Safe Control under Uncertainty with Probabilistic Signal Temporal Logic

Dorsa Sadigh*
University of California, Berkeley
dsadigh@eecs.berkeley.edu

Ashish Kapoor
Microsoft Research, Redmond
akapoor@microsoft.com

Abstract—**Safe control of dynamical systems that satisfy temporal invariants expressing various safety properties is a challenging problem that has drawn the attention of many researchers. However, making the assumption that such temporal properties are deterministic is far from the reality. For example, a robotic system might employ a camera sensor and a machine learned system to identify obstacles. Consequently, the safety properties the controller has to satisfy, will be a function of the sensor data and the associated classifier. We propose a framework for achieving safe control. At the heart of our approach is the new Probabilistic Signal Temporal Logic (PrSTL), an expressive language to define stochastic properties, and enforce probabilistic guarantees on them. We also present an efficient algorithm to reason about safe controllers given the constraints derived from the PrSTL specification. One of the key distinguishing features of PrSTL is that the encoded logic is adaptive and changes as the system encounters additional data and updates its beliefs about the latent random variables that define the safety properties. We demonstrate our approach by deriving safe control of quadrotors and autonomous vehicles in dynamic environments.**

I. Introduction

Achieving safe control for robotics and cyber-physical systems (CPS) is a challenging problem, due to various factors including uncertainty arising from the environment. For example, any safe control strategy for quadcopters need to incorporate predictive information about wind gusts and any associated uncertainty in such predictions. Similarly, in the case of autonomous driving, the controller needs a probabilistic predictive model about the other vehicles on the road to avoid collisions. Without a model of uncertainty that characterizes all possible outcomes, there is no hope in giving guarantees about the safety of the synthesized controller.

The field of Machine Learning (ML) has a rich set of tools that can characterize uncertainties. Specifically, Bayesian graphical models [19] have been popular in modeling uncertainties arising in scenarios common to robotics. For example, one of the common strategies is to build classifiers or predictors based on acquired sensor data. It is appealing to consider such predictors in achieving safe control of dynamical systems. However, it is almost impossible to guarantee a prediction system that works perfectly at all times. Consequently, we need to devise control methodologies aware of such limitations imposed by the ML systems. Specifically, we need to build a framework that is capable of achieving safe control by being aware of when the prediction system would work or fail.

* Work done while at Microsoft Research, Redmond

In this paper, we propose a framework for achieving safe control, when ML models are employed to make predictions based on sensed signals. The heart of our framework is the novel *Probabilistic Signal Temporal Logic* (PrSTL) that allows us to express safety constraints by considering the predictive models and their associated uncertainties. This logic allows specifications that embed Bayesian classification methods via *probabilistic predicates* that take random variables as parameters, thereby resulting in a powerful framework that can reason about safety under uncertainty. One of the main advantages of using Bayesian classification models is the fact that the predictions provided are full distributions associated with the quantity of interest as opposed to a point estimate. For example, a classical machine learning method might just provide a value for wind speed; however, under the Bayesian paradigm we would be recovering an entire probability distribution over all possible wind profiles. Finally, another distinguishing aspect of our framework is that these probabilistic predicates are adaptive: as the system sees more and more data, the inferred distribution over the latent variables of interest can change leading to change in the predicates themselves.

Previous efforts for achieving safe control either operate under deterministic environments or model uncertainty only as part of the dynamics of the system [16]. These approaches lack clear connections to various sources of uncertainty present in the environment. Specifically, there is no clear understanding of how uncertainty arising due to sensing and classification could be incorporated while reasoning about safe controllers.

In this paper we aim to alleviate these issues by defining a probabilistic logical specification framework that has the capacity to reason about safe control strategies by embedding various predictions and their associated uncertainty. Specifically, our contributions in this paper are:

- Framework for safe control under uncertainty.
- Formally define PrSTL, a logic for expressing probabilistic properties that can embed Bayesian graphical models.
- Solve a receding horizon control problem to satisfy PrSTL specifications using Mixed Integer SDPs.
- A toolbox implementing the framework and experiments in autonomous driving and control of quadrotors.

Next we discuss the related work and preliminaries, we then define the problem statement along with PrSTL, and show our solution with some experimental results, and then conclude.

II. Related Work

Over the years researchers have proposed different approaches for safe control of dynamical systems. Designing controllers under reachability analysis and safe learning are well-studied methods that allow specifying safety and reachability properties, while learning the optimal strategy online [32, 33, 15, 3, 1]. However, finding the reachable set is computationally expensive, which makes these approaches impractical for most interesting tasks. Controller synthesis under temporal specifications such as Linear Temporal Logic (LTL) allows expressing interesting properties of the system and environment, e. g., safety, liveness, response, stability, and has shown promising results [35, 25, 22, 49, 20, 36]. However, synthesis for LTL requires time and space discretization, which suffers from the curse of dimensionality. Also, while such approaches are effective at high level planning, they are unsuitable for lower level control of dynamical systems. Recently, synthesis for Signal Temporal Logic (STL), which allows real-valued, dense-time properties have been studied in receding horizon settings [39, 21, 14]. One downside of specifying properties in STL or LTL is that the properties of the system and environment have to be expressed deterministically. Full knowledge of the exact parameters and bounds of the specification is an unrealistic assumption for most robotics applications, where the system interacts with uncertain environments. Related to our work is the paradigms of Markov Logic Networks [40] that aim to induce probability distributions over possible worlds by considering weighted logical formulae. However, it is not clear how such networks can be used for controller synthesis. The proposed framework instead considers formulae parameterized by random variables, thereby inducing probability distribution over the set of possible formulae. Further, we also show how such formalism can be embedded in a receding horizon MPC for controller synthesis. In robust control, uncertainty is modeled as part of the dynamics, and the optimal strategy is found for the worst case disturbance, which can be a conservative assumption [24, 47]. More recently, the uncertainty is modeled in a chance constrained framework showing promising results for urban autonomous driving [28, 46, 6, 8]. Considering uncertainties while satisfying temporal logic requirements has recently been explored for controller synthesis and verification [42, 43, 12, 11, 37]. Leahy et al. maximize information gain in a distributed setting to reduce the uncertainty over belief of every state; however, the uncertainty is not considered as part of the specification [26]. Defining a formalism for modeling uncertainty as part of the requirement is also explored subsequent to our paper [41, 9]. To best of our knowledge, none of the previous studies consider scenarios, where the uncertainty and confidence in properties are originated from sensors, predictors and classifiers, and are formalized as part of the property. We propose a natural framework for safe control, by first defining a probabilistic temporal specification that allows us to express safety constraints over different sources of uncertainty, and then providing an efficient algorithm to solve for control inputs that would preserve the safety invariants.

III. Preliminaries

Hybrid Dynamical System: Lets consider a continuous time hybrid dynamical system:

$$\dot{x_t} = f(x_t, u_t), \quad y_t = g(x_t, u_t). \tag{1}$$

Here, $x_t \in \mathcal{X} \subseteq (\mathbb{R}^{n_c} \times \{0,1\}^{n_d})$ is a signal representing the continuous and discrete mode of the system at time t, $u_t \in U \subseteq (\mathbb{R}^{m_c} \times \{0,1\}^{m_d})$ is the control input and $y_t \in Y \subseteq (\mathbb{R}^{p_c} \times \{0,1\}^{p_d})$ is the output of the system at time t. This continuous system can be discretized using time intervals $dt > 0$, and every discrete time step is $k = \lfloor t/dt \rfloor$. The discrete time hybrid dynamical system is formalized as:

$$x_{k+1} = f_d(x_k, u_k), \quad y_k = g_d(x_k, u_k). \tag{2}$$

We let $x_0 \in \mathcal{X}$ denote the initial state, and express an infinite *run* of the system as: $\xi = (x_0, u_0), (x_1, u_1), \ldots$. Given x_0, and a finite length input sequence: $\mathbf{u}^H = u_0, u_1, \ldots, u_{H-1}$, the finite horizon *run* or *trajectory* of the system following the dynamics in equation (2) is $\xi^H(x_0, \mathbf{u}^H) = (x_0, u_0), (x_1, u_1), \ldots, (x_H, u_H)$. Furthermore, we let $\xi(t) = (x_t, u_t)$ be a *signal* consisting of the state and input of the system at time t; $\xi_x(t) = x_t$ is the state, and $\xi_u(t) = u_t$ is the input at time t. A cost function is defined for the finite horizon trajectory, denoted by $J(\xi^H)$, and maps $\xi^H \in \Xi$, the set of all trajectories to positive real valued costs in $\mathbb{R}^+$. The goal then is to determine the trajectory that has the lowest cost. However, not all the trajectories might be safe. The safety constraints on these trajectories are often described as logical specification.

