochastic choice function, where there is a non-zero probability of either not changing the formula, or pruning one branch of a disjunction operator. Note, that formulae that do not contain disjunctions might still change, due to the removal of temporal operators whose deadline has passed.

In case of updating for rewiring, the parent of vertex of v_2 is changed to v_1 again only if the upper bound of the RoSI associated with the new connection is positive. Additionally, we require the lower bound to be improved and the formulae associates with the two vertices to be *consistent*. Since vertices are associated with simplified versions of the original specification, we need to ensure that new connections contain formulae on the same chain of simplifications, i.e., the formula of a child vertex can be obtain as a simplification of the parent's formula. Formally, we can define a partial order over the subformulae of the specification ϕ induced by "implication" to check line 7).

Algorithm 4: $update(v_1, v_2)$

```
    // propagate RoSI
1   $I'_2 = [a'_2, b'_2] \leftarrow om(\phi_1, rosi_\mathcal{T}(v_1), steer(x_1, x_2))$
2   if $\phi_2 = \bowtie$ then
3       if $b'_2 \geq 0$ then
4           $rosi_\mathcal{T}(v_2) \leftarrow I'_2$
5           $\phi_2 \leftarrow simplify(\phi_1)$
6           $V \leftarrow V \cup \{v_2\}$, $E \leftarrow E \cup \{(v_1, v_2)\}$
7   else if $b'_2 \geq 0 \wedge a'_2 \geq \min rosi(v_2) \wedge \phi_1 \implies \phi_2$ then
8       $rosi_\mathcal{T}(v_2) \leftarrow I'_2$
9       $E \leftarrow (E \setminus \{(pa(v_2), v_2)\}) \cup \{(v_1, v_2)\}$
10      $V_{upd} = ch(v_2)$        // children of $v_2$
11      while $V_{upd} \neq \emptyset$ do
12          $v \leftarrow V_{upd}.pop()$, $v' \leftarrow pa(v)$
            $rosi_\mathcal{T}(v') \leftarrow om(\phi, rosi_\mathcal{T}(v'), steer(v', v))$
```

F. Analysis

In this section, we analyze the convergence properties of our algorithm and its complexity. In particular, we prove that the control policy $\mathbf{u}$ given by the Alg. 1 after N^{max} iterations converges to the solution of Problem 3.2 as N^{max} goes to infinity, with probability one. We provide proof sketches due to space constraints.

Theorem 4.5 (Asymptotic optimality): The probability that Alg. 1 returns a control policy $\mathbf{u}$ that converges to the solution $\mathbf{u}^*$ of Problem 3.2 in the bounded variation norm sense, approaches one as $N = |V|$ tends to infinity, i.e.,

$$\mathbb{P}\left(\left\{\lim_{N \to \infty} ||\mathbf{x}[x_{init}, \mathbf{u}] - \mathbf{x}[x_{init}, \mathbf{u}^*]||_{BV} = 0\right\}\right) = 1$$

Proof: [Sketch] The proof follows from the asymptotic optimality of the RRT*algorithm (Theorem 34 in [20]). Let $\mathbf{u}^*$ be the solution of Problem 3.2 that maximizes $\rho(\mathbf{x}[x_{init}, \mathbf{u}^*], \phi)$. Define a finite sequence of balls $B^N = \{B_1^N, \ldots, B_m^N\}$ around the optimal trajectory $\mathbf{x}^* = \mathbf{x}[x_{init}, \mathbf{u}^*]$ such that consecutive balls overlap. The radii of the balls must be set to a fraction of $\gamma(\log(N)/N)^{1/n}$ such that any state in $x \in B_i^N$ can be connected using the *steer(x, x')* function to a state x' in the successor ball B_{i+1}^N, where n is the dimension of the state space. It can then be shown that for large enough N the probability that each ball in B^N contains at least one sample is one. The convergence in probability follows from two properties of the proposed algorithm. First, all decision for biasing the sampling (Alg. 2) and guiding the steering of the system (Alg. 1, DIS and choice function (4)) are stochastic and assign non-zero probabilities to all possible choices. Thus, the probability measure of trajectories generated by Alg. 1 has the same support set as for standard RRT*, albeit skewed towards increasing robustness. Second, we used the lower bounds of RoSIs associated with the tree's branches as the cost function. This cost function is monotonic due to Lemma 2.3, continuous, and it converges to the robustness value in bounded time. Thus, there is a trajectory $\mathbf{x}^N$ that intersects all balls in B^N with probability one, and it can be shown (as in [20]) that $\mathbf{x}^N$ converges to $\mathbf{x}^*$ as $N \to \infty$. ■

