PAS: Probably Approximate Safety Verification of Reinforcement Learning Policy Using Scenario Optimization

Arambam James Singh
Nanyang Technological University
Singapore
arambam.singh@ntu.edu.sg

Arvind Easwaran
Nanyang Technological University
Singapore
arvinde@ntu.edu.sg

ABSTRACT

With the advancement of machine learning based automation in the current digital world, the problem of safety verification of such systems is becoming crucial, especially in safety-critical domains like self-driving cars, robotics, etc. Reinforcement learning (RL) is an emerging machine learning technique with many applications, including in safety-critical domains. The classical safety verification approach of making a binary decision on determining whether a system is safe or unsafe is particularly challenging for an RL system. Such an approach generally requires prior knowledge about the system, e.g., the transition model of the system, the set of unsafe states in the environment, etc., which are typically unavailable in a standard RL setting. Instead, this paper addresses the safety verification problem from a quantitative safety perspective, i.e., we quantify the safe behavior of the policy in terms of probability. We formulate the safety verification problem as a chance-constrained optimization using the technique of barrier certificate. We then use a sampling based approach called scenario optimization to solve the chance-constrained problem, which gives the desired probabilistic guarantee on the safe behavior of the policy. Our extensive empirical evaluation shows the validity and robustness of our approach in three RL domains.

KEYWORDS

Safety verification; reinforcement Learning; scenario optimization

ACM Reference Format:
Arambam James Singh and Arvind Easwaran. 2024. PAS: Probably Approximate Safety Verification of Reinforcement Learning Policy Using Scenario Optimization. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

Recent breakthroughs in the area of reinforcement learning (RL) have led to the achievement of significant milestones in the field of artificial intelligence (AI). The landmarks include playing atari video games at the human level [24], beating the best human player in the game of Go [33], and other challenging domains. However, most of these breakthroughs are mainly in the so-called *artificial domains*. Real-world applications of RL are limited [13]. In real-world scenarios, especially in safety-critical domains like self-driving cars, robotics, etc., achieving a high-performance RL objective is not sufficient; verifying the safe behavior of the RL policy is also paramount. In a typical real-world setting, training is generally done in a simulation environment. For instance, training a physical robot or a self-driving car in a real-world environment is impractical due to the risks involved. Such a learning-based AI system trained in the simulation environment needs safety verification before being deployed in a real-world scenario. There is a plethora of work [2, 16, 20, 23, 24, 31, 32] on the development of new reinforcement learning algorithms. Still, a limited number of studies address the problem of safety verification of RL policies. With this work, we intend to fill the gap.

The classical safety verification problem of a dynamical system [6, 17, 18] is generally studied under *formal methods* in which the main goal is to prove with mathematical rigor that no trajectory generated from an initial state of the dynamical system can ever enter an unsafe region in the state space. The solution method is typically a model-checking approach to verify whether the dynamical system satisfies the given safety specifications, resulting in a binary decision on whether the system is safe or unsafe. One of the key challenges in the classical approach is that the solution method generally requires prior knowledge about the system, e.g.,

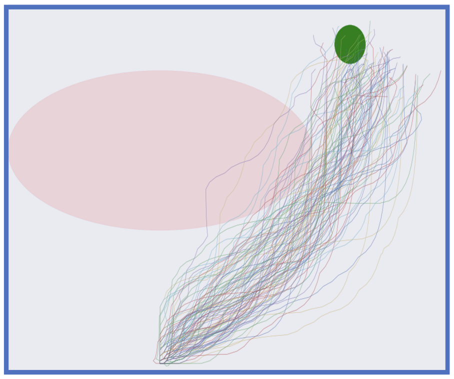


Figure 1: **A safe navigation RL environment where the objective is to reach the goal (green circle) by avoiding an unsafe region (red ellipse). The RL agent starts from an initial state below the unsafe region. Different curve lines represent different trajectory samples generated from the policy.**

This work was supported in part by AI.SG Research Grant #AISG2-RP-2020-017. This research is also part of the programme DesCartes and is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

the transition dynamics of the system, the set of unsafe states in the state space, etc.

