Elliptical Slice Sampling for Probabilistic Verification of Stochastic Systems with Signal Temporal Logic Specifications

Guy Scher
gs679@cornell.edu
Sibley School of Mechanical and Aerospace Engineering,
Cornell University
Ithaca, NY, USA

Sadra Sadraddini
sadra@dexai.com
Dexai Robotics
Boston, MA, USA

Russ Tedrake
russt@mit.edu
Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology
Cambridge, MA, USA

Hadas Kress-Gazit
hadaskg@cornell.edu
Sibley School of Mechanical and Aerospace Engineering,
Cornell University
Ithaca, NY, USA

ABSTRACT

Autonomous robots typically incorporate complex sensors in their decision-making and control loops. These sensors, such as cameras and lidars, have imperfections in their sensing and are influenced by environmental conditions. In this paper, we present a method for probabilistic verification of linearizable systems with Gaussian and Gaussian mixture noise models (e.g. from perception modules, machine learning components). We compute the probabilities of task satisfaction under Signal Temporal Logic (STL) specifications, using its robustness semantics, with a Markov Chain Monte-Carlo slice sampler. As opposed to other techniques, our method avoids over-approximations and double-counting of failure events. Central to our approach is a method for efficient and rejection-free sampling of signals from a Gaussian distribution that satisfy or violate a given STL formula. We show illustrative examples from applications in robot motion planning.

CCS CONCEPTS

• **Computer systems organization → Robotics**; • **Theory of computation → Modal and temporal logics**.

KEYWORDS

Probabilistic verification, Signal Temporal Logic

ACM Reference Format:
Guy Scher, Sadra Sadraddini, Russ Tedrake, and Hadas Kress-Gazit. 2022. Elliptical Slice Sampling for Probabilistic Verification of Stochastic Systems with Signal Temporal Logic Specifications. In *25th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '22), May 4–6, 2022, Milan, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3501710.3519506

HSCC '22, May 4–6, 2022, Milan, Italy

ACM ISBN 978-1-4503-9196-2/22/05…$15.00
https://doi.org/10.1145/3501710.3519506

1 INTRODUCTION

To deploy autonomous robots, such as self-driving cars or assistive robots, we seek formal guarantees that they can operate safely and reliably. Providing such guarantees is challenging due to the sheer amount of non-determinism in the world including noisy sensors, uncontrolled environment (humans, other robots) and different environment conditions (such as lighting, occlusions, etc.). Modern systems also include machine learning components [7] that can contribute to the uncertainty since they might be deployed in different settings than the ones they were trained on.

Sensors, from proprioceptive ones that sense the robot's internal values such as speed or joint angles, to exteroceptive ones that sense the environment such as range finders and cameras, are usually modeled with errors coming from a Gaussian distribution or bounded noises. The system designer needs to reason about the likelihood that the system will successfully perform a task and re-design it if needed. The general approach is to find all the states (e.g. the robot's positions) that the robot may reach under all circumstances, i.e. the "reachable set", and reason about the safety and task completion. Testing with hardware is limiting, impractical and intractable because of the variability of tests and environmental conditions. Finding rigorous formal mathematical guarantees is usually infeasible for complicated systems performing complex tasks. Verifying systems using simulations may be the only way, but they suffer from long computation times, especially when searching for rare and hard to find events [5, 26, 35].

Several techniques exist for verifying systems with uncertainty. Imposing hard constraints on the state will always result in violation when dealing with unbounded non-determinism such as the Gaussian noise model. As such, it makes sense to describe the constraints with the probability of satisfying them - probabilistic state constraints. One common approach to verifying such robotic systems is with **chance constraints** [3, 12]. In these formulations, it is common to do risk allocation and use Boole's inequality, which allocates the level of uncertainty for each constraint component [22], or use ellipsoidal approximations. However, both are considered to be conservative [3, 16], as they over approximate failure probabilities. A known issue with these approaches is "double counting". A constraint violation of a trajectory at time t might yield a violation at $t+1$ as well, and they will be considered as two separate violations

because of the way they are constructed. In reality, we would like to consider that trajectory as only one failure.

Another common approach is to use **Monte-Carlo methods** to verify systems. These methods are attractive because they can be applied to non-linear systems, intricate noise models and black-box simulators. However, they can be computationally inefficient, especially when trying to detect rare events [4]. The verification process needs to iterate through many (guided) simulations to find rare events [11, 26, 29] in order to produce an accurate estimate of the probability. Other more generic techniques also exist to improve the performance of a Monte-Carlo simulation [32].

An issue with Monte-Carlo simulations is when there are many non-deterministic parameters [15]; in such cases, Monte-Carlo techniques may require a prohibitively large number of simulations to adequately represent the posterior distribution. In our case, a dynamic system with multiple noise sources and a long time horizon can grow to a large parametric space quickly. We, and the work in [15] which we extend upon, show that our method can yield accurate integrations for Gaussians in high dimensions independent of the probability mass of the posterior.

Optimization techniques have been used extensively in the literature to verify systems with uncertainty. There exist numerous verification algorithms that deal with machine learning components in the loop (e.g. [31]). The authors in [33] consider dynamic systems with a neural network component as the controller. In that setting, the inputs to the neural network are discretized and a linear program over-approximates the output. With that, they compute the over-approximation of the complete system's reachable set. In [7], the authors use robust control theory to provide guarantees for a system where the perception errors can be bounded using some assumptions on the data used during the training process versus the data that is collected in real-time.

Another line of work can be categorized as **geometric** algorithms. Set propagation techniques have been applied to reachability analysis. Except for a limited number of systems, these techniques always deal with under or over-approximations because finding the reachable sets is undecidable [2]. These techniques provide efficient computation frameworks; however, they work only on uni-model disturbances such as a Gaussian model, or a bounded disturbance. When discussing systems with a large number of states, one must employ other methods, such as decomposition of the system dynamics, for the methods to be tractable. The authors in [1] combined zonotopes and support functions to create an efficient framework for calculating the reachable sets of linear and switched dynamics systems. It considers only bounded disturbances.

In this work, we focus on the verification of properties that can be expressed using Signal Temporal Logic (STL) [10] for linear or linearizable time-variant systems with Gaussian error models. We show how our verification technique performs with a Gaussian mixture noise model (Section 4.7), where the weights of each Gaussian could be either static, come from a choice model like a Markov chain, or from a black-box choice model. We provide a verification method for generic STL formulae. We also describe a special case of reach-avoid [14] type specifications for which we propose an alternate solution that, in some cases, is more computationally efficient. We leverage and extend the framework in [15] to compute the probability that the robot satisfies its task specification.

Our main **contribution** is a computation framework for verifying and computing the probability of a high-dimensional system to satisfy (or violate) complex STL specifications within a finite horizon using the STL quantitative semantics. The technique is especially useful (accurate and tractable) when dealing with low probability events and displays the following properties: 1. We provide an efficient computation framework that does not suffer from the combinatorial nature of representing the signals that satisfy an STL specification. 2. Failure modes are not double-counted and not over-approximated. The computational framework is solved efficiently and can be parallelized. 3. Sampling is done from the posterior distribution in a rejection-free manner. Thus, we can sample *new* trajectories efficiently from the target distribution for analysis purposes, control synthesis, etc. 4. The algorithm is parameter-free; no fine-tuning of hyper-parameters is required. 5. It can verify systems with more intricate noises than Gaussian errors thus capturing realistic perception models.

2 PRELIMINARIES

In this section, we provide the necessary background on elliptical slice sampling and Signal Temporal Logic.

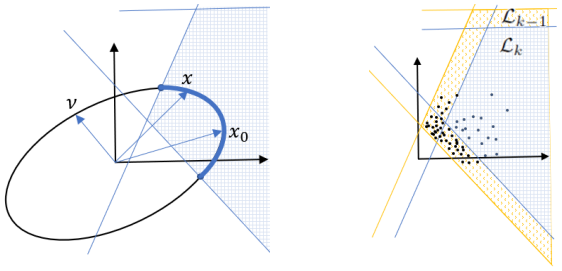
2.1 Elliptical Slice Sampling (ESS) and the Holmes-Diaconis-Ross (HDR) algorithm

An adaptive elliptical slicing method is used to sample from a linearly constrained domain under Gaussian distributions in [15]. We describe the main idea here for clarity and completeness. We extend [15] to compute the probability that the robot trajectories, represented as a multivariate Gaussian, satisfy a specification.

Elliptical slice sampling (ESS) [20] is a Markov Chain Monte Carlo technique (MCMC) for sampling from a posterior when the prior is a multivariate Gaussian $\mathcal{N}(\mu, \Sigma)$. In our case, the posterior will be a Gaussian under constrained linear domains (a truncated Gaussian). Given a single sample $x_0 \in \mathbb{R}^n$ inside the linear constrained domain $\mathcal{L} \subseteq \mathbb{R}^n$, and a new auxiliary point sampled from the same Gaussian $\nu \sim \mathcal{N}(\mu, \Sigma)$, the approach constructs an ellipse $x(\theta) = x_0 \cos(\theta) + \nu \sin(\theta)$, parameterized by the scalar $\theta \in [0, 2\pi]$. Using a closed-form solution to the intersections between the auxiliary ellipse and the hyperplanes that confine the linear domain $\mathcal{L}$, we can sample θ^* from a Uniform distribution over the ellipse arc lengths that lie within the domain, and thus obtain a new sample $x(\theta^*) \in \mathcal{L}$. A point on the ellipse is in the domain $\mathcal{L}$, when the intersection between all d constraints exceed zero, $Ax+b \geq 0$ where $A \in \mathbb{R}^{d \times n}, b \in \mathbb{R}^d$. This process is depicted in Fig.1a where the new sample x is sampled from the constrained Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ (for proof, see [15, 20]).

