Specification Mining and Robust Design under Uncertainty: A Stochastic Temporal Logic Approach

PANAGIOTIS KYRIAKIS, JYOTIRMOY V. DESHMUKH, and PAUL BOGDAN,
University of Southern California

In this paper, we propose Stochastic Temporal Logic (StTL) as a formalism for expressing probabilistic specifications on time-varying behaviors of controlled stochastic dynamical systems. To make StTL a more effective specification formalism, we introduce the quantitative semantics for StTL to reason about the robust satisfaction of an StTL specification by a given system. Additionally, we propose using the robustness value as the objective function to be maximized by a stochastic optimization algorithm for the purpose of controller design. Finally, we formulate an algorithm for parameter inference for Parameteric-StTL specifications, which allows specifications to be mined from output traces of the underlying system. We demonstrate and validate our framework on two case studies inspired by the automotive domain.

CCS Concepts: • **General and reference** → **Verification**; • **Information systems** → **Temporal data; Uncertainty**; • **Theory of computation** → **Modal and temporal logics**; • **Computer systems organization** → **Embedded software**; *Embedded and cyber-physical systems*; • **Mathematics of computing** → *Stochastic differential equations*; Stochastic control and optimization;

Additional Key Words and Phrases: Stochastic temporal logic, robust satisfaction, controller design

ACM Reference format:

Panagiotis Kyriakis, Jyotirmoy V. Deshmukh, and Paul Bogdan. 2019. Specification Mining and Robust Design under Uncertainty: A Stochastic Temporal Logic Approach. *ACM Trans. Embed. Comput. Syst.* 18, 5s, Article 96 (October 2019), 21 pages.
<https://doi.org/10.1145/3358231>

1 INTRODUCTION

To construct robust cyber-physical system (CPS) applications, we have to carefully reason about uncertainties occurring in the environment as well as in system components (e.g., unforeseen interdependencies, parameter variations, hardware aging, software updates that do not fully account for hardware variations). There are typically two divergent views on tackling uncertainty. The first assumes that uncertainty is nondeterministic, i.e., all feasible behaviors of the environment or the system can occur. However, for systems that interact with a physical environment, such an assumption can lead to over-constrained designs or overly-pessimistic assessments of system

This article appears as part of the ESWEK-TECS special issue and was presented at the International Conference on Embedded Software (EMSOFT) 2019.

Authors' addresses: P. Kyriakis, J. V. Deshmukh, and P. Bogdan, University of Southern California; emails: {pkyriaki, jyotirmoy.deshmukh, pbogdan}@usc.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.

1539-9087/2019/10-ART96 \$15.00

<https://doi.org/10.1145/3358231>

correctness. The second approach relies on a probabilistic framework for defining system correctness. This approach is suitable for systems that are stochastic in nature, or have highly uncertain environments that cannot be precisely modeled.

The simplest approach for modeling stochasticity is to model variables in the system (or environment) as random variables with a parametric probability distribution. For instance, it is common to use a normal distribution with unknown mean and variance to model a random variable. The use of such a distribution is typically justified from empirical observations of the system, and the parameters of the distribution (mean and variance in this case) are then estimated from data. Such assumptions are routinely used to model time-invariant stochasticity, such as to model noise in sensors or manufacturing variations in physical models [24].

However, modern CPS applications often operate in uncertain environments. For example, consider applications such as air traffic management where the underlying system is modeled as a stochastic hybrid system [15], probabilistic collision detection models for autonomous cars [3], or closed-loop control of blood glucose in artificial pancreas using stochastic, fractional dynamical system models [20]. For systems with such time-varying disturbances and uncertain dynamics, a better approach to modeling uncertainty is to treat the system as a stochastic dynamical system, where the probability distribution of the system variables is a function of time, described by an appropriate stochastic differential equation. Constructing such models enables reasoning about several interesting problems such as the design of controllers, probabilistic reasoning of system correctness, and quantifying the robustness of a system design. It is common for stochastic models for such systems to be data-driven, *i.e.*, empirically constructed from observations of systemic data [11].

There are two main challenging problems in CPS models operating in uncertain environments. The first is to precisely articulate their behavioral specifications. In certain settings, such as for legacy applications, or for emergent applications requiring rapid prototyping, specification-driven design is often infeasible. In such cases, *post hoc* mining of formal design specifications has proven to be an interesting approach that adds significant value to the design process [12, 23, 26]. The second challenge is to obtain controllers that ensure that the system satisfies its formal specifications under appropriate uncertainty models.

Recently, there has been considerable research on using real-time temporal logics such as Signal Temporal Logic (STL) [37] for expressing properties and abstractions of closed-loop cyber-physical system models [22, 25, 30, 39], and subsequent use of such specifications for control design. However, for effective probabilistic reasoning for stochastic system models, we need mathematical frameworks that include probabilities as part of the system specification. Recently, STL was extended to three probabilistic variants, Probabilistic STL (PrSTL) [40], Chance Constrained Temporal Logic (C2TL) [24], and Stochastic STL (StSTL) [35]. The logics differ in technicalities of how atomic predicates are defined, but they essentially provide a *logical characterization* of uncertainty. A recent paper proposes *Stochastic Temporal Logic* (StTL) as a formalism that unifies certain features from these logics [14]. The research in [24, 35, 40] crucially provides techniques to design controllers under specific uncertainty models. In this paper, we seek to use these logics to achieve the two research challenges we outlined before: designing controllers from uncertainty-aware specifications, and mining such specifications from CPS models and data. Specifically, the contributions of this paper are as follows:

- (1) We formulate a new *controller design* technique for *stochastic dynamical systems* with the goal of satisfying a given set of uncertainty-aware temporal specifications. For this problem, we seek to find the *controller parameters* such that the system satisfies the specification. To this end, we initially introduce Stochastic Temporal Logic (StTL) as a formal language to encode the design specifications.

Fig. 1. Overview of research problems and solutions.

- (2) We introduce quantitative semantics for StTL that provide us with a mathematical approach to *quantify the robustness of a system evolving in an uncertain dynamic environment*. Using the quantitative semantics of StTL, we provide a technique to estimate the *robustness estimate* of the given system. We define the robustness estimate as the maximal perturbation to a given StTL specification such that the specification continues to be satisfied (or violated) by the given stochastic system.
- (3) We develop a new controller design technique for stochastic dynamical systems, where we express the controller design specifications as StTL formulas, compute their robustness and use this quantity to guide a black-box (stochastic) optimization algorithm which explores the controller parameter space attempting to find a global maximum of the robustness estimate.
- (4) We formulate an algorithm to mine StTL specifications from CPS models and data. Given a parametric StTL formula (i.e. an StTL formula, where certain numerical constants are replaced by *formula parameters*), our mining algorithm infers values for the formula-parameters that lead to the marginal satisfaction of the StTL formula. Finally, we establish a connection between specification mining and robustness estimation, which allows us to speed-up the robustness computation in certain cases.

Paper Organization. We outline our contributions in the flow diagram in Figure 1. The rest of the paper is organized as follows: In Section 2, we introduce fundamental concepts of stochastic dynamical systems and formalize Stochastic Temporal Logic (StTL). In Section 3, we encode the degree of satisfaction of a StTL formula via a robustness estimate and we use this estimate in a stochastic optimization algorithm for controller synthesis. In Section 4, we propose a specification mining algorithm for parametric StTL formulas, which identifies marginally validating valuations. In Section 5, we demonstrate the efficacy of our framework through two case studies inspired by the automotive domain. In Section 6, we give a brief exposition of the related work and conclude this paper.

2 SIGNALS, SYSTEMS AND LOGIC

Stochastic Dynamical Systems. We now introduce fundamental concepts and terminology used throughout the rest of the paper. We consider a *controlled stochastic dynamical system* $M(C)$ as

an abstract model of a physical system whose behavior is regulated with a software controller. The controller $C(\mathbf{q})$ depends on a set of parameters $\mathbf{q}$ known as the *controller parameters*, assumed to belong to some controller parameter space $\mathcal{P}_{\mathbf{q}}$. We abbreviate $\mathcal{M}(C)$ to $\mathcal{M}$ when C is obvious from the context. We assume that, except for the controller parameters $\mathbf{q}$, $\mathcal{M}$ is a black-box model with a *numerical simulator*. This implies that, upon setting values to the controller parameters, we can obtain example behaviors of the system $\mathcal{M}$ without having any explicit representation of the dynamics using the numerical simulator. For example, $\mathcal{M}$ can be considered to be a model that uses proprietary code or model components,¹ which may not provide analytic description of the dynamics of each component. Alternatively, $\mathcal{M}$ could be a physical system whose analytic dynamics are too cumbersome to construct. In both cases the black-box assumption allows to consider a broad class of systems and study them using example behaviors obtained by numerical simulations or physical testing. The term *stochastic* implies that the dynamics of $\mathcal{M}$ is dependent on an input that can be possibly perturbed by the environment, and/or noisy sensors or actuators, and/or internal randomness in the physics that is being modeled. The stochasticity assumption implies that the simulation or testing of the system will give different outputs for the same controller parameters. This output is associated with a *transition function* $\Phi_{\mathcal{M}} : \mathcal{P}_{\mathbf{q}} \times \mathbb{R}^{\geq 0} \rightarrow X$, where $X \subseteq \mathbb{R}^n$ is the output space. Let v be a valuation function mapping the controller parameters $\mathbf{q}$ to some value in $\mathcal{P}_{\mathbf{q}}$. The output of the model is defined by the transition function $x_{v(\mathbf{q})}(t) = \Phi_{\mathcal{M}}(v(\mathbf{q}), t)$ at time $t \in \mathbb{R}^{\geq 0}$. Since $\mathcal{M}$ is dependent on uncertainties in the environment or the system itself, the output $x_{v(\mathbf{q})}(t)$ is essentially a stochastic process. The formal definition of a stochastic process is as follows.

