Chapter 1

Specification-Guided Safe Learning for Robotic Systems

CONTENTS

Specification-Guided Safe Learning for Robotic Systems

This chapter considers the problems of verification and synthesis for robotic systems with respect to complex tasks. In particular, a class of problems will be considered in which uncertainties in both system dynamics as well as environmental perturbations result in in high risk of failure. Abstraction-based methods are introduced which allow for computationally tractable high-level task planning with formal guarantees on the probability of task satisfaction. Then, Gaussian process learning techniques are incorporated into the abstraction model to enable learning of the system and environmental uncertainties. Finally, control policy synthesis algorithms are introduced which allow the robot to safely traverse its environment, learning the uncertainties online until the task can be satisfied with sufficient guarantees.

1.1 Introduction

Abstraction-based approaches for verification and synthesis of dynamical systems offer computationally tractable methods for planning with respect to complex tasks [1, 2, 3, 4]. These approaches model continuous and hybrid systems using discretized state space models, which allow for the use of graph-[5] and optimization-based [6] planning algorithms on the simplified state space with formal guarantees on the behavior of the original dynamics. In particular, Interval Markov Decision Processes (IMDP) [5, 7, 8], which allow for an interval of transition probabilities (and are also called Bounded Markov Decision Processes or Uncertain Markov Chains depending on the context [9, 10]), provide a rich abstraction model for stochastic systems. As compared to stochastic control [11], abstraction-based methods allow for more complex specifications to be considered and have been widely used for hybrid stochastic systems [12].

The transition probability intervals in IMDP abstractions have typically modeled the uncertainty which arises from abstracting the dynamics of continuous states in discrete regions [13]. Thus, best- and worst-case bounds on the behavior of the original system can be obtained through resolution of the uncertainty in the transition probability intervals [14]. However, partially-known stochastic systems, which show promise for modeling a wide range of real-world systems, add unknown dynamics which contribute further uncertainty. Previous works model this uncertainty by assuming that some prior data on the dynamics are available [15]. However, this precludes the possibility of tightening bounds on the behavior of the system as data is gathered online.

The work [16] is the first to address the problem of modeling unknown dynamics in stochastic hybrid systems via the use of IMDP abstraction in combination with *Gaussian process* (GP) regression [17]. GP regression can approximate unknown functions with arbitrary accuracy given sufficient data samples [18], and crucially, it also provides high-confidence bounds on the approximation uncertainty [19, 20, 21].

This chapter develops a method for sampling the unknown dynamics of a robotic system online in order to reduce abstraction error and increase the probability of satisfying a desired task. In particular, the class of tasks which can be represented using syntactically co-safe linear temporal logic (scLTL) specifications [5] will first be explored to illustrate the methodology. Then, the class of feasible task specifications will be generalized to the entire expressivity of Linear Temporal Logic (LTL) [22].

The goal is to find a control policy which guarantees the satisfaction of a scLTL or LTL specification with sufficient probability. However, there exists a stochastic noise in the dynamics which creates unavoidable perturbations. The system also has unknown dynamics which are estimated with Gaussian processes. This in turn creates an estimation error which increases uncertainty in state transitions. Then, the robot seeks to reduce this uncertainty by sampling the unknown dynamics online. However, executing a control policy to enable the robot to perform online sampling requires formal guarantees that the robot will not violate the specification of interest. Thus, this chapter focuses on the problem of safe learning to allow online exploration rather than a static planning problem using previously collected data samples as in [16].

The approach presented in this chapter is as follows. First, the unknown dynamics of the system are estimated using Gaussian processes and a high-confidence IMDP abstraction is developed. Next, an algorithm is detailed for finding *nonviolating cycles* in a product IMDP of the system abstraction combined with an automaton of the scLTL/LTL specification which allow the dynamics of the system to be sampled without violating the specification. A heuristic for evaluating candidate cycles is developed in order to maximize the uncertainty reduction gained from sampling. Finally, an iterative method is proposed to sample the state space, thereby decreasing the uncertainty of a GP estimation of the unknown dynamics until a satisfying control policy for the system can be synthesized or a terminating condition such as a maximum number of iterations has been reached. Sparse GPs [23] are introduced into the methodology in order to address computa-

tional complexity considerations. The methods are demonstrated on case studies of robotic motion planning.

1.2 Problem Setup

Consider a discrete-time, partially-known system

$$x[k+1] = f(x[k]) + u[k] + g(x[k]) + \nu[k] \tag{1.1}$$

where $x \in X \subseteq \mathbb{R}^n$ is the system state, $u \in \mathbb{R}^n$ is the control action, $f(x)$ is the known dynamics, $g(x)$ is the unknown dynamics to be learned via GP regression, ν is stochastic noise, and time is indexed with brackets. This system has applications in, *e.g.*, biology [24], communications [25], and robotics [26].

Assumption 1.1

1) Each dimension $\nu_i[k], i = 1, \ldots, n$ of ν, is an independent, zero mean random variable with stationary, symmetric, and unimodal distribution ρ_{ν_i} and is σ_{ν_i}-sub-Gaussian, i.e., the distribution tail decays at least as fast as a Gaussian random variable with variance $\sigma^2_{\nu_i}$.

2) Given a data set $D = \{(z^j, y^j)\}_{j=1}^m$ where y^j is an observation of $g(z^j)$ perturbed by σ_{ν_i}-sub-Gaussian noise, along each coordinate $i = 1, \ldots, n$, it is possible to construct an estimate $\hat{g}^D(x)$ of g and bound the estimation error between $g(x)$ and $\hat{g}^D(x)$ by

$$|g_i(x) - \hat{g}_i^D(x)| \leq \gamma_i^D(x) \quad \forall x \in X,\, i \in [1, \cdots, n] \tag{1.2}$$

for some high-confidence bound $\gamma^D(x)$. Thus,

$$g_-^D(x) = \hat{g}^D(x) - \gamma^D(x), \quad g_+^D(x) = \hat{g}^D(x) + \gamma^D(x) \tag{1.3}$$

are high-confidence bounds on g, i.e., $g_-^D(x) \leq g(x) \leq g_+^D(x)$ with high confidence. For simplicity, the superscript D is dropped when the dataset is clear.

Assumption 1.1 implies that the uncertainty from the noise ν and the unknown dynamics g are treated differently: the noise ν is drawn from a stationary distribution that is coordinate-wise independent and is stochastic and never learnable; on the other hand, $g(x)$ is unknown but learnable.

Assumption 1.2

The state space X is bounded and is partitioned into hyper-rectangular regions $\{X_q\}_{q \in Q}$ defined as

$$X_q = \{x \mid \ a_q \leq x \leq b_q\} \subset X, \tag{1.4}$$

where the inequality is taken elementwise for lower and upper bounds $a_q, b_q \in \mathbb{R}^n$ and Q is a finite index set of the regions. Each region has a center point defined as $c_q = (a_q + b_q)/2$. Additionally, the system possesses a labeling function L which maps hyper-rectangular regions to observations O. Additionally, it must also hold that the partition of the state space is complete, i.e., $\cup_{q \in Q} X_q = X$, and regions only intersect along their boundaries, i.e., $int(X_q \cap X_{q'}) = \emptyset$ for all $q \neq q'$, where $int(\cdot)$ denotes interior.

Further, there exists a set of observations (i.e., atomic propositions) O and a labeling function $L : Q \to O$ so that a sequence of states $\{x[0], x[1], x[2], \ldots\}$ induces a sequence of observations $\{L(q[0]), L(q[1]), L(q[2]), \ldots\}$ where $q[k]$ is the unique region index such that $x[k] \in X_{q[k]}$.

Given a state x, knowledge of the known dynamics f, and an estimate of the unknown dynamics $\hat{g}$, define feedback controllers $K_q(\cdot\,; \hat{g}) : X \to X$ as

$$K_q(x; \hat{g}) = c_q - f(x) - \hat{g}(x). \tag{1.5}$$

Assumption 1.3
For each region X_q, there exists a fixed subset $\mathcal{V}(q) \subset Q$ of available *controllers for which, for each $q' \in \mathcal{V}(q)$, the controller $K_{q'}(x; \hat{g})$ is available for any $x \in X_q$.*

The choice $u[k] = K_{q'}(x[k]; \hat{g})$ thus produces a control action which compensates for the known and estimated dynamics to reach the center of region $X_{q'}$, although the actual state update will be perturbed as shown in Figure 1.1 and may not reach partition $X_{q'}$. In practice, applying the feedback control $K_{q'}$ is only feasible from a subset of regions, in particular, those regions X_q for which $q' \in \mathcal{V}(q)$. An illustration of the feedback controllers under consideration is in Figure 1.1.

Remark 1.1 The problem setting above can be generalized in several ways while retaining the applicability of the results developed below at the expense of increased notation. In particular, (1.1) could allow for known dynamics $f(x, u)$ rather than the form $f(x) + u$ under an appropriate condition on the range of $f(x, \cdot)$ that ensures there exists feedback controllers $K_q(x; \hat{g})$ allowing the system to reach the center of, e.g., any adjacent region. Further, the unknown dynamics g and noise ν could depend on the applied feedback controller K_q. Lastly, the above formalism could be generalized to permit broader classes of state space partitions, e.g. convex polytopes [27]; however, hyper-rectangular partitions are especially suitable for accommodating unknown dynamics modeled as Gaussian processes. ■

The ultimate goal is to apply a sequence of feedback controllers so that the resulting sequence of observations satisfies a desired control objective. First, the set of objectives will be considered which can be represented as a syntactically co-safe LTL (scLTL) formula over the set of observations O.

Definition 1.1 Syntactically co-safe LTL [5, Def. 2.3] A *syntactically co-safe linear temporal logic (scLTL)* formula ϕ over a set of observations O is recursively defined as

$$\phi = \top \mid o \mid \neg o \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \bigcirc \phi \mid \phi_1 \mathcal{U} \phi_2 \mid \Diamond \phi$$

where $o \in O$ is an observation and ϕ, ϕ_1, and ϕ_2 are scLTL formulas. The *next* operator $\bigcirc$ is defined as meaning that ϕ will be satisfied in the next state transition, the *until* operator $\mathcal{U}$ as meaning that the system satisfies ϕ_1 until it satisfies ϕ_2, and the *eventually* operator $\Diamond$ as $\top \mathcal{U} \phi$.