Safe Control via Signal Temporal Logic: *Signal Temporal Logic (STL)* is an expressive framework for reasoning about real-valued dense-time functions, and has been used in defining robustness measures [30, 10].

Formally, $(\xi, t) \models \varphi$ denotes that a signal ξ satisfies the STL formula φ at time t. An atomic predicate of an STL formula is represented by inequalities of the form $\mu(\xi(t)) > 0$. The truth value of the predicate μ is equivalent to $\mu(\xi(t)) > 0$. Note that with slight abuse of notation, μ represents both the predicate and a function of the trajectory $\xi(t)$. Any STL formula consists of Boolean and temporal operations on these predicates and the syntax of STL formulae φ is defined recursively as follows:

$$\varphi ::= \mu \mid \neg\mu \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \mathbf{G}_{[a,b]}\psi \mid \varphi\, \mathbf{U}_{[a,b]}\psi \mid \mathbf{F}_{[a,b]}\psi, \tag{3}$$

where ψ and φ are STL formulae, $\mathbf{G}$ denotes the *globally* operator, $\mathbf{F}$ the *eventually* operator, and $\mathbf{U}$ is the *until* operator. For instance, $\xi \models \mathbf{G}_{[a,b]}\psi$ specifies that ψ must hold at all times in the given interval, $t \in [a, b]$ of signal ξ. Determining minimum cost trajectory that satisfy STL properties is a non-trivial task. Most promising approaches are based on Receding Horizon Control or Model Predictive Control (MPC) [34] that iteratively optimize the cost function $J(\xi^H)$. Starting with an initial state x_0, the MPC scheme aims to determine the optimal strategy $\mathbf{u}^H$ given the dynamics model of the system defined in equation (2), while satisfying the STL formula

φ. The constraints represented using STL allow expression of temporal specifications on the runs of the system and environment and limit the allowed behavior of the closed loop system [39, 38]. Prior work shows MPC optimization with STL constraints can be posed as a *Mixed Integer Linear Programs* (MILP) [38]. Note that the STL framework is only defined for deterministic signals. Furthermore, it is not clear how to incorporate ML components that use sensors to make predictions about the dynamic environment. We use Bayesian graphical models as a way to understand the environment and the associated uncertainty around the predictions.

Bayesian Classification to Model Uncertainty: Probability theory provides a natural way to represent uncertainty in the environment and recent advances in Machine Learning have relied on Bayesian methods to infer distributions over latent phenomenon of interest [13, 19]. We focus on Bayesian classifiers, which unlike other optimization based methods, provide entire distributions over the predictions. Such predictive distributions characterize the uncertainty present in the system and are crucial for achieving safe control. Formally, given a set of training points $\mathbf{X}_L = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, with observations $\mathbf{t}_L = \{t_1, \dots, t_n\}$, where $t_i \in \{+1, -1\}$, we are interested in finding a hyperplane $\mathbf{w}$ that separates the points belonging to the two classes according to $\text{sgn}(\mathbf{w}^T\mathbf{x})$. Under the Bayesian paradigm, we look for the distribution:

$$\begin{aligned} &p(\mathbf{w}|\mathbf{X}_L, \mathbf{t}_L) = p(\mathbf{w}) \cdot p(\mathbf{t}_L|\mathbf{X}_L, \mathbf{w}) = \\ &p(\mathbf{w}) \prod_i p(t_i|\mathbf{w}, \mathbf{x}_i) = p(\mathbf{w}) \prod_i \mathbb{I}\left[\text{sgn}(\mathbf{w}^T\mathbf{x}_i) = t_i\right]. \end{aligned} \tag{4}$$

The first line in the above equation stems from the Bayes rule, and the second line simply exploits the fact that given the classifier $\mathbf{w}$ the labels for each of the points in the dataset are independent. The expression $\mathbb{I}[\cdot]$ is an indicator function, which evaluates to 1, when the condition inside the brackets holds. Thus, equation (4) starts from a prior $p(\mathbf{w})$ over the classifiers and eventually by incorporating the training data points, infers a posterior distribution over the set of all the classifiers that respect the observed labels and the points. Given these statistical dependencies among the various variables, Bayesian inference techniques [31, 4, 2] aim to infer $p(\mathbf{w}|\mathbf{X}_L, \mathbf{t}_L)$ as a Gaussian distribution $\mathcal{N}(\mathbf{w}; \bar{\mathbf{w}}, \Sigma)$. Linear classification of a test data point $\mathbf{w}^T\mathbf{x}$ results in a Gaussian distribution of the prediction with the mean $\mathbf{w}^T\mathbf{x}$ and the variance $\mathbf{x}^T\Sigma\mathbf{x}$. Similarly, for the case of Bayesian linear regression the same procedure can be followed, albeit with continuous target variables $t \in \mathbb{R}$. These Bayesian linear classifiers and regressors are a fairly rich class of models, and have similar or better representation capabilities as kernel machines [48]. In this work, we specifically aim to incorporate such rich family of classification models for safe control.

IV. Problem Statement

We propose *Probabilistic Signal Temporal Logic* (PrSTL) that allows expression of uncertainty over the latent variables via probabilistic specifications. The key idea is to incorporate random variables in predicates, and then express temporal and Boolean operations on such predicates. The proposed logic provides an expressive framework for defining safety conditions under a wide variety of uncertainties, including ones that arise from application of Machine Learning classifiers.

The core ingredient in this work is the insight that when the uncertainty over the random variable is reasoned out in a Bayesian framework, we can use the inferred probability distributions to efficiently derive constraints from the PrSTL specifications. We provide a novel solution for synthesizing controllers for dynamical systems given different PrSTL properties. An interesting aspect of this framework is that the PrSTL formulae can evolve at every step. For example, a classifier associated with the dynamical system can continue to learn with time, thereby changing the inferred probability distributions on the latent random variables.