Definition 2.1 (Stochastic Process). A *probability space* is a triple $(\Omega, \mathcal{F}, \mathcal{P})$, where Ω is the set of all possible outcomes, $\mathcal{F}$ is the σ -algebra² of subsets of Ω and $\mathcal{P}$ is a probability measure that associates each subset with its probability (a real number in $[0, 1]$). Given a measurable output space $X \in \mathbb{R}^n$, a random variable x is a measurable function $x : \Omega \rightarrow X$. A *stochastic process* $x(t)$ is a collection of random variables indexed by (discrete or continuous) time. The evolution of $x(t)$ is given by the time varying probability density function (PDF) $p(x, t)$ and its corresponding cumulative distribution function (CDF) $F_x(x, t)$.

The stochastic process³ $x(t)$ gives a complete description of the output of the system. However, exact modelling of $x(t)$ is not only hard to obtain, as argued previously, but could also be impractical as the analytic description could be a highly complicated mathematical expression. Therefore, we rely on approximating $x(t)$ using the output traces given by the black-box simulation or testing of the system. Formally, the output trace is defined as follows:

Definition 2.2 (Output Trace). Let $\mathcal{M} = \mathcal{M}(C)$ be a controlled stochastic dynamical system and $x_{v(\mathbf{q})}(t) = \Phi_{\mathcal{M}}(v(K), t)$ the *output stochastic process* ($x(t)$ for brevity). An *output trace* or realization of this process is the sequence $(t_0, \bar{x}_0), \dots, (t_T, \bar{x}_T)$ obtained by sampling a realization of the stochastic process $x(t)$ at discrete time instances $t_0 < t_1 < \dots < t_T$.

An output trace or signal (the terms are often used interchangeably) is essentially a time-indexed collection of multidimensional measurements of the system variables of interest. We now describe a technique that allows us to learn a stochastic model of the system under consideration from individual realizations of its stochastic output process. Assume that we have tuned the controller parameters to the desired values and use the black-box model of the system to obtain M output

¹For example as a precompiled block within a Simulink® model.

²A collection of all subsets of Ω that is closed under complement, countable unions and intersections.

³When there is no confusion the subscript $v(\mathbf{q})$ may be omitted.

traces of length T , namely $\{(t_0, \bar{x}_0^m), \dots, (t_T, \bar{x}_T^m)\}_{m=1}^M$. Subsequently, for each time instance t_n , using the *ensemble statistics method*, we can estimate the probability distribution of the system output x at time t_n (denoted $p(x, t_n)$) as follows:

Definition 2.3 (Model based on Ensemble Statistics).

$$p(x, t_n) = \frac{1}{Mh} \sum_{m=1}^M \lambda\left(\frac{x - \bar{x}_n^m}{h}\right) \quad (2.1)$$

where λ is a *kernel density function* and $h > 0$ is a smoothing parameter that is chosen empirically. Typically, the kernel function chosen is some symmetric probability density function such as a Gaussian kernel.

This approach makes few assumptions about the dynamics of the stochastic system and relies mainly on our ability to simulate it and obtain output traces. Methods for choosing the kernel as well as the bandwidth are discussed extensively in [42]; we primarily use the Gaussian kernel and the smoothing parameter $h = 1$.

We note that the ensemble statistics method is far from the only technique to obtain a stochastic model from data. A more general method is based on using generalized master equations (GMEs) [44], that are popular modeling formalisms in statistical physics, chemistry and biology. We give a brief background of GMEs and their relation to the ensemble statistics method in the appendix.

Stochastic Temporal Logic. In the context of stochastic dynamical systems, formulating rich and expressive specifications requires that we use a formal language which takes into account the non-deterministic nature of the systemic variables. *Stochastic Temporal Logic*, first introduced in [14] is an amalgam of logics such as StSTL [35] and C2TL [24]. For a random variable x , let $P(x)$ denote its probability. StSTL formulas are defined over atomic predicates represented by *chance constraints* of the form:

$$\mu := P(f(x(t)) \leq c) \geq \epsilon \quad (2.2)$$

where $c \in \mathbb{R}$, $f : X \rightarrow \mathbb{R}$ (f is an invertible real-valued function), $x(t)$ is the random variable associated with the stochastic process x , and $\epsilon \in [0, 1]$. We assume that the time index t is defined over $\mathbb{T}$, which is some subset of positive real numbers. The syntax of StTL is identical to that of STL where atomic predicates⁴ are replaced by chance constraints of the aforementioned form:

$$\varphi := \mu \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \mathbf{U}_I \varphi \quad (2.3)$$

where I is a time interval, i.e., any convex subset of $\mathbb{T}$. The truth value of the atomic predicate is interpreted based on the satisfaction properties of the atomic predicate, namely it is true (denoted with $\top$) if and only if $f(x(t)) \leq 0$ holds with probability at least ϵ and false (denoted with $\perp$) otherwise.

Definition 2.4 (Recursive Semantics of StTL). The semantics of StTL define what it means for an StTL formula to be satisfied by a given stochastic process x :

$$\begin{aligned} (x, t) \models \mu &\equiv P(f(x(t)) \leq c) \geq \epsilon \\ (x, t) \models \neg\varphi &\equiv \neg((x, t) \models \varphi) \\ (x, t) \models \varphi_1 \wedge \varphi_2 &\equiv (x, t) \models \varphi_1 \wedge (x, t) \models \varphi_2 \\ (x, t) \models \varphi \mathbf{U}_I \psi &\equiv \exists t' \in t \oplus I : ((x, t') \models \psi \wedge \\ &\quad \forall t'' \in [t, t') : (x, t'') \models \varphi) \end{aligned} \quad (2.4)$$

⁴StTL can be generalized to include chance constraints over arbitrary Boolean conjunctions of predicates over the given stochastic process, see [14] for the precise syntax. All the algorithms in this paper can be extended to the full StTL syntax; we omit this extension for ease of exposition.

In the above definition, the $\oplus$ operator denotes the Minkowski sum of the interval with t , i.e., if the interval $I = [a, b]$, then $t \oplus I$ denotes the interval $[t + a, t + b]$. The formulas $F_I\varphi$ and $G_I\varphi$ are introduced as short-hand for $\text{true}U_I\varphi$ and $\neg F_I\neg\varphi$, respectively. Following the usual convention for logics such as STL, we write $x \models \varphi$ as a short-hand notation for $(x, 0) \models \varphi$. We now define the Boolean satisfaction signal, which we extend to the quantitative semantics in the next section.

Definition 2.5 (Satisfaction Signal). The *satisfaction signal* for an StTL formula φ is defined as a function from $\mathbb{T}$ to $\{\top, \perp\}$ such that:

$$\chi(\varphi, x, t) = \begin{cases} \top & \text{iff } (x, t) \models \varphi \\ \perp & \text{otherwise} \end{cases} \quad (2.5)$$

Parametric Stochastic STL (PSStL) is an extension of StTL introduced to define *template formulas*. Essentially, a template formula is an StTL formula in which some numeric constants are replaced by *formula parameters*. Let π be a set of *formula parameters*. The set π can be partitioned into three disjoint sets of variables π_c , π_e , and π_τ , of which at least one is nonempty. The variables in π_c , π_e and π_τ are called *value parameters*, *probability threshold parameters* and *time domain parameters*, respectively; they take values in some compact space $\mathcal{P}_{\pi_c}$, $[0, 1]$, and $[0, t_T]$. As for control parameters, we use v to denote a valuation function v that maps any of the above formula parameters to a value in their respective domains. An StTL formula φ is obtained by pairing a PSStL formula $\varphi(\pi)$ with a valuation function v .

Problem Definitions. We outline two main problems that motivate the work in this paper.

- (1) **Control Design Problem:** Given an StTL formula φ , find a valuation v^* for the *controller parameters* $\mathbf{q}$ such that the controlled stochastic output process corresponding to $\mathcal{M}(C(v^*(\mathbf{q})))$ satisfies the specification φ .
- (2) **Specification Mining Problem:** Given a PSStL formula $\varphi(\pi)$, and a **fixed valuation** v_{fixed} for the controller parameters $\mathbf{q}$ of the controlled model $\mathcal{M}$, identify valuations v^* for the *formula parameters* π such that $\mathcal{M}(C(v_{\text{fixed}}))$ satisfies $\varphi(v^*(\pi))$.