ScLTL formulas are characterized by the property that, for any infinite sequence (i.e., word) of observations satisfying the formula, there exists a finite prefix of the word such that all infinite extensions of the prefix satisfy the formula, i.e., scLTL formulas are satisfied in finite time. It is well-known that scLTL satisfaction can be checked using a finite state automaton:

Definition 1.2 Finite State Automaton [5, Def. 2.4] A *finite state automaton (FSA)* is a tuple $\mathcal{A} = (S, s_0, O, \delta, F)$, where

- S is a finite set of states,
- $s_0 \in S$ is the initial state,
- O is the input alphabet, which corresponds to observations from the scLTL specification,
- $\delta : S \times O \to S$ is a transition function, and
- $F \subseteq S$ is the set of accepting (final) states.

An input word $\{o[0], o[1], ...\}$ with each $o[k] \in O$ generates a sequence of states (i.e., a run) $\{s[0], s[1], \ldots\}$ of $\mathcal{A}$ via the relationship $s[0] = s_0$, $s[k+1] = \delta(s[k], o[k])$. A word is *accepted* by the FSA if the induced run satisfies $s[k] \in F$ for some k. A scLTL formula can always be translated into a FSA that accepts all and only those words satisfying the formula. The class of scLTL specifications is particularly

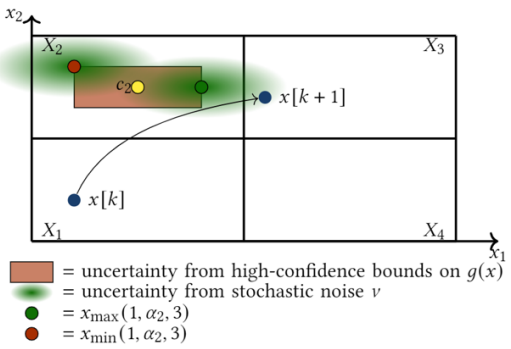

Figure 1.1: Feedback controller and calculation of transition probabilities. The controller targets the center of state X_2. The uncertainty in $\hat{g}(x)$ creates a nondeterministic region of transition (brown rectangle). The maximum probability of transitioning to state X_3 is found by centering stochastic noise at the point $x_{\max}$ closest to state X_3 (green point) and calculating the probability that the noise reaches state X_3. The minimum probability of transitioning to state X_3 under this controller is given likewise by centering stochastic noise at the point $x_{\min}$ furthest from X_3 (red point).

well-suited to robotic motion planning tasks which are satisfied in finite time. In Section 1.5, an extension to the general class of Linear Temporal Logic (LTL) specifications will be considered.

To satisfy the ultimate control objective, an abstraction-based approach will be used that takes high-confidence bounds on the unknown function g, where the bounds are improved via online data collection, to iteratively construct an Interval Markov Decision Process (IMDP) that appropriately abstracts the dynamics.

Definition 1.3 Interval Markov Decision Process An *Interval Markov Decision Process (IMDP)* is a tuple $\mathcal{I} = (Q, A, \check{T}, \hat{T}, Q_0, O, L)$ where:

- Q is a finite set of states,
- A is a finite set of actions,
- $\check{T}, \hat{T} : Q \times A \times Q' \to [0, 1]$ are lower and upper bounds, respectively, on the transition probability from state $q \in Q$ to state $q' \in Q$ under action $\alpha \in A$,
- $Q_0 \subseteq Q$ is a set of initial states,
- O is a finite set of atomic propositions or observations,

- $L : Q \to O$ is a labeling function.

The set of actions A corresponds to the set of all valid feedback controllers for the system. For some $q \in Q$ and $\alpha \in A$, if $\check{T}(q, \alpha, q') = \hat{T}(q, \alpha, q') = 0$ for all $q' \in Q$, α is said to be *disabled* at q, and is otherwise *enabled*. $A(q)$ denotes the set of actions at q. Moreover, for all $q \in Q$ and all $\alpha \in A(q)$, $\check{T}$ and $\hat{T}$ satisfy

$$\check{T}(q, \alpha, q') \leq \hat{T}(q, \alpha, q'), \sum_{q' \in Q} \check{T}(q, \alpha, q') \leq 1 \leq \sum_{q' \in Q} \hat{T}(q, \alpha, q').$$

Definition 1.4 High-Confidence IMDP Abstraction Consider stochastic system (1.1), partitions (1.4), and the family of feedback controllers (1.5) where $\hat{g}(x)$ is an estimate of $g(x)$. Further, suppose that $g_-(x)$ and $g_+(x)$ are high-confidence bounds on $\hat{g}(x)$ which satisfy (1.3). Then, an IMDP $\mathcal{I} = (Q, A, \check{T}, \hat{T}, Q_0, O, L)$ is a *high-confidence IMDP abstraction of* (1.1) given estimate $\hat{g}(x)$ and high-confidence bounds $g_-(x)$ and $g_+(x)$, or simply a *high-confidence abstraction of* (1.1), if:

- The set of states Q for the abstraction is the index set of partitions, *i.e.* partition X_q is abstracted as state q, and the set of observations O and labeling function L for the abstraction are the same as for the system,
- For all $q \in Q$, the set of actions $A(q)$ is the set of one-step reachable regions at q under its feedback controllers,
- For all $q \in Q$ and all $\alpha_{q^*} \in A(q)$:

$$\check{T}(q, \alpha_{q^*}, q') \leq \tag{1.6}$$
$$\min_{x \in X_q} \min_{g_-(x) \leq w \leq g_+(x)} \mathbb{P}_\nu(f(x) + w + K_{q^*}(x; \hat{g}) + \nu \in X_{q'}),$$
$$\hat{T}(q, \alpha_{q^*}, q') \geq \tag{1.7}$$
$$\max_{x \in X_q} \max_{g_-(x) \leq w \leq g_+(x)} \mathbb{P}_\nu(f(x) + w + K_{q^*}(x; \hat{g}) + \nu \in X_{q'})$$

where $\mathbb{P}_\nu$ denotes probability with respect to ν.

The bounds (1.6)–(1.7) ensure that $\check{T}(q, \alpha_{q^*}, q')$ and $\hat{T}(q, \alpha_{q^*}, q')$ lowerbound and upperbound the probability of transition from some state $x \in q$ to the partition q' under an action $\alpha_{q^*} \in A(q) \subseteq Q$, which prescribes a particular feedback controller K_{q^*}, given that the high-confidence bound $g_-(x) \leq g(x) \leq g_+(x)$ holds. In Section 1.3 below, an approach to constructing abstractions is presented such that (1.6) and (1.7) hold with equality, ensuring a tight abstraction.

Verification and synthesis problems for IMDP systems evaluated against scLTL specifications are often solved using graph theoretic methods on a product IMDP:

Definition 1.5 PIMDP Let $\mathcal{I} = (Q, A, \check{T}, \hat{T}, Q_0, O, L)$ be an IMDP and $\mathcal{A} = (S, s_0, O, \delta, F)$ be an FSA. The product IMDP (PIMDP) is defined as a tuple $\mathcal{P} = \mathcal{I} \otimes \mathcal{A} =$
$(Q \times S, A, \check{T}', \hat{T}', Q \times s_0, F')$, where

- $\check{T}' : (q, s) \times A \times (q', s') := \check{T}(q, \alpha, q')$ if $s' \in \delta(s, L(q))$ and 0 otherwise
- $\hat{T}' : (q, s) \times A \times (q', s') := \hat{T}(q, \alpha, q')$ if $s' \in \delta(s, L(q))$ and 0 otherwise
- $(q_0, \delta(s_0, L(q_0))) \in (Q \times S)$ is a set of initial states of $\mathcal{I} \otimes \mathcal{A}$, and
- $F' = Q \times F$ is the set of accepting (final) states.

A PIMDP admits a set of adversaries and control policies, defined as follows:

Definition 1.6 PIMDP Adversary An adversary of a PIMDP realizes transition probabilities that respect the PIMDP transition bounds, i.e., given a PIMDP state (q, s) and action α, an adversary ξ is an assignment of transition probabilities T'_ξ to all states (q', s') such that

$$\check{T}'((q, s), \alpha, (q', s')) \leq T'_\xi((q, s), \alpha, (q', s')) \leq \hat{T}'((q, s), \alpha, (q', s')).$$

Definition 1.7 Finite Memory Control Policy A finite memory control policy $\pi \in \Pi$ of a PIMDP is a mapping $(Q \times S)^+ \to A$, where $(Q \times S)^+$ is the set of finite sequences of states of the PIMDP.

With these preliminaries, the problems which will be addressed in the remainder of this chapter can be elucidated:

Problem 1.1
Given a system of the form (1.1) *with initial assumptions on* $g(x)$ *and some current dataset of samples of the system dynamics, construct a high-confidence IMDP abstraction of the system.*

Once a high-confidence IMDP abstraction has been constructed, a control policy is synthesized which allows the state space to be traversed indefinitely while data samples are collected without violating the scLTL specification of interest.

Problem 1.2
Given a system of the form (1.1) *with a corresponding high-confidence IMDP abstraction $\mathcal{I}$ and scLTL specification ϕ, synthesize a control policy which allows the state space to be sampled indefinitely without violating ϕ.*

The approach to this problem is divided into several phases. First, an algorithm is developed to find nonviolating regions in the product IMDP of the system along with corresponding control policies which do not violate ϕ.

Subproblem 1.1
Given a high confidence IMDP abstraction $\mathcal{I}$ and scLTL specification ϕ, construct a product IMDP $\mathcal{P}$ and identify regions in $\mathcal{P}$ along with corresponding control policies which do not violate ϕ.

Next, a heuristic is developed to identify an infinite cycle in the nonviolating graph constructed in Subproblem 1.1 which can be sampled to reduce the uncertainty of the GP regression:

Subproblem 1.2
Given a nonviolating sub-graph as in Subproblem 1.1, develop a heuristic to select an infinite cycle to sample to reduce uncertainty in the Gaussian process when estimating the unknown dynamics $g(x)$.

