A Learner-Verifier Framework for Neural Network Controllers and Certificates of Stochastic Systems*

Krishnendu Chatterjee[1], Thomas A. Henzinger[1](✉), Mathias Lechner[2], and Đorđe Žikelić[1]

[1] Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria
{krishnendu.chatterjee,tah,djordje.zikelic}@ist.ac.at
[2] Massachusetts Institute of Technology (MIT), Cambridge, MA, USA
mlechner@mit.edu

Abstract. Reinforcement learning has received much attention for learning controllers of deterministic systems. We consider a learner-verifier framework for stochastic control systems and survey recent methods that formally guarantee a conjunction of reachability and safety properties. Given a property and a lower bound on the probability of the property being satisfied, our framework jointly learns a control policy and a formal certificate to ensure the satisfaction of the property with a desired probability threshold. Both the control policy and the formal certificate are continuous functions from states to reals, which are learned as parameterized neural networks. While in the deterministic case, the certificates are invariant and barrier functions for safety, or Lyapunov and ranking functions for liveness, in the stochastic case the certificates are supermartingales. For certificate verification, we use interval arithmetic abstract interpretation to bound the expected values of neural network functions.

Keywords: Learning-based control · Stochastic systems · Martingales. · Formal verification

1 Introduction

Learning-based control and verification of learned controllers. Learning-based control and reinforcement learning (RL) were empirically demonstrated to have enormous potential to solve highly non-linear control tasks. However, their deployment in safety-critical scenarios such as autonomous driving or healthcare requires safety assurances. Most safety-aware RL algorithms optimize expected reward while only empirically trying to maximize safety probability. This together with the non-explainable nature of neural network controllers obtained via deep RL raise questions about the trustworthiness of learning-based methods for safety-critical applications [9,27]. To that end, formal verification of learned

*This work was supported in part by the ERC-2020-AdG 101020093, ERC CoG 863818 (FoRM-SMArt) and the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 665385.

S. Sankaranarayanan and N. Sharygina (Eds.): TACAS 2023, LNCS 13993, pp. 3–25, 2023.
https://doi.org/10.1007/978-3-031-30823-9_1

controllers as well as learning-based control with formal safety guarantees have become very active research topics.

Learning certificate functions. A classical approach to formally proving properties of dynamical systems is to compute a certificate function. A *certificate function* [26] is a function that assigns real values to system states and its defining conditions imply satisfaction of the property. Thus, in order to prove the property of interest, it suffices to compute a certificate function for that property. For instance, Lyapunov functions [46] and barrier functions [50] are standard certificate functions for proving reachability of some target set and avoidance of some unsafe set of system states, respectively, when the system dynamics are deterministic. While both Lyapunov and barrier functions are well-studied concepts in dynamical systems theory, early methods for their computation either required designing the certificates by hand or using computationally intractable numerical procedures. A more recent approach reduces certificate computation to a semi-definite programming problem by using sum-of-squares (SOS) techniques [33,49,37]. However, a limitation of this approach is that it is only applicable to polynomial systems and computation of polynomial certificate functions, whereas it is not applicable to systems with general non-linearities. Moreover, SOS methods do not scale well with the dimension of the system.

Learning-based methods are a promising approach to overcome these limitations and they have received much attention in recent years. These methods jointly learn a neural network control policy and a neural network certificate function, e.g. a Lyapunov function [53,18,3,17] or a barrier function [38,58,52,1], depending on the property of interest. The neural network certificate is then formally verified, ensuring that these methods provide formal guarantees. Both learning and verification procedures developed for verifying neural network certificates are not restricted to polynomial dynamical systems. See [26] for an overview of existing learning-based control methods that learn a certificate function to verify a system property in deterministic dynamical systems.

Prior works – deterministic dynamical systems. While the above works present significant advancements in learning-based control and verification of dynamical systems, they are predominantly restricted to *deterministic* dynamical systems. In other words, they assume that they have access to the exact dynamics function according to which the system evolves. However, for most control tasks, the underlying models used by control methods are imperfect approximations of real systems inferred from observed data. Thus, control and verification methods should also account for model uncertainty due to the noise in observed data and the approximate nature of model inference.

This survey – stochastic dynamical systems. In this work, we survey recent developments in learning-based methods for control and verification of discrete-time *stochastic* dynamical systems, based on [44,68]. Stochastic dynamical systems use probability distributions to quantify and model uncertainty. In stochastic dynamical systems, given a property of interest and a probability parameter $p \in [0, 1]$, the goal is to learn a control policy and a formal certificate which guarantees that the system under the learned policy satisfies the property of interest with probability at least p.

Supermartingale certificate functions. Lyapunov functions and barrier functions can be used to prove properties in deterministic dynamical systems, however they are not applicable to stochastic dynamical systems and do not allow reasoning about the probability of a property being satisfied. Instead, the learning-based methods of [44,68] use *supermartingale certificate functions* to formally prove properties in stochastic systems. Supermartingales are a class of stochastic processes that decrease in expected value at every time step [66]. Their nice convergence properties and concentration bounds allow their use in designing certificate functions for stochastic dynamical systems. In particular, *ranking supermartingales (RSMs)* [15,44] were used to verify probability 1 reachability and *stochastic barrier functions (SBFs)* [50] were used to verify safety with the specified probability $p \in [0, 1]$. *Reach-avoid supermartingales (RASMs)* [68] unify and extend these two concepts and were used to verify reach-avoidance properties with the specified probability $p \in [0, 1]$, i.e. a conjunction of reachability and safety properties. We define and compare these concepts in Section 3.

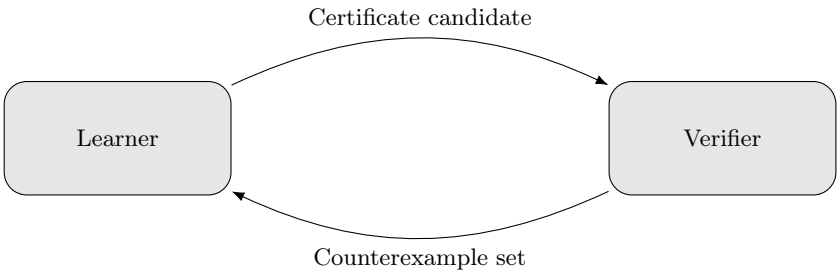

Fig. 1: Schematic illustration of the learner-verifier loop.

Learner-verifier framework for stochastic dynamical systems. In Section 4, we then present a *learner-verifier framework* of [44,68] for learning-based control and for the verification of learned controllers in stochastic dynamical systems in a counterexample guided inductive synthesis (CEGIS) fashion [55]. The algorithm jointly learns a neural network control policy and a neural network supermartingale certificate function. It consists of two modules – the learner, which learns a policy and a supermartingale certificate function candidate, and the verifier, which then formally verifies the candidate supermartingale certificate function. If the verification step fails, the verifier computes counterexamples and passes them back to the learner, which tries to learn a new candidate. This loop is repeated until a candidate is successfully verified, see Fig. 1.

This framework builds on the existing learner-verifier methods for learning-based control in deterministic dynamical systems [18,2,26]. However, the extension of this framework to stochastic dynamical systems and the synthesis of supermartingale certificate functions is far from straightforward. In particular, the methods of [18,2] use knowledge of the deterministic dynamics function to reduce the verification task to a decision procedure and use an off-the-shelf solver. However, verification of the expected decrease condition of supermartin-

gale certificates by reduction to a decision procedure would require being able to compute a closed-form expression of the expected value of a neural network function over a probability distribution and provide it to the solver. It is not clear how the closed-form expression can be computed, and it is not known whether the closed-form expression exists in the general case.

This challenge is solved by using a method for efficient computation of tight *upper and lower bounds on the expected value* of a neural network function. The verifier module then verifies the expected decrease condition by *discretizing* the state space and formally verifying a slightly stricter condition at the discretization points by using the computed expected value bounds. By carefully choosing the mesh of the discretization and adding an additional error term, we obtain a sound verification method applicable to general Lipschitz continuous systems. The expected value bound computation for neural network functions relies on interval arithmetic and abstract interpretation, and since it is of independent interest, we discuss it in detail in Section 5. We are not aware of any existing methods that tackle this problem.

Extension to general stochastic certificates. We conclude this survey with a discussion of possible extensions of the learner-verifier framework in Section 6 and of related work in Section 7.

2 Preliminaries

We consider discrete-time stochastic dynamical systems defined via

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \omega_t),\ \mathbf{x}_0 \in \mathcal{X}_0.$$

The function $f : \mathcal{X} \times \mathcal{U} \times \mathcal{N} \to \mathcal{X}$ is the dynamics function of the system and $t \in \mathbb{N}_0$ is the time index. We use $\mathcal{X} \subseteq \mathbb{R}^m$ to denote the system state space, $\mathcal{U} \subseteq \mathbb{R}^n$ the control action space and $\mathcal{N} \subseteq \mathbb{R}^p$ the stochastic disturbance space. For each $t \in \mathbb{N}_0$, $\mathbf{x}_t \in \mathcal{X}$ the state of the system, $\mathbf{u}_t \in \mathcal{U}$ the action and $\omega_t \in \mathcal{N}$ the stochastic disturbance vector at time t. The set $\mathcal{X}_0 \subseteq \mathcal{X}$ is the set of initial states. In each time step, $\mathbf{u}_t$ is chosen according to a control policy $\pi : \mathcal{X} \to \mathcal{U}$, i.e. $\mathbf{u}_t = \pi(\mathbf{x}_t)$, and ω_t is sampled according to some specified probability distribution d over $\mathbb{R}^p$. The dynamics function f, control policy π and probability distribution d together define a stochastic feedback loop system.