3 CONTROLLER DESIGN VIA ROBUSTNESS

The main objective of this section is to propose a method to address the control design problem. At the outset, we state (without proof) that the control design problem is undecidable in general. The proof follows from the undecidability of the verification problem for arbitrary dynamical systems with specifications that are simple subset of STL [5]. Hence, we develop a best effort technique that attempts to design a robust controller that maximally (but not provably) tries to satisfy a given StTL specification. A key element of this technique is to use a quantitative measure of the satisfaction of a given StTL specification. We call this measure the *robustness estimate*. The precise definition of the robustness estimate is provided by *quantitative semantics* for StTL. To define the quantitative semantics, we define a *parameterization operation* that converts a given StTL formula to a PSStL formula. We assume an unbounded supply of *formula parameter variables*, and replace each numeric constant appearing in the StTL formula with a formula parameter. For brevity, we skip the formal definition. Furthermore, for ease of exposition, we assume an empty π_τ , i.e. we do not replace any intervals appearing in the temporal operators.

Robustness Definition. We assume that the co-domain X of the random variables constituting the stochastic output process of the model $\mathcal{M}$ is a metric space. This allows us to define a metric d_X over X . We can use the standard Euclidean distance as a metric for both X and $[0, 1]$, i.e., the domain for the probability threshold parameters. Suppose the StTL formula has m value parameters and m probability threshold parameters, i.e. $\mathcal{P}_\pi \subseteq (X^m \cup [0, 1]^m)$. We can extend the metric d_X over

X to a metric $d_{\mathcal{P}_\pi}$ over the set $\mathcal{P}_\pi$ in standard fashion. Given a valuation v mapping π to $\mathcal{P}_\pi$, we define an open ε -ball (denoted $B(v, \varepsilon)$) around the valuation v as the set of all valuations v' such that $d_{\mathcal{P}_\pi}(v(\pi), v'(\pi)) < \varepsilon$. The quantitative semantics are then defined in terms of the *distance* of the given StTL formula to the *satisfaction boundary*.

Definition 3.1 (Satisfaction Boundary). Let $\varphi(\pi)$ be a PStTL formula. For a given stochastic process, the formula $\varphi(\pi)$ partitions $\mathcal{P}_\pi$ into disjoint subsets $\mathcal{P}_\pi^\top$, $\mathcal{P}_\pi^\perp$ and $\mathcal{D}^\mathcal{B}$ such that:

$$\begin{aligned} \forall v(\pi) \in \mathcal{P}_\pi^\top, \exists \varepsilon > 0 : x \models \varphi(v(\pi)) \wedge \forall v' \in B(v, \varepsilon) : x \models \varphi(v'(\pi)) \\ \forall v(\pi) \in \mathcal{P}_\pi^\perp, \exists \varepsilon > 0 : x \not\models \varphi(v(\pi)) \wedge \forall v' \in B(v, \varepsilon) : x \not\models \varphi(v'(\pi)) \\ \mathcal{P}_\pi^\mathcal{B} = \mathcal{P}_\pi \setminus (\mathcal{P}_\pi^\top \cup \mathcal{P}_\pi^\perp) \end{aligned}$$

The set $\mathcal{P}_\pi^\mathcal{B}$ is called *satisfaction boundary*. By convention, we assume that for all $v(\pi) \in \mathcal{P}_\pi^\mathcal{B}$, $x \models \varphi(v(\pi))$. In other words, points on the boundary are assumed to satisfy the formula.

Essentially, the sets $\mathcal{P}_\pi^\top$ and $\mathcal{P}_\pi^\perp$ are defined as open sets containing satisfying and violating valuations, respectively, and $\mathcal{P}_\pi^\mathcal{B}$ is the intersection of the closure of these two sets. We now define the notion of distance to boundary:

Definition 3.2 (Distance to Boundary). For a given valuation v , the *distance to boundary* $d_\mathcal{B}$ is defined as follows:

$$d_\mathcal{B}(v) = \min \{d_{\mathcal{P}_\pi}(v, v') \mid v' \in \mathcal{P}_\pi^\mathcal{B}\} \quad (3.1)$$

Observe that the above minimization is a convex optimization problem subject to equality constraints, which can be solved using Karush-Kuhn-Tucker (KKT) conditions [9]. However, solving this optimization problem requires knowledge of the satisfaction boundary, which, even in the case of STL, is a computationally challenging problem [36]. Thus, we propose a technique to approximate the quantitative semantics that allow us to estimate this distance. We first define the notion of the robustness signal for a single atomic predicate in Definition 3.3. In the definition below, we recall that $F_x(x, t)$ denotes the cumulative distribution function of x at time t . Also, f^{-1} denotes the inverse function of f that appears in the signal predicate.

Definition 3.3 (Predicate Robustness). Consider an atomic predicate of PStTL: $\mu := P(f(x(t)) \leq c) \geq \varepsilon$. Let v be some fixed valuation. We define a valuation $v^\star$ as follows:

$$v^\star(\varepsilon_B) = F_x(f^{-1}(v(c)), t) \quad (3.2)$$

$$v^\star(c_B) = f(F_x^{-1}(v(\varepsilon), t)) \quad (3.3)$$

The first expression above gives the valuation for the parameter ε when $f(x(t))$ is exactly $v(c)$. The second one gives the valuation of the parameter c for which the probability of $f(x(t)) \leq v^\star(c)$ is exactly $v(\varepsilon)$. Then, the predicate robustness is defined as follows:

$$\rho_\mu(x, t) = (\Delta c, \Delta \varepsilon) = (v^\star(c_B) - v(c), v^\star(\varepsilon_B) - v(\varepsilon)) \quad (3.4)$$

LEMMA 3.4. *The parameter valuations $v' = (v^\star(c_B), v(\varepsilon))$ and $v'' = (v(c), v^\star(\varepsilon_B))$ both belong to the satisfaction boundary.*

We omit a formal proof, but essentially observe that $v^\star$ as defined maps both c and ε to values that are marginally satisfied by the atomic predicate. Any negative perturbation to v' or positive perturbation to v'' must lead to the corresponding atomic predicates no longer being satisfied by the stochastic process x . Thus, by definition, as v' and v'' do not have an equi-satisfiable open neighborhood, they must lie on $\mathcal{D}_\mathcal{B}$. We illustrate the idea for the predicate robustness signal for a fixed time t in Figure 2. We are now ready to define the robustness signal for an arbitrary StTL formula.

Fig. 2. Distance to boundary and predicate robustness.

Definition 3.5 (Robustness Signal). Let $\rho_\mu(x, t)$ denote the predicate robustness signal for predicate μ and v be a valuation function that assigns a value to each parameter in the PStTL formula $\varphi(p)$. The robustness signal $\rho(\varphi, x, t)$ is defined recursively as follows:

$$\begin{aligned}
 \rho(\text{true}, x, t) &= (\infty, 1 - \epsilon) \\
 \rho(\mu, x, t) &= \rho_\mu(x, t) \\
 \rho(\neg\varphi, x, t) &= -\rho(\varphi, x, t) \\
 \rho(\varphi_1 \wedge \varphi_2, x, t) &= \min(\rho(\varphi_1, x, t), \rho(\varphi_2, x, t)) \\
 \rho(\varphi \mathbf{U}_I \psi, x, t) &= \sup_{t' \in t \oplus I} (\min(\rho(\psi, x, t'), \\
 &\quad \inf_{t'' \in [t, t']} \rho(\varphi, x, t''))
 \end{aligned} \tag{3.5}$$

where $\rho_\mu(x, t)$ is the predicate robustness as given by Definition 3.4.

We remark that the robustness computation looks very similar to that of STL, but there are some nuances to keep in mind. Consider the formula $\neg(P(x(t) > c) > \epsilon)$. Note that this formula is equivalent to $P(x(t) > c) \leq \epsilon$. In other words, the signal predicate remains unchanged, only the comparison to the probability threshold gets negated. However, we can show by a simple argument, that if the robustness of $P(x(t) > c) > \epsilon$ is $(\Delta c, \Delta \epsilon)$, then the robustness of $P(x(t) > c) \leq \epsilon$ is indeed $(-\Delta c, -\Delta \epsilon)$, or as defined in Equation (3.5).

In other words, the definition of robustness as a signed metric preserves the Boolean satisfaction semantics of StTL. Also for the formula *true*, the robustness (in the value domain) is infinite, and any given valuation for the probability threshold ϵ is $1 - \epsilon$ away from the satisfaction boundary. Given that the predicate robustness is a pair (Definition 3.4), the remaining operators in the definition above are applied element-wise to each element of the tuple.

Robustness estimate computation. Definition 3.5 gives a recursive recipe to compute the robustness estimate for a given StTL formula. We note that it suffices to compute the robustness for three operators: $\neg, \wedge, \mathbf{U}_I$, as all the other Boolean and temporal operators can be expressed as functions of these. We note that the quantitative semantics defines operations similar to robustness estimation algorithms for STL, and can thus we can leverage the efficiency and data structures used for monitoring of STL formulas [16]; with the crucial difference being the computation of the predicate robustness.