Finally, a proof is developed to show that these algorithms guarantee that the scLTL specification ϕ is not violated:

Subproblem 1.3
Prove that the algorithms developed in Subproblems 1.1 and 1.2 do not violate the specification ϕ of interest.

Additionally, an iterative algorithm is developed which utilizes the methods developed in Problem 1.1 and Problem 1.2 to synthesize a satisfying control policy with respect to a scLTL specification ϕ:

Problem 1.3
Design an iterative algorithm using the methods in Problems 1.1 and 1.2 to sample the unknown dynamics of system (1.1) *without violating the scLTL specification ϕ, construct a GP approximation, and synthesize a control policy which satisfies ϕ with some desired threshold probability or prove that no such control policy exists.*

1.3 Abstraction of System as IMDP

In this section, an approach is detailed for abstracting a system of the form (1.1) into a high-confidence IMDP. First, a method is described for calculating the transition probability bounds at each state. Then, these bounds are used to construct a high-confidence IMDP abstraction of the system.

In order to calculate lower and upper transition probability bounds for each state in the IMDP, an approximation of $g(x)$ (the unknown dynamics) needs to be calculated. At each time step of system (1.1), the quantities $x[k+1]$, $f(x[k])$, and $u[k]$ are known. Define the value $y[k]$ as

$$y[k] = x[k+1] - f(x[k]) - u[k] = g(x[k]) + \nu[k].$$

Then, a Gaussian process estimation $\hat{g}(x)$ for $g(x)$ can be constructed using a dataset of samples $(x[k], y[k])$.

Definition 1.8 Gaussian Process Regression Gaussian Process (GP) regression models a function $g_i : \mathbb{R}^n \to \mathbb{R}$ as a distribution with covariance $\kappa : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{>0}$. Assume a dataset of m samples $D = \{(z^j, y_i^j)\}_{j\in\{1,\dots,m\}}$, where $z^j \in \mathbb{R}^n$ is the input and y_i^j is an observation of $g_i(z^j)$ under Gaussian noise with variance $\sigma_{\nu_i}^2$. Let $K \in \mathbb{R}^{m\times m}$ be a matrix defined elementwise by $K_{j\ell} = \kappa(z^j, z^\ell)$ and for $z \in \mathbb{R}^n$, let $k(z) = [\kappa(z, z^1)\ \kappa(z, z^2) \dots \kappa(z, z^m)]^T \in \mathbb{R}^m$. Then, the predictive distribution of g_i at a test point z is the conditional distribution of g_i given D, which is Gaussian with mean $\mu_{g_i,D}$ and variance $\sigma_{g_i,D}^2$ given by

$$\mu_{g_i,D}(z) = k(z)^T (K + \sigma_{\nu_i}^2 I_m)^{-1} Y \tag{1.8}$$

$$\sigma_{g_i,D}^2(z) = \kappa(z,z) - k(z)^T (K + \sigma_{\nu_i}^2 I_m)^{-1} k(z), \tag{1.9}$$

where I_m is the identity and $Y = \begin{bmatrix} y_i^1 & y_i^2 & \dots & y_i^m \end{bmatrix}^T$.

In practice, GP regression has a complexity of $O(m^3)$. To mitigate this, sparse Gaussian process regression [23] can be used:

Definition 1.9 Sparse Gaussian Process Regression A sparse Gaussian process uses a set $D_\eta = \{(z^j, y_i^j)\}_{j\in\{1,\dots,\eta\}}$ to approximate a GP of a larger dataset D. Given *inducing points* $\{z_j\}_{j\in\{1,\dots,\eta\}}$ with $Y_\eta = \begin{bmatrix} y_i^1 & y_i^2 & \dots & y_i^\eta \end{bmatrix}^T$ and covariance matrix A_η, the predictive distribution of

the unknown function g_i has mean μ_{g_i,D_η} and variance $\sigma^2_{g_i,D_\eta}$

$$\mu_{g_i,D_\eta}(z) = k_\eta(z)^T(K_\eta + \sigma^2_{\nu_i} I_\eta)^{-1} Y_\eta$$
$$\sigma^2_{g_i,D_\eta}(z) = \kappa(z,z) - k_\eta(z)^T K_\eta^{-1}(K_\eta - A_\eta)K_\eta^{-1} k_\eta(z)$$

where $K_\eta \in \mathbb{R}^{\eta\times\eta}$ is a matrix defined elementwise by $K_{\eta,j\ell} = \kappa(z^j, z^\ell)$ for all $z \in D_\eta$. For $z \in \mathbb{R}^n$, let $k_\eta(z) = [\kappa(z,z^1)\ \kappa(z,z^2)\dots\ \kappa(z,z^\eta)]^T \in \mathbb{R}^\eta$. The parameters $\{z^j\}_{j\in\{1,\dots,\eta\}}$, $\{y_i^j\}_{j\in\{1,\dots,\eta\}}$, and A_η are optimized to minimize the Kullback-Leibler divergence (evaluated at the inducing points) between $\mathcal{N}(\mu_{g_i,D_\eta}, \sigma^2_{g_i,D_\eta})$, the posterior of g_i under the sparse GP; and $p(g_i|Y)$, the posterior of g_i under a GP with the full dataset D. The reader is referred to [23] for a detailed treatment of sparse Gaussian process theory. The computational complexity of sparse GP regression is $O(m\eta^2)$, so by fixing η the algorithm is linear in m. Note that sparse GP regression introduces error into the estimation; however, in practice this error does not affect the validity of the methods detailed in this chapter.

Given some dataset D, an estimation of the unknown dynamics is constructed independently in each coordinate and high-confidence bounds on the estimation error are determined using the equations

$$\hat{g}_i^D(x) := \mu_{g_i,D}(x),$$
$$\gamma_i(x) := \beta\sigma_{g_i,D}(x) \geq |g_i(x) - \hat{g}_i^D(x)|$$

for each $i = 1,\dots,n$. Additionally, high-confidence lower and upper bounds on $g(x)$ are given by the equations

$$g_-(x) = \hat{g}^D(x) - \beta\sigma_{g,D}(x), \quad g_+(x) = \hat{g}^D(x) + \beta\sigma_{g,D}(x)$$

The parameter β is calculated as

$$\beta = \left(\frac{\sigma_\nu}{\sqrt{1+(2/m)}}(B_i + \sigma_\nu\sqrt{2(\gamma_k^m + 1 + \log\frac{1}{\delta})})\right) \tag{1.10}$$

for noise σ_ν-sub-Gaussian, m the number of GP samples, high-confidence parameter δ, information gain constant γ_k^m, and RKHS constant B_i as detailed in Lemma 1, [16]. Note that the same parameter $\beta\sigma_{g,D}$ is used to determine high-confidence bounds on both the estimation error and $g(x)$ itself.
For each region q in the state space, a high-confidence error bound for the unknown dynamics is given by

$$\gamma_i(q) = \max_{x\in X_q} \gamma_i(x)$$

In practice, this bound is computed by sampling $\gamma_i(x)$ throughout the

state space, introducing a trade-off between approximation error and computation complexity. Next, transition probability intervals are constructed assuming that the high-confidence bounds on the unknown dynamics hold:

Theorem 1.1 Construction of Transition Probabilities

Consider $q, q' \in Q$ and action $\alpha_{q^*} \in A(q)$. Lower bound $\check{T}$ and upper bound $\hat{T}$ transition probabilities from q to q' under α_{q^*} are given by

$$\check{T}(q, \alpha_{q^*}, q') = \prod_{i=1}^{n} \int_{a'_i}^{b'_i} \rho_{\nu_i}(z - x_{\min,i}(q, \alpha_{q^*}, q'))dz, \tag{1.11}$$

$$\hat{T}(q, \alpha_{q^*}, q') = \prod_{i=1}^{n} \int_{a'_i}^{b'_i} \rho_{\nu_i}(z - x_{\max,i}(q, \alpha_{q^*}, q'))dz, \tag{1.12}$$

where $x_{\min,i}$ is the i-th coordinate of $x_{\min}$ and similarly for $x_{\max,i}$, ρ_{ν_i} is the probability density function of the stochastic noise ν_i, and a' and b' are the lower and upper boundary points for region q'. The points $x_{\min}$ and $x_{\max}$ are defined with the equations

$$x_{\min}(q, \alpha_{q^*}, q') = \underset{x \in X}{\operatorname{argmax}} \; ||x - c_{q'}||_1 \tag{1.13}$$
$$s.t.\ c_{q^*} - \gamma(q) \leq x \leq c_{q^*} + \gamma(q),$$

$$x_{\max}(q, \alpha_{q^*}, q') = \underset{x \in X}{\operatorname{argmin}} \; ||x - c_{q'}||_1 \tag{1.14}$$
$$s.t.\ c_{q^*} - \gamma(q) \leq x \leq c_{q^*} + \gamma(q),$$

where $||\cdot||_1$ is the 1-norm and $\gamma(q)$ is a high-confidence error bound on the unknown dynamics satisfying Assumption 1.1.
Then, the transition probability bounds (1.11)–(1.12) *satisfy the constraints for high-confidence IMDP abstractions in* (1.6)–(1.7).