A trajectory of the system is a sequence $(\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t \in \mathbb{N}_0}$ such that, for each $t \in \mathbb{N}_0$, we have $\mathbf{u}_t = \pi(\mathbf{x}_t)$, $\omega_t \in \mathsf{support}(d)$ and $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \omega_t)$. For each initial state $\mathbf{x}_0 \in \mathcal{X}$, the system induces a Markov process. This gives rise to the probability space over the set of all trajectories of the system that start in $\mathbf{x}_0$ [51]. We denote the probability measure and the expectation in this probability space by $\mathbb{P}_{\mathbf{x}_0}$ and $\mathbb{E}_{\mathbf{x}_0}$, respectively.

Assumptions. We assume that $\mathcal{X} \subseteq \mathbb{R}^m$, $\mathcal{X}_0 \subseteq \mathbb{R}^m$, $\mathcal{U} \subseteq \mathbb{R}^n$ and $\mathcal{N} \subseteq \mathbb{R}^p$ are all Borel-measurable. This is necessary for the probability space of the set of all system trajectories starting in some initial state to be mathematically well-defined. We also assume that $\mathcal{X} \subseteq \mathbb{R}^m$ is compact (i.e. closed and bounded) and that the dynamics function f is Lipschitz continuous, which are common assumptions in

control theory. Finally, we assume that the probability distribution d is a product of independent univariate probability distributions, which is necessary for efficient sampling and expected value computation.

2.1 Brief Overview of Martingale Theory

In this subsection, we provide a brief overview of definitions and results from martingale theory that lie at the core of formal reasoning about supermartingale certificate functions. We assume that the reader is familiar with the mathematical definitions of probability space, measurability and random variables, see [66] for the necessary background. The results in this subsection will help in building an intuition on supermartingale certificate functions, but omitting them would not prevent the reader from following the rest of this paper.

Probability space. A probability space is a triple $(\Omega, \mathcal{F}, \mathbb{P})$ where Ω is a state space, $\mathcal{F}$ is a sigma-algebra and $\mathbb{P}$ is a probability measure which is required to satisfy Kolmogorov axioms [66]. A random variable is a function $X : \Omega \to \mathbb{R}$ that is $\mathcal{F}$-measurable. We use $\mathbb{E}[X]$ to denote the *expected value* of X. A *(discrete-time) stochastic process* is a sequence $(X_i)_{i=0}^{\infty}$ of random variables in $(\Omega, \mathcal{F}, \mathbb{P})$.

Conditional expectation. Let X be a random variable in a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Given a sub-$\sigma$-algebra $\mathcal{F}' \subseteq \mathcal{F}$, a *conditional expectation* of X given $\mathcal{F}'$ is an $\mathcal{F}'$-measurable random variable Y such that, for each $A \in \mathcal{F}'$, we have

$$\mathbb{E}[X \cdot \mathbb{I}(A)] = \mathbb{E}[Y \cdot \mathbb{I}(A)].$$

Here, $\mathbb{I}(A) : \Omega \to \{0, 1\}$ is an *indicator function* of A defined via $\mathbb{I}(A)(\omega) = 1$ if $\omega \in A$, and $\mathbb{I}(A)(\omega) = 0$ if $\omega \notin A$. Intuitively, conditional expectation of X given $\mathcal{F}'$ is an $\mathcal{F}'$-measurable random variable that behaves like X whenever its expected value is taken over an event in $\mathcal{F}'$. Conditional expectation of a random variable X given $\mathcal{F}'$ is guaranteed to exist if X is real-valued and non-negative [66]. Moreover, for any two conditional expectations Y and Y' of X given $\mathcal{F}'$, we have that $\mathbb{P}[Y = Y'] = 1$. Therefore, the conditional expectation is almost-surely unique and we may pick one such random variable as a canonical conditional expectation and denote it by $\mathbb{E}[X \mid \mathcal{F}']$.

Supermartingales. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \cdots \subseteq \mathcal{F}$ be an increasing sequence of sub-σ-algebras in $\mathcal{F}$ with respect to inclusion. A non-negative *supermartingale* with respect to $(\mathcal{F}_i)_{i=0}^{\infty}$ is a stochastic process $(X_i)_{i=0}^{\infty}$ such that each X_i is $\mathcal{F}_i$-measurable, and $X_i(\omega) \geq 0$ and $\mathbb{E}[X_{i+1} \mid \mathcal{F}_i](\omega) \leq X_i(\omega)$ hold for each $\omega \in \Omega$ and $i \geq 0$. Intuitively, the second condition says that the expected value of X_{i+1} given the value of X_i has to decrease. This condition is formalized by using conditional expectation.

The following two results that will be key technical ingredients in our design of supermartingale certificate functions. The first theorem shows that nonnegative supermartingales have nice convergence properties and converge almost-surely to some finite value. The second theorem bounds the probability that the value of the supemartingale ever exceeds some threshold, and it will allow us to bound from above the probability of occurrence of some bad event.

Theorem 1 (Supermartingale convergence theorem [66]). *Let* $(X_i)_{i=0}^{\infty}$ *be a nonnegative supermartingale with respect to* $(\mathcal{F}_i)_{i=0}^{\infty}$*. Then, there exists a random variable* X_∞ *in* $(\Omega, \mathcal{F}, \mathbb{P})$ *to which the supermartingale converges to with probability* 1*, i.e.* $\mathbb{P}[\lim_{i\to\infty} X_i = X_\infty] = 1$.

Theorem 2 ([41]). *Let* $(X_i)_{i=0}^{\infty}$ *be a nonnegative supermartingale with respect to* $(\mathcal{F}_i)_{i=0}^{\infty}$*. Then, for every real* $\lambda > 0$*, we have* $\mathbb{P}[\sup_{i\geq 0} X_i \geq \lambda] \leq \mathbb{E}[X_0]/\lambda$.

2.2 Problem Statement

We now formally define the properties and control tasks that we focus on in this work. In what follows, let $\mathcal{X}_t, \mathcal{X}_u \subseteq \mathcal{X}$ be disjoint Borel-measurable sets and $p \in [0, 1]$ be a lower bound on the probability with which the system under the learned controller needs to satisfy the property:

- *Reachability.* Let $\text{Reach}(\mathcal{X}_t) = \{(\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t\in\mathbb{N}_0} \mid \exists t \in \mathbb{N}_0 . \mathbf{x}_t \in \mathcal{X}_t\}$ be the set of all trajectories that reach the target set $\mathcal{X}_t$. The goal is to learn a control policy under which the system reaches $\mathcal{X}_t$ with probability at least p, i.e. $\mathbb{P}_{\mathbf{x}_0}[\text{Reach}(\mathcal{X}_t)] \geq p$ holds for every initial state $\mathbf{x}_0 \in \mathcal{X}_0$.
- *Safety (or avoidance).* Let $\text{Safe}(\mathcal{X}_u) = \{(\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t\in\mathbb{N}_0} \mid \forall t' \leq t . \mathbf{x}_{t'} \notin \mathcal{X}_u\}$ be the set of all trajectories that do not visit the unsafe set $\mathcal{X}_u$. The goal is to learn a control policy under which the system stays away from $\mathcal{X}_u$ with probability at least p, i.e. $\mathbb{P}_{\mathbf{x}_0}[\text{Safe}(\mathcal{X}_u)] \geq p$ holds for every initial state $\mathbf{x}_0 \in \mathcal{X}_0$.
- *Reach-avoidance.* Let $\text{ReachAvoid}(\mathcal{X}_t, \mathcal{X}_u) = \{(\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t\in\mathbb{N}_0} \mid \exists t \in \mathbb{N}_0 . \mathbf{x}_t \in \mathcal{X}_t \wedge (\forall t' \leq t . \mathbf{x}_{t'} \notin \mathcal{X}_u)\}$ be the set of all trajectories that reach $\mathcal{X}_t$ without reaching $\mathcal{X}_u$. The goal is to learn a control policy under which the system reaches $\mathcal{X}_t$ while staying away from $\mathcal{X}_u$ with probability at least p, i.e. $\mathbb{P}_{\mathbf{x}_0}[\text{ReachAvoid}(\mathcal{X}_t, \mathcal{X}_u)] \geq p$ holds for every initial state $\mathbf{x}_0 \in \mathcal{X}_0$.

3 Supermartingale Certificate Functions

We now overview three classes of supermartingale certificate functions that formally prove reachability, safety and reach-avoidance properties. *Supermartingale certificate functions* do not refer to a single class of certificate functions. Rather, we use this term to refer to all certificate functions that exhibit a supermartingale-like behavior and can formally verify properties in stochastic dynamical systems. In what follows, we assume that the control policy π is fixed. In the following section, we will then present a learner-verifier framework for jointly learning a control policy and a supermartingale certificate function.