Remark 3.1 (Spatial vs. Temporal Robustness). We observe that by omitting time-domain parameters, our definition of the robustness signal currently includes only spatial robustness. If we include time-domain parameters, we can also recover a notion of time-robustness of the given stochastic model to an StTL formula. This notion of time-robustness would support dilations or contractions

of time intervals of temporal operators to find the satisfaction boundary. For ease of exposition, we defer a full treatment to future work.

Robustness-Based Synthesis. We now present our solution for synthesizing control parameters for a given controller. We make a crucial observation: the robustness estimate preserves Boolean satisfaction of an StTL formula φ , i.e. if both elements of the robustness estimate pair are positive, then φ is satisfied by the given stochastic process x . Thus, given a controlled stochastic model $\mathcal{M}(C)$, maximizing the robustness estimate of its output w.r.t. a given StTL specification φ , (by exploring the control parameter space) pushes the model $\mathcal{M}(C)$ towards satisfying φ .

Formally, let $x_{v(q)}(t)$ denote the output stochastic process, then the controller synthesis problem is formalized as follows:

$$v^*(q) = \arg \max_{v(q) \in \mathcal{P}_q} \rho(\varphi, x_{v(q)}) \quad (3.6)$$

To avoid requiring to place restrictions on the system or the formula, we address this problem using stochastic optimization, more specifically we adopt a Particle Swarm Optimization (PSO) approach. PSO is a prominent optimization method that does not require the objective function to be differentiable, which perfectly fits our problem as the robustness signal is not differentiable in general. PSO works by keeping track of a population of S candidate controller parameters (called particles). These particles are moved around in the controller parameter space using simple formulas. These movements are guided by each particle's best known position and velocity as well as the swarm's best known position. Algorithm 1 gives the pseudo-code of PSO for the robustness maximization problem given by Equation (3.6). PSO is a fairly simple algorithm: we pick a certain number of particles randomly initialized to a "position". We use the function `uniform_random` to indicate uniform random sampling in the interval represented by its arguments. The position of each particle represents a valuation of the controller parameters. For each particle we track its best known position, i.e., the valuation for the particle that yields the best robustness estimate. Additionally, we also track the best position in the entire swarm of particles. Each particle position is updated randomly based on hyper-parameters ω_v, ω_p and ω_g . Although PSO can deal with unbounded controller parameter spaces, in practice it is often preferable to place a lower bound v_ℓ and an upper v_u bound on the controller parameter valuations. Additionally, the hyper-parameters $\omega_v, \omega_p, \omega_g$ are design choices that control the performance and efficacy of the method. For methods to choose these parameters the interested reader could look into [41]. Finally, the termination condition could be the maximum number of iterations or a suitable threshold ε on the robustness value.

4 SPECIFICATION MINING FOR STTL

In this section, we address the problem of specification mining for a given PStTL formula. Recall that in this problem, we are given a controlled stochastic system $\mathcal{M}(C)$, where the controller parameters are fixed, i.e., $C = C(v^*(q))$, for some fixed v^* . Recall that we denote the stochastic output process of the model by $x_{v^*(q)}(t)$, or $x(t)$ in brief. Given a PStTL formula $\varphi(\pi)$, the specification mining problem tries to infer the valuation v^* for π such that the resulting StTL formula $\varphi(v^*(\pi))$ is marginally satisfied by $x(t)$. This problem is closely related to finding *under-approximations* for the sets $\mathcal{P}_\pi^\top$ and $\mathcal{P}_\pi^\perp$, and thus, an *over-approximation* for $\mathcal{P}_\pi^\mathcal{B}$. A brute-force approach for finding the satisfaction boundary is to consider a finite set of points G in the parameter space (for example a grid), and define a function S that maps each valuation v corresponding to a point in G to $\top$ if it belongs to $\mathcal{P}_\pi^\top$, and to $\perp$ if it belongs to $\mathcal{P}_\pi^\perp$. Consider the following sets:

$$\begin{aligned} \widehat{\mathcal{P}}_\pi^\top &= \{v \mid v \in G \wedge x \models \varphi(v(\mathbf{p}))\} \\ \widehat{\mathcal{P}}_\pi^\perp &= \{v \mid v \in G \wedge x \not\models \varphi(v(\mathbf{p}))\} \end{aligned} \quad (4.1)$$

ALGORITHM 1: PSORobustness($\rho, S, v_\ell, v_u, \omega_v, \omega_p, \omega_g$)**Procedure :**

```

1: numIterations  $\leftarrow 0$ 
2: Initialize swarm's best position  $v^* \sim \text{uniform\_random}(v_\ell, v_u)$ 
3: for each particle  $i = 1, \dots, S$  do
4:   Initialize particle position  $v_i \sim \text{uniform\_random}(v_\ell, v_u)$ 
5:   Initialize best particle position  $v_i^* \leftarrow v_i$ 
6:   if  $\rho(\varphi, x_{v_i(q)}) > \rho(\varphi, x_{v^*(q)})$  then
7:     Update swarm's best position  $v^* \leftarrow v_i^*$ 
8:   end if
9:   Initialize velocity  $v_i \sim \text{uniform\_random}(-|v_u - v_\ell|, |v_u - v_\ell|)$ 
10: end for
11: while ( $\rho(\varphi, x_{v^*(q)}) < \varepsilon$ ) or (numIterations < maxIterations) do
12:   for each particle  $i = 1, \dots, S$  do
13:     for each dimension  $d = 1, \dots, |q|$  do
14:       Generate random numbers  $r_p, r_g \sim \text{uniform\_random}(0, 1)$ 
15:        $\bar{v}_{i,d} \leftarrow \omega_p r_p (v_{i,d}^* - v_{i,d}) + \omega_g r_g (v^* - v_{i,d})$ 
16:        $v_{i,d} \leftarrow \omega_v v_{i,d} + \bar{v}_{i,d}$ 
17:     end for
18:      $v_i \leftarrow v_i + v_i$ 
19:     if ( $\rho(\varphi, x_{v_i(q)}) > \rho(\varphi, x_{v_i^*(q)})$ ) then
20:       Update particle's best position  $v_i^* \leftarrow v_i$ 
21:       if ( $\rho(\varphi, x_{v_i^*(q)}) > \rho(\varphi, x_{v^*(q)})$ ) then
22:         Update swarm's best position  $v^* \leftarrow v_i^*$ 
23:       end if
24:     end if
25:   end for
26:   numIterations  $\leftarrow$  numIterations + 1
27: end while
28: return  $v^*$ 

```

An attractive option is then to somehow seek to generalize the sets $\widehat{\mathcal{P}}_\pi^\top$ and $\widehat{\mathcal{P}}_\pi^\perp$ to approximate the satisfaction boundary. However, for general PStTL formulas, given two given satisfying (resp. violating) valuations, there may be convex combinations of these valuations that are not satisfying (resp. violating).

Example 4.1. Consider the formula $G_{[0,10]}P(x_1 > c) \geq 0.5 \wedge P(x_2 \leq c) \geq 0.5$. Let $u(x)$ denote the Heaviside step function, i.e. $u(x) = 1$ if $x \geq 0$ and 0 otherwise. Let CDFs for x_1 and x_2 be respectively $u(x_1 - 2)$ and $u(x_2 - 1)$ for all time t . Then, the formula is satisfied if $v(c) \in (1, 2]$, but is not satisfied otherwise. A sampling method that only considers valuations $v(c) \leq 1$ and $v(c) > 2$, would miss the satisfying region altogether!

In addition to the unsatisfying mathematical properties of the sampling-based approach, it can be a computationally expensive procedure. For example, using a grid-based approach, where we pick M grid points along each parameter dimension, requires the satisfaction of $\varphi(\pi)$ to be evaluated at $M^{|x|}$ points. An identical problem has been identified for parameter inference for STL, where it has been shown in [4, 26] that for the *monotonic* fragment of Parametric-STL, the inference problem can be efficiently solved. Furthermore, in [34, 36], the authors have shown that if

the satisfaction function is monotonic the satisfaction boundary can be efficiently approximated to a desired degree of precision. We now show that for the monotonic fragment of PStTL, the satisfaction boundary can also be computed efficiently. We also propose a new specification mining algorithm for a reduced subset of PStTL (where each parameter name is unique and there are no time-domain or probability threshold parameters), the satisfaction boundary can be computed *exactly*.

Definition 4.2 (Monotonic PStTL). A PStTL formula φ is said to be monotonically increasing in parameter π_i if condition (4.2) holds for any stochastic process x and is said to be monotonically decreasing in parameter π_i if condition (4.3) holds for every x .

$$v(\pi_i) \leq v'(\pi_i) \wedge x \models \varphi(v(\pi_i)) \Rightarrow x \models \varphi(v'(\pi_i)) \quad (4.2)$$

$$v(\pi_i) \geq v'(\pi_i) \wedge x \models \varphi(v(\pi_i)) \Rightarrow x \models \varphi(v'(\pi_i)) \quad (4.3)$$

A multi-parameter formula $\varphi(\mathbf{p})$ is said to be monotonic if and only if it is monotonic in each parameter $\pi_i \in \mathbf{p}$.