In a standard RL setting, using a classical approach of safety verification is challenging because, typically, the transition model is unknown either due to a black-box model of the system or the model is not in a closed form for formal verification, and the set of unsafe states is also not available. Under such a setting, a classical formal methods based approach to verify the safe behavior of an RL policy is not feasible. In this work, we address the safety verification problem from a quantitative safety perspective, i.e., we quantify the safe or unsafe behavior of the RL policy in terms of probability. Formally, we seek to determine the probability of unsafe behavior of the policy remaining below a user-specified quantitative safety target with some specified confidence. We call such a system probably approximately safe. Our proposed method of *probably approximate safety* (PAS) verification is motivated by the PAC learning framework [34] as it also provides safety guarantees in terms of error probability and confidence. Thus, our approach can also be regarded as statistical safety verification of RL policies.

In Fig.1, we present a simple didactic RL task; the agent starts from an initial state below the unsafe region. The objective is to reach the goal (green circle) by avoiding an unsafe region (red ellipse) in the state space. We observe that many trajectories can reach the goal by preventing the unsafe region, but some of them do enter the region. In this scenario, depending on a specified safety tolerance level, the RL policy may be classified as a safety risk even though it fulfills the objective of reaching the goal. With our proposed PAS verification approach, given a safety threshold, we aim to compute the probability of safe behavior for the given policy with high confidence. In our method, we use the technique of barrier certificates [26, 27], a popular tool used in certifying the stability of dynamical systems. Using the barrier certificate, we find a region in the state space called a *barrier region* for the policy. An important feature of the barrier region is that it exhibits the property of *forward invariance* [36], i.e., any trajectory originating from this region stays within the same region. This property is key in ensuring the probabilistic safety guarantees associated with the barrier region would also apply to any trajectory within the region. Thus, our main objectives of this work are to find the policy's barrier region and obtain the probabilistic safety guarantees associated with it. Our key contributions are summarised below:

- We introduce a novel statistical safety verification problem of a given test RL policy.
- We also propose a novel approach to address the safety verification problem. We use the technique of barrier certificates to formulate the problem of finding the barrier region of the policy as a chance-constrained optimization with safety constraints.
- We use a sampling-based approach called scenario optimization [10] to solve the chance-constrained problem, which gives the barrier region of the policy along with the desired probabilistic safety guarantees.

The rest of the paper is outlined as follows. In Section 2, we discuss some related works in the safety verification of RL systems. Section 3 provides the relevant background for our proposed method. In Section 4, we describe our proposed probably approximate safety verification method of an RL policy. Extensive empirical evaluations of our method are discussed in Section 5, and we summarize the work in Section 6.

2 RELATED WORK

This section discusses some recent works in the safety verification of RL systems. Fulton et al. [14] proposed a formal verification based method for provably safe learning of RL systems. They provide formal proof that incorporating safe control mechanisms obtained by formal verification techniques into the learning system preserves safety guarantees. However, their method relies on learning an accurate RL environment model, which can be challenging. Bacci et al. [1] proposed an approach that gives probabilistic guarantees on the safety aspects of an RL system. The authors introduce a probabilistic model-checking technique on an abstraction of the original Markov Decision Process (MDP) to compute an upper-bound probability of reaching a failure state. The work of Bastani et al. [4, 5], Gupta et al. [15] and Berkenkamp et al. [7] also proposed safety verification approaches in which either a *nominal dynamics model* (an approximation of the true dynamics model) is known in advance, or the dynamics model is learned from scratch. Verma et al. [35] propose a program synthesis based approach to learn interpretable and verifiable policy from a pre-trained RL policy. However, the approach may not be scalable to complex domains with large state and action spaces in which the policy is required to learn intricate behaviors.