RSMs for probability 1 *reachability.* We start with *ranking supermartingales (RSMs)*, which can prove probability 1 reachability of some target set $\mathcal{X}_t$. Intuitively, an RSM is a continuous function that maps system states to nonnegative real values and is required to strictly decrease in expectation by some $\epsilon > 0$ in

every time step until the target $\mathcal{X}_t$ is reached. Due to the strict expected decrease as well as the Supermartingale Convergence Theorem (Theorem 1), one can show that the existence of an RSM guarantees that the system under policy π reaches $\mathcal{X}_t$ with probability 1. RSMs can be viewed as a stochastic extension of Lyapunov functions. Note that RSMs can only be used to prove probability 1 reachability, but cannot be used to reason about probabilistic reachability. RSMs were originally used for proving almost-sure termination in probabilistic programs [15] and were used to certify probability 1 reachability in stochastic dynamical systems in [44].

Definition 1 (Ranking supermartingales [44]). *Let $\mathcal{X}_t \subseteq \mathcal{X}$ be a target set. A continuous function $V : \mathcal{X} \to \mathbb{R}$ is a* ranking supermartingale (RSM) *with respect to $\mathcal{X}_t$ if it satisfies:*

1. Nonnegativity condition. $V(\mathbf{x}) \geq 0$ *for each* $\mathbf{x} \in \mathcal{X}$.
2. Expected Decrease condition. *There exists* $\epsilon > 0$ *such that, for each* $\mathbf{x} \in \mathcal{X} \backslash \mathcal{X}_t$, *we have* $V(\mathbf{x}) \geq \mathbb{E}_{\omega \sim d}[V(f(\mathbf{x}, \pi(\mathbf{x}), \omega))] + \epsilon$.

Theorem 3 ([44]). *Suppose that there exists an RSM with respect to $\mathcal{X}_t$. Then, for every $\mathbf{x}_0 \in \mathcal{X}_0$, we have $\mathbb{P}_{\mathbf{x}_0}[\text{Reach}(\mathcal{X}_t)] = 1$.*

SBFs for probabilistic safety. On the other hand, *stochastic barrier functions (SBFs)* can prove probabilistic safety. Given an unsafe set $\mathcal{X}_u$ and probability $p \in [0, 1)$, an SBF is also a continuous function mapping system states to nonnegative real values, which is required to decrease in expectation at each time step. However, unlike RSMs, the expected decrease need not be strict and there is no target set. In addition, its initial value must be at most 1, whereas its value upon reaching an unsafe set must be at least $1/(1-p)$. Thus, for the system under policy π to violate the safety constraint, the value of the SBF needs to increase from at most 1 to at least $1/(1-p)$ even though it is required to decrease in expectation. The probability of this event can be bounded from above and shown to be at most $1-p$ by using Theorem 2. We highlight the assumption that $p < 1$, which is necessary for the safety constraint to be mathematically defined. As the name suggests, SBFs are a stochastic extension of barrier functions.

Definition 2 (Stochastic barrier functions [50]). *Let $\mathcal{X}_u \subseteq \mathcal{X}$ be an unsafe set and $p \in [0, 1)$. A continuous function $V : \mathcal{X} \to \mathbb{R}$ is a* stochastic barrier function (SBF) *with respect to $\mathcal{X}_u$ and p if it satisfies:*

1. Nonnegativity condition. $V(\mathbf{x}) \geq 0$ *for each* $\mathbf{x} \in \mathcal{X}$.
2. Initial condition. $V(\mathbf{x}) \leq 1$ *for each* $\mathbf{x} \in \mathcal{X}_0$.
3. Safety condition. $V(\mathbf{x}) \geq \frac{1}{1-p}$ *for each* $\mathbf{x} \in \mathcal{X}_u$.
4. Expected Decrease condition. *For each* $\mathbf{x} \in \mathcal{X}$, *if* $V(\mathbf{x}) \leq \frac{1}{1-p}$ *then* $V(\mathbf{x}) \geq \mathbb{E}_{\omega \sim d}[V(f(\mathbf{x}, \pi(\mathbf{x}), \omega))]$.

Theorem 4 ([50]). *Suppose that there exists an SBF with respect to $\mathcal{X}_u$ and p. Then, for every $\mathbf{x}_0 \in \mathcal{X}_0$, we have $\mathbb{P}_{\mathbf{x}_0}[\text{Safe}(\mathcal{X}_u)] \geq p$.*

RASMs for probabilistic reach-avoidance. Finally, *reach-avoid supermartingales (RASMs)* unify and extend RSMs and SBFs in the sense that they allow simultaneous reasoning about reachability and safety and proving a conjunction of

these properties, i.e. reach-avoid properties. Let $\mathcal{X}_t$ and $\mathcal{X}_u$ be disjoint target and unsafe sets and let $p \in [0, 1)$. Similarly to SBFs, an RASM is a continuous nonnegative function which is required to be initially at most 1 but needs to attain a value that is at least $1/(1-p)$ for the unsafe region to be reached. On the other hand, similarly to RSMs, it is required to strictly decrease in expectation by $\epsilon > 0$ at every time step until either the target set $\mathcal{X}_t$ or a state in which the value is at least $1/(1-p)$ is reached. Thus, RASMs can be viewed as a stochastic extension of both Lyapunov functions and barrier functions, which combines the strict decrease of Lypaunov functions and the level-set reasoning of barrier functions.

Definition 3 (Reach-avoid supermartingales [68]). *Let $\mathcal{X}_t \subseteq \mathcal{X}$ and $\mathcal{X}_u \subseteq \mathcal{X}$ be a target set and an unsafe set, respectively, and let $p \in [0, 1]$ be a probability threshold. Suppose that either $p < 1$ or that $p = 1$ and $\mathcal{X}_u = \emptyset$. A continuous function $V : \mathcal{X} \to \mathbb{R}$ is a* reach-avoid supermartingale (RASM) *with respect to $\mathcal{X}_t$, $\mathcal{X}_u$ and p if it satisfies:*

1. Nonnegativity condition. $V(\mathbf{x}) \geq 0$ *for each* $\mathbf{x} \in \mathcal{X}$.
2. Initial condition. $V(\mathbf{x}) \leq 1$ *for each* $\mathbf{x} \in \mathcal{X}_0$.
3. Safety condition. $V(\mathbf{x}) \geq \frac{1}{1-p}$ *for each* $\mathbf{x} \in \mathcal{X}_u$.
4. Expected Decrease condition. *There exists* $\epsilon > 0$ *such that, for each* $\mathbf{x} \in \mathcal{X} \backslash \mathcal{X}_t$ *at which* $V(\mathbf{x}) \leq \frac{1}{1-p}$, *we have* $V(\mathbf{x}) \geq \mathbb{E}_{\omega \sim d}[V(f(\mathbf{x}, \pi(\mathbf{x}), \omega))] + \epsilon$.

Theorem 5 ([68]). *Suppose that there exists an RASM with respect to $\mathcal{X}_t$, $\mathcal{X}_u$ and p. Then, for every $\mathbf{x}_0 \in \mathcal{X}_0$, we have $\mathbb{P}_{\mathbf{x}_0}[\text{ReachAvoid}(\mathcal{X}_t, \mathcal{X}_u)] \geq p$.*

Note that RASMs indeed unify and generalize the definitions of RSMs and SBFs. First, by setting $\mathcal{X}_u = \emptyset$ and $p = 1$ (so $1/(1-p) = \infty$), RASMs reduce to RSMs as the Initial condition that can be enforced without loss of generality by rescaling. Second, by setting $\mathcal{X}_t = \emptyset$, RASMs reduce to SBFs. In this case, the Expected Decrease condition is strengthened as it requires strict decrease by $\epsilon > 0$. However, the proof of Theorem 5 which we outline below also implies Theorem 4 and $\epsilon > 0$ is only necessary to reason about the reachability of $\mathcal{X}_t$.

We also note that RASMs strictly extend the applicability of RSMs, since RASMs can be used to prove reachability with any lower bound $p \in [0, 1]$ on probability and not only probability 1 reachability. Indeed, if we set $\mathcal{X}_u = \emptyset$ and $p \in [0, 1]$, in order to prove reachability of $\mathcal{X}_t$ with probability at least p the RASMs require strict expected decrease in expectation by $\epsilon > 0$ until either $\mathcal{X}_t$ is reached or the RASM value exceeds $1/(1-p)$ (with $1/(1-p) = \infty$ if $p = 1$).

In the rest of this section, we outline the proof of Theorem 5 that was presented in [68]. This proof also implies Theorem 3 and Theorem 4. We do this to highlight the connection of RSMs, SBFs and RASMs to the mathematical notion of supermartingale processes. We also do this to illustrate the tools from martingale theory that are used in proving soundness of supermatingale certificate functions, as we envision that they may be useful in designing supermatingale certificate functions for more general classes of properties.