The Holmes-Diaconis-Ross (HDR) algorithm [8], a multi-level splitting algorithm, estimates the probability of sampling from a constrained region under any distribution. Direct Monte-Carlo methods may be inefficient because most candidate samples may be rejected (low probability distribution function or high dimensional domain). With HDR, the probability $p(\mathcal{L})$ of sampling from $\mathcal{L} \subseteq \mathbb{R}^n$ is estimated using the product of conditional probabilities:

$$p(\mathcal{L}) = p(\mathcal{L}_0) \prod_{k=1}^{K} p(\mathcal{L}_k | \mathcal{L}_{k-1}) \tag{1}$$



(a) Elliptical slice sampling **(b) HDR**

Figure 1: (a) Sampling a new point $x\,(\theta^*)$ from the constrained domain (blue grid) given an initial point x_0, an auxiliary point $v \sim \mathcal{N}(\mu, \Sigma)$ and $\theta^* \sim \mathcal{U}(\theta_{min}, \theta_{max})$. The active intersection is the bold blue line section of the ellipse where all points within $[\theta_{min}, \theta_{max}]$ are in the linearly constrained domain. (b) Original constrained domain $\mathcal{L}_K$ in the blue grid and shifted domain $\mathcal{L}_{K-1}$ in yellow divot after producing samples from $\mathcal{N}(\mu, \Sigma)$ using the ESS procedure under $\mathcal{L}_{K-1}$.

where $\mathcal{L}_0 = \mathbb{R}^n$, $p(\mathcal{L}_0) = 1$. Each domain $\mathcal{L}_k$ (also referred to as a nesting) is shifted (enlarged, see Fig.1b) to $\mathcal{L}_{k-1}$ by a scalar $\gamma_k > 0$ such that the conditional probabilities $p(\mathcal{L}_k|\mathcal{L}_{k-1}) \approx 0.5$ and $\gamma_K = 0$ is exactly the target domain $\mathcal{L}_K = \mathcal{L}$. Fig.1b depicts this process where the target domain $\mathcal{L}$ (blue grid) is expanded until it contains enough samples - when the probability to sample from the shifted region is about 0.5. Then, the algorithm iteratively shrinks the shifted region to keep the proportion of samples within the new domain to the previous domain at about half. n_k samples are drawn from each domain $\mathcal{L}_{k-1}$ with the ESS algorithm. The probability $p(\mathcal{L}_k|\mathcal{L}_{k-1}) = N(k)/n_k$, is the ratio between the number of samples $N(k) = \sum_{j=1}^{n_k} I(x_j \in \mathcal{L}_k)$ (I is the indicator function, equals one if the argument is true, zero otherwise) to the total number of samples, n_k, drawn at that nesting.

General closed-form solutions to the integral of a Gaussian under a linear constrained domain do not exist when the domain is not axis-aligned with the Gaussian. Numerical methods, such as quadrature algorithms, do not scale with the problem dimension [24].

2.2 Signal Temporal Logic

Signal temporal logic (STL) [19] enables specifying a broad range of temporal constraints over real-valued signals. Here we consider STL for discrete-time signals. Continuous-time logics and their properties can be found in, e.g., [10, 13].

Consider a discrete-time real-valued signal $\mathbf{s} = s_0, s_1, s_2, \cdots$, where $s_t \in \mathbb{R}^n, \forall t \in \mathbb{N}$. A *predicate* over $\mathbb{R}^n$ is denoted by $\mu = (h(s) \geq 0)$, where $h : \mathbb{R}^n \to \mathbb{R}$. A predicate is called *linear* if h is an affine function of s. Given a set of predicates, STL formulae are defined recursively using the following operators:

$$\mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \mathcal{U}_{[t_1,t_2]} \varphi_2 \mid \Diamond_{[t_1,t_2]} \varphi \mid \Box_{[t_1,t_2]} \varphi \quad (2)$$

where $\varphi, \varphi_1, \varphi_2$ are STL formulae, $\neg$ is the negation operator, $\wedge, \vee$ are conjunction and disjunction, respectively, and $\mathcal{U}_{[t_1,t_2]}, \Diamond_{[t_1,t_2]}, \Box_{[t_1,t_2]}$ are bounded temporal operators, over the time interval $[t_1, t_2]$, that stand for "until", "eventually", and "always", respectively.

Example 2.1. Consider a signal with values in $\mathbb{R}^2$, where $s = (s_{(1)}, s_{(2)})'$. The specification

$$\varphi = \Box_{[0,9]}(s_{(1)} + s_{(2)} - 10 \geq 0) \vee \Diamond_{[0,15]} \Box_{[0,5]}(-s_{(1)} \geq 0)$$

encodes "for all times in the interval [0,9], the value of $s_{(1)} + s_{(2)}$ stays above 10, *or*, for some time in the interval $[0, 15]$, the value of $s_{(1)}$ stays below 0 for 5 consecutive time steps".

Definition 2.2. *The STL score, or quantitative semantics [9], $\rho(\mathbf{s}, \varphi, t)$ is recursively defined as:*

- $\rho(\mathbf{s}, \mu, t) = h(s_t)$,
- $\rho(\mathbf{s}, \neg\varphi, t) = -\rho(\mathbf{s}, \varphi, t)$,
- $\rho(\mathbf{s}, \varphi_1 \wedge \varphi_2, t) = \min(\rho(\mathbf{s}, \varphi_1, t), \rho(\mathbf{s}, \varphi_2, t))$,
- $\rho(\mathbf{s}, \varphi_1 \vee \varphi_2, t) = \max(\rho(\mathbf{s}, \varphi_1, t), \rho(\mathbf{s}, \varphi_2, t))$,
- $\rho(\mathbf{s}, \Diamond_{[t_1,t_2]}\varphi, t) = \max_{\tau \in t+[t_1,t_2]} \rho(\mathbf{s}, \varphi, \tau)$,
- $\rho(\mathbf{s}, \Box_{[t_1,t_2]}\varphi, t) = \min_{\tau \in t+[t_1,t_2]} \rho(\mathbf{s}, \varphi, \tau)$,
- $\rho(\mathbf{s}, \varphi_1 \mathcal{U}_{[t_1,t_2]} \varphi_2, t) = \max_{\tau \in t+[t_1,t_2]} (\min(\rho(\mathbf{s}, \varphi_2, \tau), \min_{\tau' \in [t,\tau]} \rho(\mathbf{s}, \varphi_1, \tau')))$.

The STL score provides a metric for the distance to satisfaction of an STL formula. A positive STL score indicates satisfaction and a negative one stands for violation. To remove ambiguity, we consider the STL score of $\rho(s, \varphi, t) = 0$ as satisfying. We define the STL score of a signal $\mathbf{s}$ and specification φ as $\rho(\mathbf{s}, \varphi, 0)$.

Example 2.3. In Example 2.1, let $s_t = (t-8, 2)', \forall t \in \mathbb{N}$. Applying Definition 2.2, we obtain

$$\rho(\mathbf{s}, \Box_{[0,9]}(s_{(1)} + s_{(2)} - 10 \geq 0), 0) = \min_{t \in [0,9]} (t - 16) = -16$$

$$\rho(\mathbf{s}, \Diamond_{[0,15]} \Box_{[0,5]}(-s_{(1)} \geq 0), 0) = \max_{t \in [0,15]} \min_{\tau' \in [0,5]} (8 - (t + \tau')) = 3,$$

thus $\rho(\mathbf{s}, \varphi, 0) = \max(-16, 3) = 3$. Therefore signal $\mathbf{s}$ satisfies φ and its STL score is 3.

Definition 2.4. *The ϱ-level set of an STL formula φ is defined as:*

$$\mathcal{L}(\varphi, \varrho) = \{\mathbf{s} | \rho(\mathbf{s}, \varphi, 0) \geq \varrho\}. \quad (3)$$

Definition 2.5. *The horizon of the STL formula φ, denoted by H^φ, is the minimum length of truncated signal $\mathbf{s} = s_0, s_1, \cdots, s_{H^\varphi - 1}$ that is required to evaluate $\rho(\mathbf{s}, \varphi, 0)$. It is recursively given by:*

- $H^{(h(s) \geq 0)} = 1$,
- $H^{\neg\varphi} = H^\varphi$,
- $H^{\varphi_1 \wedge \varphi_2} = H^{\varphi_1 \vee \varphi_2} = \max(H^{\varphi_1}, H^{\varphi_2})$,
- $H^{\Diamond_{[t_1,t_2]}\varphi} = H^{\Box_{[t_1,t_2]}\varphi} = t_2 + H^\varphi$
- $H^{\varphi_1 \mathcal{U}_{[t_1,t_2]} \varphi_2} = t_2 + \max(H^{\varphi_1}, H^{\varphi_2})$.

Example 2.6. In Example 2.1, $H^\varphi = \max(9 + 1, 15 + 5 + 1) = 21$. The values of $s_{21}, s_{22}, \cdots$ do not affect $\rho(\mathbf{s}, \varphi, 0)$.