A. Probabilistic Signal Temporal Logic

PrSTL supports probabilistic temporal properties on real-valued, dense-time signals. Specifically, $(\xi, t) \models \varphi$ denotes the signal ξ satisfies the PrSTL formula φ at time t. We introduce the notion of a probabilistic atomic predicate $\lambda_{\alpha_t}^{\epsilon_t}$ of a PrSTL formula that is parameterized with a time-varying random variable α_t drawn from a distribution $p(\alpha_t)$:

$$(\xi, t) \models \lambda_{\alpha_t}^{\epsilon_t} \iff P(\lambda_{\alpha_t}(\xi(t)) < 0) > 1 - \epsilon_t. \tag{5}$$

Similar to STL, with a slight abuse of notation, we let $\lambda_{\alpha_t}^{\epsilon_t}$ denote the probabilistic atomic predicate, and λ_{α_t} be a function of the signal $\xi(t)$. Here, $P(\cdot)$ represents the probability of the event, and $1 - \epsilon_t$ defines the *tolerance* level in satisfaction of the probabilistic properties. The parameter $\epsilon_t \in [0, 0.5]$ is a small time-varying positive number and represents the threshold on satisfaction probability of $\lambda_{\alpha_t}(\xi(t)) < 0$. Small values of ϵ_t favors high tolerance satisfaction of formulas, which also facilitates solving the controller synthesis problem as discussed later in this section. A signal $\xi(t)$ satisfies the PrSTL predicate $\lambda_{\alpha_t}^{\epsilon_t}$ with confidence $1 - \epsilon_t$ if and only if:

$$\int_{\alpha_t} \mathbb{I}[\lambda_{\alpha_t}(\xi(t)) < 0]\, p(\alpha_t)\, d\alpha_t > 1 - \epsilon_t. \tag{6}$$

Here, $\mathbb{I}[\cdot]$ is an indicator function, and the equation marginalizes out the random variable α_t with the probability density $p(\alpha_t)$. The truth value of the PrSTL predicate $\lambda_{\alpha_t}^{\epsilon_t}$ is equivalent to satisfaction of the probabilistic constraint in equation (5). Computing such integrals as in equation (6) for general distributions is computationally difficult; however, there are many parameterized distributions (e.g., Gaussian and other members of the exponential family) for which there exists either a closed form solution or efficient numerical procedures.

Note that $\lambda_{\alpha_t}(\xi(t))$ is a stochastic function of the signal ξ at time t and expresses a model of the uncertainty in environment based on the observed signals. As the system evolves and observes more data about the environment, the distribution over the random variable α_t changes over time, thereby leading to an adaptive PrSTL predicate. The PrSTL formula consists of Boolean and temporal operations over their predicates. We recursively define the syntax of PrSTL as:

$$\varphi ::= \lambda_{\alpha_t}^{\epsilon_t} \mid \tilde{\neg}\lambda_{\alpha_t}^{\epsilon_t} \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \mathbf{G}_{[a,b]}\psi \mid \varphi\, \mathbf{U}_{[a,b]}\psi \mid \mathbf{F}_{[a,b]}\psi.$$

Here, φ is a PrSTL formula, which is built upon predicates $\lambda_{\alpha_t}^{\epsilon_t}$ defined in equation (5), *propositional formulae* φ composed of the predicates and Boolean operators such as $\wedge$ (and), $\tilde{\neg}$ (negation), and *temporal operators* on φ such as $\mathbf{G}$ (globally), $\mathbf{F}$ (eventually) and $\mathbf{U}$ (until). Note, that in these operations the PrSTL predicates can have different probabilistic parameters, i. e., α_t and ϵ_t. In addition, satisfaction of the PrSTL formulae for each of the Boolean and temporal operations based on the predicates is defined as:

$$\begin{array}{lcl}
(\xi,t) \models \lambda_{\alpha_t}^{\epsilon_t} & \Leftrightarrow & P(\lambda_{\alpha_t}(\xi(t)) < 0) > 1-\epsilon_t \\
(\xi,t) \models \tilde{\neg}\lambda_{\alpha_t}^{\epsilon_t} & \Leftrightarrow & P(-\lambda_{\alpha_t}(\xi(t)) < 0) > 1-\epsilon_t \\
(\xi,t) \models \varphi \wedge \psi & \Leftrightarrow & (\xi,t) \models \varphi \wedge (\xi,t) \models \psi \\
(\xi,t) \models \varphi \vee \psi & \Leftrightarrow & (\xi,t) \models \varphi \vee (\xi,t) \models \psi \\
(\xi,t) \models \mathbf{G}_{[a,b]}\varphi & \Leftrightarrow & \forall t' \in [t+a,t+b], (\xi,t') \models \varphi \\
(\xi,t) \models \mathbf{F}_{[a,b]}\varphi & \Leftrightarrow & \exists t' \in [t+a,t+b], (\xi,t') \models \varphi \\
(\xi,t) \models \varphi\; \mathbf{U}_{[a,b]}\; \psi & \Leftrightarrow & \exists t' \in [t+a,t+b] \text{ s.t. } (\xi,t') \models \psi \\
 & & \wedge \forall t'' \in [t,t'], (\xi,t'') \models \varphi.
\end{array}$$

Remark 1: PrSTL does not follow the *law of excluded middle*, while it follows the *law of noncontradiction*. This means it will never be the case for a formula and its negation $\tilde{\neg}$, as defined above, to both satisfy a specification at the same time. However, there exists situations, where they both can violate a specification.

Remark 2: The PrSTL framework reduces to STL, when the distribution $p(\alpha_t)$ is a Dirac distribution. A Dirac or a point distribution over α_t enforces $\lambda_{\alpha_t}(\xi(t)) < 0$ to be deterministic and equivalent to an STL predicate μ defined in Section III.

B. Controller Synthesis for Probabilistic STL

We now formally define the controller synthesis problem in the MPC framework with PrSTL specifications.

Problem 1: Given a hybrid dynamical system as in equation (2), initial state x_0, PrSTL formula φ, and cost function $J(\xi^H)$ defined over a finite horizon trajectory ξ^H, find:

$$\underset{\mathbf{u}^H}{\operatorname{argmin}}\, J(\xi^H(x_0,\mathbf{u}^H)) \quad \text{subject to} \quad \xi^H(x_0,\mathbf{u}^H) \models \varphi. \tag{7}$$

Problem (1) formulates a framework for finding a control strategy $\mathbf{u}^H$ that optimizes a given cost function, and satisfies a PrSTL formula. Finding the best strategy for this optimization given only deterministic PrSTL formulae, where α_t is drawn from a Dirac distribution is the same as solving a set of mixed integer linear constraints. We now show how the optimization is solved for the general case of PrSTL. Specifically, we provide full solution for Gaussian distributions, where the optimization reduces to mixed integer semi-definite programs.

1) Mixed Integer Constraints: We first discuss how every PrSTL formula generates a set of integer constraints. Given a PrSTL formula, we introduce two integer variables for every time step t, p_{t}^{φ} and $q_{\mathrm{t}}^{\varphi} \in \{0,1\}$, which correspond to the truth value of the PrSTL formula and its negation respectively. These variables enforce satisfaction of the formula φ as:

$$(p_{\mathrm{t}}^{\varphi} = 1) \rightarrow (\xi,t) \models \varphi \text{ and } (q_{\mathrm{t}}^{\varphi} = 1) \rightarrow (\xi,t) \models \tilde{\neg}\varphi \tag{8}$$

The formula φ holds true if $p_{\mathrm{t}}^{\varphi} = 1$, and its negation $\tilde{\neg}\varphi$ (defined in Section IV-A) holds true if $q_{\mathrm{t}}^{\varphi} = 1$. Due to the definition of negation, and Remark 1, these implications are only one-way, i. e., there always exist a satisfiable solution when p_{t}^{φ} and q_{t}^{φ} are zero. Using both integer variables, we define the constraints required for logical and temporal operations of PrSTL on p_{t}^{φ} and q_{t}^{φ}. It is important to note that p_{t}^{φ} and q_{t}^{φ} are not functions of the truth value of the formula φ, so their values are not meant to be uniquely determined. Instead, the integer variables enforce the truth value of the formula φ. We refer to them as *truth value enforcers*:

- **Negation** $(\varphi = \tilde{\neg}\psi)$: $p_{\mathrm{t}}^{\varphi} \le q_{\mathrm{t}}^{\psi}$ and $q_{\mathrm{t}}^{\varphi} \le p_{\mathrm{t}}^{\psi}$
- **Conjunction** $(\varphi = \wedge_{i=1}^{N}\psi_i)$: $p_{\mathrm{t}}^{\varphi} \le p_{\mathrm{t}}^{\psi_i}$ and $q_{\mathrm{t}}^{\varphi} \le \sum_{i=1}^{N} q_{\mathrm{t}}^{\psi_i}$
- **Disjunction** $(\varphi = \vee_{i=1}^{N}\psi_i)$: $\varphi = \tilde{\neg} \wedge_{i=1}^{N} \tilde{\neg}\psi_i$
- **Globally** $(\varphi = \mathbf{G}_{[a,b]}\psi)$:
$p_{\mathrm{t}}^{\varphi} \le p_{t'}^{\psi} \qquad \forall t' \in [t+a,\, \min(t+b, H-1)]$,
$q_{\mathrm{t}}^{\varphi} \le \sum_{t'=t+a}^{t+b} q_{t'}^{\psi} \qquad$ (Only for $t < H-b$).
- **Eventually** $(\varphi = \mathbf{F}_{[a,b]}\psi)$: $\varphi = \tilde{\neg}\, \mathbf{G}_{[a,b]}\tilde{\neg}\psi$.
- **Unbounded Until** $(\varphi = \psi_1\, \tilde{\mathbf{U}}_{[0,\infty)}\psi_2)$:
$\bigvee_{t=0}^{H-1} \big((\mathbf{G}_{[0,t]}\psi_1) \wedge (\mathbf{G}_{[t,t]}\psi_2) \big) \vee \mathbf{G}_{[0,H-1]}\psi_1$
- **Bounded Until** $(\varphi = \psi_1\, \mathbf{U}_{[a,b]}\psi_2)$:
$\varphi = \mathbf{G}_{[0,a]}\psi_1 \wedge \mathbf{F}_{[a,b]}\psi_2 \wedge \mathbf{G}_{[a,a]}(\psi_1 \tilde{\mathbf{U}}_{[0,\infty)}\psi_2)$

Here, we have shown how p_{t}^{φ} and q_{t}^{φ} are defined for every logical property such as *negation*, *conjunction*, and *disjunction*, and every temporal property such as *globally*, *eventually*, and *until*. We use $\tilde{\mathbf{U}}$ to refer to unbounded until with infinite time interval, and $\mathbf{U}$ for bounded until.

While synthesizing controllers for PrSTL formulae in an MPC scheme, we sometimes are required to evaluate satisfaction of the formula outside of the horizon range. For instance, a property $\mathbf{G}_{[a,b]}\varphi$ might need to be evaluated beyond H for some $t' \in [t+a, t+b]$. In such cases, our proposal is to act optimistically, i. e., we assume the formula holds true for the time steps outside the horizon of *globally* operator, and similarly assume the formula does not hold true for the negation of the *globally* operator. This optimism is evident in formulating the truth value enforcers of the *globally* operator, and based on that, it is specified for other temporal properties. With the recursive definition of PrSTL, and the above encoding, the truth value enforcers of every PrSTL formula is defined using a set of integer inequalities involving a composition of the truth value enforcers of the inner predicates.

2) Satisfaction of PrSTL predicates: We have defined the PrSTL predicate $\lambda_{\alpha_t}^{\epsilon_t}$ for a general function, $\lambda_{\alpha_t}(\xi(t))$ of the signal ξ at time t. This function allows a random variable $\alpha_t \sim p(\alpha_t)$ to be drawn from any distribution at every time step. The general problem of controller synthesis satisfying the PrSTL predicates is computationally difficult since the evaluation of the predicates boils down to computing the integration in equation (6). Consequently, to solve the problem in equation (7), we need to enforce a structure on the predicates of φ. In this section, we explore the linear-Gaussian structure of the predicates that appears in many real-world scenarios, and show how it translates to Mixed Integer SDPs. Formally, if $\varphi = \lambda^{\epsilon_t}_{\alpha_t}$ is only a single predicate, the optimization in equation (7) will reduce to:

$$\begin{aligned} \underset{\mathbf{u}^H}{\text{argmin}} \quad & J(\xi^H(x_0, \mathbf{u}^H)) \\ \text{subject to} \quad & (\xi, t) \models \lambda^{\epsilon_t}_{\alpha_t} \qquad \forall t \in \{0, \dots, H-1\}. \end{aligned} \tag{9}$$

This optimization translates to a chance constrained problem [5, 7, 45, 28, 6] at every time step of the horizon, based on the definition of PrSTL predicates in equation (5):

$$\begin{aligned} \underset{\mathbf{u}^H}{\text{argmin}} \quad & J(\xi^H(x_0, \mathbf{u}^H)) \\ \text{subject to} \quad & P(\lambda_{\alpha_t}(\xi(t)) < 0) > 1 - \epsilon_t \\ & \forall t \in \{0, \dots, H-1\}. \end{aligned} \tag{10}$$

An important challenge with such chance constrained optimization is there are no guarantees that equation (10) is convex. The convexity of the problem depends on the structure of the function λ_{α_t}, and the distribution $p(\alpha_t)$. It turns out that the problem takes a simple convex form when λ_{α_t} is linear-Gaussian, i. e., the random variable α_t comes from a Gaussian distribution and the function itself is linear in α_t:

$$\lambda_{\alpha_t}(\xi(t)) = {\alpha_t}^\top \xi_x(t) = {\alpha_t}^\top x_t, \quad \alpha_t \sim \mathcal{N}(\mu_t, \Sigma_t). \tag{11}$$

It is easy to show that for this structure of λ_{α_t}, i.e., a weighted sum of the states with Gaussian weights α_t, the chance constrained optimization in equation (10) is convex [44, 23]. Specifically, the optimization problem can be transformed to a second-order cone program (SOCP). First, consider a normally distributed random variable $\nu \sim \mathcal{N}(0, 1)$, and its cumulative distribution function (CDF) $\Phi = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} dt$. Then, the chance constrained optimization reduces to SOCP:

$$P(\lambda_{\alpha_t}(\xi(t)) < 0) > 1 - \epsilon_t \Leftrightarrow P(\alpha_t^\top x_t < 0) > 1 - \epsilon_t \Leftrightarrow$$

$$P(\nu < \frac{-\mu_t^\top x_t}{x_t^\top \Sigma_t x_t}) > 1 - \epsilon_t \Leftrightarrow \int_{-\infty}^{\frac{-\mu_t^\top x_t}{x_t^\top \Sigma_t x_t}} \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} dt > 1 - \epsilon_t$$

$$\Leftrightarrow \Phi(\frac{\mu_t^\top x_t}{x_t^\top \Sigma_t x_t}) < \epsilon_t \Leftrightarrow \mu_t^\top x_t - \Phi^{-1}(\epsilon_t) ||\Sigma_t^{1/2} x_t||_2 < 0.$$

In this formulation, $\mu_t^\top x_t$ is the linear term, where μ_t is the mean of the random variable α_t at every time step, and $||\Sigma_t^{1/2} x_t||_2$ is the l_2-norm representing a quadratic term, where Σ_t is the variance of α_t. This quadratic term is scaled by $\Phi^{-1}(\epsilon_t)$, the inverse of the Normal CDF function, which is negative for small values of $\epsilon_t \leq 0.5$. Thus, every chance constraint can be reformulated as a SOCP, and as a result with a convex cost function $J(\xi^H)$, we can efficiently solve the following convex optimization for every predicate of PrSTL:

$$\begin{aligned} \underset{\mathbf{u}^H}{\text{minimize}} \quad & J(\xi^H(x_0, \mathbf{u}^H)) \\ \text{subject to} \quad & \mu_t^\top x_t - \Phi^{-1}(\epsilon_t) ||\Sigma_t^{1/2} x_t||_2 < 0 \\ & \forall t \in \{0, \dots, H-1\}. \end{aligned} \tag{12}$$

Assuming a linear-Gaussian form of the function, we generate the SOCP above, and translate it to a semi-definite program (SDP) by introducing auxiliary variables [7]. We use this SDP that solves the problem in equation (9) with a single constraint $\varphi = \lambda^{\epsilon_t}_{\alpha_t}$ as a building block, and use it multiple times to handle complex PrSTL formulae. Specifically, any PrSTL formula can be decomposed to its predicates by recursively introducing integer variables that correspond to the truth value enforcers of the formula at every step as discussed in Section IV-B1.