Proof (proof sketch of Theorem 5). Here we outline the main ideas behind the proof, and for the full proof we refer the reader to [68]. Let $\mathbf{x}_0 \in \mathcal{X}_0$. We need to

show that $\mathbb{P}_{\mathbf{x}_0}[\text{ReachAvoid}(\mathcal{X}_t, \mathcal{X}_u)] \geq p$. To do this, we consider the probability space $(\Omega_{\mathbf{x}_0}, \mathcal{F}_{\mathbf{x}_0}, \mathbb{P}_{\mathbf{x}_0})$ of trajectories that start in $\mathbf{x}_0$ and for each time step $t \in \mathbb{N}_0$ define a random variable in this probability space via

$$X_t(\rho) = \begin{cases} V(\mathbf{x}_t), & \text{if } \mathbf{x}_i \notin \mathcal{X}_t \text{ and } V(\mathbf{x}_i) < \frac{1}{1-p} \text{ for each } 0 \leq i \leq t \\ 0, & \text{if } \mathbf{x}_i \in \mathcal{X}_t \text{ for some } 0 \leq i \leq t, V(\mathbf{x}_j) < \frac{1}{1-p} \text{ for each } 0 \leq j \leq i \\ \frac{1}{1-p}, & \text{otherwise} \end{cases}$$

for each trajectory $\rho = (\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t \in \mathbb{N}_0} \in \Omega_{\mathbf{x}_0}$. Hence, $(X_t)_{t=0}^{\infty}$ defines a stochastic process whose value at each time step is equal to the value of V at the current system state unless either the target set $\mathcal{X}_t$ has been reached after which future values of $\mathcal{X}_t$ are set to 0, or a state in which V exceeds $1/(1-p)$ has been reached after which future values of $\mathcal{X}_t$ are set to $1/(1-p)$. It can be shown that $(X_t)_{t=0}^{\infty}$ is a nonnegative supermartingale $(\Omega_{\mathbf{x}_0}, \mathcal{F}_{\mathbf{x}_0}, \mathbb{P}_{\mathbf{x}_0})$. This claim can be proved by using the Nonnegativity and the Expected Decrease condition of RASMs. Here we do not yet need that the expected decrease is strict, i.e. $\epsilon \geq 0$ in the Expected Decrease condition of RASMs is sufficient.

Since $(X_t)_{t=0}^{\infty}$ is a nonnegative supermartingale, substituting $\lambda = 1/(1-p)$ into the inequality in Theorem 2 shows that

$$\mathbb{P}_{\mathbf{x}_0}\Big[\sup_{i \geq 0} X_i \geq \frac{1}{1-p}\Big] \leq (1-p) \cdot \mathbb{E}_{\mathbf{x}_0}[X_0] \leq 1-p.$$

The second inequality follows since $X_0(\rho) = V(\mathbf{x}_0) \leq 1$ for every $\rho \in \Omega_{\mathbf{x}_0}$ by the Initial condition of RASMs. Hence, by the Safety condition of RASMs it follows that the system under policy π reaches the unsafe set $\mathcal{X}_u$ with probability at most $1-p$. Note that here we can already conclude the claim of Theorem 4.

Finally, as $(X_t)_{t=0}^{\infty}$ is a nonnegative supermartingale, by Theorem 1 its value converges with probability 1. One can then prove that this value has to be either 0 or $\geq 1/(1-p)$ by using the fact that the expected decrease in the Expected Decrease condition of RASMs is strict. But we showed above that a state in which V is $\geq 1/(1-p)$ is reached with probability at most $1-p$. Hence, the probability that the system under policy π reaches the target set $\mathcal{X}_t$ without reaching the unsafe set $\mathcal{X}_u$ is at least p, i.e. $\mathbb{P}_{\mathbf{x}_0}[\text{ReachAvoid}(\mathcal{X}_t, \mathcal{X}_u)] \geq p$. □

4 Learner-Verifier Framework for Stochastic Systems

We now present the learner-verifier framework of [44,68] for the learning-based control and verification of learned controllers in stochastic dynamical systems. We focus on the probabilistic reach-avoid problem, assume that we are given a target set $\mathcal{X}_t$, unsafe set $\mathcal{X}_u$ and a probability parameter $p \in [0,1]$, and learn a control policy π and an RASM which certifies that $\mathbb{P}_{\mathbf{x}_0}[\text{ReachAvoid}(\mathcal{X}_t, \mathcal{X}_u)] \geq p$ for all $\mathbf{x}_0 \in \mathcal{X}_0$. The algorithm for learning RSMs and SBFs can be obtained analogously, since we showed that RASMs unify and generalize RSMs and SBFs.

The algorithm behind the learner-verifier framework consists of two modules – the learner, which learns a neural network control policy π_θ and a neural

network supermartingale certificate function V_ν, and the verifier, which then formally verifies the learned candidate function. If the verification step fails, the verifier produces counterexamples that are passed back to the learner to fine-tune its loss function. Here, θ and ν are vectors of neural network parameters. The loop is repeated until either a certificate function is successfully verified, or some specified timeout is reached. By incorporating feedback from the verifier, the learner is able to tune the policy and the certificate function towards ensuring that the resulting policy meets the desired reach-avoid specification.

Applications. As outlined above, the learner-verifier framework can be used for *learning-based control* with formal guarantees that a property of interest is satisfied by jointly learning a control policy and a supermartingale certificate function for the property. On the other hand, it can also be used to *formally verify* a previously learned control policy by fixing policy parameters and only learning a supermartingale certificate function. Finally, if one uses a different method to learn a policy that turns out to violate the desired property, one can use the learner-verifier framework to *fine-tune an unsafe policy* towards repairing it and obtaining a safe policy for which a supermartingale certificate function certifies that the property of interest is satisfied.

4.1 Algorithm Initialization

As mentioned in Section 1, the key challenge for the verifier is to check the Expected Decrease condition of supermartingale certificates. Our algorithm solves this challenge by discretizing the state space and verifying a slightly stricter condition at discretization vertices which we show to imply the Expected Decrease condition over the whole region required by Definition 3. On the other hand, learning two neural networks in parallel while simultaneously optimizing several objectives can be unstable due to inherent dependencies between two networks. Thus, proper initialization of networks is important. We allow all neural network architectures so long as all activation functions are continuous functions. Furthermore, we apply the softplus activation function to the output neuron of V_ν, in order to ensure that the value of V_ν is always nonnegative.

Discretization. A *discretization* $\tilde{\mathcal{X}}$ of $\mathcal{X}$ with mesh $\tau > 0$ is a set of states such that, for every $\mathbf{x} \in \mathcal{X}$, there exists a state $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}$ such that $||\mathbf{x} - \tilde{\mathbf{x}}||_1 < \tau$. The algorithm takes mesh τ as a parameter and computes a finite discretization $\tilde{\mathcal{X}}$ with mesh τ by simply taking a hyper-rectangular grid of the sufficiently small cell size. Since $\mathcal{X}$ is compact, this yields a finite discretization.

Network initialization. The policy network π_θ is initalized by running proximal policy optimization (PPO) [54] on the Markov decision process (MDP) defined by the stochastic dynamical system with a reward function $r_t = 1[\mathcal{X}_t](\mathbf{x}_t) - [\mathcal{X}_u](\mathbf{x}_t)$.

The discretization $\tilde{\mathcal{X}}$ is used to define three sets of states which are then used by the learner to initialize the certificate network V_ν and to which counterexamples computed by the verifier will be added later. In particular, the algorithm initializes $C_{\text{init}} = \tilde{\mathcal{X}} \cap \mathcal{X}_0$, $C_{\text{unsafe}} = \tilde{\mathcal{X}} \cap \mathcal{X}_u$ and $C_{\text{decrease}} = \tilde{\mathcal{X}} \cap (\mathcal{X} \backslash \mathcal{X}_t)$.

4.2 The Learner module

The Learner updates the parameters θ of the policy and ν of the neural network certificate function candidate V_ν with the objective of the candidate satisfying the supermartingle certificate conditions. The parameter updates happen incrementally via gradient descent of the form $\theta \leftarrow \theta - \alpha \frac{\partial \mathcal{L}(\theta,\nu)}{\partial \theta}$ and $\nu \leftarrow \nu - \alpha \frac{\partial \mathcal{L}(\theta,\nu)}{\partial \nu}$, where $\alpha > 0$ is the learning rate and $\mathcal{L}$ is a loss function that corresponds to a differentiable optimization objective of the supermartingle certificate conditions. Ideally, the global minimum of $\mathcal{L}$ should correspond to a policy π and a neural network V_ν that fulfills all certificate conditions. In practice, however, due to the non-convexity of the network V_ν, gradient descent is not guaranteed to converge to the global minimum. As a result, the learner is not monotone, i.e. a new iteration does not guarantee improvement over the previous iteration. The training process usually applies a fixed number of gradient descent iterations or, alternatively, continues until a certain threshold on the loss value is achieved.