An immediate corollary of Theorem 4.5 is that the optimal trajectory returned by Alg. 1 converges to the maximum robustness value as N goes to infinity, i.e.,

$$\mathbb{P}\left(\left\{\lim_{N\to\infty} ||\rho(\mathbf{x}[x_{init}, \mathbf{u}], \phi) - \rho(\mathbf{x}[x_{init}, \mathbf{u}^*], \phi)||_{BV} = 0\right\}\right) = 1.$$

Theorem 4.6: The computational complexity of Alg. 1 is $O(|\phi|\log(N))$ per iteration, where $|\phi|$ is the size of the formula, i.e., the number of predicates and operators.

This iteration complexity follows from a quick inspection of Alg. 1. The sampling procedure takes $O(|\phi|)$ to compute the constraints (i.e., active predicates) of the biased region. There are $O(\log(N))$ vertices in a ball of radius $\gamma(\log(N)/N)^{1/d}$ that affect the number of nearest-neighbor queries. Moreover, it follows that the steering and rewiring functions are called for $O(\log(N))$ samples as well. The update procedure takes $O(|\phi|)$ time due to the computation of the DMIS and the RoSI for each vertex, and checking the consistency of two simplified STL formulae. Overall, the complexity of each iteration is $O(|\phi|\log(N))$.

V. CASE STUDY

In this section, we demonstrate the practical effectiveness of our approach via representative case studies. We implemented our algorithms in Python, and all experiments were performed on an Intel®Core$^{\text{TM}}$ i7-5500U CPU with a clock speed of 2.40GHz, 8GB RAM and 4 cores.

1) Case 1: Consider a system $\mathcal{R}_1$ whose dynamics are represented by a double integrator.

$$\ddot{q} = u, \tag{9}$$

The input u is bounded as $\|u(t)\|_1 \leq u_{\max}$. The system can be rewritten as a linear control system

$$\dot{x}_1 = x_2,\ \dot{x}_2 = u,\ y = x_1 \tag{10}$$

where $x_i(t) \in \mathbb{R}^n$ for time t and $i = 1, 2$.

We ran our approach to synthesize a control policy for $\mathcal{R}_1$ subject to the specification $\phi_1 = \diamondsuit_{[2,10]}(3.5 < x_1 \leq 4 \wedge -0.2 < x_2 \leq 0.2) \wedge \Box_{[0,2]}(-0.5 < x_2 \leq 0.5) \wedge \Box_{[0,10]}((2 < x_1 \leq 3) \implies (x_2 > 0.5 \vee x_2 \leq -0.5))$. We set N_{max} to 500.

The mean time per iteration was 16ms, and the total time for 500 iterations was 7.8s. Along the returned best control policy, the RoSI chain was:

$$\begin{aligned}&[(-4.000, 0.500), (-4.000, 0.349), (-4.000, 0.300),\\ &(-4.000, 0.207), (-4.000, 0.207), (-4.000, 1.143),\\ &(-4.000, 0.902), (-4.000, 0.734), (-4.000, 0.441),\\ &(-4.000, 0.272), (-4.000, 0.086), (-4.000, 0.034),\\ &(-4.000, 0.005), (-4.000, 0.005), (-4.000, 0.005),\\ &(-4.000, 0.005), (-4.000, 0.005), (\mathbf{0.005}, \mathbf{0.005})]\end{aligned}$$

The final robustness of satisfaction was thus 0.005. Figure 2 shows the generated tree and best policy at three time steps (100, 200 and 300, respectively). Note that the policy has already converged by time step 200.

2) Case 2: Consider a rear wheel car $\mathcal{R}_2$ whose dynamics are given by

$$\begin{aligned}\dot{x}_1 = x_4\cos(x_3),\ \dot{x}_2 = x_4\sin(x_3),\ \dot{x}_3 = x_5,\\ \dot{x}_4 = u_1,\ \dot{x}_5 = u_2,\end{aligned} \tag{11}$$

where x_1, x_2, x_3 represent the pose (position and orientation) of the vehicle, x_4, x_5 the linear and angular velocities, and the vehicle is controlled by bounded linear and angular acceleration u_1 and u_2, respectively, with $\mid u_1 \mid < 0.2$, $\mid u_2 \mid < 0.4$.