In case of STL, it has been shown by several previous works [4, 22, 26, 39] that the monotonic fragment of STL captures a large number of useful properties in practice. We argue that by extension monotonic PStTL formulas also capture a broad class of interesting uncertainty-aware temporal specifications. We note that by restricting our attention to monotonic PStTL formulas, it is not possible to express formulas such as those that reuse a single formula parameter in several signal predicates, or interval bounds on temporal operators where both bounds of the interval are timed formula parameters.

In what follows, we assume without loss of generality that the given formula is monotonically increasing with respect to each parameter. (There is no loss of generality because we can always replace a monotonically decreasing parameter π_j with a new parameter $\pi'_j = -\pi_j$.) We now present our specification mining algorithm, which is based on the observation that, given the monotonicity assumption, it suffices to find the parameter valuation at which the validity of the formula changes.

Least Permissive Valuation Signal. We assume that the index set of the stochastic process is some sequence of discrete time-indices, i.e., $\mathbb{T} = \{t_0, \dots, t_T\}$, where $t_0 = 0$. Our algorithm performs a recursive traversal of the PStTL formula. For each subformula, the algorithm iterates over all the time-points in the *horizon* of the subformula to compute the *least permissive valuation* signal. This signal, denoted $\rho_{lpv}(\psi, \pi, t)$ gives the *least valuation* v^* for parameters in π such that $\psi(v^*(\pi))$ is satisfied by x . We include the formula parameter π in defining $\rho_{lpv}(\psi, \pi, t)$ as a way of indicating that the ρ_{lpv} function will return a valuation for the specified formula parameters in π at time t . We now present the details of computing $\rho_{lpv}(\psi, \pi, t)$.

If the subformula μ is an atomic predicate $\mu = P(x(t) \leq c) \geq 1 - \delta$, (where $\delta \in [0, 1]$ is some numeric constant), then, $\rho_{lpv}(\mu, c, t)$ is the smallest valuation $v(c)$ such that $P(x(t) \leq v(c)) \geq 1 - \delta$ is true. Such a valuation can be easily computed from the inverse CDF of x for the $1 - \delta$ value, as shown in the first case. For other subformulas ψ of the given formula, the $\rho_{lpv}(\psi, \pi, t)$ signal is calculated as shown in Algorithm 2. For a negation, the set of formula parameters for which ρ_{lpv} is returned remains unchanged, but the sign of the least permissive values is flipped. For Boolean combinations, as the formula parameters in each subformula are disjoint, ρ_{lpv} returns the concatenation of least permissive valuations for both subformulas. For temporal operators, the least permissive valuation requires further explanation. For ease of exposition, we first explain the special cases for G and F operators. Consider the formula $G_I P(x \leq c) > 1 - \delta$, where $\delta \in [0, 1]$, and $I \subseteq \mathbb{T}$ are known constants. At each time, the ρ_{lpv} signal for the atomic predicate (μ for short) will return the smallest value of c that makes the predicate satisfied at that time. Now, to

satisfy the outer G operator, we need to pick the *maximum* among the least permissive valuations for c in the interval I . This will guarantee that the $G\mu$ formula holds at time 0. Similarly, for a formula of the type F_I , we should choose the *minimum* value for c over the interval I . Now, to compute the least permissive valuation for a subformula of the type $\varphi_1(\pi_1)U_I\varphi_2(\pi_2)$, we first note that the ρ_{lpv} signal will consist of concatenation of least permissive values of π_1 and π_2 . These values can be computed from the semantics of U from a fashion similar to that for G and F .

ALGORITHM 2: Compute $\rho_{lpv}(\psi, \pi, t)$

Procedure :

```

1: switch  $\psi(\pi)$  do
2:   case  $P(f(x) < c) \geq 1 - \delta$ 
3:      $\forall t : \rho_{lpv}(\psi, c, t) = f(F_x^{-1}(1 - \delta, t))$ 
4:   case  $\neg\varphi(\pi)$ 
5:      $\forall t : \rho_{lpv}(\psi, \pi, t) = -\rho_{lpv}(\varphi, \pi, t)$ 
6:   case  $\varphi_1(\pi_1) \wedge \varphi_2(\pi_2)$ 
7:      $\forall t : \rho_{lpv}(\psi, (\pi_1 \cup \pi_2), t) =$ 
8:        $\rho_{lpv}(\varphi_1, \pi_1, t) \cup \rho_{lpv}(\varphi_2, \pi_2, t)$ 
9:   case  $\varphi_1(\pi)U_I\varphi_2(\pi)$ 
10:     $\forall t : v_1(t) = \rho_{lpv}(\varphi_1, \pi_1, t)$ 
11:     $\forall t : v_2(t) = \rho_{lpv}(\varphi_2, \pi_2, t)$ 
12:     $\forall t : \rho_{lpv}(\psi, (\pi_1 \cup \pi_2), t) =$ 
13:       $(\max_{t'' \in [t, t']} v_1(t''), \min_{t' \in t \oplus I} v_2(t'))$ 

```

THEOREM 4.3. Let $x(t)$ be a stochastic process, $\psi(\pi)$ be a monotonically increasing stochastic STL formula parametrized by a set of formula parameters belonging to $\mathcal{P}_\pi$, and let $[0, T]$ be a discretized time interval. Let $<$ denote a total order on the formula parameter space that respects the monotonicity order (i.e. $v(\pi) < v'(\pi)$ is equivalent to $v(\pi) < v'(\pi)$ when π is monotonically increasing, and is equivalent to $v(\pi) > v'(\pi)$ when it is monotonically decreasing). Algorithm 2 gives a unique partitioning of $\mathcal{P}_\pi$ defined as follows:

$$\mathcal{P}_\pi = \mathcal{P}_\pi^\top \cup \mathcal{P}_\pi^\perp \cup \mathcal{P}_\pi^\mathcal{B} \quad (4.4)$$

$$\forall v \in \mathcal{P}_\pi^\top, \forall v' \in \mathcal{P}_\pi^\perp : v' < v \quad (4.5)$$

$$\mathcal{P}_\pi^\mathcal{B} = \rho_{lpv}(\psi, \pi, 0) \quad (4.6)$$

PROOF. The proof follows from the definition of the $<$ relation, and the following observation:

$$\forall v \in \mathcal{P}_\pi^\top, \forall v' \in \mathcal{P}_\pi^\perp, \forall v^* \in \mathcal{P}_\pi^\mathcal{B} : v' < v^* < v \quad \square$$

The complexity of the proposed algorithm is $O(|\pi|T)$, radically better than the $O(M^{|\pi|})$ complexity of the aforementioned grid method. Our algorithm exploits the monotonicity in order to find the boundary at which the formula transits from true to false without redundantly searching over all the parameter space. In addition to that, we should also note that in real-world applications we are primarily interested in finding the satisfaction boundary rather than any validating or falsifying parameter valuation. For example, consider the formula:

$$\varphi = G_{[0, \infty]}(P[x < c] > 0.99) \quad (4.7)$$

where x is a stochastic process that models the velocity of a car. Obviously, by setting $c = 200$ mph, we would obtain a valid formula. However, such a valuation is not informative about the behavior

of the car. It would be much more informative to know that the velocity never exceeds 60 *mph* rather than it never exceeds 200 *mph*. Therefore, the most informative valuations are those that lie on the satisfaction boundary, which is exactly what Algorithm 2 returns.

Robustness and Boundary estimation. We remark that computing the robustness w.r.t. a given parameter valuation for a PStTL formula is a straightforward implementation of the quantitative schematics given in Equation (3.5). In other words, it computes $(\Delta c, \Delta \epsilon)$ for a *single* point $(v(c), v(\epsilon))$ in the parameter space. However, suppose we are given a PStTL formula that effectively defines a family of StTL formulas (by changing the valuations for the formula parameters). Suppose we wish to compute the robustness estimate of the given stochastic model at each of these StTL formulas. Then, repeated application of the above procedure would be an inefficient way to obtain the robustness values. Instead, if we have full knowledge of the satisfaction boundary of the PStTL formula, then obtaining the robustness algorithm for *any* single valuation in the formula parameter space becomes a constant-time lookup operation from the known boundary.

5 EXPERIMENTAL RESULTS

To validate the above-mentioned formalism, we consider an experimental analysis consisting of two parts. In the first part, we verify the correctness of the specification mining algorithm proposed in Section 4. In the second case study, we apply our unified framework for mining specifications and designing a controller in a real-world system, more specifically in a Model Predictive Adaptive Cruise Controller (block diagram shown in Figure 4).