Loss functions. The particular type of loss function $\mathcal{L}$ depends on the type of supermartingale certificate function that should be learned by the network, but is of the general form

$$\mathcal{L}(\theta,\nu) = \mathcal{L}_{\text{Certificate}}(\theta,\nu) + \lambda \cdot \big(\mathcal{L}_{\text{Lipschitz}}(\theta) + \mathcal{L}_{\text{Lipschitz}}(\nu)\big), \tag{1}$$

where $\mathcal{L}_{\text{Certificate}}$ is the specification-specific loss. The auxiliary loss terms $\mathcal{L}_{\text{Lipschitz}}$ regularize the training to obtain networks π_θ and V_ν that have a low upper bound of their Lipschitz constant. The purpose of this regularization is that networks with low Lipschitz upper bound are easier to check by the verifier module, i.e. requiring a coarser discretization grid. The value of $\lambda > 0$ decides the strength of the regularization that is applied. The regularization loss is based on the upper bound derived in [57] and defined as

$$\mathcal{L}_{\text{Lipschitz}}(\theta) = \max\Big\{L_{V_\theta} - \frac{\delta}{\tau \cdot (L_f \cdot (L_\pi + 1) + 1)}, 0\Big\}. \tag{2}$$

In the case of a reach-avoid specification, the RASM certificate loss is

$$\mathcal{L}_{\text{Certificate}}(\theta,\nu) = \mathcal{L}_{\text{Expected}}(\theta,\nu) + \mathcal{L}_{\text{Unsafe}}(\nu) + \mathcal{L}_{\text{Init}}(\nu), \tag{3}$$

with

$$\begin{aligned}
\mathcal{L}_{\text{Expected}}(\theta,\nu) = \frac{1}{|C_{\text{decrease}}|} \cdot \sum_{\mathbf{x}\in C_{\text{expected}}} \Big(\max\Big\{ \\
\sum_{\omega_1,\ldots,\omega_N \sim \mathcal{N}} \frac{V_\nu\big(f(\mathbf{x},\pi_\theta(\mathbf{x}),\omega_i)\big)}{N} - V_\theta(\mathbf{x}) + \tau \cdot K, 0\Big\}\Big) \\
\mathcal{L}_{\text{Init}}(\nu) = \max_{\mathbf{x}\in C_{\text{init}}} \{V_\nu(\mathbf{x}) - 1, 0\} \\
\mathcal{L}_{\text{Unsafe}}(\nu) = \max_{\mathbf{x}\in C_{\text{unsafe}}} \{\frac{1}{1-p} - V_\nu(\mathbf{x}), 0\}.
\end{aligned}$$

The sets C_{expected}, C_{init} and C_{unsafe} are the training sets for achieving the expected decrease, initial and unsafe RASM conditions. Each of the three sets is

initialized with a coarse discretization of the state space to guide the learning toward learning a correct RASM already in the first loop iteration. In the subsequent calls to the learner, these sets are extended by counterexamples computed by the verifier. In [68] it was shown that, if V_θ is a RASM and satisfies all conditions checked by the verifier below, then $\mathcal{L}_{\text{Certificate}}(\theta, \nu) \to 0$ as the number of samples N used to estimate expected values in $\mathcal{L}_{\text{Expected}}(\theta, \nu)$ increases.

4.3 The Verifier module

Verification task. The verifier now formally checks whether the learned RASM candidate V_ν satisfies the four RASM defining conditions in Definition 3. Since we applied the softplus activation function to the output neuron of V_ν, we know that the Nonnegativity condition is satisfied by default. Thus, the verifier only needs to check the Initial, Safety and Expected Decrease conditions in Definition 3.

Expected Decrease condition. To check the Expected Decrease condition, we utilize the fact that the dynamics function f is Lipschitz continuous and that the state space $\mathcal{X}$ is compact to show that it suffices to check a slightly stricter condition at the discretization points. Let L_f be a Lipschitz constant of f. Since π_θ and V_ν are continuous functions defined over the compact domain $\mathcal{X}$, we know that they are also Lipschitz continuous. Let L_π and L_V be their Lipschitz constants. We assume that L_f is provided to the algorithm, and use the method of [57] for computing neural network Lipschitz constants to compute L_π and L_V.

To verify the Expected Decrease condition, the verifier collects a subset $\tilde{\mathcal{X}}_e \subseteq \tilde{\mathcal{X}}$ of all discretization vertices whose adjacent grid cells contain a non-target state and over which V_ν attains a value that is smaller than $\frac{1}{1-p}$. To compute this set, the algorithm first collects all grid cells that intersect $\mathcal{X} \backslash \mathcal{X}_t$. For each collected cell, it then uses interval arithmetic abstract interpretation (IA-AI) [24,30] to propagate interval bounds across neural network layers towards bounding from below the minimal value that V_ν attains over the cell. Finally, it adds to $\tilde{\mathcal{X}}_e$ vertices of those cells at which the computed lower bound is less than $1/(1-p)$.

Finally, the verifier checks if the following condition is satisfied at each $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}_e$

$$\mathbb{E}_{\omega \sim d}\Big[V_\nu\Big(f(\tilde{\mathbf{x}}, \pi_\theta(\tilde{\mathbf{x}}), \omega)\Big)\Big] < V_\nu(\tilde{\mathbf{x}}) - \tau \cdot K, \tag{4}$$

where $K = L_V \cdot (L_f \cdot (L_\pi + 1) + 1)$. Note that this condition is a strengthened version of the Expected Decrease condition, where instead of strict decrease by arbitrary $\epsilon > 0$ we require strict decrease by at least $\tau \cdot K$ which depends on the discretization mesh τ and Lipschitz constants of f, π_θ and V_ν. To compute $\mathbb{E}_{\omega \sim d}[V_\nu(f(\tilde{\mathbf{x}}, \pi_\theta(\tilde{\mathbf{x}}), \omega))]$ in eq. (4), we cannot simply evaluate the expected value in state $\tilde{\mathbf{x}}$ by substituting $\tilde{\mathbf{x}}$ into some expression, as we do not know a closed-form expression for the expected value of a neural network function. Instead, the algorithm uses the method of [44] to compute upper and lower bounds on the expected value of a neural network function, which we describe in Section 5. This upper bound is then plugged it into eq. (4).

If no violations to eq. (4) are found, the verifier concludes that the Expected Decrease condition is satisfied. Otherwise, for any counterexample $\tilde{\mathbf{x}}$ to eq. (4),

the algorithm checks if $\tilde{\mathbf{x}} \in \mathcal{X} \backslash \mathcal{X}_t$ and $V_\nu(\mathbf{x}) < 1/(1-p)$ and if so adds it to the counterexample set C_{decrease}.

Initial and safety conditions. The Initial and Safety conditions are checked using IA-AI. To check the Initial condition, the verifier collects the set $\text{Cells}_{\mathcal{X}_0}$ of all grid cells that intersect the initial set $\mathcal{X}_0$, and for each cell in $\text{Cells}_{\mathcal{X}_0}$ checks if

$$\sup_{\mathbf{x} \in \text{cell}} V_\nu(\mathbf{x}) > 1. \tag{5}$$

The supremum is bounded from above via IA-AI by propagating interval bounds across neural network layers. If no violations are found, the verifier concludes that V_ν satisfies the Initial condition. Otherwise, vertices of any grid cells which are counterexamples to eq. (5) and which are contained in $\mathcal{X}_0$ are added to C_{init}. Analogously, to check the Safety condition, the verifier collects the set $\text{Cells}_{\mathcal{X}_u}$ of all grid cells that intersect the unsafe set $\mathcal{X}_u$, and for each cell checks if

$$\inf_{\mathbf{x} \in \text{cell}} V_\nu(\mathbf{x}) < \frac{1}{1-p}. \tag{6}$$

If no violations are found, the verifier concludes that V_ν satisfies the Safety condition. Otherwise, vertices of any grid cells which are counterexamples to eq. (6) and which are contained in $\mathcal{X}_u$ are added to C_{unsafe}.

Algorithm output and correctness. If all three checks are successful and no counterexample is found, the algorithm concludes that π_θ guarantees reach-avoidance with probability at least p and outputs the policy p_θ. Otherwise, it proceeds to the next learner-verifier iteration where computed counterexamples are added to sets C_{init}, C_{unsafe} and C_{decrease} to be used by the learner. The following theorem establishes correctness of the verifier module, and its proof can be found in [68].

Theorem 6 ([68]). *Suppose that the verifier verifies that the certificate V_ν satisfies eq. (4) for each $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}_e$, eq. (5) for each cell $\in \text{Cells}_{\mathcal{X}_0}$ and eq. (6) for each cell $\in \text{Cells}_{\mathcal{X}_u}$. Then the function V_ν is an RASM for the system with respect to $\mathcal{X}_t$, $\mathcal{X}_u$ and p.*

Optimizations. The verification task can be made more efficient by a discretization refinement procedure. In particular, the verifier may start with a coarse grid and decomposes each grid cell on demand into a finer discretization in case the check when some RASM condition fails. This procedure can be used recursively to refine further in the case when elements of the decomposed grid cannot be verified. In case the recursion encounters a grid element that violates Eq. 4 even for $\tau = 0$, the refinement procedure terminates unsuccessfully with the grid center point as a counterexample of the RASM condition. This optimization with a maximum recursion depth of 1 has been applied in [68].

5 Bounding Expected Values of Neural Networks

We now present the method for computing upper and lower bounds on the expected value of a neural network function over a given probability distribution.