We would like to point out that assuming linear-Gaussian form of λ_{α_t} is not too restrictive. The linear-Gaussian form subsumes the case of Bayesian linear classifiers, and consequently the framework can be applied to a wide variety of scenarios where a classification or regression function needs to estimate quantities of interest that are critical for safety. Furthermore, the framework is applicable to all random variables whose distributions exhibit unimodal behavior and aligned with the large law of numbers. For non-Gaussian random variables, there are many approximate inference procedures that effectively estimate the distributions as Gaussian.

3) Convex Subset of PrSTL: As discussed in the previous section IV-B2, at the predicate level of φ, we create a chance constrained problem for predicates $\lambda^{\epsilon_t}_{\alpha_t}$. These predicates of the PrSTL formulae can be reformulated as a semi-definite program, where the predicates are over intersections of cone of positive definite matrices with affine spaces. Semi-definite programs are special cases of convex optimization; consequently, solving Problem 1, only for PrSTL predicates is a convex optimization problem. Note that in Section IV-B1 we introduced integer variables for temporal and Boolean operators of the PrSTL formula. Construction of such integer variables increases the complexity of Problem 1, and results in a mixed integer semi-definite program (MISDP). However, we are not always required to create integer variables. Therefore, we define *Convex PrSTL* as a subset of PrSTL formulae that can be solved without constructing integer variables.

Definition 1: Convex PrSTL is a subset of PrSTL such that it is recursively defined over the predicates by applying Boolean conjunctions, and the globally temporal operator. Satisfaction of a convex PrSTL formulae is defined as:

$$\begin{aligned} (\xi, t) \models \lambda^{\epsilon_t}_{\alpha_t} \quad & \Leftrightarrow \quad P(\lambda_{\alpha_t}(\xi(t)) < 0) > 1 - \epsilon_t \\ (\xi, t) \models \varphi \wedge \psi \quad & \Leftrightarrow \quad (\xi, t) \models \varphi \wedge (\xi, t) \models \psi \\ (\xi, t) \models \mathbf{G}_{[a,b]}\varphi \quad & \Leftrightarrow \quad \forall t' \in [t + a, t + b], (\xi, t') \models \varphi \end{aligned}$$

Theorem 1: Given a convex PrSTL formula φ, a hybrid dynamical system as defined in equation (2), and an initial state x_0; the controller synthesis problem (Problem 1) is convex under a convex cost function J.

Proof: We have shown that the predicates of φ, i. e., $\lambda^{\epsilon_t}_{\alpha_t}$ create a set of convex constraints. The Boolean conjunction of convex programs are also convex; therefore, $\varphi \wedge \psi$ result in convex constraints. In addition, the *globally* operator is defined as a set of finite conjunctions over its time interval: $\mathbf{G}_{[a,b]}\varphi = \bigwedge_{t=a}^{b} \varphi_t$. Thus, the *globally* operator retains the convexity property of the constraints. Consequently, Problem 1, with a convex PrSTL constraint φ is a convex program. ■

Theorem 1 allows us to efficiently reduce the number of integer variables required for solving Problem 1. We only introduce integer variables when disjunctions, eventually, or until operators appear in the PrSTL constraints. Even when a formula is not completely part of the Convex PrSTL, integer variables are introduced only for the non-convex segments.

Algorithm 1 Controller Synthesis with PrSTL Formulae

```
1: procedure PROB. SYNTHESIS($(f, x_0, H, \tau, J, \varphi)$)
2: Let $\tau = [t_1, t_2]$ is the time interval of interest.
3: past $\leftarrow$ Initialize($t_1$)
4:     for $t = t_1 : dt : t_2$
5:         $f_{\text{lin}} = \text{linearize}(f, \xi(t))$
6:         $\alpha_t \leftarrow$ Update Distributions($\alpha_{t-dt}$, sense($\xi_x(t)$))
7:         $\varphi \leftarrow \varphi(\alpha_t, \epsilon_t)$
8:         $\mathtt{C}_{\mathtt{PrSTL}} = \text{MISDP}(\varphi)$
9:         $\mathtt{C} = \mathtt{C}_{\mathtt{PrSTL}} \wedge f_{\text{lin}}$
10:            $\wedge\, [\, \xi(\max(t_1, t-H) \cdots t-dt) = \mathtt{past}\, ]$
11:        $\mathbf{u}^H = \text{optimize}\big(J(\xi^H), \mathtt{C}\big)$
12:        $x_{t+1} = f(x_t, u_t)$
13:        past $\leftarrow$ [past $\xi(t)$]
14:        Drop the first element of past if len(past) $> \frac{H}{dt}$
15:    end for
16: end procedure
```

We show our complete method of controlling dynamical systems in uncertain environments in Algorithm 1. At the first time step t_1, we run an open-loop control algorithm to populate `past` in line 3. We then run the closed-loop algorithm, finding the optimal strategy at every time step of the time interval $\tau = [t_1, t_2]$. In the closed-loop algorithm, we linearize the dynamics at the current local state and time in line 5, and then update the distributions over the random variables in the PrSTL formula based on new sensor data in line 6. Then, we update the PrSTL formulae, based on the updated distributions. If there are any other dynamic parameters that change at every time step, they can also be updated in line 7. In line 8, we generate the mixed integer constraints in $\mathtt{C}_{\mathtt{PrSTL}}$, and populate $\mathtt{C}$ with all the constraints including the PrSTL constraints, linearized dynamics, and enforcing to keep the past horizon's trajectory. Note that we do not construct integer variables if the formula is Convex PrSTL. Then, we call the finite horizon optimization algorithm under the cost function $J(\xi^H)$, and the constraints $\mathtt{C}$ in line 11, which provides a length H strategy $\mathbf{u}^H$. We advance the state with the first element of $\mathbf{u}^H$, and update the previous horizon's history in `past`. The size of this problem does not grow, and keeping the past horizon history is crucial in satisfaction of fairness properties, e. g., enforcing a signal to oscillate. We continue running this loop and synthesizing controllers for all times in interval τ. Algorithm 1 solves MISDP problems, which is NP-hard. However, the actual runtime depends on the size of the MISDP, which is linear in the number of predicates and operators in the PrSTL specification. For convex PrSTL, the complexity is the same as an SDP, which is cubic in the number of constraints [27].

V. Experimental Results

We implemented our controller synthesis algorithm for PrSTL formulae as a Matlab toolbox available at: `https://github.com/dsadigh/CrSPrSTL`.

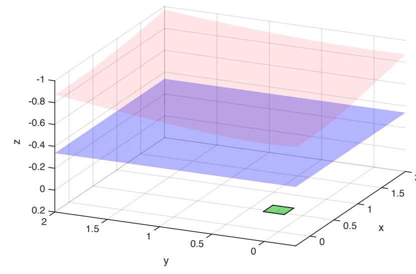


Fig. 1: A quadrotor (shown by green square) starting a trajectory from the origin. The purple surface represents the ceiling, and the orange surface is the quadrotors belief of the ceiling's location is based on the current sensor data.

We uses Yalmip [29] and Gurobi [17] as the optimization engines. In all of our examples discussed below, the optimization at every step is completed in less than 2 seconds on a 2.3 GHz Intel Core i7 processor with 16 GB RAM.

A. Quadrotor Control

Controlling quadrotors in dynamic uncertain environments is a challenging task due to different sources of uncertainty, e.g., the position of the obstacles estimated based on classification methods, distributions over wind or battery profiles. We show how to characterize different models of uncertainty over time, and then find an optimal strategy using the framework.