5.1 Evaluation of the Specification Mining Algorithm

For the first case study, we performed experimental measurements of sensor readings, actuator inputs, driver actions and environmental conditions in a vehicle driven by one of the co-authors. We recorded several descriptive quantities of the vehicle's behavior, including *position, velocity, acceleration, fuel consumption, throttle position, power and torque output*. We use a standard off-the-shelf OBD (On-board Diagnostics) port logging device to obtain this data. We illustrate our proposed specification mining algorithm by using the measured vehicle speed, which is modeled as a stochastic process $x(t)$. To obtain an approximation of the PDF $p(x, t)$ of the vehicle's velocity we applied the ensemble statistics method using a Gaussian function as kernel. Following that, we define the following parametric StTL formula for the vehicle's speed:

$$\phi_1(c_1, c_2) = G_{[0,30]}(P[x \geq c_1] \geq 0.9 \implies F_{[0,10]}P[x \leq c_2] \geq 0.9) \quad (5.1)$$

To find the parameter valuations which validate or falsify the formula, we implemented and tested two methods:

- (1) *Grid Method*: We chose c_1 and c_2 to be in the range $[0, 20]$ and $[0, 45]$ respectively and sampled 100 points along each dimension. Then, we created a 2 dimensional grid points and tested the validity of the formula at each one of them. This required 100^2 computational steps.
- (2) *Specification Mining (Algorithm 2)*: We can easily verify that the formula is monotonic with respect to both c_1 and c_2 . Therefore, the assumptions of Algorithm 2 are satisfied. To test the algorithm we discretized the time horizon ($T = 30s$) into 100 time instances and applied Algorithm 2. Since the formula contains $|P| = 2$ parameters, this required $2 \cdot 100$ computational steps.

Fig. 3. Comparison of Algorithm 2 against the grid method. Red denotes the satisfaction boundary.

Fig. 4. Block diagram of the MPC-based ACC system.

The results of the two methods are shown in Figure 3. Observe that the satisfaction boundary given by Algorithm 2 approximates the one given by the grid method almost perfectly⁵ This experimental result validates the accuracy of our specification mining algorithm.

5.2 MPC-based Adaptive Cruise Controller

The second case study is inspired by the challenges arising in the semi-autonomous vehicle domain. Our experimental setup consists of a *Lead* and an *Ego* car that are traveling on the same lane of the freeway. The state of the ego (resp. lead) car is the position $p_e(t)$ (resp. $p_l(t)$) and the velocity $v_e(t)$ (resp. $v_l(t)$). The control input to the ego car is the acceleration $a_e(t)$ which is given by a Model Predictive Controller (MPC) based Adaptive Cruise Controller (ACC). Although the actual acceleration $a_l(t)$ of the lead vehicle is unknown, the values of the relative distance $d_{rel}(t)$ and relative velocity $v_{rel}(t)$ are estimated by on-board sensors installed on the ego car. These estimates are corrupted by Gaussian sensor noise of unknown statistics. For both the ego vehicle and the lead vehicle, the dynamics between acceleration and velocity are modeled by the following transfer function:

$$G(s) = \frac{1}{s(0.5s + 1)} \quad (5.2)$$

which approximates the dynamics of the throttle body and vehicle inertia [43]. The ACC operates in two distinct modes:

- (1) if $d_{rel}(t) > d_{safe}(t)$ the speed control mode is active. The goal is to reach and maintain a reference velocity v_{ref}

⁵There are a couple of discrepancies between satisfaction boundary computed and its depiction in the figure. First, in plotting the results of the grid method, we assumed the formula to be conservatively false at all points where the satisfaction value was unknown. Thus, the boundary indicated by the colors underapproximates the validity domain. The second discrepancy is that at a few points, the red line seems to be above a point indicated as a true valuation by the grid method. This is due to the resolution of plotting.

Fig. 5. Mined Parameters for a conservative ($K = 0.1$), balanced ($K = 0.5$) and aggressive ($K = 0.9$) controller.

- (2) if $d_{rel}(t) < d_{safe}(t)$ the spacing control mode is active. The goal is to maintain the safe distance d_{safe}

The safe distance between the lead car and the ego car is a function of the ego car velocity:

$$d_{safe}(t) = d_{default} + t_g v_{ego}(t) \quad (5.3)$$

where $d_{default}$ is the preset default spacing between the cars and t_g is the gap time. A common rule of thumb for the gap time is the *two second rule* which states that the ego car should ideally stay 2 seconds behind the lead car. The default spacing was set at 10 meters. The MPC-based ACC computes optimal control actions such that the system obeys the constraints imposed by physical restrictions on the velocity and acceleration and adapts to the driving behavior of the lead car according to the two modes of operation. The behaviour of the controller $C(K)$ is adjusted by the parameter $K \in [0, 1]$, whereby smaller values produce a more conservative controller with smoother control actions and larger values produce a more aggressive controller with a faster response time. We implemented the MPC ACC system in Simulink and extracted the PDFs of the quantities of interest (position and velocity) by simulating the model $M = 100$ times and using the ensemble statistics method.

Specification Mining. In this part of the experimental study, we address the specification mining problem. Since safety is a critical component of such a system, we chose to test our specification mining algorithm using the safety formula φ_{safe} as indicated in Equation (5.5).

$$\psi := \left(P[d_{rel} \leq c_1] \geq 0.9 \implies F_{[0,10]} G_{[0,\infty]} (P[v_e \leq c_2] \geq 0.9) \right) \quad (5.4)$$

$$\varphi_{safe} := G_{[0,80]} \psi \quad (5.5)$$

This formula states that whenever the relative distance drops below the threshold c_1 with probability at least 0.9, then in the subsequent 10s the velocity of the ego vehicle should drop below c_2 with probability at least 0.9. Note that this formula is monotonically increasing with respect to both parameters. For the purposes of this case study we tested three different controller behaviors, namely a conservative ($K = 0.1$), a balanced ($K = 0.5$) and an aggressive ($K = 0.9$) controller. For purpose of illustrating our mining algorithm, we indicate the least permissive valuations for the parameters of the sub-formula ψ shown in Equation (5.4) in Figure 5, i.e., instead of plotting the satisfaction boundary in the c_1, c_2 -parameter space, we plot the pair of parameters that lead to a valid formula over time. This implies that, at each time instance, every pair of values c_1, c_2 as shown in Figure 5 is a validating valuation. The relation between the controller parameter K and the formula parameters c_1 and c_2 can be observed empirically through the figures. The balanced

Fig. 6. Satisfaction and Robustness signals for Equation (5.5) ($c_1 = 60m$, $c_2 = 30m/s$). The satisfaction signals overlap, and hence only one of them is visible.

controller is safer with respect to this specification as for higher bounds on the relative distance it leads to lower bounds on the velocity of the ego vehicle.

Robustness Computation. In addition to mining specifications, we also tested the robustness computation according to the schematics given by Equation (3.5) for the three aforementioned controller parameter valuations. To this end, we fixed the parameters of the formula in Equation (5.5) to $c_1 = 60m$, $c_2 = 30m/s$. These values are chosen such that they obey the *two second rule*. Looking at Figure 5 we observe that the mined parameter c_1 reaches the value of $60m$ in the time interval $[40, 50]$ and in that interval the value of mined parameter c_2 lies well below $30m/s$ for all three controllers. Given that the formula is monotonically increasing we expect that the aforementioned chosen parameters will lead to a valid formula. This is indeed verified by the satisfaction and robustness signals shown in Figure 6. In addition to that, we verify that the balanced controller performs better with respect to this specification as it leads to higher robustness values.

Robust Design. For the last part of the experimental results we test the controller design method. Specifically, we solve the optimization problem stated in Equation (3.6) using the Particle Swarm Optimization approach. We utilize the robustness of the PStL formula given by Equation (5.5) in the objective function and we seek to find the controller parameter valuation K^* which maximizes the average robustness. To test the design method for different formula parameters, we discretize c_1 and c_2 by sampling uniformly 10 points for each of them in the intervals $[10, 100]$ and $[5, 30]$, respectively. We set the maximum tolerance on the objective function change to 10^{-6} and run all the simulations on an Intel Core i7-7700HQ @ 2.8GHz with 16 GB of RAM. The run-time for all parameter values ranged from 0.2247s to 0.2840s. The results for the optimal controller parameter value K^* are shown in Figure 7. Note that the points have been interpolated for the ease of presentation. Figure 7 shows the power of using a robustness-guided control design; in spite of uncertainty in the environment, for each parameterized specification, we can pick a controller parameter K^* for which the system demonstrates good behavior.

6 RELATED WORK AND CONCLUSIONS

Signal Temporal Logic [37] has been used for expressing properties and abstractions of closed-loop cyber-physical system models [22, 25, 30, 39]. There is also growing interest in using temporal logic specifications for *controller synthesis* of systems with uncertainty. Under the assumption of a known stochastic dynamical model, there have been proposed approaches to find optimal control policies that maximize the probability of satisfying a given temporal logic specification by planning over stochastic abstractions [1, 19, 28, 29]. When the underlying model is unknown, reinforcement learning based methods have been used for specifications given as LTL [38] and STL [2] formulas.

Fig. 7. Controller Design using PSO to maximize the robustness of Equation (5.5) for different values of c_1, c_2 .