Another line of study is motivated by the technique of barrier certificates. In classical safety verification problems, barrier certificate [19, 26, 27] is a popular tool for certifying the stability of a dynamical system. Luo et al. [22] proposed a safety verification technique in which the barrier certificate is learned along with the policy and the dynamics model of the MDP in an iterative fashion. The policy optimized within the barrier region is certified safe without any safety violations. In their method, the set of unsafe states is given in advance, which may not be available in many complex domains with large state space. Cheng et al. [12] also use a similar methodology of using barrier certificates to ensure the safety of policy learned using any off-the-shelf RL algorithm. However, they use a handcrafted barrier function and Gaussian Processes to model the system dynamics, which requires domain knowledge.

In contrast to previous methods, our proposed *probably approximate safety* (PAS) verification approach is sampling-based. It does not require knowledge of the model dynamics or knowledge of the unsafe states in advance. The main requirement in our method is access to the simulator that can provide the trajectory samples along with a cost value at every time step, like the reward, which is a common setting in a standard RL problem. Our approach is also scalable because the sampling process can be easily parallelized. To the best of our knowledge, our proposed model-free sampling-based approach is first in the safety verification of RL policies.

3 BACKGROUND

3.1 Markov decision process

A standard reinforcement learning problem is formally described by a Markov Decision Process (MDP) [28]. An MDP is defined

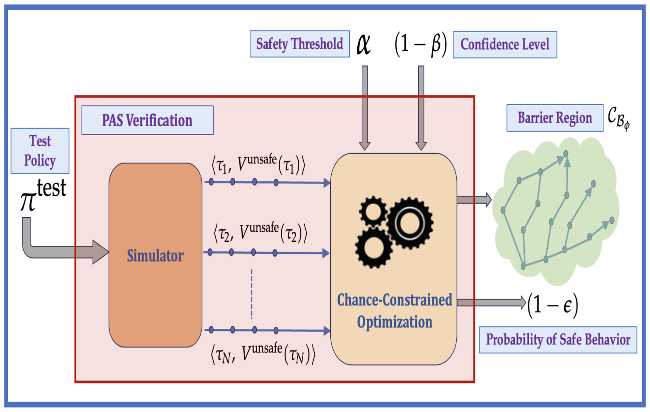

Safety Threshold
α
$(1-\beta)$
Confidence Level
PAS Verification
Test Policy
π^{test}
Simulator
$\langle\tau_1, V^{\text{unsafe}}(\tau_1)\rangle$
$\langle\tau_2, V^{\text{unsafe}}(\tau_2)\rangle$
$\langle\tau_N, V^{\text{unsafe}}(\tau_N)\rangle$
Chance-Constrained Optimization
Barrier Region
$C_{\mathcal{B}_\phi}$
$(1-\epsilon)$
Probability of Safe Behavior

Figure 2: The figure shows the system diagram of our proposed *probably approximate safety* (PAS) verification approach. The trajectory samples of the policy π^{test} are obtained from the simulator. Like the reward value, the simulator also provides a cost value c_t at each time step, which is used to compute the safety value of trajectory $V^{\text{unsafe}}(\tau)$. For given values of the safety threshold α and confidence level $(1-\beta)$, our PAS verification approach provides the barrier region $(C_{\mathcal{B}_\phi})$ along with the probability of safe behavior $(1-\epsilon)$ of the policy.

by a tuple $\langle S, A, \mathcal{T}, r, \gamma, H\rangle$, where $s_t \in S$ and $a_t \in A$ denote the state and action of the agent, respectively. At each time step t, the agent transitions to a next state s_{t+1} following a transition model $\mathcal{T}(s_{t+1}|s_t, a_t)$. $r(s_t, a_t)$ is a reward function, $\gamma \in [0,1)$ is a discount factor and H is a planning horizon. S^{ini} denotes a set of initial states, and $p_0(\cdot)$ is the initial state distribution. The agent gets action a_t from a policy $\pi(\cdot|s_t)$. The main goal in a standard RL problem is to find a policy π that maximizes the expected long-term reward $\mathbb{E}[\sum_{t=0}^{H-1}\gamma^t r(s_t, a_t)]$.