We follow the derivation of the dynamics model of a quadrotor in [18]. We consider a 12 dimensional system, where the state consists of the position and velocity of the quadrotor x, y, z and $\dot{x}, \dot{y}, \dot{z}$, as well as the Euler angles ϕ, θ, ψ, i. e., roll, pitch, yaw, and the angular velocities p, q, r. Let $\mathbf{x}$ be:

$$\mathbf{x} = [x \;\; y \;\; z \;\; \dot{x} \;\; \dot{y} \;\; \dot{z} \;\; \phi \;\; \theta \;\; \psi \;\; p \;\; q \;\; r]^\top. \tag{13}$$

The system has a 4 dimensional control input $\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^\top$, where u_1, u_2 and u_3 are the control inputs about each axis for roll, pitch and yaw respectively. u_4 represents the thrust input to the quadrotor in the vertical direction (z-axis). The nonlinear dynamics of the system is:

$$\begin{aligned}
&f_1(x,y,z) = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^\top \\
&f_2(\dot{x},\dot{y},\dot{z}) = \begin{bmatrix} 0 & 0 & g \end{bmatrix}^\top - R_1(\dot{x},\dot{y},\dot{z}) \begin{bmatrix} 0 & 0 & 0 & u_4 \end{bmatrix}^\top / m \\
&f_3(\phi,\theta,\psi) = R_2(\dot{x},\dot{y},\dot{z}) \begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^\top \\
&f_4(p,q,r) = I^{-1} \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}^\top - R_3(p,q,r) I \begin{bmatrix} p & q & r \end{bmatrix}^\top,
\end{aligned}$$

where R_1 and R_2 are rotation matrices, R_3 is a skew-symmetric matrix, and I is the inertial matrix of the rigid body. Here, g and m denote gravity and mass of the quadrotor, and for all our studies the mass and inertia matrix used are based on small sized quadrotors. Thus, the dynamics equation is $f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 \end{bmatrix}^\top$.

1) Control in an Uncertain Environments: We first demonstrate obstacle avoidance for a quadrotor. Figure 1, shows the quadrotor as a green square at its initial position $(0, 0, 0)$, and its objective is to reach the coordinates $(1, 1, 0)$ smoothly. If we let $z = 0$ represent the ground level ($z < 0$ is above the ground level), the objective of the quadrotor is to take off and travel the distance, and then land on the ground at the destination coordinate through the following:

$$J(\xi^H) = \sum_{t=0}^{H-1} ||(x_t, y_t, z_t) - (1,1,0)||_2^2 + c||(\phi_t, \theta_t, \psi_t)||_2^2.$$

Here, we penalize the l_2-norm of the Euler angles, which enforces a resulting smooth trajectory. Besides initializing the state and control input at zero, we bound the control inputs via the following deterministic PrSTL formulae (here only shown for roll, similar form follows for pitch and thrust):

$$\varphi_{\text{roll}} = \mathbf{G}_{[0,\infty)}(||u_1|| \leq 0.3) \text{ Bound on Roll Input}$$

In Figure 1, the purple surface is a ceiling that the quadrotor should not collide with as it is taking off and landing at the final position. However, the quadrotor does not have a full knowledge of where the ceiling is exactly located. We define a sensing mechanism for the quadrotor, which consists of a meshgrid of points around the body of the quadrotor. As the system moves in the space, a Bayesian binary classifier is updated by providing a single label -1 (no obstacles present) or 1 (obstacle present) for each of the sensed points.

The Bayesian classifier is the same as the Gaussian Process based method described in Section III and has the linear-Gaussian form. Applying this classifier results in a Gaussian distribution for every point in the 3D-space. We define our classifier with confidence $1 - \epsilon_t = 0.95$, as the stochastic function $\lambda_{\alpha_t}^{0.05}(\xi(t)) = \alpha_t^\top [x_t \;\; y_t \;\; z_t]$. Here, x_t, y_t, z_t are the coordinates of the sensing points, and $\alpha_t \sim \mathcal{N}(\mu_t, \Sigma_t)$ is the Gaussian weight inferred over time using the sensed data. We define a time-varying probabilistic constraint that needs to be held at every step as its value changes over time. Our constraint specifies that given a classifier based on the sensing points parameterized by α_t, we would enforce the quadrotor to stay within a safe region (defined by the classifier) with probability $1 - \epsilon_t$ at all times. Thus the probabilistic formula is:

$$\varphi_{\text{classifier}} = \quad \mathbf{G}_{[0.1,\infty)}\big(P(\alpha_t^\top [x_t \;\; y_t \;\; z_t] < 0) > 0.95\big).$$

We enforce this probabilistic predicate at $t \in [0.1, \infty)$, which verifies the property starting from a small time after the initial state, so the quadrotor has gathered some sensor data. In Figure 1, the orange surface represents the second order cone created based on $\varphi_{\text{classifier}}$, at every step characterized by:

$$\mu_t^\top \begin{bmatrix} x_t & y_t & z_t \end{bmatrix} - \Phi^{-1}(0.05)||\Sigma_t^{1/2} \begin{bmatrix} x_t & y_t & z_t \end{bmatrix}||_2 < 0.$$

Note that the surface shown in Figure 1, at the initial time step is not an accurate estimate of where the ceiling is, and it is based on a distribution learned from the initial values of the sensors. Thus, if the quadrotor was supposed to follow this estimate without updating, it would have collided with the ceiling. In Figure 2, the blue path represents the trajectory the quadrotor has already taken, and the dotted green line is the future planned trajectory based on the current state of the classifier. The dotted green trajectory at the initial state goes through the ceiling since the belief of the location of the ceiling is incorrect; however, the trajectory is modified at every step as the Bayesian inference updates the distribution over the

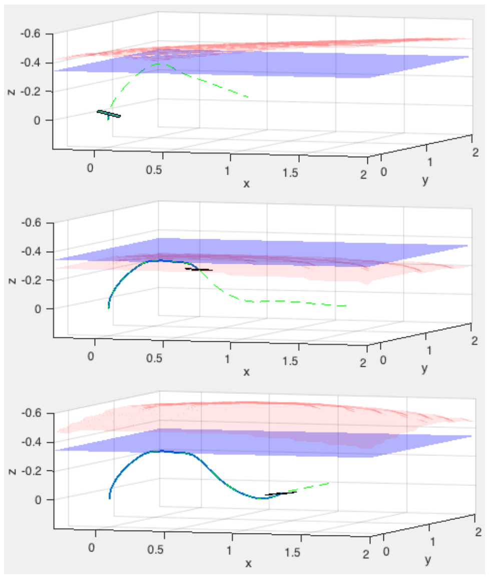
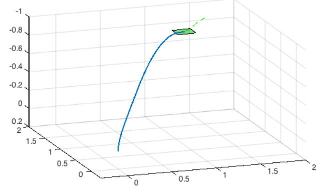


Fig. 2: The quadrotor in Fig. 1 taking the optimal trajectory to reach the goal, while avoiding collisions with the ceiling. The figures from the top correspond to $t = 0.18$ s $t = 0.63$ s, and $t = 1.02$ s.

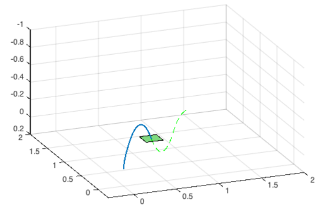
classifier. As shown in Figure 2, the orange surface changes at every time step, and the learned parameters μ_t, and Σ_t are updated, so the quadrotor safely reaches the final position. We solve the optimization using our toolbox, with $dt = 0.03$, and horizon of $H = 20$. These choices describe a common setting for solving MPC problems. We emphasize that some of the constraints are time-varying, and we need to update them at every step of the optimization. We similarly update the dynamics at every step, since we locally linearize the dynamics around the current position of the quadrotor.