Proof 1.1 *The righthand side of the bound in equation* (1.6) *can be rewrit-*

ten as

$$\min_{x\in X_q} \min_{\substack{g_-(x)\le w\\ \le g_+(x)}} \mathbb{P}_\nu(f(x)+w+K_{q^*}(x;\hat{g})+\nu\in X_{q'}) \tag{1.15}$$

$$= \min_{x\in X_q} \min_{g_-(x)\le w\le g_+(x)} \mathbb{P}_\nu(c_{q^*}+w-\hat{g}(x)+\nu\in q') \tag{1.16}$$

$$= \min_{x\in X_q} \min_{-\gamma(x)\le\omega\le\gamma(x)} \mathbb{P}_\nu(c_{q^*}+\omega+\nu\in X_{q'}) \tag{1.17}$$

$$= \min_{x\in X_q} \min_{\substack{-\gamma(x)\le\omega\\ \le\gamma(x)}} \prod_{i=1}^{n} \mathbb{P}_{\nu_i}(c_{q^*,i}+\omega_i+\nu_i\in[a_{q',i},b_{q',i}]) \tag{1.18}$$

$$= \min_{x\in X_q} \prod_{i=1}^{n} \min_{\substack{-\gamma_i(x)\le\omega_i\\ \le\gamma_i(x)}} \mathbb{P}_{\nu_i}(c_{q^*,i}+\omega_i+\nu_i\in[a_{q',i},b_{q',i}]) \tag{1.19}$$

$$\ge \prod_{i=1}^{n} \min_{\substack{-\gamma_i(q)\le\omega_i\\ \le\gamma_i(q)}} \mathbb{P}_{\nu_i}(c_{q^*,i}+\omega_i+\nu_i\in[a_{q',i},b_{q',i}]) \tag{1.20}$$

where (1.15) *is the righthand side of* (1.6)*;* (1.16) *follows after expanding the feedback controller expression* $K_{q^*}(x;\hat{g})$ *using* (1.5) *and simplifying;* (1.17) *follows by assumption of high-confidence error bound* $\gamma(x)$ *and the definition of* $g_-(x)$ *and* $g_+(x)$ *from Assumption 1.1 and taking* $\omega = w-\hat{g}(x)$*;* (1.18) *follows by assumption that each* ν_i *is independent and* $\mathbb{P}_{\nu_i}$ *denotes probability with respect to* ν_i*, where* $a_{q'}$ *and* $b_{q'}$ *are the lower and upper corners of the region* $X_{q'}$*, and* $a_{q',i}$ *is the* i*-th coordinate of* $a_{q'}$ *and similarly for* $c_{q*,i}$ *and* $b_{q',i}$*;* (1.19) *follows from the fact that the hyper-rectangular constraint* $-\gamma(x)\le\omega\le\gamma(x)$ *is equivalent to independent constraint* $-\gamma_i(x)\le\omega_i\le\gamma_i(x)$ *along each coordinate; and* (1.20) *follows from the definition* $\gamma_i(q)=\max_{x\in X_q}\gamma_i(x)$*.*

Now, because the probability distribution for each random variable ν_i *is assumed unimodal and symmetric,* $\mathbb{P}_{\nu_i}(c_{q^*,i}+\omega_i+\nu_i\in[a_{q',i},b_{q',i}])$ *is minimized when the distance between* $(c_{q^*,i}+\omega_i)$ *and the midpoint of* $[a_{q',i},b_{q',i}]$ *is maximized, i.e., when* $|c_{q^*,i}+\omega_i-c_{q',i}|$ *is maximized, subject to the constraint* $-\gamma_i(q)\le\omega_i\le\gamma_i(q)$*. Substituting* $x=c_{q^*}+\omega$*, and observing that* $\|x-c_{q'}\|_1=\sum_{i=1}^n|x_i-c_{q',i}|$*, this is exactly the maximizing point specified by* $x_{\min}(q,\alpha_{q^*},q')$ *in* (1.13)*. Thus, the expression in* (1.20) *is equivalent to*

$$\prod_{i=1}^{n} \mathbb{P}_{\nu_i}(x_{\min,i}(q,\alpha_{q^*},q')+\nu_i\in[a_{q',i},b_{q',i}]), \tag{1.21}$$

which in turn is equivalent to the righthand side of (1.11)*, establishing the bound in* (1.6)*. An analogous argument establishes that* (1.12) *satisfies* (1.7)*.*

An illustration of transition probability interval calculations is shown in Figure 1.1, above in Section 1.2.

A high-confidence IMDP abstraction of the system is constructed

using the hyper-rectangular partition regions as states, high-confidence bounds on the unknown dynamics obtained via GP regression, and transition probability intervals calculated using Theorem 1.1, solving Problem 1.1.

1.4 Safe Sampling of PIMDP

In this section, an iterative learning algorithm is detailed which synthesizes control policies to learn the unknown dynamics and satisfy a given specification with sufficient probability. First, a PIMDP is constructed and a method is described to calculate specification satisfaction probabilities for PIMDP states, which will be necessary for later algorithms in this section. Then, a method is detailed to generate a sub-graph of the PIMDP which guarantees safety (i.e., nonviolation) with respect to a given scLTL specification. Next, a heuristic is developed to calculate infinite cycles within the nonviolating sub-graph and choose a particular cycle to execute. Finally, the complete algorithm is detailed which samples the state space iteratively, improving the GP approximation of the unknown dynamics until the given scLTL specification can be satisfied or the GP uncertainty is sufficiently small.

1.4.1 Probability of Satisfaction Calculation

Given a high-confidence IMDP abstraction of the system and a FSA of a desired scLTL specification, a PIMDP is constructed using Definition 1.5. Then, to find safe sampling cycles in the PIMDP, the probability that a random path w starting at PIMDP state (q, s) satisfies the scLTL specification ϕ under a maximizing control policy π and minimizing adversary ξ is calculated:

$$\check{P}_{\max}((q,s) \models \phi) = \max_{\pi \in \Pi} \min_{\xi \in \Xi} P(w \models \phi \mid \pi, \xi, w[0] = (q,s)),$$

Additionally, the best case probability of satisfaction under a maximizing control policy and adversary will also be used:

$$\hat{P}_{\max}((q,s) \models \phi) = \max_{\pi \in \Pi} \max_{\xi \in \Xi} P(w_i \models \phi \mid \pi, \xi, w[0] = (q,s))$$

To calculate these probabilities, a method proposed in Section V of [14] can be used, which is briefly summarized here. First, an ordering of all the states in the PIMDP is constructed based on their initial probability of satisfying the specification of interest. Then, for each state-action pair, ϕ-minimizing adversary is constructed by assigning

the maximum probability to one-step reachable states which have the lowest probabilities of satisfying ϕ. Similarly, a ϕ-maximizing adversary is constructed by assigning as much probability as possible to transition states which have the highest probabilities of satisfying ϕ.

The state ordering method for resolving adversarial transition probabilities naturally lends itself to a value iteration method in order to determine $\check{P}_{\max}$ and $\hat{P}_{\max}$ for every state with arbitrary accuracy. First, the probabilities of satisfaction for each state are initialized by setting $\check{P}_{\max} = \hat{P}_{\max} = 1$ for states (q, s) such that s is an accepting state in the FSA, and $\check{P}_{\max} = \hat{P}_{\max} = 0$ for all other states. Then, adversarial transition probabilities for each state are selected using the state-ordering method described above. At each step, the next iteration of the probability of satisfaction $\hat{P}^{+}_{\max}$ or $\check{P}^{+}_{\max}$ for each state-action pair is calculated by multiplying the appropriate transition probability to each one-step reachable state by that state's current probability of satisfaction and summing across all reachable states:

$$\hat{P}_{\max}((q,s),a)^{+} = \sum_{i=1}^{n} T((q,s),a,r_i) \cdot \hat{P}_{\max}(r_i)$$

The new probability of satisfaction for each state corresponds to the maximizing control action:

$$\hat{P}_{\max}((q,s))^{+} = \max_{a \in A} \hat{P}_{\max}((q,s),a)^{+}$$

The same process can be used to find $\check{P}_{\max}$ with appropriate state-ordered transition probabilities.

1.4.2 Nonviolating Sub-Graph Generation

Note that scLTL specifications may generate FSA states which are absorbing and non-accepting, i.e., it is impossible to satisfy the specification once one of these states is reached. Such states may also exist in PIMDP constructions even without appearing in the corresponding FSA. In this chapter, these so-called *failure states* are defined as those which have zero probability of satisfying the scLTL specification under any control policy and adversary:

$$\text{Failure States} = \{(q,s) \in Q \times S \mid \hat{P}_{\max}((q,s) \models \phi) = 0\}.$$

Example 1.1 Failure States

Figure 1.2 shows a simple example of failure states. State 4 is the accepting state for the system, and the transitions are denoted in the form "action:(lower

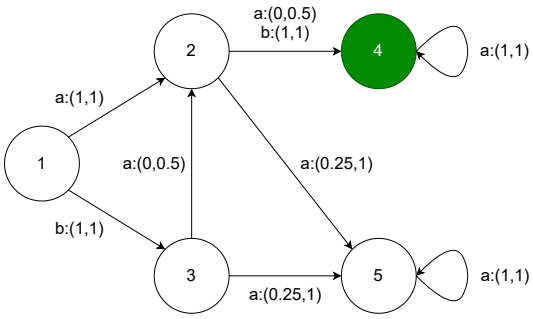

Figure 1.2: Illustration of failure states in a PIMDP. State 5 can never reach accepting state 4 (in green), so state 5 is considered a failure state.

bound, upper bound)". Since state 5 can never reach state 4, it is a failure state. Note that while state 3 transitions to state 5 with probability 1 under a minimizing adversary and thus has $\check{P}_{\max} = 0$*, it can reach state 4 under a maximizing adversary with* $\hat{P}_{\max} > 0$ *and thus is not considered a failure state.*

Then, a notion of specification nonviolation can be defined:

Definition 1.10 Nonviolating PIMDP A PIMDP $\mathcal{P}$ is *nonviolating* with respect to a scLTL specification ϕ if there exists no failure states in $\mathcal{P}$.

An algorithm for calculating a nonviolating PIMDP is detailed in Algorithm 1, solving Subproblem 1.1. It takes as input a PIMDP construction along with upper bounds $\hat{P}_{\max}$ on the maximum probability of satisfying the specification for each state. In lines 1–2, failure states which have $\hat{P}_{\max} = 0$ are identified. Next, in lines 4–10, non-failure states are looped through and any of their actions which have a nonzero upper bound probability of reaching the set of failure states are removed. In lines 12–17, states which have no actions remaining are added to the set of failure states and the algorithm returns to line 3. The loop from lines 3–18 repeats until no new failure states are identified. The algorithm returns the remaining non-failure states and actions as the nonviolating sub-graph.