Given φ, we only need the truncated signal $s_0, s_1, \cdots, s_{H^\varphi - 1}$ to check whether it satisfies φ. Thus, we can stack the truncated signal into a vector denoted by $s^\varphi := (s_0', s_1', \cdots, s_{H^\varphi - 1}')' \in \mathbb{R}^{n \cdot H^\varphi}$. With a slight abuse of notation we extend the STL score and level-set definitions to the following function and set in $\mathbb{R}^{n \cdot H^\varphi}$:

$$\rho(s^\varphi) := \rho(\mathbf{s}, \varphi, 0). \quad (4)$$

$$\mathcal{L}(\varphi, \varrho) := \{s^\varphi \in \mathbb{R}^{n \cdot H^\varphi} | \rho(s^\varphi) \geq \varrho\}. \quad (5)$$

It is straightforward to show that given φ with linear predicates on $\mathbb{R}^n$, we have the following properties:

- $\rho : \mathbb{R}^{n \cdot H^\varphi} \to \mathbb{R}$ is piecewise affine and Lipschitz continuous.
- For a given ϱ, the set $\mathcal{L}(\varphi, \varrho)$ is a union of polyhedra in $\mathbb{R}^{n \cdot H^\varphi}$.

3 PROBLEM SETUP

3.1 System

We consider discrete linear(izable), possibly time-varying, systems (LTV) with the dynamic and measurement equations:

$$\begin{aligned} x_{t+1} &= A_t x_t + B_t u_t + w_t, \\ y_t &= C_t x_t + v_t \end{aligned} \tag{6}$$

where $x_t \in \mathbb{R}^n$ is the state at time t, $u_t \in \mathbb{R}^m$ is the control input and $y_t \in \mathbb{R}^q$ is the measurement vector. The process noise $w_t \in \mathbb{R}^n$ and the measurement noise $v_t \in \mathbb{R}^q$ are described in more detail in Section 3.2. A_t, B_t and C_t are the mappings between the states and measurements and are assumed known. The discrete system has a time step Δt. The system can be open loop (6), or have a linear state observer and a closed loop feedback controller for tracking a reference trajectory r_t:

$$\hat{x}_{t+1} = A_t \hat{x}_t + B_t u_t + L_t(y_t - C_t \hat{x}_t) \tag{7}$$

$$u_t = r_t - K_t \hat{x}_t \tag{8}$$

or, directly using the measurement for feedback:

$$u_t = r_t - K_t y_t. \tag{9}$$

3.2 Noise model

In this paper we focus on Gaussian errors, $v_t \sim \mathcal{N}(\mu_t^v, \Sigma_t^v)$ and $w_t \sim \mathcal{N}(\mu_t^w, \Sigma_t^w)$. We assume that all the noises are independent and identically distributed (iid). Note that one can augment the system's states if the noise is colored.

In Sec. 4.7 we consider noise modelled as a Gaussian mixture, $v_t \sim \sum_{m=1}^{M_v} \pi_m^v \mathcal{N}(\mu_m^v, \Sigma_m^v)$ where π_m^v is the probability of choosing Gaussian distribution m (similarly for w_t). While a single Gaussian is a special case of the mixture, we separate the discussion because we can provide stricter guarantees for this case.

3.3 Specification

We consider STL specifications where the underlying signal is the system trajectories: $\mathbf{x} = x_0, x_1, \cdots$ We consider only linear predicates on the system's state $\mu = (a'x + b \geq 0), a \in \mathbb{R}^n, b \in \mathbb{R}$. The assumption of linearity is essential since later in the paper we will use a closed-form solution for intersections of an ellipse and a hyperplane (Section 4.4). While it is possible to consider specific forms of nonlinear predicates and still retain closed-form solutions, we leave that to future work.

Example 3.1. A common STL formula in robotics is *reach-avoid*:

$$\varphi_{R/A} := \phi_0 \wedge \bigwedge_{i=1}^{N_{\text{unsafe}}} \Box_{[0,T]} \neg \phi_{\text{unsafe},i} \wedge \bigwedge_{j=1}^{N_{\text{goals}}} \Diamond_{[T_{0_j}, T_{1_j}]} \phi_{\text{goal},j}, \tag{10}$$

here $T \geq T_{1_j}, j = 1, \cdots, N_{\text{goals}}$, and $\phi_0 = \bigwedge_{i=1}^{N_0}(a_i'^0 x + b_i^0 \geq 0)$ defines a polyhedron in the state-space with N_0 hyperplanes each represented by a linear predicate. Similar notation is used to define sets of polyhedra for the unsafe sets (e.g. obstacles) and the goals. In words, the system satisfies the specification when it is able to start in the set defined by ϕ_0, avoid all obstacles $\phi_{\text{unsafe},i}$ for the entire trajectory and reach each $\phi_{\text{goal},j}$ at some $t \in [T_{0_j}, T_{1_j}]$. Given Δt and T, a trajectory of the system contains $t_H = \lceil T/\Delta t \rceil$ discrete time steps. To verify the specification φ, we require $t_H \geq H^\varphi$.

Given the number of possible ways to satisfy this formula, a strength of our approach is the ability to efficiently address the combinatorial aspect of all possible trajectory classes that may satisfy or violate the specification, without double counting them.

3.4 Problem formulation

Problem 1. *Given a linear system in the form of* (6)-(9)*, a Gaussian (mixture) noise model and an STL formula φ with linear predicates over x, find the probability that φ is satisfied.*

4 APPROACH

We illustrate our approach through an example of a holonomic robot navigating in a 2-D workspace (Fig. 2).

Example 4.1. A holonomic robot's state is $\xi = [x, y, \dot{x}, \dot{y}]'$ with the discrete-time dynamics:

$$\begin{aligned} \xi_{t+1} &= \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \xi_t + \begin{bmatrix} \Delta t^2/2m & 0 \\ 0 & \Delta t^2/2m \\ \Delta t/m & 0 \\ 0 & \Delta t/m \end{bmatrix} u_t + w_t \\ u_t &= r_t - K_{fb}\eta_t \end{aligned} \tag{11}$$

We use the discrete Linear Quadratic Regulator (LQR) algorithm [17] with the desired Δt to compute the optimal controller K_{fb}. We assume full-state measurement:

$$\eta_t = \xi_t + v_t \tag{12}$$

The noise v_t is normally distributed and w_t is omitted for brevity. We consider an arbitrary STL specification φ for the rest of the section unless otherwise specified.

4.1 Integral over Trajectory Space

The first step is to incorporate process and measurement noises into system trajectories by turning the trajectories into Gaussians in a higher dimensional space $x_{traj} \triangleq [x_0', \ldots, x_{t_H-1}']' \in \mathbb{R}^{n \cdot t_H}$. In our robot example, this means the concatenated states for every time step in the horizon, $\mathbf{x}_{traj} \in \mathbb{R}^{4 \cdot t_H}$. We consider $w_t = 0$ and $A_t = A, B_t = B$ and $C_t = C$ without loss of generality to simplify the following expressions. Based on Eq. (6)-(9), we can express the full trajectory in vector form with (13) by iteratively substituting the states and controls:

$$\mathbf{x}_{traj} = \Phi_0 x_0 + \Phi_r \mathbf{R} + \Phi_v \mathbf{V} \tag{13}$$

Where $\mathbf{R} = [r_0', \ldots, r_{t_H-1}']' \in \mathbb{R}^{m \cdot t_H}$, and $\mathbf{V} = [v_0', \ldots, v_{t_H-1}']' \in \mathbb{R}^{q \cdot t_H}$. Φ_0, Φ_r, Φ_v are the matrix coefficients that transfer the initial state, the extended reference inputs and measurement noises to the full trajectory, respectively. All components in (13) are deterministic except for the stochastic $\mathbf{V}$ with the noise model:

$$V \sim \mathcal{N}([\mu_0^{v'}, \cdots, \mu_{t_H-1}^{v'}]', diag([\Sigma_0^v, \cdots, \Sigma_{t_H-1}^v]) \tag{14}$$

We can extract the multivariate Gaussian in the trajectory space which is the distribution over which we integrate:

$$\mathbf{x}_{traj} \sim \mathcal{N}\left(\Phi_0 x_0 + \Phi_r \mathbf{R} + \Phi_v \, \mathrm{M}, \Phi_v \Sigma \Phi_v'\right) \tag{15}$$

$M = [\mu_0^{v'}, \cdots, \mu_{t_H-1}^{v'}]'$. Similarly, it is possible to derive the Gaussian of a trajectory with both v_t and w_t (and possibly, $x_0 \sim \mathcal{N}$).

Other work, e.g. chance constraints that are typically implemented and over-approximated with Boole's inequality, deal with constraints on the state-level. We work with the full trajectory. This difference is one of the reasons we do not double count events. When computing the probability of failures, the trajectory Gaussian is integrated with respect to the trajectory-level constraints.

The evaluation of the probability in Problem 1 is equivalent to computing the following integral:

$$p(\varphi) = \int_{x^\varphi \in \mathcal{L}(\varphi,0)} \text{pdf}(x^\varphi) dx^\varphi, \tag{16}$$

where $\text{pdf}(x^\varphi)$ is the probability density function of the trajectories, the Gaussian in this case from (15). $\mathcal{L}(\varphi, 0)$ is the set where the trajectories satisfy the specification φ.