We are not aware of any existing methods for solving this problem, so believe that this is a result of independent interest.

To define the setting of the problem at hand, let $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ be a system state and suppose that we want to compute upper and lower bounds the expected value $\mathbb{E}_{\omega \sim d}[V(f(\mathbf{x}, \pi(\mathbf{x}), \omega))]$. Here d is a probability distribution over the stochastic disturbance space $\mathcal{N} \subseteq \mathbb{R}^p$ from which the stochastic disturbance is sampled independently at each time step. As noted in Section 2, we assume that d is a product of independent univariate probability distributions. Alternatively, the method is also applicable if the support of d is bounded.

The method first partitions the stochastic disturbance space $\mathcal{N} \subseteq \mathbb{R}^p$ into finitely many cells $\text{cell}(\mathcal{N}) = \{\mathcal{N}_1, \ldots, \mathcal{N}_k\}$. Let $\text{maxvol} = \max_{\mathcal{N}_i \in \text{cell}(\mathcal{N})} \mathsf{vol}(\mathcal{N}_i)$ and $\text{minvol} = \min_{\mathcal{N}_i \in \text{cell}(\mathcal{N})} \mathsf{vol}(\mathcal{N}_i)$ denote the maximal and the minimal volume of any cell in the partition with respect to the Lebesgue measure over $\mathbb{R}^p$, respectively. Also, for each $\omega \in \mathcal{N}$ let $F(\omega) = V(f(\mathbf{x}, \pi(\mathbf{x}), \omega))$. The upper and the lower boundd on the expected value are computed as follows

$$\mathbb{E}_{\omega \sim d}\Big[V\Big(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\Big)\Big] \leq \sum_{\mathcal{N}_i \in \text{cell}(\mathcal{N})} \text{maxvol} \cdot \sup_{\omega \in \mathcal{N}_i} F(\omega),$$

$$\mathbb{E}_{\omega \sim d}\Big[V\Big(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\Big)\Big] \geq \sum_{\mathcal{N}_i \in \text{cell}(\mathcal{N})} \text{minvol} \cdot \inf_{\omega \in \mathcal{N}_i} F(\omega).$$

Each supremum (resp. infimum) in the sum is then bounded from above (resp. from below) via interval arithmetic abstract interpretation by using the method of [30].

If the support of d is bounded, then no further adjustments are needed. However, if the support of d is unbounded, maxvol and minvol may not be finite. In this case, since we assume that d is a product of univariate distributions, the method first applies the probability integral transform [48] to each univariate probability distribution in d in order to reduce the problem to the case of a probability distribution of bounded support.

6 Discussion on Extension to General Certificates

The focus of this survey has primarily been on three concrete classes of supermartingale certificate functions in stochastic systems, namely RSMs, SBFs and RASMs, and the learner-verifier framework for their computation. For each class of supemartingale certificate functions, the learner module encodes the defining conditions of the certificate as a differentiable loss function whose minimization leads to a candidate certificate function. The verifier module then formally checks whether the defining conditions of the certificate function are satisfied. These checks are performed by discretizing the state space and using interval arithmetic abstract interpretation and the previously discussed method for computing bounds on expected values of neural network functions.

It should be noted that the design of both the learner and the verifier modules was not specifically tailored to any of the three certificate functions. Rather, both the learner and the verifier follow very general design principles that we envision

are applicable to more general classes of certificate functions. In particular, we hypothesize that as long as the state space of the system is compact and a certificate function can be defined in terms of

- exact and expected value evaluations of Lipschitz continuous functions, and
- inequalities between such evaluations imposed over state space regions,

then the learner-verifier framework in Section 4 may present a promising approach to learning and verifying the certificate function. In particular, the learner-verifier framework presents a natural candidate for automating the computation of *any supermartingale certificate function* that may be designed for other properties in the future. Furthermore, while RSMs, SBFs and RASMs exhibit a supermartingale-like behavior which is fundamental for their soundness, the learner-verifier framework does not rely or depend on their supermartingale-like behavior. Hence, we envision that the learner-verifier framework could also be used to compute other classes of *stochastic certificate functions*.

Even more generally, note that all certificate functions that we have considered so far are of the type $\mathcal{X} \to \mathbb{R}$. One could also consider extensions of the learner-verifier framework to learning certificate functions of different datatypes. For instance, the work [43] uses a learner-verifier framework to learn an inductive transition invariant of type $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that certifies safety in deterministic systems. On the other hand, lexicographic ranking supermartingales are a multi-dimensional generalization of RSMs of type $\mathcal{X} \to \mathbb{R}^k$ that provide a more efficient and compositional approach to proving probability 1 termination in probabilistic programs [5,22]. Studying possible extensions of the learner-verifier framework for stochastic systems to learn certificate functions of different arity of both domain and codomain is a very interesting direction of future work.

7 Related Work

Existing learning-based methods for learning and verification of certificate functions in deterministic and stochastic systems have been discussed in Section 1. In this section, we overview some other existing methods for verification and control of stochastic dynamical systems, as well as some other uses of martingale theory in stochastic system verification.

Abstraction-based methods. Another class of approaches to stochastic dynamical system control with formal safety guarantees are abstraction based methods [56,42,14,63,60,25]. These methods consider finite-time horizon systems and approximate them via a finite-state Markov decision process (MDP). The control problem is then solved for the obtained MDP and the computed policy is used to exhibit a policy for the original stochastic dynamical system. The key difference in applicability between abstraction based methods and our framework is that abstraction based methods consider *finite-time horizon* systems, whereas we consider *infinite-time horizon* systems.

Safe control via shielding. Shielding is an RL framework that ensures safety in the context of avoidance of unsafe regions by computing two control policies – the main policy that optimizes the expected reward, and the backup policy that the system falls back to whenever the safety constraint may be violated [7,36,29].

Constrained MDPs. A standard approach to safe RL is to solve constrained MDPs (CMDPs) [8,28] which impose hard constraints on expected cost for one or more auxiliary cost functions. Several efficient RL algorithms for solving CMDPs have been proposed [59,4], however their constraints are only satisfied in expectation, hence constraint satisfaction is not formally guaranteed.

RL reward specification and neurosymbolic methods. There are several works on solving model-free RL tasks under logic specifications. In particular, several works propose methods for designing reward functions that encode temporal logic specifications [6,12,32,31,45,34,13,40,39]. Formal methods have also been used for extraction of interpretable policies [62,61,35] and safe RL [10,67,11].

Deterministic systems with stochastic controllers. Another way to give rise to a stochastic dynamical system is to consider a dynamical system with deterministic dynamics function and use a stochastic controller, which helps in quantifying uncertainty in the controller's prediction. Formal verification of deterministic dynamical systems with Bayesian neural network controllers has been considered in [43]. In particular, this work also uses a learner-verifier method to learn an inductive invariant for the deterministic system which formally proves safety.

Supermartingales for probabilistic program analysis. Supermartingales have also been used for the analysis of probabilistic programs (PPs). In particular, RSMs were originally introduced in the setting of PPs to prove almost-sure termination [15] and have since been extensively used, see e.g. [19,20,5,47,22]. The work [1] proposed a learner-verifier method to learn an RSM in the PP. Supermartingales were also used for safety [23,64,21], cost [65] and recurrence and persistence [16] analysis in PPs.

8 Conclusion

This paper presents a framework for learning-based control with formal reachability, safety and reach-avoidance guarantees in stochastic dynamical systems. We present a learner-verifier framework in which a neural network control policy is learned together with a neural network certificate function that formally proves that the property of interest holds with at least some desired probability $p \in [0, 1]$. For certification, we use supermartingale certificate functions. The learner module encodes the defining certificate function conditions into a differentiable loss function which is then minimized to learn a candidate certificate function. The verifier then formally verifies the candidate by using interval arithmetic abstract interpretation and a novel method for computing bounds on expected values of neural networks.

The learner-verifier framework presented in this work opens several interesting directions for future work. The first is the design of supermartingale certificates for more general properties of stochastic systems and the use of our learner-verifier framework for their computation. The second is to study and understand the general class of certificate functions in stochastic systems that the learner-verifier can be used to compute, possibly going beyond supermartingale certificate functions. Finally, on the practical side, a venue for future work is to explore methods for reducing the computational cost of the framework and extensions that can handle more complex and higher dimensional systems.