Example 1.2 Pruning
An illustration of the pruning technique for the system in Figure 1.2 is shown

Algorithm 1 Nonviolating Sub-Graph Generation

Input: PIMDP $\mathcal{P}$, $\hat{P}_{\max}$ for each state in $\mathcal{P}$
Output: PIMDP $\mathcal{P}$' which is a nonviolating subset of $\mathcal{P}$
Initialize $R = \{(q, s) \in \mathcal{P} \mid \hat{P}_{\max}(q, s \models \phi) = 0\}$
Initialize $U = R$
while $R \neq \emptyset$ **do**
    for $(q, s) \in \mathcal{P} \setminus U$ **do**
        for $\alpha \in A$ **do**
            if $\hat{T}((q, s), \alpha, U) \neq 0$ **then**
                Remove α from available actions at (q, s)
            end if
        end for
    end for
    $R = \emptyset$
    for $(q, s) \in \mathcal{P} \setminus U$ **do**
        if $A((q, s)) = \emptyset$ **then**
            $R = R \cup (q, s)$
            $U = U \cup (q, s)$
        end if
    end for
end while
Return $\mathcal{P}' = \mathcal{P} \setminus U$

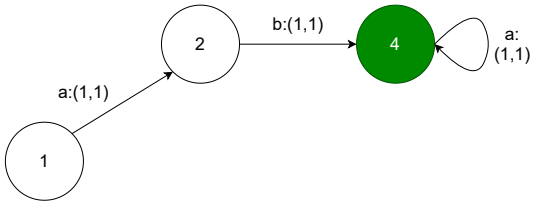

Figure 1.3: Illustration of the nonviolating sub-graph for the example shown in Figure 1.2. State 5 along with transitions to it were pruned, leaving state 3 with no actions. Therefore, state 3 also became a failure state, so it was pruned along with the actions leading to it, leaving the nonviolating sub-graph shown here.

in Figure 1.3. In this example, state 5 was found to be the only failure state, so it was pruned from the graph along with the actions with transitions to it from states 2 and 3. However, pruning action "a" left state 3 with no actions, so it in turn also became a failure state and the transition to it from state 1 was also pruned, leaving the nonviolating sub-graph shown in the figure.

1.4.3 Candidate Cycle Selection

Now that a nonviolating sub-graph of the PIMDP has been calculated, the next problem is to select a path which the robot can traverse in order to sample the state space indefinitely while maximizing the information gain of the Gaussian process learning of the unknown dynamics. To do this, the concept of *maximal end components* [28] is introduced:

Definition 1.11 End Component [28] An *end component* of a finite PIMDP $\mathcal{P}$ is a pair $(\mathcal{T}, Act)$ with $\mathcal{T} \subseteq (Q \times S)$ and $Act : \mathcal{T} \to A$ such that

- $\emptyset \neq Act(q,s) \subseteq A(q)$ for all states $(q,s) \in \mathcal{T}$,
- $(q,s) \in \mathcal{T}$ and $\alpha \in Act(q,s)$ implies $\{(q',s') \in \mathcal{T} \mid \hat{T}(q,\alpha,q')) > 0, s' \in \delta(s, L(q))\} \subseteq \mathcal{T}$,
- The digraph $G_{(\mathcal{T},Act)}$ induced by $(\mathcal{T}, Act)$ is strongly connected.

Definition 1.12 Maximal End Component (MEC) [28] An end component $(\mathcal{T}, Act)$ of a finite PIMDP $\mathcal{P}$ is *maximal* if there is no end component $(\mathcal{T}^*, Act^*)$ such that $(\mathcal{T}, Act) \neq (\mathcal{T}^*, Act^*)$ and $\mathcal{T} \subseteq \mathcal{T}^*$ and $Act(q,s) \subseteq Act^*(q,s)$ for all $(q,s) \in \mathcal{T}$.

PIMDP abstractions have the property that any infinite path will eventually stay in a single MEC. Then, the following heuristic is used to select a MEC to cycle within.

First, the value $\check{P}_{\max}$ is calculated from the initial state to each candidate MEC using the value iteration method detailed previously. Any MEC which cannot be reach with probability 1 is rejected, or, in case no MECs can be reached with probability 1, the MEC with the highest reachability probability is selected. If multiple candidate MECs remain, the Gaussian process covariance $\kappa(c_q, c_{q^*})$ is then calculated between the centers of the IMDP states q in each remaining candidate MEC and the accepting IMDP state q^*. The covariances for all states in each MEC are summed and the MEC with the highest total covariance score is selected, which corresponds to maximum information gain [29], defined as reduction of GP uncertainty at the accepting state. A control policy is generated by selecting the actions at each state which give the maximum probability of reaching the MEC. Once in the MEC, a controller is used which cycles through the available actions. This method was shown in [28] to guarantee fair reachability of each state within the MEC. The heuristic is summarized in Algorithm 2.

Algorithm 2 takes as input a nonviolating sub-graph of a PIMDP. In line 1, the maximal end components of the sub-graph are identified. In lines 3–8, a lower bound $\check{P}_{\max}$ on the maximum probability of reaching each MEC from the initial state of the original PIMDP is calculated. Those MECs which have $\check{P}_{\max} = 1$ are added to the list of candidate MECs. In lines 9–11, if there are no candidate MECs found, then the algorithm selects the MEC with the highest $\check{P}_{\max}$ as the MEC to cycle in. If there are candidate MECs, then in lines 12–17 each candidate MEC is assigned a score equal to the sum of the covariances between each state in the MEC and the accepting state q^* of the PIMDP. The MEC with the maximum covariance score is selected as the MEC to cycle in. In line 18, a control policy for the selected MEC is calculated which selects the actions at each state outside the MEC which have maximum probability of reaching the MEC. For states within the MEC, the control policy cycles through the actions at each state which are available in the MEC. The algorithm returns the selected MEC along with its corresponding control policy. By applying the algorithms detailed above to calculate a non-violating PIMDP and MEC, a control policy which samples the state space indefinitely without violating the specification is generated, solving Subproblem 1.2.

Algorithm 2 Nonviolating Cycle Selection

Input: Nonviolating Sub-PIMDP $\mathcal{P}'$
Output: Selected MEC $(\mathcal{T}^*, Act^*)$ and control policy π^*
Initialize M as the MECs of $\mathcal{P}'$
Initialize $C = \emptyset$ as the set of candidate MECs
for $(\mathcal{T}^\dagger, Act^\dagger) \in M$ **do**
    Calculate reachability probability $\check{\mathcal{P}}_{\max}$ from initial state r_0 of PIMDP $\mathcal{P}'$ to $\mathcal{T}^\dagger$
    if $\check{P}_{\max} = 1$ **then**
        $C = C \cup (\mathcal{T}^\dagger, Act^\dagger)$
    end if
end for
if $C = \emptyset$ **then**
    $(\mathcal{T}^*, Act^*) = \underset{(\mathcal{T}^\dagger, Act^\dagger) \in M}{\operatorname{argmax}} \check{P}_{\max}(r_0 \models \Diamond \mathcal{T}^\dagger)$
else
    for $(\mathcal{T}^\dagger, Act^\dagger) \in C$ **do**
        $H = \text{Sum } \kappa(c_q, c_{q^*})$ for all IMDP states $q \in \mathcal{T}^\dagger$ w.r.t. accepting state q^*
    end for
    Find $(\mathcal{T}^*, Act^*) \in C$ with maximum score H
end if
Construct Control policy π^* which selects available actions in $\mathcal{P}'$ with maximum probability of reaching $\mathcal{T}^*$ for states not in $\mathcal{T}^*$ and cycles through actions Act^* for all states in $\mathcal{T}^*$
Return Selected MEC $(\mathcal{T}^*, Act^*)$, control policy π^*

1.4.4 Safety Guarantees

The exploration of Problem 1.2 is completed by addressing Subproblem 1.3, guaranteeing the safety of Algorithms 1 and 2. A proof is given for the property that any control policy generated using these algorithms is guaranteed never to visit any failure states which have maximum probability 0 of satisfying a given scLTL specification ϕ:

Theorem 1.2 Safety of Algorithms 1 and 2
Following the control policy π^* generated by Algorithms 1 and 2 guarantees that the system will never enter into a failure state which almost surely violates a given scLTL specification ϕ.

Proof 1.2 Failure states are characterized as those states which are absorbing and non-accepting, i.e. they have probability 0 of satisfying ϕ under any path regardless of the actions or transition probabilities chosen. Therefore, failure states can be identified as states which have $\hat{P}_{\max} = 0$ of satisfying ϕ under a maximizing adversary. By construction, Algorithm 1 identifies these failure states and removes them from the nonviolating sub-graph $\mathcal{P}'$ along with any transitions which lead to them. Therefore, in the sub-graph $\mathcal{P}'$ it is impossible to reach a failure state under any available control action. Since Algorithm 2 calculates a control policy using only actions in $\mathcal{P}'$, the path generated by Algorithm 2 is guaranteed to avoid failure states.

1.4.5 Iterative Sampling Algorithm

In this section the complete iterative sampling algorithm is detailed which solves Problem 1.3.

Given a scLTL specification ϕ with desired probability of satisfaction P_{sat}, a PIMDP is first constructed using a high-confidence IMDP abstraction of the system in Eq. (1.1) and an FSA which models ϕ. Then, reachability probabilities under a minimizing adversary $\check{P}_{\max}$ are calculated from the initial states in the PIMDP to the accepting states. If $\check{P}_{\max} \geq P_{\text{sat}}$, then the control policy selects the actions which produce $\check{P}_{\max}$ at each state and the problem is solved. Otherwise, Algorithms 1 and 2 are used to calculate a control policy to sample the state space without violating the specification ϕ using the methods in previous sections. The robot executes the calculated control policy for a predetermined number of steps and samples the unknown dynamics at each step. The GP is batch updated with the data collected, transition probability intervals are reconstructed for each state, and reachability

probabilities $\check{P}_{\max}$ are recalculated for the initial states. If $\check{P}_{\max} \geq P_{\text{sat}}$, a satisfying control policy is found; otherwise, the process above is repeated.

The iterative algorithm ends when $\check{P}_{\max} \geq P_{\text{sat}}$; the GP approximation has low enough uncertainty to know that a successful control policy cannot be synthesized, *i.e.*, when the reachability probability $\hat{P}_{\max}$ under a maximizing adversary is less than the desired P_{sat}; or a maximum number of iterations has been reached. The complete method is detailed in Algorithm 3.