4.2 Monte-Carlo Sampling

We propose a guided Monte-Carlo approach for verifying a dynamical system with fixed controls (they can be time-varying but not state-dependent) and Gaussian noise sources over a fixed horizon with respect to STL specification. We represent the full trajectories as Gaussian, and use the HDR and ESS algorithms to integrate the probability density function under the domains that satisfy the STL formula.

The advantages of the approach are threefold: 1. Efficient (rejection-free and parameter-free) sampling of trajectories that satisfy an STL specification and computation of the probability of satisfaction, without over-approximations and double-counting, such as with the use of Boole's inequality on each separate state in the trajectory. 2. Efficiently finding events with low probability that would be otherwise intractable to compute with naive Monte-Carlo simulations. 3. It enables longer horizons and more random variables without suffering from the dimension explosion problem (the ill-sampling of the posterior distribution).

The first point is achieved by sampling, with ESS, trajectories that are within the set of trajectories that satisfy the specification. The second point is achieved using the HDR algorithm as we can construct the required number of nestings to evaluate the probability. In fact, once all nestings are set up, the sampling time of the rejection-free ESS algorithm is not influenced by the probability mass. Regarding the third point, the variance of the error of the quantity we wish to estimate with a Monte-Carlo simulation $\sigma_{\bar{x}}^2 = \sigma_x^2 / n_{sim}$ decreases with the number of simulations n_{sim}. However, we cannot accurately estimate the value of σ_x^2 from the sampled simulations when we ill-sample the posterior distribution. We do not know the true variance a priori and in fact, the variance itself may increase rapidly as the number of variables increase. Intuitively, there are more combinations of noise errors which may cause the robot to violate the specification and it is harder to sample "useful" (for the purpose of correctly estimating the probability) combinations. Our approach, on the other hand, is sampling rejection-free from the constrained posterior distribution to the requisite level of accuracy.

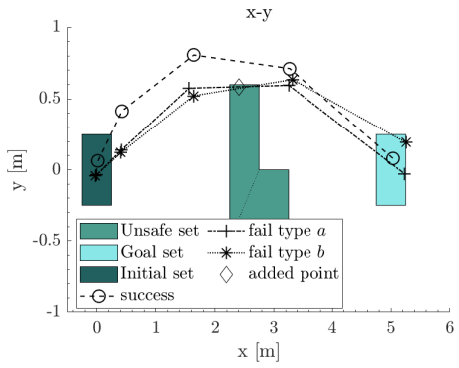


Figure 2: Example of a robot in 2-D. The initial, unsafe and goal sets are depicted in the figure. Successful and failed trajectories of a 5-step horizon are shown with an added intermediate point. We define failed trajectories that hit an obstacle, but reach the goal on time, as type a, and those that do not reach the goal on time as type b. In Section 4.5 we make use of these failures for reach-avoid specifications.

4.3 STL-Score-Guided Elliptical Slice Sampling

Here we describe how we draw sample points from $\text{pdf}(x)$ that are inside $\mathcal{L}(\varphi, \varrho)$ - trajectories that have an STL score $\geq \varrho$. As mentioned earlier, the naive way is to draw samples from $\text{pdf}(x)$ and reject those that fall outside of $\mathcal{L}(\varphi, \varrho)$. However, if the probability mass inside $\mathcal{L}(\varphi, \varrho)$ is too small, the procedure will be inefficient as most of the samples will be rejected.

We use ESS as described in Section 2. The explicit representation of $\mathcal{L}(\varphi, 0)$ - the domain of the integral in (16) - as a union of polyhedra requires an enumeration of all of the possible convex sets. The number of such sets can grow exponentially in the size of the formula (see, e.g., [28]). We avoid explicit enumeration of the polyhedra in $\mathcal{L}(\varphi, 0)$ while computing the integral in (16). The key insight is that we only need the STL score function [21].

Theorem 4.2. *Given an STL formula φ with a set of linear predicates $\mu_i = (a_i' x + b_i \geq 0), i = 1, \cdots, N_\varphi$, where N_φ is the total number of predicates. Given an existing sample trajectory $x_e^\varphi \in \mathcal{L}(\varphi, \varrho)$ and free sample trajectory x_f^φ (not necessarily in $\mathcal{L}(\varphi, \varrho)$), construct the ellipse $\mathcal{E} = \{x_e^\varphi \cos\theta + x_f^\varphi \sin\theta \mid \theta \in [0, 2\pi]\}$ in $\mathbb{R}^{n.H^\varphi}$. Then construct the following sorted list of real numbers in $[0, 2\pi]$:*

$$\Theta = sorted\Big\{ \begin{array}{l} \theta \mid \exists t \in \{0, 1, \cdots, H^\varphi - 1\}, i \in \{1, \cdots, N_\varphi\}, \\ \quad s.t.\ a_i' x_t + b_i = \pm\varrho,\ x^\varphi = x_e^\varphi \cos\theta + x_f^\varphi \sin\theta, \\ x^\varphi = (x_0', x_1', \cdots, x_{H^\varphi-1}')' \Big\}. \end{array} \tag{17}$$

Then for any two consecutive elements $\theta_1, \theta_2 \in \Theta$ (cyclic), one of the following statements is correct:

$$\forall \theta \in [\theta_1, \theta_2], \rho(x_e^\varphi \cos\theta + x_f^\varphi \sin\theta) \geq \varrho, \text{ or} \tag{18}$$

$$\forall \theta \in [\theta_1, \theta_2], \rho(x_e^\varphi \cos\theta + x_f^\varphi \sin\theta) \leq \varrho. \tag{19}$$

Proof. In order for $\rho(x^\varphi) = \varrho$, the value inside the function of at least one of the predicates should be equal to $\pm\varrho$ - this predicate becomes the maximizer/minimizer in the STL score function. Note

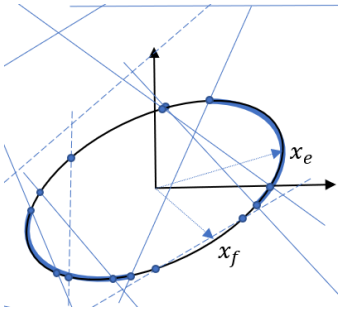


Figure 3: Hyperplanes, representing predicates in the STL formula, projected on the ellipse. Some may not intersect, and some may intersect but not change $\mathcal{L}(\varphi, \varrho)$ thus are in the domain. Dashed lines represent hyperplanes of predicates that are not in the time bounds described in φ.

that we have ± as negation might be in the formula. Therefore, the set Θ contains all the roots for $\rho(x^\varphi) - \varrho = 0$ but can contain spurious elements. Since ρ is Lipschitz continuous, $\rho(x^\varphi) - \varrho$ is sign-stable on $\mathcal{E}$ between two consecutive roots. □

Theorem 4.2 paves our way to compute portions of the ellipse that fall into $\mathcal{L}(\varphi, 0)$ by only computing the roots of the robustness function on the ellipse. Furthermore, (17), provides all the candidates with the complexity of solving $2 \cdot H^\varphi N_\varphi$ intersections of the ellipse with a hyperplane, for which closed-form solutions exist [15]. Then, using Theorem 4.2, we can sample $\theta \in [\theta_1, \theta_2]$ within each pair, to assign if $[\theta_1, \theta_2]$ is in $\mathcal{L}(\varphi, 0)$. Fig. 3 illustrates the adjusted ESS procedure. The complexity of intersecting the ellipse and $\mathcal{L}(\varphi, \varrho)$ is $O(H^\varphi N_\varphi)$, and we avoid the exponential blow up associated with explicit representation of $\mathcal{L}(\varphi, \varrho)$.

4.4 Holmes-Diaconis-Ross for STL

Now that we have a method to draw samples from $\mathcal{L}(\varphi, \varrho)$, we use it for our HDR-based Monte-Carlo method.

4.4.1 Nesting partitioning. To perform HDR where the probability density function is low, we need to account for the multi-level splitting described in the preliminaries. This means that when sampling from the nesting k, a larger domain than what we would like to evaluate, we shift ϱ to a new value (usually it will be a negative value, allowing more trajectories that violate φ, where about half the trajectories have robustness greater than ϱ) as the new cutoff level instead of 0. On the other hand, we also need to shift the linear predicates, to get the new intersections of $\mathcal{L}(\varphi, \varrho)$. For a general specification, we do not know whether to shift the predicates with a positive or a negative ϱ due to the structure of the sub-formulas. For example, consider the difference between $\varphi_1 := \mu$ versus $\varphi_2 := \neg\mu$. If $\rho(x, \varphi_1, 0) < 0$, we must enlarge μ to allow more "satisfying" trajectories while if $\rho(x, \varphi_2, 0) < 0$, we must make μ smaller. Instead of analyzing each component of the specification, we shift each predicate by $+\varrho$ and by $-\varrho$ as shown in (17).

4.4.2 Error Analysis. Monte-Carlo methods by nature give different results every time they are performed. It is necessary to have an estimate on the variance of the computed probability $p(\varphi)$. For the HDR nesting k, we sample n_k samples, and as discussed in Section 2, we aim for the conditional probability to be $p_{k \mid k-1} \approx 0.5$. In principal, ESS is a MCMC method, thus the samples are by definition dependent and the central limit theorem (CLT) does not apply. To mitigate this limitation, we keep only every n_d-th sample from the ESS, thus making the dependency between the sampled x_i to x_{i+n_d} practically non-existent (in all our examples we used $n_d = 4$). This is sometimes referred to as the "burn-in" phase, and its purpose here is to weaken the dependency between samples. In practice, our experience shows (see Fig. 6) that the dependency is weak, due to the "burn-in" process and sampling θ independently from a uniform distribution, and the following error analysis applies.