References

1. Abate, A., Ahmed, D., Edwards, A., Giacobbe, M., Peruffo, A.: FOSSIL: a software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks. In: Bogomolov, S., Jungers, R.M. (eds.) HSCC '21: 24th ACM International Conference on Hybrid Systems: Computation and Control, Nashville, Tennessee, May 19-21, 2021. pp. 24:1–24:11. ACM (2021). https://doi.org/10.1145/3447928.3456646, https://doi.org/10.1145/3447928.3456646
2. Abate, A., Ahmed, D., Giacobbe, M., Peruffo, A.: Formal synthesis of lyapunov neural networks. IEEE Control. Syst. Lett. **5**(3), 773–778 (2021). https://doi.org/10.1109/LCSYS.2020.3005328, https://doi.org/10.1109/LCSYS.2020.3005328
3. Abate, A., Giacobbe, M., Roy, D.: Learning probabilistic termination proofs. In: Silva, A., Leino, K.R.M. (eds.) Computer Aided Verification - 33rd International Conference, CAV 2021, Virtual Event, July 20-23, 2021, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12760, pp. 3–26. Springer (2021). https://doi.org/10.1007/978-3-030-81688-9_1, https://doi.org/10.1007/978-3-030-81688-9_1
4. Achiam, J., Held, D., Tamar, A., Abbeel, P.: Constrained policy optimization. In: International Conference on Machine Learning. pp. 22–31. PMLR (2017)
5. Agrawal, S., Chatterjee, K., Novotný, P.: Lexicographic ranking supermartingales: an efficient approach to termination of probabilistic programs. Proc. ACM Program. Lang. **2**(POPL), 34:1–34:32 (2018). https://doi.org/10.1145/3158122, https://doi.org/10.1145/3158122
6. Aksaray, D., Jones, A., Kong, Z., Schwager, M., Belta, C.: Q-learning for robust satisfaction of signal temporal logic specifications. In: 55th IEEE Conference on Decision and Control, CDC 2016, Las Vegas, NV, USA, December 12-14, 2016. pp. 6565–6570. IEEE (2016). https://doi.org/10.1109/CDC.2016.7799279, https://doi.org/10.1109/CDC.2016.7799279
7. Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: McIlraith, S.A., Weinberger, K.Q. (eds.) Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018. pp. 2669–2678. AAAI Press (2018), https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17211
8. Altman, E.: Constrained Markov decision processes, vol. 7. CRC Press (1999)
9. Amodei, D., Olah, C., Steinhardt, J., Christiano, P.F., Schulman, J., Mané, D.: Concrete problems in AI safety. CoRR **abs/1606.06565** (2016), http://arxiv.org/abs/1606.06565
10. Anderson, G., Verma, A., Dillig, I., Chaudhuri, S.: Neurosymbolic reinforcement learning with formally verified exploration. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual (2020), https://proceedings.neurips.cc/paper/2020/hash/448d5eda79895153938a8431919f4c9f-Abstract.html
11. Bacci, E., Giacobbe, M., Parker, D.: Verifying reinforcement learning up to infinity. In: Zhou, Z. (ed.) Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021. pp. 2154–2160. ijcai.org (2021). https://doi.org/10.24963/ijcai.2021/297, https://doi.org/10.24963/ijcai.2021/297

12. Brafman, R.I., Giacomo, G.D., Patrizi, F.: Ltlf/ldlf non-markovian rewards. In: McIlraith, S.A., Weinberger, K.Q. (eds.) Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018. pp. 1771–1778. AAAI Press (2018), https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17342
13. Camacho, A., Icarte, R.T., Klassen, T.Q., Valenzano, R.A., McIlraith, S.A.: LTL and beyond: Formal languages for reward function specification in reinforcement learning. In: Kraus, S. (ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019. pp. 6065–6073. ijcai.org (2019). https://doi.org/10.24963/ijcai.2019/840, https://doi.org/10.24963/ijcai.2019/840
14. Cauchi, N., Abate, A.: Stochy-automated verification and synthesis of stochastic processes. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control. pp. 258–259 (2019)
15. Chakarov, A., Sankaranarayanan, S.: Probabilistic program analysis with martingales. In: Sharygina, N., Veith, H. (eds.) Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8044, pp. 511–526. Springer (2013). https://doi.org/10.1007/978-3-642-39799-8_34, https://doi.org/10.1007/978-3-642-39799-8_34
16. Chakarov, A., Voronin, Y., Sankaranarayanan, S.: Deductive proofs of almost sure persistence and recurrence properties. In: Chechik, M., Raskin, J. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9636, pp. 260–279. Springer (2016). https://doi.org/10.1007/978-3-662-49674-9_15, https://doi.org/10.1007/978-3-662-49674-9_15
17. Chang, Y., Gao, S.: Stabilizing neural control using self-learned almost lyapunov critics. In: IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021. pp. 1803–1809. IEEE (2021). https://doi.org/10.1109/ICRA48506.2021.9560886, https://doi.org/10.1109/ICRA48506.2021.9560886
18. Chang, Y., Roohi, N., Gao, S.: Neural lyapunov control. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada. pp. 3240–3249 (2019), https://proceedings.neurips.cc/paper/2019/hash/2647c1dba23bc0e0f9cdf75339e120d2-Abstract.html
19. Chatterjee, K., Fu, H., Goharshady, A.K.: Termination analysis of probabilistic programs through positivstellensatz's. In: Chaudhuri, S., Farzan, A. (eds.) Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9779, pp. 3–22. Springer (2016). https://doi.org/10.1007/978-3-319-41528-4_1, https://doi.org/10.1007/978-3-319-41528-4_1
20. Chatterjee, K., Fu, H., Novotný, P., Hasheminezhad, R.: Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. In: Bodík, R., Majumdar, R. (eds.) Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016. pp. 327–

342. ACM (2016). https://doi.org/10.1145/2837614.2837639, https://doi.org/10.1145/2837614.2837639
21. Chatterjee, K., Goharshady, A.K., Meggendorfer, T., Zikelic, D.: Sound and complete certificates for quantitative termination analysis of probabilistic programs. In: Shoham, S., Vizel, Y. (eds.) Computer Aided Verification - 34th International Conference, CAV 2022, Haifa, Israel, August 7-10, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13371, pp. 55–78. Springer (2022). https://doi.org/10.1007/978-3-031-13185-1_4, https://doi.org/10.1007/978-3-031-13185-1_4
22. Chatterjee, K., Goharshady, E.K., Novotný, P., Zárevúcky, J., Zikelic, D.: On lexicographic proof rules for probabilistic termination. In: Huisman, M., Pasareanu, C.S., Zhan, N. (eds.) Formal Methods - 24th International Symposium, FM 2021, Virtual Event, November 20-26, 2021, Proceedings. Lecture Notes in Computer Science, vol. 13047, pp. 619–639. Springer (2021). https://doi.org/10.1007/978-3-030-90870-6_33, https://doi.org/10.1007/978-3-030-90870-6_33
23. Chatterjee, K., Novotný, P., Zikelic, D.: Stochastic invariants for probabilistic termination. In: Castagna, G., Gordon, A.D. (eds.) Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017. pp. 145–160. ACM (2017). https://doi.org/10.1145/3009837.3009873, https://doi.org/10.1145/3009837.3009873
24. Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Graham, R.M., Harrison, M.A., Sethi, R. (eds.) Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977. pp. 238–252. ACM (1977). https://doi.org/10.1145/512950.512973, https://doi.org/10.1145/512950.512973
25. Crespo, L.G., Sun, J.: Stochastic optimal control via bellman's principle. Autom. **39**(12), 2109–2114 (2003). https://doi.org/10.1016/S0005-1098(03)00238-3, https://doi.org/10.1016/S0005-1098(03)00238-3
26. Dawson, C., Gao, S., Fan, C.: Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods. CoRR **abs/2202.11762** (2022), https://arxiv.org/abs/2202.11762
27. García, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. J. Mach. Learn. Res. **16**, 1437–1480 (2015), http://dl.acm.org/citation.cfm?id=2886795
28. Geibel, P.: Reinforcement learning for mdps with constraints. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4212, pp. 646–653. Springer (2006). https://doi.org/10.1007/11871842_63, https://doi.org/10.1007/11871842_63
29. Giacobbe, M., Hasanbeig, M., Kroening, D., Wijk, H.: Shielding atari games with bounded prescience. In: Dignum, F., Lomuscio, A., Endriss, U., Nowé, A. (eds.) AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021. pp. 1507–1509. ACM (2021). https://doi.org/10.5555/3463952.3464141, https://www.ifaamas.org/Proceedings/aamas2021/pdfs/p1507.pdf
30. Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T.A., Kohli, P.: On the effectiveness of interval bound propagation for training verifiably robust models. CoRR **abs/1810.12715** (2018), http://arxiv.org/abs/1810.12715
31. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Omega-regular objectives in model-free reinforcement learning. In: Vojnar, T., Zhang,

L. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11427, pp. 395–412. Springer (2019). https://doi.org/10.1007/978-3-030-17462-0_27, https://doi.org/10.1007/978-3-030-17462-0_27
32. Hasanbeig, M., Kantaros, Y., Abate, A., Kroening, D., Pappas, G.J., Lee, I.: Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees. In: 58th IEEE Conference on Decision and Control, CDC 2019, Nice, France, December 11-13, 2019. pp. 5338–5343. IEEE (2019). https://doi.org/10.1109/CDC40024.2019.9028919, https://doi.org/10.1109/CDC40024.2019.9028919
33. Henrion, D., Garulli, A.: Positive polynomials in control, vol. 312. Springer Science & Business Media (2005)
34. Icarte, R.T., Klassen, T.Q., Valenzano, R.A., McIlraith, S.A.: Using reward machines for high-level task specification and decomposition in reinforcement learning. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 2112–2121. PMLR (2018), http://proceedings.mlr.press/v80/icarte18a.html
35. Inala, J.P., Bastani, O., Tavares, Z., Solar-Lezama, A.: Synthesizing programmatic policies that inductively generalize. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net (2020), https://openreview.net/forum?id=S1l8oANFDH
36. Jansen, N., Könighofer, B., Junges, S., Serban, A., Bloem, R.: Safe reinforcement learning using probabilistic shields (invited paper). In: Konnov, I., Kovács, L. (eds.) 31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference). LIPIcs, vol. 171, pp. 3:1–3:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.CONCUR.2020.3, https://doi.org/10.4230/LIPIcs.CONCUR.2020.3
37. Jarvis-Wloszek, Z., Feeley, R., Tan, W., Sun, K., Packard, A.: Some controls applications of sum of squares programming. In: 42nd IEEE international conference on decision and control (IEEE Cat. No. 03CH37475). vol. 5, pp. 4676–4681. IEEE (2003)
38. Jin, W., Wang, Z., Yang, Z., Mou, S.: Neural certificates for safe control policies. CoRR **abs/2006.08465** (2020), https://arxiv.org/abs/2006.08465
39. Jothimurugan, K., Alur, R., Bastani, O.: A composable specification language for reinforcement learning tasks. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada. pp. 13021–13030 (2019), https://proceedings.neurips.cc/paper/2019/hash/f5aa4bd09c07d8b2f65bad6c7cd3358f-Abstract.html
40. Jothimurugan, K., Bansal, S., Bastani, O., Alur, R.: Compositional reinforcement learning from logical specifications. In: Ranzato, M., Beygelzimer, A., Dauphin, Y.N., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual. pp. 10026–10039 (2021), https://proceedings.neurips.cc/paper/2021/hash/531db99cb00833bcd414459069dc7387-Abstract.html
41. Kushner, H.J.: A partial history of the early development of continuous-time nonlinear stochastic systems theory. Autom. **50**(2), 303–334 (2014). https:

//doi.org/10.1016/j.automatica.2013.10.013, https://doi.org/10.1016/j.automatica.2013.10.013
42. Lavaei, A., Khaled, M., Soudjani, S., Zamani, M.: AMYTISS: parallelized automated controller synthesis for large-scale stochastic systems. In: Lahiri, S.K., Wang, C. (eds.) Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12225, pp. 461–474. Springer (2020). https://doi.org/10.1007/978-3-030-53291-8_24, https://doi.org/10.1007/978-3-030-53291-8_24
43. Lechner, M., Zikelic, D., Chatterjee, K., Henzinger, T.A.: Infinite time horizon safety of bayesian neural networks. In: Ranzato, M., Beygelzimer, A., Dauphin, Y.N., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual. pp. 10171–10185 (2021), https://proceedings.neurips.cc/paper/2021/hash/544defa9fddff50c53b71c43e0da72be-Abstract.html
44. Lechner, M., Zikelic, D., Chatterjee, K., Henzinger, T.A.: Stability verification in stochastic control systems via neural network supermartingales. In: Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022. pp. 7326–7336. AAAI Press (2022), https://ojs.aaai.org/index.php/AAAI/article/view/20695
45. Li, X., Vasile, C.I., Belta, C.: Reinforcement learning with temporal logic rewards. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017. pp. 3834–3839. IEEE (2017). https://doi.org/10.1109/IROS.2017.8206234, https://doi.org/10.1109/IROS.2017.8206234
46. Lyapunov, A.M.: The general problem of the stability of motion. International journal of control **55**(3), 531–534 (1992)
47. McIver, A., Morgan, C., Kaminski, B.L., Katoen, J.: A new proof rule for almost-sure termination. Proc. ACM Program. Lang. **2**(POPL), 33:1–33:28 (2018). https://doi.org/10.1145/3158121, https://doi.org/10.1145/3158121
48. Murphy, K.P.: Machine learning - a probabilistic perspective. Adaptive computation and machine learning series, MIT Press (2012)
49. Parrilo, P.A.: Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization. California Institute of Technology (2000)
50. Prajna, S., Jadbabaie, A., Pappas, G.J.: A framework for worst-case and stochastic safety verification using barrier certificates. IEEE Trans. Autom. Control. **52**(8), 1415–1428 (2007). https://doi.org/10.1109/TAC.2007.902736, https://doi.org/10.1109/TAC.2007.902736
51. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley Series in Probability and Statistics, Wiley (1994). https://doi.org/10.1002/9780470316887, https://doi.org/10.1002/9780470316887
52. Qin, Z., Zhang, K., Chen, Y., Chen, J., Fan, C.: Learning safe multi-agent control with decentralized neural barrier certificates. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net (2021), https://openreview.net/forum?id=P6_q1BRxY8Q
53. Richards, S.M., Berkenkamp, F., Krause, A.: The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In: 2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings. Proceedings of Machine Learning Research, vol. 87, pp. 466–476. PMLR (2018), http://proceedings.mlr.press/v87/richards18a.html

54. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
55. Solar-Lezama, A., Tancau, L., Bodík, R., Seshia, S.A., Saraswat, V.A.: Combinatorial sketching for finite programs. In: Shen, J.P., Martonosi, M. (eds.) Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2006, San Jose, CA, USA, October 21-25, 2006. pp. 404–415. ACM (2006). https://doi.org/10.1145/1168857.1168907, https://doi.org/10.1145/1168857.1168907
56. Soudjani, S.E.Z., Gevaerts, C., Abate, A.: FAUST ^{2} : Formal abstractions of uncountable-state stochastic processes. In: Baier, C., Tinelli, C. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9035, pp. 272–286. Springer (2015). https://doi.org/10.1007/978-3-662-46681-0_23, https://doi.org/10.1007/978-3-662-46681-0_23
57. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: Bengio, Y., LeCun, Y. (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (2014), http://arxiv.org/abs/1312.6199
58. Taylor, A.J., Singletary, A., Yue, Y., Ames, A.D.: Learning for safety-critical control with control barrier functions. In: Bayen, A.M., Jadbabaie, A., Pappas, G.J., Parrilo, P.A., Recht, B., Tomlin, C.J., Zeilinger, M.N. (eds.) Proceedings of the 2nd Annual Conference on Learning for Dynamics and Control, L4DC 2020, Online Event, Berkeley, CA, USA, 11-12 June 2020. Proceedings of Machine Learning Research, vol. 120, pp. 708–717. PMLR (2020), http://proceedings.mlr.press/v120/taylor20a.html
59. Uchibe, E., Doya, K.: Constrained reinforcement learning from intrinsic and extrinsic rewards. In: 2007 IEEE 6th International Conference on Development and Learning. pp. 163–168. IEEE (2007)
60. Vaidya, U.: Stochastic stability analysis of discrete-time system using lyapunov measure. In: American Control Conference, ACC 2015, Chicago, IL, USA, July 1-3, 2015. pp. 4646–4651. IEEE (2015). https://doi.org/10.1109/ACC.2015.7172061, https://doi.org/10.1109/ACC.2015.7172061
61. Verma, A., Le, H.M., Yue, Y., Chaudhuri, S.: Imitation-projected programmatic reinforcement learning. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada. pp. 15726–15737 (2019), https://proceedings.neurips.cc/paper/2019/hash/5a44a53b7d26bb1e54c05222f186dcfb-Abstract.html
62. Verma, A., Murali, V., Singh, R., Kohli, P., Chaudhuri, S.: Programmatically interpretable reinforcement learning. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 5052–5061. PMLR (2018), http://proceedings.mlr.press/v80/verma18a.html
63. Vinod, A.P., Gleason, J.D., Oishi, M.M.K.: Sreachtools: a MATLAB stochastic reachability toolbox. In: Ozay, N., Prabhakar, P. (eds.) Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019. pp. 33–38.

ACM (2019). https://doi.org/10.1145/3302504.3311809, https://doi.org/10.1145/3302504.3311809
64. Wang, J., Sun, Y., Fu, H., Chatterjee, K., Goharshady, A.K.: Quantitative analysis of assertion violations in probabilistic programs. In: Freund, S.N., Yahav, E. (eds.) PLDI '21: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20-25, 2021. pp. 1171–1186. ACM (2021). https://doi.org/10.1145/3453483.3454102, https://doi.org/10.1145/3453483.3454102
65. Wang, P., Fu, H., Goharshady, A.K., Chatterjee, K., Qin, X., Shi, W.: Cost analysis of nondeterministic probabilistic programs. In: McKinley, K.S., Fisher, K. (eds.) Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22-26, 2019. pp. 204–220. ACM (2019). https://doi.org/10.1145/3314221.3314581, https://doi.org/10.1145/3314221.3314581
66. Williams, D.: Probability with Martingales. Cambridge mathematical textbooks, Cambridge University Press (1991)
67. Zhu, H., Xiong, Z., Magill, S., Jagannathan, S.: An inductive synthesis framework for verifiable reinforcement learning. In: McKinley, K.S., Fisher, K. (eds.) Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22-26, 2019. pp. 686–701. ACM (2019). https://doi.org/10.1145/3314221.3314638, https://doi.org/10.1145/3314221.3314638
68. Zikelic, D., Lechner, M., Henzinger, T.A., Chatterjee, K.: Learning control policies for stochastic systems with reach-avoid guarantees. To appear at the Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23) (2023)