Algorithm 3 takes as input the system dynamics, a scLTL specification, and a desired probability of satisfaction P_{sat}. In line 1, an IMDP abstraction of the system and a FSA of the specification are constructed. Then, the IMDP and FSA are combined into a PIMDP construction. Finally, a GP estimation $\hat{g}(x)$ of the unknown dynamics is initialized with its hyperparameters. In lines 2–3, lower and upper bounds $\check{P}_{\max}$ and $\hat{P}_{\max}$ are calculated for the initial state in the PIMDP. If the lower bound probability $\check{P}_{\max}$ is less than the desired P_{sat}, the loop in lines 4–18 is entered. In lines 5–7, if the upper bound probability $\hat{P}_{\max}$ is less than P_{sat}, then the specification cannot be satisfied with sufficient probability regardless of how well the unknown dynamics are learned. Thus, the algorithm returns that no satisfying control policy exists. Otherwise, in lines 8–9, a nonviolating sub-graph of the PIMDP is calculated using Algorithm 1 and a MEC to cycle in along with its corresponding control policy is calculated from this sub-graph using Algorithm 2. In lines 10–13, this control policy is used to take a predefined number of steps to sample the unknown dynamics. In lines 14–15, the GP estimation $\hat{g}(x)$ is updated using these samples, and transition probability intervals are recalculated for each state in the PIMDP. In lines 16–17, $\check{P}_{\max}$ and $\hat{P}_{\max}$ are recalculated for the initial state. If $\check{P}_{\max} \geq P_{\text{sat}}$, the loop terminates and the algorithm returns a control policy calculated in lines 19–21 which selects the actions at each state which have maximum probability of satisfying the specification. If $\check{P}_{\max} < P_{\text{sat}}$, the loop repeats from line 4. If the maximum number of iterations of the loop is reached, the algorithm terminates without determining a satisfying control policy or the nonexistence thereof.

Algorithm 3 Iterative Synthesis Algorithm

Input: System dynamics in (1.1), scLTL specification ϕ, P_{sat}
Output: Satisfying control policy $\pi^\dagger$ or proof of nonexistence
Construct IMDP $\mathcal{I}$ from System 1.1, FSA $\mathcal{A}$ from ϕ, PIMDP $\mathcal{P}$ from $\mathcal{I}$ and $\mathcal{A}$, initial GP regression $\hat{g}(x)$
Calculate $\check{P}_{\max}((q_0, \delta(q_0, s_0)) \models \phi)$ for initial state
Calculate $\hat{P}_{\max}((q_0, \delta(q_0, s_0)) \models \phi)$ for initial state
while $(\check{P}_{\max} < P_{\text{sat}})$ **and** (Count< MaxIterations) **do**
    if $\hat{P}_{\max} < P_{\text{sat}}$ **then**
        Return No satisfying control policy exists
    end if
    Find nonviolating sub-PIMDP $\mathcal{P}'$ using Algorithm 1
    Find MEC to cycle in with corresponding control policy π^* using Algorithm 2
    for NumInnerIterations **do**
        Take action $\pi^*(q)$ at current state q
        Sample $y[k] = x[k+1] - f(x[k]) - u[k]$
    end for
    Construct GP $\hat{g}(x)$ using collected samples $y[k]$
    Recalculate transition probability intervals for each state in $\mathcal{P}$
    $\check{P}_{\max}((q_0, \delta(q_0, s_0)) \models \phi)$ for initial state
    $\hat{P}_{\max}((q_0, \delta(q_0, s_0)) \models \phi)$ for initial state
    if $\check{P}_{\max} \geq P_{\text{sat}}$ **then**
        Return Control policy $\pi^\dagger = \underset{\alpha \in A(q)}{\text{argmax}}\ \check{P}_{\max}((q, s) \models \phi)\ \forall (q, s) \in \mathcal{P}$
    end if
end while

1.5 Extension to LTL

In this section, the methods detailed above for scLTL specifications are generalized to the entire semantics of Linear Temporal Logic, defined as follows:

Definition 1.13 Linear Temporal Logic [5, Def. 2.1, 2.2] A *linear temporal logic (LTL)* formula ϕ over a set of observations O is recursively defined as

$$\phi = \top \mid o \mid \neg o \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \bigcirc \phi \mid \\ \phi_1 \mathcal{U} \phi_2 \mid \Diamond \phi \mid \Box \phi \mid \phi_1 \rightarrow \phi_2 \mid \phi_1 \leftrightarrow \phi_2$$

LTL encompasses the syntax of scLTL with the addition of the *always* operator $\Box$ defined as $\neg\Diamond\neg\top$.
Additionally, LTL includes the operators

$$\phi_1 \rightarrow \phi_2 := \neg\phi_1 \vee \phi_2$$
$$\phi_1 \leftrightarrow \phi_2 := (\phi_1 \rightarrow \phi_2) \wedge (\phi_2 \rightarrow \phi_1)$$

Finally, LTL operators can be combined to form new operators such as $\Diamond\Box$ ("eventually always") and $\Box\Diamond$ ("always eventually").

As compared to scLTL, LTL includes classes of tasks which capture limit behavior of the system as well as infinite-time tasks. Thus, the methods detailed above which relied on checking finite end behavior in a finite state automaton are not directly applicable when generalizing to LTL specifications.

However, analogously to scLTL-FSA equivalence, satisfaction of LTL formulas can be checked using an equivalent deterministic Rabin automaton (DRA):

Definition 1.14 Deterministic Rabin Automaton [5, Def. 2.7] A deterministic Rabin automaton is a tuple $\mathcal{R} = (S, s_0, O, \delta, F)$, where

- S is a finite set of states,
- $s_0 \subset S$ is a singleton initial state,
- O is the input alphabet, which corresponds to observations from the LTL formula,
- $\delta : S \times O \rightarrow 2^S$ is a transition map which is either $\emptyset$ or a singleton for all $s \in S$ and $o \in O$, and

- $F = \{(G_1, B_1), \cdots, (G_n, B_n)\}$, where $G_i, B_i \subseteq S, i = 1, 2, \cdots, n$ is the acceptance condition.

The semantics of a Rabin automaton are defined over infinite input words in O^ω (the set of infinite sequences of observations). A run of $\mathcal{R}$ over an infinite word $w_O = w_O(1), w_O(2)w_O(3) \cdots \in O^\omega$ is a sequence $w_S(1)w_S(2)w_S(3) \cdots \in S^\omega$, where $w_S(1) = s_0$ and $w_S(k+1) = \delta(w_S(k), w_O(K))$ for all $k \geq 1$.
A run w_S admits a set $\inf(w_S) = \{w_S(i) : \forall m \in \mathbb{N}\ \exists k > m \text{ s.t. } w_S(k) = w_S(i)\}$, defined as the set of observations in w_S which appear infinitely often. Similarly, there exists a set of infinitely appearing observations $\inf w_O$ in any infinite word w_O. Then, a run w_S is *accepted* by $\mathcal{R}$ if $\inf(w_S) \cap G_i \neq \emptyset \wedge \inf(w_S) \cap B_i = \emptyset$ for some $i \in \{1, \cdots, n\}$, *i.e.* any of the acceptance conditions are met. If w_S is accepted by $\mathcal{R}$, the equivalent notation is $w_s \models \mathcal{R}$. Intuitively, acceptance condition i stipulates that all observations from G_i are seen infinitely often and all observations from B_i are seen only finitely often.

Then, similarly to the scLTL case, the next step is to construct a product IMDP composed of the IMDP of the system dynamics and the DRA of the LTL specification:

Definition 1.15 PIMDP (LTL) Let $\mathcal{I} = (Q, A, \check{T}, \hat{T}, Q_0, O, L)$ be an IMDP and $\mathcal{A} = (S, s_0, O, \delta, F)$ be an DRA. The product IMDP (PIMDP) is defined as a tuple $\mathcal{P} = \mathcal{I} \otimes \mathcal{A} =$
$(Q \times S, A, \check{T}', \hat{T}', Q \times s_0, F')$, where

- $\check{T}' : (q, s) \times A \times (q', s') := \check{T}(q, \alpha, q')$ if $s' \in \delta(s, L(q))$ and 0 otherwise
- $\hat{T}' : (q, s) \times A \times (q', s') := \hat{T}(q, \alpha, q')$ if $s' \in \delta(s, L(q))$ and 0 otherwise
- $(q_0, \delta(s_0, L(q_0))) \in (Q \times S)$ is a set of initial states of $\mathcal{I} \otimes \mathcal{A}$, and
- $F' = Q \times F = \{Q \times (G_1, B_1), \cdots, Q \times (G_n, B_n)\}$, where $G_i, B_i \subseteq S, i = 1, 2, \cdots, n$ is the acceptance condition.

Unlike in the scLTL case, where the acceptance condition of the PIMDP was a set of accepting states, in the LTL version of the PIMDP the acceptance condition requires checking multiple pairs of accepting and rejecting states. The key insight to addressing this complication is the use of MECs, which were introduced earlier in this chapter to address the problem of selecting paths to traverse during the online sampling process. Drawing on ideas in Chapter 10 of [13], checking the acceptance condition of a PIMDP formed using a LTL specification corresponds to checking reachability of accepting MECs in the PIMDP. Specifically, the accepting MECs of the PIMDP are those for which at least one acceptance condition i holds, i.e. none of the states in the

MEC are contained in the set of rejecting states B_i and the MEC contains all of the states in the accepting set G_i. Then, calculating the lower and upper bound probability of satisfying the LTL specification from any state in the PIMDP can be done by calculating the probability of reaching any accepting MEC from the state under the appropriate (minimizing or maximizing) adversary and a maximizing control policy.

With this insight, the methods for scLTL-based PIMDP verification and synthesis detailed in previous sections can adapted to accommodate LTL specifications by replacing the accepting states in the scLTL-based PIMDP with accepting MECs in the LTL-based PIMDP and modifying the calculations of satisfaction probabilities accordingly. With the modified satisfaction probability calculations, the remainder of Algorithms 1, 2, and 3 can be executed as detailed for the scLTL cases, as the PIMDP structures in the scLTL and LTL cases differ only in their accepting conditions. Thus, the introduction of a LTL-based PIMDP structure allows for the generalization of the iterative synthesis algorithm to the entire class of LTL specifcations.