2) Control under Battery Constraints: Next we demonstrate a scenario that addresses safety, when there is uncertainty around the battery level. The battery level is a stochastic variable due to uncertain environment and usage factors, such as radio communications, etc. Our goal is to derive a safe strategy that considers such stochasticity. We start by augmenting the logarithm of battery level (b_t) to the state space of the system discussed above. Thus, the quadrotor is a 13 dimensional system, where the first 12 states follow the same order and system dynamics as before and $\mathbf{x}_t(13) = \log(b_t)$. In our model, the state of $\log(b_t)$ evolves with the negative thrust value: $\frac{d\log(b_t)}{dt} = -|u_4|$. We enforce the same constraints to bound the control as earlier, and the objective of the quadrotor is to start from the origin and reach the top diagonal corner of the space with coordinates $(1, 1, -0.9)$ smoothly.

We then consider adding a safety invariant that would limit high altitude flights if the battery level is low. The following formulae describe these constraints: (i) $\phi := \mathbf{F}_{[0,0.3]}(z_t \leq -0.1)$ encodes that eventually in the next 0.3 s, the quadrotor will fly above a threshold level of -0.1 m. (ii) $\psi := P(\log(b_t) + \mathcal{N}(0, t\sigma^2) \geq b_{\text{min}}) \geq 1 - \epsilon_t$ represents the constraint that the quadrotor has to be confident that the logarithm of its battery state perturbed by a time-varying variance is above b_{min}. Now, we can combine the two formulae to specify the condition that the quadrotor needs to fly low if



(a) Deterministic state: $\sigma = 0$.



(b) Probabilistic state: $\sigma = 10$.

Fig. 3: Quadrotor flying to reach a goal while being confident in its battery level. On the right, the quadrotor has low confidence in its battery state, so it avoids flying higher than $z = -0.1$ m.

the battery state is low: $\varphi_{\text{battery}} := \mathbf{G}_{[0,\infty)}\big(\phi \to \psi)\big)$. The $\to$ in the formula φ_{battery} is the implication relation, and intuitively states that anytime the quadrotor flies above the threshold then it means that there is sufficient battery reserve.

We synthesize a controller for the specifications with $\epsilon_t = 0.2$. The trajectory of the quadrotor is shown in Figure 3. Figure 3a corresponds to $\sigma = 0$, i. e., the battery state changes deterministically, and Figure 3b, corresponds to $\sigma = 10$, when the quadrotor is more cautious about the state of the battery. Note the safe trajectory does not pass the -0.1 m height level whenever the confidence in the battery level is low.

B. Autonomous Driving

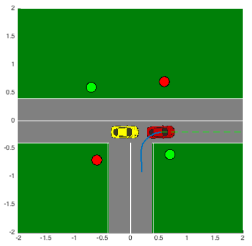
We now consider an autonomous driving scenario. We use a simple point-mass model to define the dynamics of the vehicles. Let the state of the system be $\mathbf{x} = [x \;\; y \;\; \theta \;\; v]^\top$, where x, y denote the coordinates of the vehicle, θ is the heading, and v is the speed. If $\mathbf{u} = [u_1 \;\; u_2]^\top$ are the steering and acceleration control inputs respectively, then the dynamics of the vehicle are: $[\dot{x} \;\; \dot{y} \;\; \dot{\theta} \;\; \dot{v}] = [v \cdot \cos(\theta) \quad v \cdot \sin(\theta) \quad \frac{v}{m} u_1 \quad u_2]$.

Figure 4, shows a scenario for an autonomous vehicle making a right turn at a signalized intersection. Here the red car is the *ego* vehicle that is autonomous, and the yellow car as the *environment* vehicle. Our goal is to find a strategy for the ego vehicle (red), so it makes a safe right turn when the traffic light is red, while yielding to the oncoming traffic (yellow).

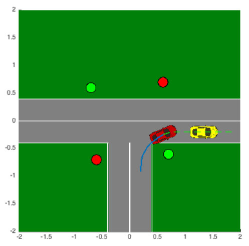
Note the ego vehicle only has a probabilistic model of the velocity of the environment car. We refer to the states of the environment vehicle as: $[x^{\text{env}} \;\; y^{\text{env}} \;\; \theta^{\text{env}} \;\; v^{\text{env}}]^\top$. To synthesize a safe strategy, we need to ensure collision avoidance despite the uncertainty in the estimates of the states. We define collision avoidance as the following PrSTL property:

$$\begin{aligned}
\varphi_{\text{crash}} = \mathbf{G}_{[t_0,\infty)}\Big(\; & P\big(x_t - (x^{\text{env}}_{t_0} + s_{t_0,x}(t-t_0)) \geq \delta\big) \;\; \geq 1-\epsilon_t \\
\vee \;\; & P\big(x_t - (x^{\text{env}}_{t_0} + s_{t_0,x}(t-t_0)) \leq -\delta\big) \;\; \geq 1-\epsilon_t \\
\vee \;\; & P\big(y_t - (y^{\text{env}}_{t_0} + s_{t_0,y}(t-t_0)) \geq \delta\big) \;\; \geq 1-\epsilon_t \\
\vee \;\; & P\big(y_t - (y^{\text{env}}_{t_0} + s_{t_0,y}(t-t_0)) \leq -\delta\big) \;\; \geq 1-\epsilon_t \;\Big)
\end{aligned}$$

Here, φ_{crash} consists of a global operator at all times over the disjunction of four PrSTL predicates. The four probabilistic predicates encode all possible crash scenarios between the two vehicles. The minimum permissible distance between the vehicles is represented as δ that generates the four disjunctions on the predicates. The estimate of the distance between x and y coordinates of the two vehicles is encoded in each predicate as the difference between the coordinates of the ego vehicle,



(a) A deterministic model of the yellow car's velocity with $\sigma = 0$.



(b) A probabilistic model of the yellow car's velocity with $\sigma = 0.4$.

Fig. 4: Red car making a right turn at a signalized intersection. On the right, the strategy computed performs a safer trajectory with a probabilistic model of the environment, where it first waits for the yellow car to pass. The blue lines show the trajectory followed by the ego vehicle, and the dotted green line is the future planned trajectory.

and the propagated coordinates of the environment vehicle based on the value of its velocity. We assume the magnitude of the velocity is distributed as a Gaussian distribution $v^{\text{env}}_t \sim \mathcal{N}(\bar{v_t}^{\text{env}}, \sigma^2)$. Then the vector of Gaussian random variables $[s_{t_0,x}, s_{t_0,y}]^T = v^{\text{env}}_{t_0}[\cos(\theta^{\text{env}}_{t_0}), \sin(\theta^{\text{env}}_{t_0}]^T$ consists of projections of the velocity v^{env}_t on the x and y directions based on the current heading $\theta^{\text{env}}_{t_0}$. The predicates in φ_{crash} take the familiar linear Gaussian form representing the coordinates of the ego vehicle, and are parameterized by the random variable characterizing the velocity of the environment vehicle.

We use $dt = 0.1$ s as sampling time, $H = 20$ as the horizon, $\sigma = 0.4$, $\delta = 0.4$ and $\epsilon_t = 0.2$. We synthesize a strategy for the autonomous vehicle by solving Problem 1, and following the steps in Algorithm 1. The trajectory generated is shown by the solid blue line in Figure 4. The dotted green line is the future trajectory computed by the MPC scheme. In Figure 4a, the ego vehicle has a deterministic model of the environment vehicle as $\sigma = 0$; therefore, it turns right before letting the environment vehicle pass. However, as shown in Figure 4b, for $\sigma = 0.4$, and $\epsilon_t = 0.2$, the ego vehicle is not confident enough in avoiding collisions, so it acts in a conservative manner and waits for the environment car to pass first before turning right.

VI. Conclusion

We presented a framework to achieve safe control under uncertainty. The key contributions include defining PrSTL, a logic for expressing probabilistic properties that can embed Bayesian graphical models. We also show how to synthesize receding horizon controllers under PrSTL specifications that express Bayesian linear classifiers. Further, the resulting logic adapts as more data is observed with the evolution of the system. We demonstrate the approach by synthesizing safe strategies for a quadrotor and an autonomous vehicle traveling in uncertain environments.