We synthesized a control policy for $\mathcal{R}_2$ subject to the reach-avoid specification with timing constraints $\phi_2 = \diamondsuit_{[0,18]}(3 < x_1 \leq 4 \wedge 2 < x_2 \leq 3) \wedge \Box_{[0,6]}\neg(1 < x_1 \leq 2 \wedge 2 < x_2 \leq 3)$.

The mean execution time per iteration was 260ms, and the total time for 1600 iterations was 400.55s. The final robustness of satisfaction value was 0.421 corresponding to the policy shown in the rightmost subfigure in Figure 2.

VI. CONCLUSION AND FUTURE WORK

We presented a sampling-based method for incrementally synthesizing a motion control policy such that resulting trajectories satisfy an STL specification. We defined the Direction of Increasing Satisfaction, and used it to construct a class of heuristic functions to bias the sampling of controlled edges in an RRT. We also used the robust interval semantics, which bounds the quantitative satisfaction of the specification given a partial policy, to choose which node of the tree to extend. We prove and demonstrate experimentally that this allows us to both converge to a path that satisfies the specification, and improve upon an existing path, asymptotically converging to a solution that satisfies the specification with maximum robustness. In future work, we will demonstrate this approach on a wider variety of domains, including robotic manipulation, to demonstrate its effectiveness.

Additionally, in this paper, we consider a stochastic choice function that depends only on robustness intervals of various subformulae associated with the partial trajectory. Another option is to learn this choice function while building the RRT*. Finally, while the approach we present deals with open-loop control policy synthesis for a closed system over a bounded time horizon, receding horizon approaches have been developed for extending STL synthesis to indefinite time

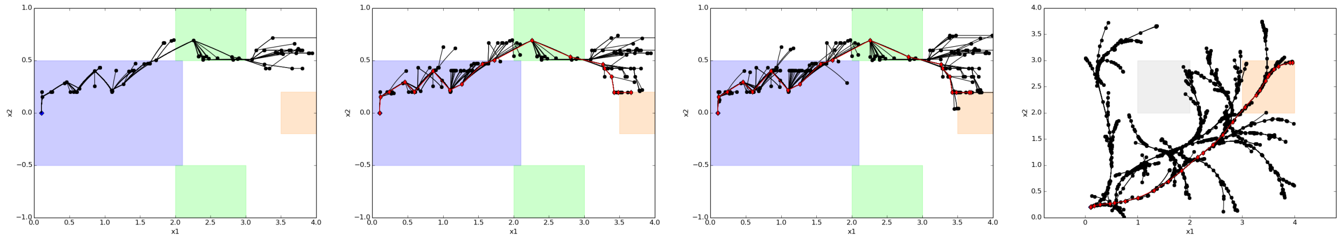

Fig. 2. The three figures on the left show the generated tree for a double integrator with $\phi_1 = \Diamond_{[2,10]}(3.5 < x_1 \le 4 \wedge -0.2 < x_2 \le 0.2) \wedge \Box_{[0,2]}(-0.5 < x_2 \le 0.5) \wedge \Box_{[0,10]}((2 < x_1 \le 3) \implies (x_2 > 0.5 \vee x_2 \le -0.5))$. Trees are shown after 100, 200 and 300 iterations. The rightmost figure shows the tree for a rear wheel car with $\phi_2 = \Diamond_{[0,18]}(3 < x_1 \le 4 \wedge 2 < x_2 \le 3) \wedge \Box_{[0,6]} \neg(1 < x_1 \le 2 \wedge 2 < x_2 \le 3)$.

horizons and open systems [32], [16], [33], [34]. We note that our approach can also be combined with such a receding horizon scheme to incorporate changes in the environment at runtime.

References

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

[2] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Trans. Program. Lang. Syst.*, vol. 8, no. 2, pp. 244–263, Apr. 1986.

[3] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science*, Oct 1977, pp. 46–57.

[4] S. Karaman and E. Frazzoli, "Vehicle routing problem with metric temporal logic specifications," in *IEEE Conference on Decision and Control*, December 2008, pp. 3953 – 3958.