1.6 Case Studies

Two case studies are considered to demonstrate the algorithms detailed in previous sections. The first case study illustrates a basic reach-avoid task a robot might be expected to accomplish, while the second case study uses a more complicated specification which demonstrates the expressivity of the LTL specifications which can be accommodated by these methods.

1.6.1 Reach-Avoid Specification

Consider a mobile robot in a 2D state space with position $x \in X := [0, 5]^2 \subset \mathbb{R}^2$. The state space is partitioned into a set of 25 hyper-rectangular regions corresponding to IMDP states. The dynamics of the robot are

$$x[k+1] = x[k] + u[k] + g(x[k]) + v \tag{1.22}$$

where $g(x)$ models the unknown effect of the slope of the terrain. The use of $g(x)$ to model unknown dynamics for locomotion systems is motivated by previous explorations in the literature, e.g. [30, 31, 32]. The control action u is generated by the family of controllers in Section 1.2 where the set of available target regions are those left, right, above, or below each region.

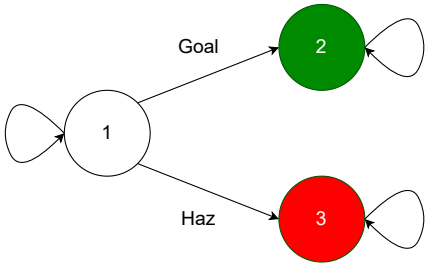

Figure 1.4: Finite state automaton of the scLTL specification for the reach-avoid specification. The system starts in state 1, which has a self-transition until a hazard or goal state is reached. State 2 is the accepting state, and state 3 is a non-accepting absorbing state. Therefore, all states (q, s) in the product IMDP such that $q \in Q$ and $s = 3$ are failure states, as are any states which have nonzero probability of transitioning to failure states under all actions.

Within the state space, there exists one goal region with the atomic proposition `Goal` and a set of hazard regions labeled with `Haz`. These yield the scLTL specification

$$\phi_1 = \neg \texttt{Haz}\, \mathcal{U}\, \texttt{Goal}. \tag{1.23}$$

The FSA generated by this specification is shown in Figure 1.4.

An illustration of the state space is shown in Figure 1.5.

The true $g(x)$ is sampled from two randomly generated Gaussian processes (one for each dimension) with bounded support $[-0.4, 0.4]$ and squared exponential kernel κ,

$$\kappa(x, x') = \sigma_g^2 e^{-\frac{(x-x')^2}{2l^2}}. \tag{1.24}$$

The hyperparameters are chosen to be $\sigma_g = 0.45$ and $l = 1.75$.

The unknown dynamics are estimated with two sparse Gaussian processes with the same kernel as the true dynamics, but which have no knowledge of the system initially. The GPs are sampled at 100 points in each region to determine error bounds. The number of inducing points is set at $\eta = 250$ and the high-confidence-bound parameter is chosen to be $\beta = 2$. Each iteration of the algorithm takes 250 steps, so the total number of data samples m is the number of iterations times 250. The stochastic noise ν is independently drawn from two truncated Gaussian distributions, one for each dimension, and both with $\sigma_\nu = 0.1$ and bounded support $[-0.2, 0.2]$. Note that all distributions with bounded support are sub-Gaussian.

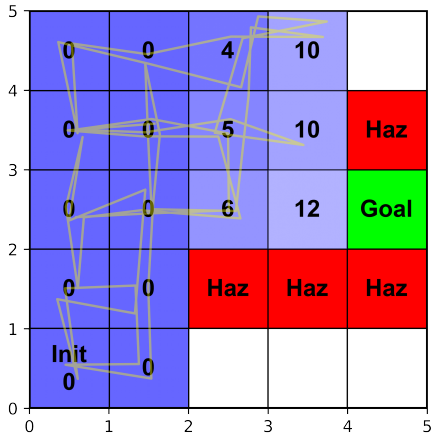

Figure 1.5: State space of the reach-avoid case study. The initial region is labeled with "Init", the (green) target region is labeled with "Goal", and the (red) hazard regions are labeled with "Haz". States that eventually enter the safe cycle are blue, and the number in the region indicates the iteration of the algorithm at which the state enters the safe cycle. States which are not numbered do not enter the safe cycle. The yellow trace is an example of a sampling run.

Remark 1.2 The rationale for the selection of the GP hyperparameters is as follows. The length parameter l is large enough that sampling states gives information about the adjacent states, thus enabling the iterative learning process. Increasing the variance of the learned dynamics or the stochastic noise results in larger uncertainty in the system, which can be overcome at the expense of more iterations of the algorithm to collect data samples. ▮

Next, the iterative algorithm described in Section 1.4.5 is executed, setting the desired probability of satisfying the specification to 1. The algorithm successfully finds a satisfying feedback control strategy in an average of 15 iterations (calculated over 10 runs). The algorithm is implemented in Python on a 4.5 GHz AMD Ryzen 7700X machine with 32 GB of RAM and a Nvidia RTX 3090 GPU, and requires on average 48 seconds to complete.

Figure 1.5 depicts the expansion of the safe cycle used to sample the state space. Initially, only the left two columns of states are safe and reachable. As the algorithm progresses, more states and actions are added to the safe cycle, moving the system closer to the goal until the unknown dynamics can be estimated with enough certainty to achieve a probability of satisfying the specification of 1.

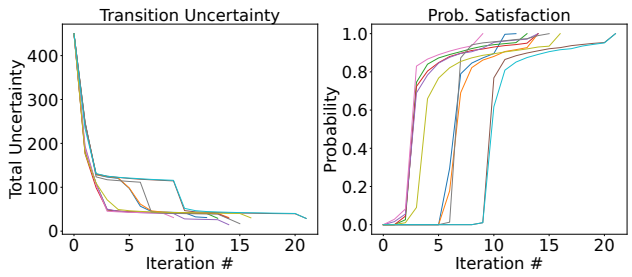

Figure 1.6: The left plot shows the total uncertainty in transition probability intervals after each iteration of the algorithm, and the right plot shows the probability of satisfying the specification after each iteration. Results are plotted over 10 runs of the algorithm. The uncertainty decreases as more data samples are collected, and likewise the probability of satisfaction increases once the safe cycle has expanded close enough to the goal.

The left plot in Figure 1.6 depicts the total transition probability uncertainty for the system after each iteration

$$T_{\text{unc,total}} = \sum_{q \in Q} \sum_{\alpha \in A(q)} \sum_{q' \in Q} \hat{T}(q, \alpha, q') - \check{T}(q, \alpha, q'). \tag{1.25}$$

As the number of iterations increases, the transition probability uncertainty decreases as the Gaussian process estimation of the unknown dynamics becomes more precise.

The right plot in Figure 1.6 shows the probability of satisfying the specification after each iteration. A control policy which satisfied the specification with probability 1 was successfully calculated for all runs of the algorithm.

1.6.2 Branching Objective

Consider the same mobile robot described in the previous case study, this time in a 10x10 state space. In this case, the robot still needs to reach a goal state while avoiding hazards, but it must also reach one of two alternative intermediate checkpoints. It can either choose to reach the checkpoint A, which is isolated by hazards, or reach the checkpoints B1 and B2 sequentially, which are individually easier to reach but requires traversal of more of the state space. Figure 1.8 shows the case study environment.

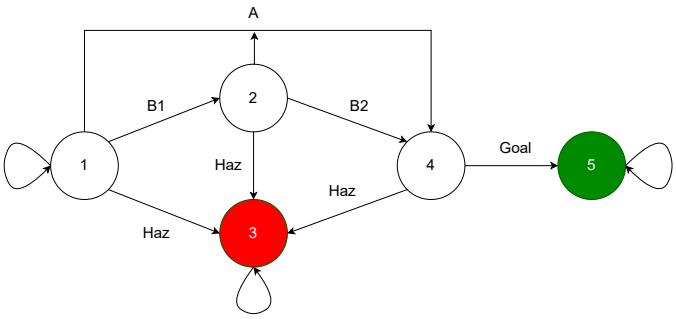

Figure 1.7: Finite state automaton of the scLTL specification for the second case study. The system starts in state 1, which has a self-transition until a hazard or the intermediate checkpoints A or B1 are reached. State 5 is the accepting state, and state 3 is a non-accepting absorbing state. States 2 and 4 are intermediate states which keep track of the checkpoints already reached.

The specification can be written using scLTL as

$$\phi_2 = \neg \texttt{Haz}\, \mathcal{U}\, \texttt{Goal} \ \wedge\ (\neg \texttt{Goal}\, \mathcal{U}\, [\texttt{A} \vee (\texttt{B1} \ \wedge\ \bigcirc \texttt{B2})]). \tag{1.26}$$

Figure 1.7 depicts the FSA of the specification. The hyperparameters for this case study are $\sigma_g = 0.24$ and $l = 1$. Each iteration of the algorithm, the robot takes 50 steps, and correspondingly the number of inducing points in the sparse GPs are set to 50. Otherwise, the parameters of the setup remain the same as in the first case study.

The algorithm takes seven iterations to satisfy the specification with probability 1, and requires 35 minutes and 14 seconds to complete the computations. The state space of the PIMDP for this case study is approximately 12 times the size of the state space in the previous case study but the computation time is 44 times as long, illustrating the curse of dimensionality inherent to abstraction-based approaches.

Figure 1.8 depicts the trajectory of the robot through the state space. Initially, only the left two columns of the state space are in the safe subgraph, so the robot samples these states until it learns the uncertainty sufficiently well to explore states further to the right. The robot chooses to visit the checkpoints B1 and B2 before traversing towards the goal, perhaps because it has more knowledge about the states closer to these checkpoints than the states closer to checkpoint A. Ultimately, the robot is able to safely traverse the environment to learn

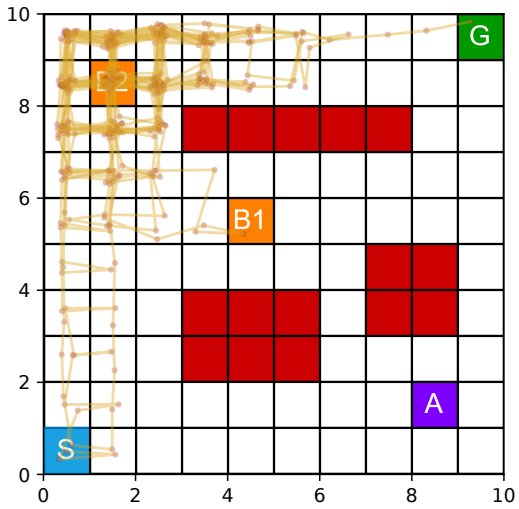

Figure 1.8: State space of the complex case study. The initial region is labeled with "S", the (green) target region is labeled with "G", and the hazard regions are red. Additionally, there exists intermediate checkpoints "A" (purple) and the pair "B1" and "B2" (orange). The yellow trace depicts the robot's trajectory, which initially stays in the left two columns until it ventures further right in order to collect sufficient information to satisfy the specification.

the uncertainties and then synthesize a control policy which satisfies the specification despite the complex nature of the objective.