Our work builds on recent work to extend STL to systems with uncertainty, specifically work on Probabilistic STL (PrSTL) [40], Chance Constrained Temporal Logic (C2TL) [24], and Stochastic STL (StSTL) [35]. The atomic predicates in the logic C2TL are affine constraints on variables (or arbitrary Boolean combinations of such constraints), where the coefficients in the constraints are assumed to be random variables with a Gaussian distribution. The signal variables themselves are assumed to evolve deterministically. For this logic, the authors show how the synthesis problem with a C2TL objective can be conservatively encoded as a mixed integer linear program. In [40], the authors define PrSTL that uses probabilistic atomic predicates parameterized with a time-varying random variable drawn from a given distribution. The authors then synthesize a receding horizon controller that satisfies PrSTL specifications using a mixed integer semi-definite encoding of the synthesis problem. The work in [35] focuses on using StSTL specifications as contracts for verification using a similar mixed integer linear programming formulation. Each of these logics separately contains some of the elements that we desire for a Stochastic Temporal Logic; our logic fundamentally differs in spirit. While previous logics focus on specific uncertainties in an otherwise deterministic system, the target of our proposed logic is reasoning about arbitrary stochastic dynamical systems, where the source of uncertainty could be in the environment or the system itself.

Our work heavily draws on our previous work in [14], which introduces Stochastic Temporal Logic. While our prior work introduces Boolean semantics for StTL, in this paper, we propose a new quantitative semantics for StTL. While [14] also provides a parameter inference algorithm, we believe that our presentation in this paper formalizes many of the concepts. Furthermore, our paper also makes a connection between parameter inference and different notions of robust satisfaction for StTL. Finally, this paper suggests a practical use for leveraging robustness values as a heuristic for feedback control-based satisfaction of StTL specifications, which is not considered in [14].

Our work on robust satisfaction semantics for StTL draws on previous work on quantitative semantics for MTL [18] and STL [17]. The crucial difference is that our calculation of robustness is w.r.t. a stochastic process, and not a signal. This difference is crucial as it requires knowledge of the time-varying PDF (through a model based on first-principles or a data-driven model), and the robustness of atomic predicates is significantly different than that for STL or MTL.

Our work on parameter inference proposed in this paper was developed independently of the work that appears in [7]. However, our algorithm closely mimics the algorithm in [7]. The signal that we call the *least permissible valuation* signal is called the *parameter validity signal* in [7]. A key difference is the parameter inference for atomic predicates, which in our case are chance constraints, while that in [7] is for signal predicates. The algorithm in [7] is however more general than

the one presented here, as it reasons about propagation of Pareto fronts of satisfying parameter valuations, while the restrictions we impose on PStTL allows us to not have to consider this.

In contrast to extensions to the logical formalism adopted by the previous works, the approach in [8] uses a different interpretation of a given STL formula to define probabilistic quantitative semantics. The key idea is to extend the notion of robustness for STL formulas to stochastic systems, by considering distributions over robustness values induced by the stochasticity in the system. The authors then define quantitative semantics using notions of average and conditional average robustness.

Also of note are approaches that use Probabilistic Computation Tree Logic (PCTL), that was introduced to express properties over the realizations of Markov chains and Markov Decision Processes [21]. A significant amount of research has focused on using PCTL for verification and optimal control of Markov Decision Processes [6, 10, 32, 45]. We remark that these approaches rely on the system structure to perform model checking or to design control strategies w.r.t. a given PCTL formula.

Limitations and Future Work. Our approach for control design essentially considers only parameterized control laws where the controller parameters are unknown. Furthermore, the optimization procedure that we use is a best effort, black-box optimization procedure that does not provide any theoretical guarantees on convergence or error bounds. It would be interesting to consider approaches such as statistical model checking [13, 33] to prove the properties of the controllers synthesized using our procedure. It would be also interesting to consider extensions to machine learning based control architectures such as [2]. Our technique for mining specifications is also currently limited to template specifications that are specified by the user. In the future, we will consider combining structure learning techniques such as [27, 31] with the formula parameter inference procedure.

One direction for further investigation relates to the estimation of the PDFs, where we currently use the ensemble statistics method. The precision of estimating the PDFs is affected by the selection of the kernels. In order to bound the error on the robustness or determine the number of required simulation traces to attain a specific precision in estimating the PDFs and implicitly attain an error bound in the robustness computation, we will investigate advanced probability theory techniques such as the Hoeffding's inequality, the concentration inequality or the generalized central limit theorem arguments.

Conclusion. In this paper, we present Stochastic Temporal Logic (StTL) as an amalgam of previous efforts to extend Signal Temporal Logic to stochastic system. Our definition allows us to use StTL as a specification formalism for arbitrary controlled stochastic dynamical systems. After defining StTL, we introduce the quantitative schematics to reason about the degree of satisfaction of an StTL specification, we present an algorithm for their computation and we propose a method for designing optimal controllers by maximizing the robustness of the given specification using a Particle Swarm Optimizer. Additionally, we develop an efficient specification mining algorithm which is based on iterating over the time horizon of each sub-formula and finding the least permissive parameter valuation that yields a valid formula. Our algorithm correctly identifies the satisfaction boundary in the parameter space where the validity of the formula changes and outperforms the grid method. We demonstrate and validate our framework and algorithms in two case studies from the automotive domain.

APPENDIX

Timed traces are measurements that are obtained from the physical system under consideration, whereas a stochastic process is a mathematical model that summarizes the timed traces. Our

objective is to map the traces to the random process that describes the system. For this purpose, we introduce a process that describes the random state transitions of a system in an n -dimensional Hilbert space. We assume the system is at state $\mathbf{x}(t) \in \mathbb{R}^n$ at time t . Subsequently, it waits dt seconds⁶ and then makes a transition to state $\mathbf{x}(t) + d\mathbf{x}$. The *inter-event times* dt and *magnitude increment* $d\mathbf{x}$ are random variables drawn from the joint PDF $\lambda(\mathbf{x}, t)$. Using these definitions, we enumerate all possible transitions that could have led to the system being at state $\mathbf{x}$ at time t and using simple probabilistic arguments we obtain the following *generalized master equation* (GME):

$$p(\mathbf{x}, t) = \Psi(t)\delta(\mathbf{x}) + \int_0^t d\tau \int_{\mathcal{R}^n} p(\mathbf{y}, \tau)\lambda(\mathbf{x} - \mathbf{y}, t - \tau)d\mathbf{y} \quad (6.1)$$

where $p(\mathbf{x}, t)$ is the probability of finding the system at state $\mathbf{x}$ at time t , $\delta(\mathbf{x})$ is the multivariate Dirac delta function and $\Psi(t)$ is the probability of making no transitions until time t . The above equation provides the desired stochastic process that describes the system. In this paper, we construct a data-driven solution to the above GME using a black-box, simulation-based method, whereby we assume access to computational model of the system, which we can stimulate and measure its output. We implement a method which relies on the *ensemble statistics*. Using the computational model we are able to obtain M timed-traces $\{(t_0, \bar{\mathbf{x}}_0^i), \dots, (t_n, \bar{\mathbf{x}}_n^i)\}_{i=1}^M$ of the stochastic process described by Equation (6.1). Subsequently, an estimate of the time-dependant probability distribution $p(\mathbf{x}, t)$ is obtained by calculating the empirical PDF at each time instance; namely for each time t_k we estimate the empirical PDF of the samples $\{\mathbf{x}_k^i\}_{i=1}^M$. Naturally, as M increases, so does the accuracy of the estimation. Obviously, this approach makes no assumption about the system dynamics or the type of stochasticity; given enough realizations, any complicated distribution can be estimated.

ACKNOWLEDGMENT

We thank the anonymous reviewers who carefully read our paper and gave valuable feedback that has helped improve it. The authors gratefully acknowledge the support by the National Science Foundation Career award under Grant No. CPS/CNS-1453860, the NSF Grant CCF-1837131, the U.S. Army Research Office (ARO) under Grant No. W911NF-17-1-0076, the Okawa Foundation award, and the Defense Advanced Research Projects Agency (DARPA) Young Faculty Award under Grant No. N66001-17-1-4044 support. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied by the Defense Advanced Research Projects Agency, the Department of Defense or the National Science Foundation.

REFERENCES

- [1] A. Abate, A. D’Innocenzo, and M. D. Di Benedetto. 2011. Approximate abstractions of stochastic hybrid systems. *IEEE Trans. Automat. Control* 56, 11 (Nov 2011), 2688–2694.
- [2] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta. 2016. Q-learning for robust satisfaction of signal temporal logic specifications. In *IEEE 55th Conference on Decision and Control (CDC)*. 6565–6570.
- [3] Matthias Althoff, Olaf Stursberg, and Martin Buss. 2009. Model-based probabilistic collision detection in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems* 10 (2009), 299–310.
- [4] Eugene Asarin, Alexandre Donzé, Oded Maler, and Dejan Nickovic. 2011. Parametric identification of temporal properties. In *Proceedings of Runtime Verification*. 147–160.
- [5] Eugene Asarin and Oded Maler. 1998. Achilles and the tortoise climbing up the arithmetical hierarchy. *J. Comput. System Sci.* 57, 3 (1998), 389–398.
- [6] Christel Baier, Marcus Größer, Martin Leucker, Benedikt Bollig, and Frank Ciesinski. 2004. Controller synthesis for probabilistic systems. In *Exploring New Frontiers of Theoretical Informatics*. Springer, 493–506.