Using the CLT ($n_k p_{k|k-1} \gg 1$), we can assess that the variance for nesting k is $\sigma_k^2 \approx p_{k|k-1}(1 - p_{k|k-1})/n_k$ where $p_{k|k-1} = \mathrm{N}(k)/n_k$ and $\mathrm{N}(k)$ is the number of points sampled within $\mathcal{L}_k$. Therefore, for nestings $k = \{1, \ldots, K-1\}$, a good approximation for the variance is $\sigma_k^2 \approx \frac{1}{4n_k}$, such that $p_{k|k-1} \sim \mathcal{N}(\frac{1}{2}, \frac{1}{4n_k})$. With a slight abuse of notation, we define $p_k \equiv p_{k \mid k-1}$ for clarity. Each p_k is a Gaussian iid, therefore we can compute the variance of the product of the conditionals:

$$\begin{aligned} \mathrm{Var}[p_1 \cdots p_K] &= \mathbb{E}[(p_1 \cdots p_K)^2] - (\mathbb{E}[p_1 \cdots p_K])^2 \\ &= \mathbb{E}[p_1^2 \cdots p_K^2] - (\mathbb{E}[p_1] \cdots \mathbb{E}[p_K])^2 \\ &= \mathbb{E}[p_1^2] \cdots \mathbb{E}[p_K^2] - (\mathbb{E}[p_1])^2 \cdots (\mathbb{E}[p_K])^2 \\ &= \prod_{k=1}^{K} \left(\mathrm{Var}[p_k] + (\mathbb{E}[p_k])^2\right) - \prod_{k=1}^{K} (\mathbb{E}[p_k])^2 \end{aligned} \tag{20}$$

Substituting for our nominal parameters we obtain the approximation:

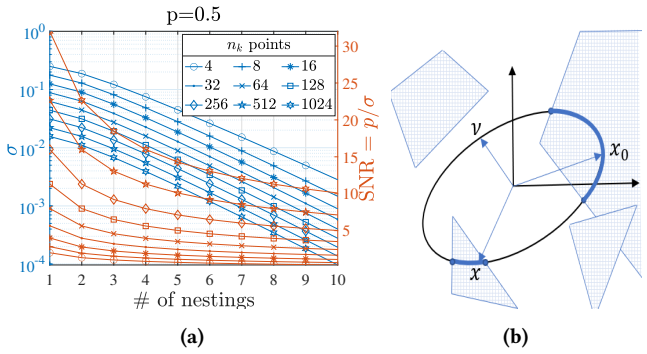
$$\mathrm{Var}[p_1 \cdots p_K] = \prod_{k=1}^{K} \left(\frac{1}{4n_k} + \frac{1}{4}\right) - \prod_{k=1}^{K} \left(\frac{1}{4}\right) \tag{21}$$

In practice, we compute the variance with the actual sampled values and not (21). The key point is that when the number of points in a nesting is large enough, the variance is proportional to $(1/4)^{K-1}$. For example, selecting 64 points per nesting with $K = 5$ yields a standard deviation $\sigma = 0.031$.

4.4.3 Adaptive nesting samples. Using (20), we can compute an expected minimal number of samples for a desired value of $p(\mathcal{L}_k | \mathcal{L}_{k-1})$. Fig.4a shows how increasing the number of samples decreases the uncertainty. Increasing the number of nestings, decreases the uncertainty as well; however, the number of nestings is not a design parameter but rather depends on the problem at hand. The number of nestings is approximately $K = \lceil -\log_2 p \rceil$. We can automatically select the number of samples to use per nesting by utilizing (20) and Fig.4a. The benefit of using this is shorter computation times; in problems with many nestings (i.e. low probability), we can get the desired confidence interval with less samples.

4.5 Special case: Reach-avoid specifications

In Eq. (10) we showed a common specification for robotic applications. In some cases, it might be more efficient to construct the union of polyhydra that represent the predicates in $\varphi_{R/A}$ explicitly, rather than use the STL score-based ESS and HDR (due to the number of hyperplanes and STL score computations discussed in Section 4.3). Consider finding the probability of violating Example 3.1, i.e.



(a) (b)

Figure 4: (a) The effect of the number of nestings and the number of sampled points on the confidence interval of the HDR algorithm on a nominal $p_k = 0.5$. (b) Extending [15] to a union of polytopes.

$\mathcal{L}(\neg\varphi_{R/A}, 0)$. We define two possible ways of violating $\varphi_{R/A}$ (i) type a: $\varphi_a := \phi_0 \wedge \Diamond_{[0,T]}\phi_{\text{unsafe}} \wedge \Diamond_{[T_0,T_1]}\phi_{\text{goal}}$, "hit an obstacle and reach the goals on time". (ii) type b: $\varphi_b := \phi_0 \wedge \Box_{[T_0,T_1]}\neg\phi_{\text{goal}}$, "do not reach the goals on time". Thus, $\mathcal{L}(\varphi_a, 0) \bigcup \mathcal{L}(\varphi_b, 0) = \mathcal{L}(\neg\varphi_{R/A}, 0)$.

We compute the integral of the Gaussian under the constrained domain by modifying the procedure in [15]; we consider a union of polytopes instead of only one. This means that as long as $\exists l \in \text{Set}(poly)$ such that the intersection of all its constraints exceed zero, then it is a valid point in the domain. There may be numerous intersections of the constructed ellipse with the faces of the different polytopes. We extend [15] (Fig. 4b) to find the active segments of the union of polytopes and sample points from the active domain.

4.6 Discrete-time implementation

We consider discrete time systems; however, there is a gap when verifying the system that is in fact continuous. There could be situations where all the discrete states in the horizon satisfy the specification, yet the system might collide with an obstacle in between the states. See Fig. 2 for an example near the obstacle.

There are several ways to address this. First, we can either increase the sampling rate or bloat the obstacles. The former will increase the dimension of the problem, while the latter would constrain the problem even more. In both cases, it will increase the computational load by increasing the dimensions in the trajectory space or by reducing the probability mass function under the domains (need more nestings).

The second approach, if we assume constant velocity between two consecutive states (valid in short time spans), we can introduce more constraints without increasing the problem dimensions. These intermediate points will add robustness by adding more area of the polytopes without as many computations as increasing the number of states. It can also be introduced only in parts of the trajectory that are susceptible to failure. Fig. 2 shows an example of a point added in the middle between t_2 to t_3 and the numerical simulation in Sec. 5.1. This additive technique can also be applied to compute $\rho(x^\varphi)$ if the STL library can compute the score of signals with dense time steps. The added constraints for an intermediate point is $a_i'(0.5x_t + 0.5x_{t+1}) + b_i \geq 0$

4.7 Gaussian Mixture models

Our approach may also be used to verify systems where the underlying noise model is better described with a Gaussian mixture. For example, a common model for range finders is the Beam model [30] (Ch. 6). It incorporates several modes of sensing errors that depend on the physical interaction of the sensor with its environment and may be approximated by a Gaussian mixture. Another example is a camera that is tracking cars but due to occlusions or errors in its neural net, it starts to track clutter or a different car in its field of view. The noise distribution at time t might depend on the distribution at $t-1$, making π_m^v come from a Markov chain or a black-box choice model:

$$v_t \sim \sum_{m=1}^{M} \pi_m^v \mathcal{N}(\mu_m^v, \Sigma_m^v) \tag{22}$$

Where M is the number of distributions; it can vary between time steps. In this case, computing the tree of possible combinations of the Gaussian distributions and noises throughout the trajectory, and their weights, is intractable. However, we provide a procedure for computing the total probability. The first step samples just the mixing factors π_m from their underlying distributions, for the entire horizon. When the mixing factors are fixed, the problem reduces to Problem 1. We compute the probability $p(\varphi)$ and variance, and repeat this procedure for N iterations. Then, we compute the unbiased mean estimate and the variance of the N iterations. This method still relies on Monte-Carlo simulation to compute the probability and variance estimation; however, only the trajectory modes are sampled, thus reducing the problem's input dimensions considerably. A full Monte-Carlo simulation will have the modes and the actual values to sample from and can thus be susceptible to the curse of dimensionality. We show an example with Gaussian Mixture noises in Sec. 5.2.

5 CASE STUDIES

We implemented the following simulations on a standard desktop Linux machine using Python. The robustness score is calculated with the `rtamt` library [21]. We compare our results with the MC approach because at the limit it provides the ground truth.