Figure 1.9 shows the plots of the total transition probability uncertainty and iterative probability of satisfaction for this case study, illustrating the effectiveness of the iterative learning process.

1.7 Conclusion

This chapter explored the use of abstraction-based methods in concert with Gaussian process learning to enable complex task planning for robotic systems in the presence of uncertainties. We saw that IMDPs are well-suited to formally model uncertainties in system dynamics, and we incorporated Gaussian process learning of uncertainties into the IMDP model. Then, using a product IMDP construction of the system IMDP combined with either a finite state automaton of a scLTL specification or a deterministic Rabin automaton of a LTL specification,

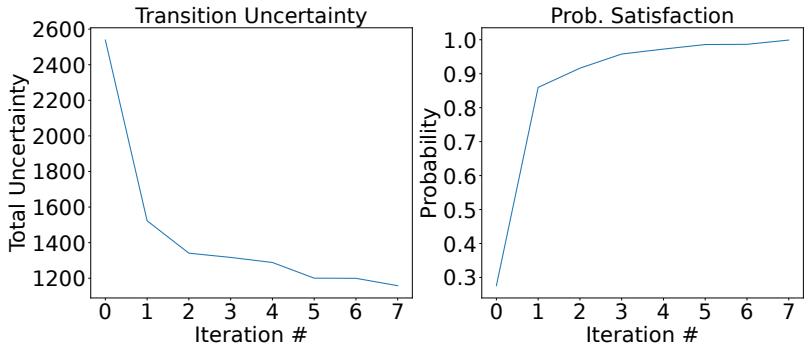

Figure 1.9: The left plot shows the total uncertainty in transition probability intervals after each iteration of the algorithm, and the right plot shows the probability of satisfying the specification after each iteration. The uncertainty decreases as more data samples are collected, and likewise the probability of satisfaction increases once the safe cycle has expanded close enough to the goal.

we developed algorithms which allow the robot to safely traverse its environment, sampling and learning uncertainties online to improve its probability of satisfying the specification.

The methodology developed in this chapter is applicable to a wide range of robotic systems, allowing for the use of learning methods in concert with formal guarantees on system behavior from abstraction-based methods. Future work in this area remains in addressing computational complexity considerations arising from operations on the abstracted system, integrating more complex learning structures (e.g. stacked Gaussian processes [33]), and extending these methods to complex, high-order robotic systems (e.g. legged robots [26]).

IMDP methods show great promise in combining techniques from the fields of formal methods, learning, and locomotion to enable novel capabilities for real-world robotic operation. Thus, it is our hope that the work detailed in this chapter will inspire further explorations in this area to enable safe, efficient, and effective learning and control for robotic systems.

References

[1] M. L. Bujorianu, J. Lygeros, and M. C. Bujorianu, "Bisimulation for General Stochastic Hybrid Systems," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, M. Morari and L. Thiele, Eds. Berlin, Heidelberg: Springer, 2005, pp. 198–214.

[2] N. Cauchi, L. Laurenti, M. Lahijanian, A. Abate, M. Kwiatkowska, and L. Cardelli, "Efficiency through Uncertainty: Scalable Formal Synthesis for Stochastic Hybrid Systems," *arXiv:1901.01576 [cs]*, Jan. 2019, arXiv: 1901.01576. [Online]. Available: http://arxiv.org/abs/1901.01576

[3] A. Abate, A. D'Innocenzo, M. D. Di Benedetto, and S. S. Sastry, "Markov set-chains as abstractions of stochastic hybrid systems," in *Hybrid systems: Computation and control*, M. Egerstedt and B. Mishra, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1–15.

[4] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Computer aided verification*, G. Gopalakrishnan and S. Qadeer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 585–591.

[5] C. Belta, B. Yordanov, and E. Göl, *Formal Methods for Discrete-Time Dynamical Systems*, ser. Studies in Systems, Decision and Control. Springer International Publishing, 2017.

[6] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 5319–5325.

[7] R. Givan, S. Leach, and T. Dean, "Bounded-parameter Markov decision processes," *Artificial Intelligence*, vol. 122, no. 1, pp. 71–109, Sep. 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370200000473

[8] E. M. Hahn, V. Hashemi, H. Hermanns, M. Lahijanian, and A. Turrini, "Interval markov decision processes with multiple objectives: From robust strategies to pareto curves," *ACM Trans. Model. Comput. Simul.*, vol. 29, no. 4, Nov. 2019. [Online]. Available: https://doi.org/10.1145/3309683

[9] K. Chatterjee, K. Sen, and T. A. Henzinger, "Model-checking ω-regular properties of interval markov chains," in *Foundations of Software Science and Computational Structures*, R. Amadio, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 302–317.

[10] K. Sen, M. Viswanathan, and G. Agha, "Model-checking markov chains in the presence of uncertainties," in *Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS'06. Berlin, Heidelberg: Springer-Verlag, 2006, p. 394–410.

[11] C. Mark and S. Liu, "Stochastic MPC with distributionally robust chance constraints," *IFAC*, vol. 53, no. 2, pp. 7136–7141, 2020.

[12] A. Lavaei, S. Soudjani, A. Abate, and M. Zamani, "Automated verification and synthesis of stochastic hybrid systems: A survey," 2021.

[13] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen, "Model-checking algorithms for continuous-time Markov chains," *IEEE Transactions on Software Engineering*, vol. 29, no. 6, pp. 524–541, Jun. 2003.

[14] M. Lahijanian, S. B. Andersson, and C. Belta, "Formal Verification and Synthesis for Discrete-Time Stochastic Systems," *IEEE Transactions on Automatic Control*, vol. 60, no. 8, pp. 2031–2045, Aug. 2015.

[15] M. Ahmadi, A. Israel, and U. Topcu, "Safety assessemt based on physically-viable data-driven models," in *56th IEEE CDC*, Dec. 2017, pp. 6409–6414.

[16] J. Jackson, L. Laurenti, E. Frew, and M. Lahijanian, "Strategy Synthesis for Partially-known Switched Stochastic Systems," *Proceedings of the 24th International Conference on Hybrid*

Systems: Computation and Control, pp. 1–11, May 2021, arXiv: 2104.02172. [Online]. Available: http://arxiv.org/abs/2104.02172

[17] C. K. I. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Advances in neural information processing systems 8*. MIT press, 1996, pp. 514–520.

[18] V. I. Paulsen and M. Raghupathi, *An Introduction to the Theory of Reproducing Kernel Hilbert Spaces*, ser. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2016.

[19] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-based safe learning with Gaussian processes," in *53rd IEEE CDC*, Dec. 2014, pp. 1424–1431.

[20] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe Model-based Reinforcement Learning with Stability Guarantees," *arXiv:1705.08551 [cs, stat]*, Nov. 2017, arXiv: 1705.08551. [Online]. Available: http://arxiv.org/abs/1705.08551

[21] A. Blaas, A. Patane, L. Laurenti, L. Cardelli, M. Kwiatkowska, and S. Roberts, "Adversarial Robustness Guarantees for Classification with Gaussian Processes," *arXiv:1905.11876 [cs, stat]*, Mar. 2020, arXiv: 1905.11876. [Online]. Available: http://arxiv.org/abs/1905.11876

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

[23] F. Leibfried, V. Dutordoir, S. John, and N. Durrande, "A tutorial on sparse gaussian processes and variational inference," 2021, arXiv: 2012.13962.

[24] A. A. Julius, A. Halasz, M. S. Sakar, H. Rubin, V. Kumar, and G. J. Pappas, "Stochastic modeling and control of biological systems: The lactose regulation system of *Escherichia Coli*," *IEEE Transactions on Automatic Control*, vol. 53, no. Special Issue, pp. 51–65, 2008.

[25] E. Altman, T. Başar, and R. Srikant, "Congestion control as a stochastic control problem with action delays," *Automatica*, vol. 35, no. 12, pp. 1937–1950, 1999.

[26] A. Shamsah, J. Warnke, Z. Gu, and Y. Zhao, "Integrated Task and Motion Planning for Safe Legged Navigation in Partially Observable Environments," 2021, arXiv: 2110.12097.

[27] Y. Z. Lun, J. Wheatley, A. D'Innocenzo, and A. Abate, "Approximate abstractions of markov chains with interval decision processes," *IFAC-PapersOnLine*, vol. 51, no. 16, pp. 91–96, 2018.

[28] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[29] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Information-theoretic regret bounds for gaussian process optimization in the bandit setting," *IEEE Transactions on Information Theory*, vol. 58, no. 5, p. 3250–3265, May 2012.

[30] C. O. Saglam and K. Byl, "Quantifying the trade-offs between stability versus energy use for underactuated biped walking," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2014, pp. 2550–2557, iSSN: 2153-0866.

[31] K. Sreenath, C. R. Hill, and V. Kumar, "A partially observable hybrid system model for bipedal locomotion for adapting to terrain variations," in *Proceedings of the 16th international conference on Hybrid systems: computation and control*, ser. HSCC '13. New York, NY, USA: Association for Computing Machinery, Apr. 2013, pp. 137–142. [Online]. Available: https://doi.org/10.1145/2461328.2461352

[32] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "An experimental comparison of Bayesian optimization for bipedal locomotion," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 1951–1958, iSSN: 1050-4729.

[33] M. Neumann, K. Kersting, Z. Xu, and D. Schulz, "Stacked gaussian process learning," in *2009 Ninth IEEE International Conference on Data Mining*, 2009, pp. 387–396.