⁶The units are mentioned only for convenience.

- [7] Alexey Bakhirkin, Thomas Ferrère, and Oded Maler. 2018. Efficient parametric identification for STL. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control*. ACM, 177–186.
- [8] Ezio Bartocci, Luca Bortolussi, Laura Nenzi, and Guido Sanguinetti. 2013. On the robustness of temporal properties for stochastic models. In *Proceedings of the Second International Workshop on Hybrid Systems and Biology, HSB 2013, Taormina, Italy, 2nd September 2013*:3–19.
- [9] Stephen Boyd and Lieven Vandenbergh. 2004. *Convex Optimization*. Cambridge University Press, New York, NY, USA.
- [10] Tomáš Brázdil, Krishnendu Chatterjee, Martin Chmelík, Vojtěch Forejt, Jan Křetínský, Marta Kwiatkowska, David Parker, and Mateusz Ujma. 2014. Verification of Markov decision processes using learning algorithms. In *Automated Technology for Verification and Analysis*. 98–114.
- [11] Chris Chatfield. 2016. *The Analysis of Time Series: An Introduction*. CRC press.
- [12] Gang Chen, Zachary Sabato, and Zhaodan Kong. 2016. Active learning based requirement mining for cyber-physical systems. In *IEEE 55th Conference on Decision and Control (CDC'16)*. 4586–4593.
- [13] Edmund M Clarke and Paolo Zuliani. 2011. Statistical model checking for cyber-physical systems. In *International Symposium on Automated Technology for Verification and Analysis*. Springer, 1–12.
- [14] Jyotirmoy V. Deshmukh, Panagiotis Kyriakis, and Paul Bogdan. 2018. Stochastic temporal logic abstractions: Challenges and opportunities. In *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 3–16.
- [15] Jerry Ding, Maryam Kamgarpour, Sean Summers, Alessandro Abate, John Lygeros, and Claire Tomlin. 2013. A stochastic games framework for verification and control of discrete time stochastic hybrid systems. *Automatica* 49, 9 (2013), 2665–2674.
- [16] Alexandre Donzé, Thomas Ferrère, and Oded Maler. 2013. Efficient robust monitoring for STL. In *Proceedings of the 25th International Conference on Computer Aided Verification (Proceedings of Computer-Aided Verification'13)*. Springer-Verlag, Berlin, Heidelberg, 264–279.
- [17] Alexandre Donzé, Thomas Ferrère, and Oded Maler. 2013. Efficient robust monitoring for STL. In *Proceedings of Computer-Aided Verification*. 264–279.
- [18] 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.
- [19] Jie Fu and Ufuk Topcu. 2015. Computational methods for stochastic control with metric interval temporal logic specifications. In *Proceedings of the International Conference on Decision and Control (CDC)*. IEEE, 7440–7447.
- [20] Mahboobeh Ghorbani and Paul Bogdan. 2014. Reducing risk of closed loop control of blood glucose in artificial pancreas using fractional calculus. In *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*. IEEE, 4839–4842.
- [21] Hans Hansson and Bengt Jonsson. 1994. A logic for reasoning about time and reliability. *Formal Aspects of Computing* 6, 5 (1 Sep 1994), 512–535.
- [22] Bardh Hoxha, Houssam Abbas, and Georgios Fainekos. 2015. Benchmarks for temporal logic requirements for automotive systems. In *ARCH14-15. 1st and 2nd International Workshop on Applied Verification for Continuous and Hybrid Systems (EPiC Series in Computing)*, Goran Frehse and Matthias Althoff (Eds.), Vol. 34. EasyChair, 25–30.
- [23] Bardh Hoxha, Adel Dokhanchi, and Georgios Fainekos. 2018. Mining parametric temporal logic properties in model-based design for cyber-physical systems. *International Journal on Software Tools for Technology Transfer* 20, 1 (2018), 79–93.
- [24] Susmit Jha, Vasumathi Raman, Dorsa Sadigh, and Sanjit A. Seshia. 2018. Safe autonomy under perception uncertainty using chance-constrained temporal logic. *J. Autom. Reason.* 60, 1 (Jan. 2018), 43–62.
- [25] Xiaoping Jin, Jyotirmoy V. Deshmukh, James Kapinski, Koichi Ueda, and Ken Butts. 2014. Powertrain control verification benchmark. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*. ACM, 253–262.
- [26] Xiaoping Jin, Alexandre Donzé, Jyotirmoy V. Deshmukh, and Sanjit A. Seshia. 2015. Mining requirements from closed-loop control models. *IEEE Trans. Comp. Aided Design* 34, 11 (2015), 1704–1717.
- [27] Austin Jones, Zhaodan Kong, and Calin Belta. 2014. Anomaly detection in cyber-physical systems: A formal methods approach. In *53rd IEEE Conference on Decision and Control*. IEEE, 848–853.
- [28] A. A. Julius and G. J. Pappas. 2009. Approximations of stochastic hybrid systems. *IEEE Trans. Automat. Control* 54, 6 (June 2009), 1193–1203.
- [29] M. Kamgarpour, J. Ding, S. Summers, A. Abate, J. Lygeros, and C. Tomlin. 2011. Discrete time stochastic hybrid dynamical games: Verification & controller synthesis. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*. 6122–6127.
- [30] J. Kapinski, X. Jin, J. Deshmukh, A. Donzé, T. Yamaguchi, H. Ito, T. Kaga, S. Kobuna, and S. A. Seshia. 2016. ST-Lib: A library for specifying and classifying model behaviors. In *SAE Technical Paper*. SAE.

- [31] Zhaodan Kong, Austin Jones, Ana Medina Ayala, Ebru Aydin Gol, and Calin Belta. 2014. Temporal logic inference for classification and prediction from data. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*. ACM, 273–282.
- [32] M. Lahijanian, S. B. Andersson, and C. Belta. 2011. Control of Markov decision processes from PCTL specifications. In *Proceedings of the 2011 American Control Conference*. 311–316.
- [33] Axel Legay, Benoît Delahaye, and Saddek Bensalem. 2010. Statistical model checking: An overview. In *International Conference on Runtime Verification*. Springer, 122–135.
- [34] Julien Legriel, Colas Le Guernic, Scott Cotton, and Oded Maler. 2010. Approximating the pareto front of multi-criteria optimization problems. In *Proceedings of TACAS*. 69–83.
- [35] Jiwei Li, Pierluigi Nuzzo, Alberto Sangiovanni-Vincentelli, Yugeng Xi, and Dewei Li. 2017. Stochastic contracts for cyber-physical system design under probabilistic requirements. In *ACM/IEEE Int. Conf. on Formal Methods and Models for System Design*.
- [36] Oded Maler. 2017. Learning Monotone Partitions of Partially-Ordered Domains (Work in Progress). (July 2017). <https://hal.archives-ouvertes.fr/hal-01556243> working paper or preprint.
- [37] Oded Maler and Dejan Nickovic. 2004. Monitoring temporal properties of continuous signals. In *Proceedings of FORMATS/FTRTFT*. 152–166.
- [38] Aurélien Rizk, Grégory Batt, François Pages, and Sylvain Soliman. 2008. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In *Computational Methods in Systems Biology*, Monika Heiner and Adeline M. Uhrmacher (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 251–268.
- [39] Hendrik Roehm, Rainer Gmehlich, Thomas Heinz, Jens Oehlerking, and Matthias Woehrle. 2015. Industrial examples of formal specifications for test case generation. In *Workshop on Applied verification for Continuous and Hybrid Systems, ARCH@CPSWeek 2015*. 80–88.
- [40] Dorsa Sadigh and Ashish Kapoor. 2016. Safe control under uncertainty with probabilistic signal temporal logic. In *Robotics Science and Systems*.
- [41] Yuhui Shi and Russell C. Eberhart. 1998. Parameter selection in particle swarm optimization. In *Evolutionary Programming VII*, V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 591–600.
- [42] Bernard W. Silverman. 1986. *Density Estimation for Statistics and Data Analysis*. Vol. 26. CRC press.
- [43] R. Verma, D. D. Vecchio, and H. K. Fathy. 2008. Development of a scaled vehicle with longitudinal dynamics of an HMMWV for an ITS testbed. *IEEE/ASME Transactions on Mechatronics* 13, 1 (Feb 2008), 46–57. DOI: <https://doi.org/10.1109/TMECH.2008.915820>
- [44] Yuankun Xue and Paul Bogdan. 2017. Constructing compact causal mathematical models for complex dynamics. In *Proceedings of the 8th International Conference on Cyber-Physical Systems (ICCPs'17)*. 97–107.
- [45] Xiaobin Zhang, Bo Wu, and Hai Lin. 2015. Learning based supervisor synthesis of pomdp for petl specifications. In *IEEE 54th Annual Conference on Decision and Control (CDC'15)*. IEEE, 7470–7475.

Received April 2019; revised June 2019; accepted July 2019