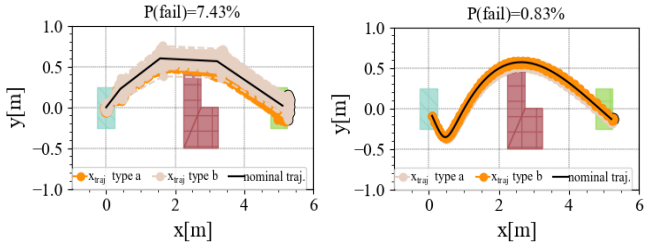
5.1 Robot navigation - reach-avoid

We demonstrate verification for Example 4.1 with $v_x \sim \mathcal{N}(0, 0.06^2)$, $v_y \sim \mathcal{N}(0, 0.06^2)$, $v_{\dot{x}} \sim \mathcal{N}(0.0, 0.04^2)$, $v_{\dot{y}} \sim \mathcal{N}(0.0, 0.04^2)$ and $w_t = 0$. Fig. 5a presents the static obstacles, goal, and the reference trajectory. To increase the fidelity of the simulation, we add intermediate points as discussed in Section 4.5. In the first scenario, the horizon $T = 5$sec with $\Delta t = 1$sec. The STL specification:

$$\varphi_1 := \phi_0 \wedge \Box_{[0,5]}\neg\phi_{obs1} \wedge \Diamond_{[5,5]}\phi_{goal} \tag{23}$$

Where ϕ_z = True if the intersection of all predicates of z over the state x is greater than zero. In this case, ϕ_{obs1} is non-convex thus we use Delaunay triangulation [18] to decompose it into two convex polytopes $\phi_{obs1:1}, \phi_{obs1:2}$. We have not considered any other restrictions on the state except on the pose.

5.1.1 Setup. We compute $p(fail) = p(\neg\varphi_1)$. We construct a disjunction between the trajectory-space $\mathcal{H}$-polytopes of failing trajectories of type $\mathcal{L}(\varphi_a, 0)$ and $\mathcal{L}(\varphi_b, 0)$ as discussed in Section 4.5.



(a) Scenario 1: $\Delta t = 1$sec. $p(fail) = 7.43\% \pm 0.8\%$.

(b) Scenario 2: $\Delta t = 0.1$sec. $p(fail) = 0.83\% \pm 0.26\%$.

Figure 5: X-Y projection of a robot maneuvering in a field with obstacles (red) and goal (green). In orange, the failing trajectories of type (a). In gray, the failing trajectories of type (b). In black, the reference trajectory.

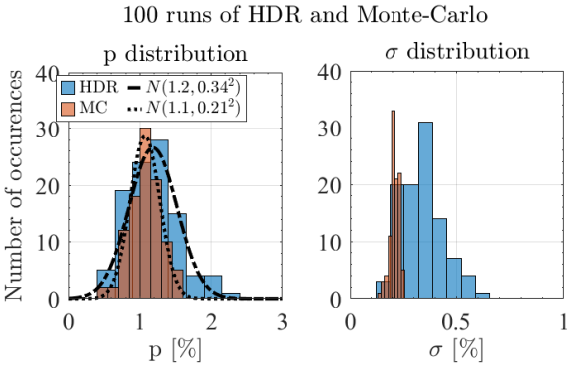


Figure 6: Running our approach (HDR) and MC (mean of 2400 runs) a 100 times each, and the comparison of the mean $\bar{p}(fail)$ estimation and its standard deviation.

5.1.2 Results. Fig.5a shows a sample of the trajectories of both failure modes. Computing the probability with our proposed algorithm yields $p(fail) = 7.43\% \pm 0.8\%$ and took 18sec. Verifying the same system with Monte-Carlo simulations yields $p_{MC}(fail) = 7.66\% \pm 0.5\%$ and took 13.0sec. To estimate the probability of failing with Monte-Carlo, we use $n_{MC} = 2400$ simulations, where $\sigma^2 = \hat{p}(1-\hat{p})/n_{MC}$. We use $n_k = n = 256$ samples in each nesting of the HDR algorithm (to get a comparable standard deviation). Monte-Carlo yields faster results because the probability to fail is relatively high.

Using a horizon of 5 steps yields a simulation of 20 random variables which can be considered a relatively small parameter space. Fig.5b depicts the second scenario with similar settings (initial conditions were changed to induce failures because the LQR controller with the new time step performs differently) for a horizon of $T = 5$sec and $\Delta t = 0.1$sec. This time, the problem dimension is 200. With our algorithm, $p(fail) = 0.83\% \pm 0.26\%$ and took 110sec to complete with $n_k = 64$. Monte-Carlo simulation takes 327sec and yields $p_{MC}(fail) = 1.2\% \pm 0.2\%$ in 2400 simulations. Fig.6 shows the distribution of running our algorithm and MC 100 times with different seeds and the results match for $\bar{p}(\neg\varphi_1)$, σ depends on n_k.

5.2 Car passing an intersection - reach-avoid

5.2.1 Setup. We consider a controlled car (Ego-vehicle, E) driving along the x-axis and an uncontrolled car (Other vehicle, O) driving along the y-axis, as shown in Fig. 7a. Each car's dynamic equations follow the holonomic robot in (11). Since O is uncontrolled we model its dynamics with a process noise $w^O_{\dot{y}} \sim \mathcal{N}(0, 0.2^2)$. E is measuring the distance d to O using a Lidar and has errors [30]. The error modes are a Gaussian about the true value, and a maximum range error, $v^E_d \sim \pi_1\mathcal{N}(0, 0.04^2) + \pi_2\mathcal{N}(5, 0.6^2)$. The transitions between the Gaussians are expressed with a Markov chain $p(\pi_1(t) \mid \pi_1(t-1)) = 0.98$, $p(\pi_2(t) \mid \pi_1(t-1)) = 0.02$, $p(\pi_1(t) \mid \pi_2(t-1)) = 0.6$, $p(\pi_2(t) \mid \pi_2(t-1)) = 0.4$ indicating the probability of having a bad measurement after a previous bad measurement is higher (occlusion, multipath). E needs to cross the intersection safely and uses the control law: $u^E_t = u_0 - Kd = u_0 - K(\sqrt{(x^O_t - x^E_t)^2 - (y^O_t - y^E_t)^2} + v_d)$. The time horizon is $T = 3$sec and $\Delta t = 0.1$sec. The cars' lengths are $L = 1.0$m and widths $W = 0.5$m. $K = -0.1$ and $u_0 = 0.075$ such that when the distance between the cars $d \leq 0.5(L+W)$, the control yields $u_k = 0$ and E stops until O crosses the intersection.

We derive a new state variable $z = [x^E_k - x^O_k, y^E_k - y^O_k, \dot{x}^E_k - \dot{x}^O_k, \dot{y}^E_k - \dot{y}^O_k]'$ with the initial conditions $z_0 = [-5, 5, 2, -2]'$, as shown in Fig. 7a. Given this new state variable, it is easy to show that the unsafe set (the "obstacle") is a square centered at the origin of z_x, z_y where the lengths of all the sides are $L + W$. The goal is for E to cross to the other side, i.e. $z_x \geq 0.5(L+W)$. The polytope sets are shown in Fig. 7b. The STL specification:

$$\varphi_{int} := \Box_{[0,30]}\neg\phi_{unsafe} \wedge \Diamond_{[29,30]}\phi_{goal} \tag{24}$$

Here, the measurement equation is non-linear. We find the trajectory's Gaussian distribution by linearizing the distance measurement in (25) evaluated at the expected value of the state, (26).

$$C_t = \frac{\partial d}{\partial z} \mid_{z=E[z_t]} = \frac{1}{d}[z_x, z_y, 0, 0] \tag{25}$$

$$\mathbb{E}[z_{t+1}] = (A - BKC_t)\,\mathbb{E}[z_t] + Bu_0 + \mathbb{E}[w^O_t] - BK\,\mathbb{E}[v^E_t] \tag{26}$$

5.2.2 Results. Fig. 7c shows the failing trajectory samples of one iteration of sampled noise mixing factors (Sec. 4.7). Total probability to fail $p(fail) = 54.68\%$ with 95% confidence level $[52.73\%, 56.63\%]$. We compare with Simple Random Sampling (SRS) with $n_{MC} = 2500$ for the Monte-Carlo simulations of the full non-linear system. The probability estimate $p_{MC}(fail) = 54.08\% \pm 1.0\%$ took $T = 32$sec to run, while using our method took $T = 130$sec (again, the times are due to the high probability of failure).

5.3 Data-based simulation - reach-avoid

In this example we show how this technique can be used in a scenario where the noises or system dynamics are not known. For this demonstration we run the Jackal [6] robot in the Gazebo simulator [25] with the Robot Operating System (ROS) [23]. We use the built-in controllers and estimation algorithms provided with and for the robot, and send it a goal command. With probability of 5%, a maximum range noise [30] is injected to any of the Lidar's ray measurements. Fig. 8 shows the environment the robot is navigating through. Our purpose is to verify that the system can reach the goal

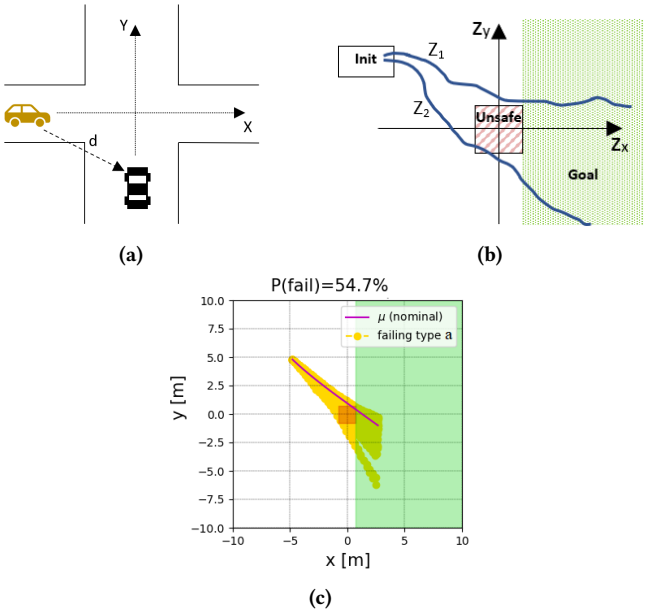


Figure 7: (a) Controlled car (yellow, left) entering an intersection with an uncontrolled car (black, bottom). (b) A satisfying trajectory (z_1), and a violating trajectory (z_2) in the Z coordinate frame. (c) A single noise sequence example of failing trajectories that intersect with the unsafe set.