[5] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot planning and control in polygonal environments," *IEEE Trans. on Robotics*, vol. 21, no. 5, pp. 864–874, 2005.

[6] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding Horizon Temporal Logic Planning for Dynamical Systems," in *IEEE Conference on Decision and Control (CDC)*, 2009, pp. 5997 –6004.

[7] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Trans. on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[8] K. Leahy, D. Zhou, C.-I. Vasile, K. Oikonomopoulos, M. Schwager, and C. Belta, "Provably Correct Persistent Surveillance for Unmanned Aerial Vehicles Subject to Charging Constraints," in *Int. Symposium on Experimental Robotics*, Morocco, June 2014.

[9] M. Lahijanian, M. Maly, D. Fried, L. Kavraki, H. Kress-Gazit, and M. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 583–599, 2016.

[10] M. Lahijanian, S. Almagor, D. Fried, L. Kavraki, and M. Vardi, "This Time the Robot Settles for a Cost: A Quantitative Approach to Temporal Logic Planning with Partial Satisfaction," in *AAAI*, 2015, pp. 3664–3671.

[11] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-time systems*, vol. 2, no. 4, pp. 255–299, 1990.

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

[13] C.-I. Vasile, D. Aksaray, and C. Belta, "Time Window Temporal Logic," *Theoretical Computer Science*, p. (submitted), http://arxiv.org/abs/1602.04294.

[14] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, vol. 19, no. 3, pp. 291–314, 2001.

[15] A. Donzé and O. Maler, *Robust satisfaction of temporal logic over real-valued signals*. Springer, 2010.

[16] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control (HSCC)*, Seattle, WA, USA, April 2015, pp. 239–248.

[17] A. Donzé, T. Ferrere, and O. Maler, "Efficient robust monitoring for STL," in *Computer Aided Verification*. Springer, 2013, pp. 264–279.

[18] J. V. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. A. Seshia, "Robust online monitoring of signal temporal logic," in *Runtime Verification - 6th International Conference, RV 2015 Vienna, Austria, September 22-25, 2015. Proceedings*, 2015, pp. 55–70.

[19] S. M. LaValle and J. J. Kuffner, "Randomized Kinodynamic Planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400.

[20] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.

[21] ——, "Sampling-based Motion Planning with Deterministic μ-Calculus Specifications," in *IEEE Conference on Decision and Control*, 2009.

[22] ——, "Sampling-based Optimal Motion Planning with Deterministic μ-Calculus Specifications," in *American Control Conference*, 2012.

[23] C. Vasile and C. Belta, "Sampling-Based Temporal Logic Path Planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[24] ——, "Reactive Sampling-Based Temporal Logic Path Planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, 2014.

[25] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[26] A. Bhatia, L. Kavraki, and M. Vardi, "Sampling-based motion planning with temporal goals," in *IEEE International Conference on Robotics and Automation*, 2010.

[27] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Motion Planning with Dynamics by a Synergistic Combination of Layers of Planning," *IEEE Trans. on Robotics*, vol. 26, no. 3, pp. 469–482, 2010.

[28] E. M. Wolff, U. Topcu, and R. M. Murray, "Automaton-guided controller synthesis for nonlinear systems with temporal logic," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 4332–4339.

[29] E. Aydin Gol, M. Lazar, and C. Belta, "Language-Guided Controller Synthesis for Linear Systems," *IEEE Trans. on Automatic Control*, vol. 59, no. 5, pp. 1163–1176, 2014.

[30] A. Dokhanchi, B. Hoxha, and G. Fainekos, *5th International Conference on Runtime Verification, Toronto, ON, Canada*. Springer, 2014, ch. On-Line Monitoring for Temporal Logic Robustness, pp. 231–246.

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

[32] V. Raman, A. Donze, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *CDC*, Dec 2014, pp. 81–87.

[33] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," *CoRR*, vol. abs/1511.00347, 2015.

[34] S. S. Farahani, V. Raman, and R. M. Murray, "Robust model predictive control for signal temporal logic synthesis," in *Analysis and Design of Hybrid Systems (ADHS)*, M. Egerstedt and Y. Wardi, Eds., vol. 48, no. 27, 2015, pp. 323–328.