The approach extends easily to distributions other than Gaussians via Bayesian approximate inference techniques [31, 4] that can project distributions to the Gaussian densities. Future work includes extending controller synthesis for arbitrary distributions via sampling based approaches; we are also exploring using the proposed framework for complex robotic tasks that need to invoke higher level planning algorithms.

References

[1] Anayo K Akametalu, Shahab Kaynama, Jaime F Fisac, Melanie N Zeilinger, Jeremy H Gillula, and Claire J Tomlin. Reachability-based safe learning with gaussian processes. In *2014 IEEE 53rd Annual Conference on Decision and Control*.

[2] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*.

[3] Anil Aswani, Humberto Gonzalez, S Shankar Sastry, and Claire Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*.

[4] Matthew James Beal. *Variational algorithms for approximate Bayesian inference*. University of London, 2003.

[5] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Princeton University Press, 2009.

[6] Lars Blackmore, Masahiro Ono, and Brian C Williams. Chance-constrained optimal path planning with obstacles. *2011 IEEE Transactions on Robotics*.

[7] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[8] Ashwin Carvalho, Yiqi Gao, Stéphanie Lefevre, and Francesco Borrelli. Stochastic predictive control of autonomous vehicles in uncertain environments. In *12th International Symposium on Advanced Vehicle Control*, 2014.

[9] Yoo Chanyeol and Calin Belta. Control with probabilistic signal temporal logic. http://arxiv.org/pdf/1510.08474, October 2015.

[10] Alexandre Donzé, Thomas Ferrere, and Oded Maler. Efficient robust monitoring for STL. In *Computer Aided Verification*, 2013.

[11] Jie Fu and Ufuk Topcu. Integrating active sensing into reactive synthesis with temporal logic constraints under partial observations. *arXiv:1410.0083*, 2014.

[12] Jie Fu and Ufuk Topcu. Computational methods for stochastic control with metric interval temporal logic specifications. *arXiv:1503.07193*, 2015.

[13] Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*. 2014.

[14] Shromona Ghosh, Dorsa Sadigh, Pierluigi Nuzzo, Vasumathi Raman, Alexandre Donzé, Alberto Sangiovanni-Vincentelli, S. Shankar Sastry, and Sanjit A. Seshia. Diagnosis and repair for synthesis from signal temporal logic specifications. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*. ACM, 2016.

[15] Jeremy H Gillula and Claire J Tomlin. Guaranteed safe online learning via reachability: tracking a ground target using a quadrotor. In *2012 IEEE International Conference on Robotics and Automation*.

[16] Michael Green and David JN Limebeer. *Linear robust control*. Courier Corporation, 2012.

[17] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2015. URL http://www.gurobi.com.

[18] Haomiao Huang, Gabriel M Hoffmann, Steven L Waslander, and Claire J Tomlin. Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. In *2009 IEEE International Conference on Robotics and Automation*.

[19] Michael Jordan. Learning in graphical models (adaptive computation and machine learning). 1998.

[20] Sertac Karaman and Emilio Frazzoli. Sampling-based motion planning with deterministic μ-calculus specifications. In *Proceedings of the 48th IEEE Conference on Decision and Control, 2009.*

[21] Sertac Karaman and Emilio Frazzoli. Vehicle routing problem with metric temporal logic specifications. In *Proceedings of the 47th IEEE Conference on Decision and Control.*, 2008.

[22] Sertac Karaman and Emilio Frazzoli. Linear temporal logic vehicle routing with applications to multi-uav mission planning. *International Journal of Robust and Nonlinear Control*, 2011.

[23] Shinji Kataoka. A stochastic programming model. *Econometrica: Journal of the Econometric Society*, 1963.

[24] Mayuresh V Kothare, Venkataramanan Balakrishnan, and Manfred Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 1996.

[25] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. Temporal-logic-based reactive mission and motion planning. *2009 IEEE Transactions on Robotics*.

[26] Kevin Leahy, Austin Jones, Mac Schwager, and Calin Belta. Distributed information gathering policies under temporal logic constraints. In *2015 IEEE 54th Annual Conference on Decision and Control (CDC)*.

[27] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. *arXiv preprint arXiv:1508.04874*, 2015.

[28] David Lenz, Tobias Kessler, and Alois Knoll. Stochastic model predictive controller with chance constraints for comfortable and safe driving behavior of autonomous vehicles. In *2015 IEEE Intelligent Vehicles Symposium*.

[29] Johan Löfberg. Yalmip: A toolbox for modeling and optimization in matlab. In *2004 IEEE International Symposium on Computer Aided Control Systems Design*.

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

[31] Thomas P Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

[32] Ian Mitchell and Claire J Tomlin. Level set methods for computation in hybrid systems. In *Hybrid Systems: Computation and Control*. 2000.

[33] Ian Mitchell, Alexandre Bayen, and Claire J Tomlin. A time-dependent hamilton-jacobi formulation of reachable

sets for continuous dynamic games. *2005 IEEE Transactions on Automatic Control*.

[34] Manfred Morari, CE Garcia, JH Lee, and DM Prett. *Model predictive control*. 1993.

[35] Nir Piterman, Amir Pnueli, and Yaniv Sa′ar. Synthesis of reactive (1) designs. In *Verification, Model Checking, and Abstract Interpretation, 2006*.

[36] Erion Plaku and Sertac Karaman. Motion planning with temporal-logic specifications: Progress and challenges. *AI Communications*.

[37] Alberto Puggelli, Wenchao Li, Alberto L Sangiovanni-Vincentelli, and Sanjit A Seshia. Polynomial-time verification of pctl properties of mdps with convex uncertainties. In *Computer Aided Verification*, 2013.

[38] Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M Murray, Alberto Sangiovanni-Vincentelli, Sanjit Seshia, et al. Model predictive control with signal temporal logic specifications. In *2014 IEEE 53rd Annual Conference on Decision and Control*.

[39] Vasumathi Raman, Alexandre Donzé, Dorsa Sadigh, Richard M. Murray, and Sanjit A. Seshia. Reactive synthesis from signal temporal logic specifications. In *18th International Conference on Hybrid Systems: Computation and Control*, 2015.

[40] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 2006.

[41] Dorsa Sadigh and Ashish Kapoor. Safe control under uncertainty. http://arxiv.org/pdf/1510.07313v1, October 2015.

[42] Dorsa Sadigh, Eric S Kim, Samuel Coogan, S Shankar Sastry, and Sanjit Seshia. A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications. In *2014 IEEE 53rd Annual Conference on Decision and Control*.

[43] Mária Svoreňová, Jan Křetínskỳ, Martin Chmelík, Krishnendu Chatterjee, Ivana Černá, and Calin Belta. Temporal logic control for stochastic linear systems using abstraction refinement of probabilistic games. In *18th International Conference on Hybrid Systems: Computation and Control*, 2015.

[44] Cornelis Van de Panne and W Popp. Minimum-cost cattle feed under probabilistic protein constraints. *Management Science*, 9(3):405–430, 1963.

[45] Michael P Vitus and Claire J Tomlin. A probabilistic approach to planning and control in autonomous urban driving. In *2013 IEEE 52nd Annual Conference on Decision and Control*.

[46] Michael Peter Vitus. *Stochastic Control Via Chance Constrained Optimization and its Application to Unmanned Aerial Vehicles*. PhD thesis, Stanford University, 2012.

[47] Youyi Wang, Lihua Xie, and Carlos E de Souza. Robust control of a class of uncertain nonlinear systems. *Systems & Control Letters*, 1992.

[48] Christopher KI Williams and Carl Edward Rasmussen. Gaussian processes for machine learning. *the MIT Press*, 2006.

[49] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M Murray. Receding horizon control for temporal logic specifications. In *13th ACM international conference on Hybrid systems: computation and control*, 2010.