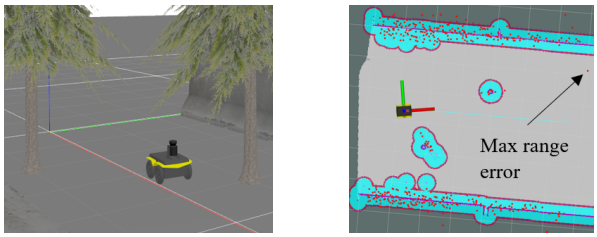


Figure 8: Jackal navigating in the environment (left). Lidar measurements and robot's current mapping (right).

safely. Since a single run takes approximately 15sec, Monte-Carlo simulation becomes intractable when the failure rate is low.

In our approach, we first run n simulations and fit a multivariate Gaussian (e.g. *robustcov* in Matlab) to the set of (ground truth) trajectories. n must be at least twice the number of variables (states times time steps, [27]). We now have $x_{traj} \sim \mathcal{N}(\mu, \Sigma)$ and we directly compute the probability to collide with a tree, miss the goal or violate any other temporal constraint.

In Fig. 9 we show the verification results for the system with

$$\varphi_{Gazebo} := \Box_{[0,33]} (\neg \text{Tree}_1 \wedge \neg \text{Tree}_2) \wedge \Box_{[33,33]} \text{Goal}$$

where Goal is the region defined by the box $5.5 \leq x \leq 7.5$, $0 \leq y \leq 0.5$. The time step in this scenario is $\Delta t = 0.4$sec and the horizon 13.2sec. We see that 16.0% of the trajectories fail to reach the goal on time (or overshoot it). We stopped the computation of the probability of hitting a tree ("type a") at $k = 24$ nestings, which means that a crash is less likely than about $6 \cdot 10^{-6}$%.

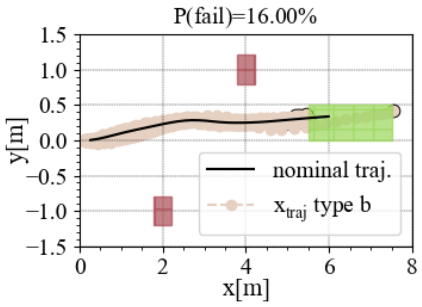


Figure 9: Data-based verification - x^{φ} comes from simulations. 16.0% of the trajectories are not at the goal at 13.2sec.

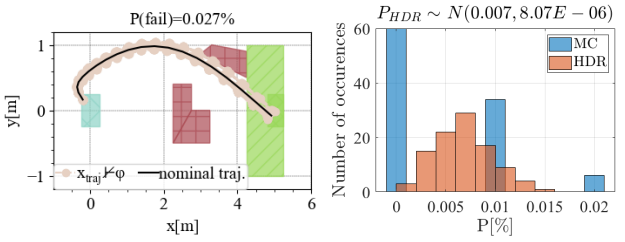
5.4 Robot navigation - Full STL

In this example we consider the robot in Example 4.1 and a complex STL specification:

$$\varphi_{STL} := \Box_{[0,T]} \neg (\phi_{\text{Obs}_1} \vee \phi_{\text{Obs}_2}) \wedge \Box_{[0,T]} (\phi_{\text{Goal}_1} \implies \Diamond_{[0,0.25]} \phi_{\text{Goal}_2}) \quad (27)$$

where $(a \implies b) = (\neg a \vee b)$. Since our technique computes the probability of *satisfying* the STL formula, to find the probability of failure, we use $\neg\varphi_{STL}$ in our computations. Following a single run of our method, we are able to find violating trajectories (Fig. 10a) even though $p(fail) = 0.027\%$. To find just one event with this probability we would need to run approx. 4000 simulations with MC. We ran 100 trials with our technique, and 100 trials with MC with 10^4 simulations each. The results are shown in Fig. 10b. 60% of the MC runs end with no failing examples, and about a third end with one failing example. The mean time to run MC is 626 ± 10sec and our method is 333 ± 35sec. The minimal probability computed by our method is $p(fail) = 0.001\%$.

Due to the use of the STL score for full STL, one cannot identify the specific cause of the failure. Furthermore, the sampled trajectories that fail the specification do not necessarily represent the proportions of the different failure causes. This is due to two reasons - first, we cannot guarantee how many trajectories are present in the final nesting, as explained in Section 2. Second, because this is a MCMC approach, the samples might be biased towards a certain region given an initial sample within that region. However, we show that if we sample new trajectories, we will get the correct proportions on average given enough samples. For example, in the previous scenario, the ratio between the probability mass for hitting the obstacle at t_{14} and not making the second goal on time, is approximately 4:1. We ran our approach 100 times. After each iteration finished, we sampled five sets of 1000 samples that violate φ_{STL} and computed how many of those hit the obstacle and how many violated the goal requirement. Results are shown in Fig. 11; although at specific instances we can get even more trajectories of goal violations than obstacle violations, we see that on average, we sample the correct proportions. This means that with our method, we are able to "jump" from an active domain to another active domain even if it is clearly distinct (different predicates and different time bounds). Of course, regions may be overshadowed by regions with considerably higher probability mass and if one



(a) (b)

Figure 10: (a) Failing trajectories of the STL formula φ_{STL}. (b) Statistics for a 100 trials for HDR, and for 100 MC, each with 10^4 simulations.

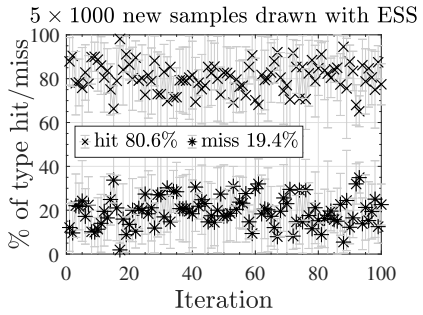


Figure 11: The assignation between trajectories that fail due to obstacle collision and due to missing the goal in time. We collect five ×1000 new trajectories with ESS after each round of $p(\varphi_{STL})$ computation. The markers are the means, and the error bars are 1 standard deviation.

wants to check those too, then they might need to decompose the specification to capture only those.

5.5 Adversarial Scenarios - Full STL

In [34] the authors developed a synthesis guided approach to find adversarial examples that falsify a dynamical system with respect to reach-avoid type specifications. An example from that paper (*Example 2*) finds a series of measurement noises that causes the system (28) and its regulator to enter the unsafe zone.

$$\xi_{t+1} = \begin{bmatrix} 0.9745 & 0.2132 \\ 0.002547 & 1.151 \end{bmatrix} \xi_t + \begin{bmatrix} 0.01959 \\ 0.1961 \end{bmatrix} u_t + \begin{bmatrix} 0.01959 \\ -0.04509 \end{bmatrix} w_t \tag{28}$$

$$u_t = -\begin{bmatrix} 1 & 1 \end{bmatrix} \eta_t \; ; \; \eta_t = \xi_t + v_t$$

In [34], the noises $v_t = [-0.1, 0.1]^2$ and $w_t = [-0.2, 0.2]$ are uniform and bounded. Here we approximate them with an appropriate Gaussian. The unsafe set is defined as $\text{Unsafe}(\xi) = [1, 2] \times [-0.5, 0.5]$ and the system starts in $\text{Init}(\xi) = [-0.15 \times 0.15]^2$. To find adversarial trajectories, we consider the STL formula:

$$\varphi_{adv} := \text{Init}(\xi) \wedge \Diamond_{[0,115]} \text{Unsafe}(\xi) \tag{29}$$

In addition to finding the probability, our approach can find adversarial examples, as done in [34]. In Fig. 12 we show a trajectory, sampled from the set of satisfying trajectories, that eventually enters the unsafe zone. In this example, the probability that the system may enter the unsafe zone is 0.09% where the different trajectories may enter the unsafe zone at different times; our apporach can provide several such examples.

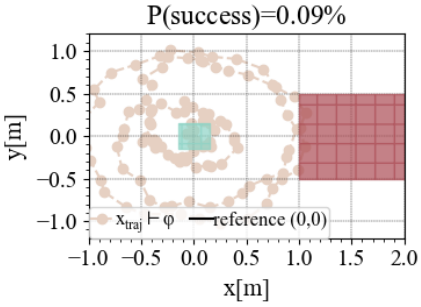


Figure 12: Discovering adversarial noise sequences that lead to unsafe behaviors.

6 DISCUSSIONS AND CONCLUSIONS

In this paper we introduced a method to accurately compute the probability that a linearizable system will satisfy an STL specification. The framework is general and can accommodate various sensor, estimator and perception errors. We provide two methods for calculating the probability - for full STL and for reach-avoid specifications.

Our method, while including computation overhead, is scalable to high dimensions (longer horizons or models with more states) and its computational complexity does not depend on the combinatorially many solutions of the specification. The sampling is efficient, especially in low probability events where a naive Monte-Carlo approach may not be tractable. The latter may suffer from dimension explosion, leading to the need for a large number of simulations to adequately sample the posterior. Our method is sampling from the posterior in a rejection-free and parameter-free (no hyper parameters needed for the slice sampler) manner.

Our method lends itself to parallel implementation, thereby reducing the computation time. Every nesting from the ESS and HDR can be run in parallel. By increasing the computation speed, our method can potentially be used as a step in motion planning, where, for example, we can check the output of a rapidly exploring random tree (RRT) generated path to check feasibility given noises or complex specifications.

In future work we will use this framework to synthesize controllers that can minimize the probability of failure. Another direction is to use the sampled failed trajectories to gain insight and requirements on the perception system that would be the most beneficial in reducing failure.

ACKNOWLEDGMENTS

This work is supported by ONR PERISCOPE MURI award N00014-17-1-2699.

REFERENCES

[1] Matthias Althoff and Goran Frehse. 2016. Combining zonotopes and support functions for efficient reachability analysis of linear systems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, Las Vegas, NV, USA, 7439–7446. https://doi.org/10.1109/CDC.2016.7799418

[2] Matthias Althoff, Goran Frehse, and Antoine Girard. 2021. Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems* 4 (2021), 369–395.

[3] Lars Blackmore and Masahiro Ono. 2009. Convex chance constrained predictive control without sampling. In *AIAA Guidance, Navigation, and Control Conference*. AIAA, Chicago, IL, USA, 5876.

[4] Drive Tesla Canada. 2021. *Tesla Q1 2021 Safety Report: Autopilot nearly 10X safer than Humanpilot*. https://driveteslacanada.ca/news/tesla-q1-2021-safety-report-autopilot-nearly-10x-safer-than-humanpilot

[5] Glen Chou, Yunus Emre Sahin, Liren Yang, Kwesi J Rutledge, Petter Nilsson, and Necmiye Ozay. 2018. Using control synthesis to generate corner cases: A case study on autonomous driving. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37, 11 (2018), 2906–2917.

[6] Clearpath Robotics. 2020. Jackal UGV - Small Weatherproof Robot - Clearpath. https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle

[7] Sarah Dean, Nikolai Matni, Benjamin Recht, and Vickie Ye. 2020. Robust Guarantees for Perception-Based Control. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control (Proceedings of Machine Learning Research, Vol. 120)*, Alexandre M. Bayen, Ali Jadbabaie, George Pappas, Pablo A. Parrilo, Benjamin Recht, Claire Tomlin, and Melanie Zeilinger (Eds.). PMLR, 350–360. https://proceedings.mlr.press/v120/dean20a.html

[8] Persi Diaconis and Susan Holmes. 1995. Three Examples of Monte-Carlo Markov Chains: At the Interface Between Statistical Computing, Computer Science, and Statistical Mechanics. In *Discrete Probability and Algorithms*, David Aldous, Persi Diaconis, Joel Spencer, and J. Michael Steele (Eds.). Springer New York, New York, NY, 43–56.

[9] Alexandre Donzé, Thomas Ferrère, and Oded Maler. 2013. Efficient Robust Monitoring for STL. In *Computer Aided Verification*, Natasha Sharygina and Helmut Veith (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 264–279.

[10] Alexandre Donzé and Oded Maler. 2010. Robust Satisfaction of Temporal Logic over Real-Valued Signals. In *Formal Modeling and Analysis of Timed Systems*, Krishnendu Chatterjee and Thomas A. Henzinger (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 92–106.

[11] Tommaso Dreossi, Daniel J. Fremont, Shromona Ghosh, Edward Kim, Hadi Ravanbakhsh, Marcell Vazquez-Chanlatte, and Sanjit A. Seshia. 2019. VerifAI: A Toolkit for the Formal Design and Analysis of Artificial Intelligence-Based Systems. In *Computer Aided Verification*, Isil Dillig and Serdar Tasiran (Eds.). Springer International Publishing, Cham, 432–442.

[12] Noel E Du Toit and Joel W Burdick. 2011. Probabilistic collision checking with chance constraints. *IEEE Transactions on Robotics* 27, 4 (2011), 809–815.

[13] Georgios E Fainekos and George J Pappas. 2009. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science* 410, 42 (2009), 4262–4291.

[14] Chuchu Fan, Umang Mathur, Sayan Mitra, and Mahesh Viswanathan. 2018. Controller Synthesis Made Real: Reach-Avoid Specifications and Linear Dynamics. In *Computer Aided Verification*, Hana Chockler and Georg Weissenbacher (Eds.). Springer International Publishing, Cham, 347–366.

[15] Alexandra Gessner, Oindrila Kanjilal, and Philipp Hennig. 2020. Integrals over Gaussians under Linear Domain Constraints. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 108)*, Silvia Chiappa and Roberto Calandra (Eds.). PMLR, 2764–2774. https://proceedings.mlr.press/v108/gessner20a.html

[16] Nikolaos Kariotoglou, Sean Summers, Tyler Summers, Maryam Kamgarpour, and John Lygeros. 2013. Approximate dynamic programming for stochastic reachability. In *2013 European Control Conference (ECC)*. IEEE, Zurich, Switzerland, 584–589. https://doi.org/10.23919/ECC.2013.6669603

[17] Huibert Kwakernaak and Raphael Sivan. 1972. *Linear optimal control systems*. Vol. 1. Wiley-interscience New York.

[18] Der-Tsai Lee and Bruce J Schachter. 1980. Two algorithms for constructing a Delaunay triangulation. *International Journal of Computer & Information Sciences* 9, 3 (1980), 219–242.

[19] Oded Maler and Dejan Nickovic. 2004. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 152–166.

[20] Iain Murray, Ryan Adams, and David MacKay. 2010. Elliptical slice sampling. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 9)*, Yee Whye Teh and Mike Titterington (Eds.). PMLR, Chia Laguna Resort, Sardinia, Italy, 541–548. https://proceedings.mlr.press/v9/murray10a.html

[21] Dejan Ničković and Tomoya Yamaguchi. 2020. RTAMT: Online robustness monitors from STL. In *International Symposium on Automated Technology for Verification and Analysis*. Springer, 564–571.

[22] Masahiro Ono and Brian C. Williams. 2008. An Efficient Motion Planning Algorithm for Stochastic Dynamic Systems with Constraints on Probability of Failure. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3* (Chicago, Illinois) *(AAAI'08)*. AAAI Press, 1376–1382.

[23] Open Robotics. 2020. ROS.org | Powering the world's robots. https://www.ros.org

[24] Ramón Orive, Juan C Santos-León, and Miodrag M Spalević. 2020. Cubature formulae for the Gaussian weight. Some old and new rules. *Electronic Transactions on Numerical Analysis* 53 (2020), 426–439.

[25] OSRF. 2020. Gazebo. http://gazebosim.org

[26] Matthew O' Kelly, Aman Sinha, Hongseok Namkoong, Russ Tedrake, and John C Duchi. 2018. Scalable End-to-End Autonomous Vehicle Testing via Rare-event Simulation. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/653c579e3f9ba5c03f2f2f8cf4512b39-Paper.pdf

[27] Peter J Rousseeuw and Katrien Van Driessen. 1999. A fast algorithm for the minimum covariance determinant estimator. *Technometrics* 41, 3 (1999), 212–223.

[28] Sadra Sadraddini and Calin Belta. 2016. Feasibility envelopes for metric temporal logic specifications. In *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 5732–5737.

[29] Edward Schmerling and Marco Pavone. 2016. Evaluating trajectory collision probability through adaptive importance sampling for safe motion planning. *arXiv preprint arXiv:1609.05399* (2016).

[30] Sebastian Thrun. 2002. Probabilistic robotics. *Commun. ACM* 45, 3 (2002), 52–57.

[31] Hoang-Dung Tran, Weiming Xiang, and Taylor T. Johnson. 2020. Verification Approaches for Learning-Enabled Autonomous Cyber-Physical Systems. *IEEE Design Test* (2020), 1–1. https://doi.org/10.1109/MDAT.2020.3015712

[32] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. 2018. An Introduction to Probabilistic Programming. *arXiv e-prints*, Article arXiv:1809.10756 (Sept. 2018), arXiv:1809.10756 pages. arXiv:1809.10756 [stat.ML]

[33] Weiming Xiang and Taylor T. Johnson. 2018. Reachability Analysis and Safety Verification for Neural Network Control Systems. *arXiv e-prints*, Article arXiv:1805.09944 (May 2018), arXiv:1805.09944 pages. arXiv:1805.09944 [cs.SY]

[34] Liren Yang and Necmiye Ozay. 2021. Synthesis-Guided Adversarial Scenario Generation for Gray-Box Feedback Control Systems with Sensing Imperfections. *ACM Trans. Embed. Comput. Syst.* 20, 5s, Article 102 (Sept. 2021), 25 pages. https://doi.org/10.1145/3477033

[35] Huafeng Yu, Xin Li, Richard M Murray, S Ramesh, and Claire J Tomlin. 2018. *Safe, Autonomous and Intelligent Vehicles*. Springer.