Note that the objective of this paper is not to find an optimal policy for a given MDP, but to propose a new methodology for the safety verification of a test RL policy. Thus, we assume the test policy for the given MDP is provided.

3.2 Problem setup

Let, $\pi^{\text{test}}(a_t|s_t)$ denote the test RL policy, $s_0 \sim p_0(\cdot)$ be an initial state sampled from an initial state distribution p_0, and $\tau = (s_0, a_0, s_1, a_1, ..., s_{H-1})$ be a trajectory induced by the test policy π^{test}. The distribution of a trajectory τ is given by:

$$P_{\pi^{\text{test}}}(\tau) = p(s_0) \cdot \prod_{t=0}^{H-1} \pi^{\text{test}}(a_t|s_t) \cdot \mathcal{T}(s_{t+1}|s_t, a_t) \tag{1}$$

Since the transition model $\mathcal{T}(s_{t+1}|s_t, a_t)$ is not available, we obtain the trajectory samples through a simulator. Our goal in this work is to verify the safe behavior of the test policy π^{test}. The classical safety verification problem is concerned with proving that no trajectory originating from an initial state can ever enter an unsafe region in the state space. However, such a problem is very challenging to solve because addressing it would require prior knowledge of the unsafe states and access to the transition model $\mathcal{T}$ of the MDP, which are generally not available in a standard RL setting. In many complex domains with large state spaces, defining the set of unsafe states may not be feasible. It would require enumeration over the state space with a safety or cost function. Instead, in this paper, we measure the safety aspect of the policy in a more general and tractable way. Like in a standard RL setting, the simulator provides a reward value for each state and action; we also receive a cost value for each state and action at each time step. The cost value is then used to compute the safety value of the trajectory. Our motivation for this work is quantitative safety, i.e., determining the probability of unsafe behavior of the policy staying below a quantitative safety target with some specified confidence. We now formally define the problem.

Problem 1: Given a parameter $\beta \in (0,1)$, the objective is to compute a probability $\epsilon \in (0,1)$ such that, with confidence at least $(1-\beta)$, an unsafe behavior value $V^{\text{unsafe}}(\tau)$ of any trajectory $\tau \sim P_{\pi^{\text{test}}}(\cdot)$ starting from an initial state $s_0 \in S^{\text{ini}}$ and following the dynamics of the MDP induced by the policy π^{test} as per (1), is below a given safety threshold α with a probability larger than $(1-\epsilon)$.

$$P\Big(V^{\text{unsafe}}(\tau) \le \alpha\Big) \ge (1-\epsilon) \tag{2}$$

where a trajectory τ is a sample from the distribution $P_{\pi^{\text{test}}}(\cdot)$, $V^{\text{unsafe}}(\tau) = \sum_{t=0}^{H-1}\gamma^t \cdot c_t$, and c_t is a cost value for each state and action at each time step from the trajectory. The cost value is provided by the simulator.

3.3 Barrier certificate

A barrier certificate (BC) [26, 27] is a common technique in safety verification for dynamical systems. It is generally employed for evaluating stability in continuous-time dynamical systems. Recently, [22, 29] have used BC in a discrete-time dynamical system. In this work, we use it to evaluate a test policy's safe behavior in a discrete-time MDP. A barrier certificate is formally defined as a real-valued function $\mathcal{B}: S \longrightarrow \mathbb{R}$ if for any state $s \in S$ such that $\mathcal{B}(s) \ge 0, \mathcal{B}(s') \ge 0$, where s' is the next state. Below are the two conditions a BC must satisfy:

- **C1:** For $s_0 \in S^{\text{ini}}$, $\mathcal{B}(s_0) \ge 0$ with probability 1.
- **C2:** For any s such that $B(s) \ge 0, \min_{s' \in \mathcal{S}'_s} \mathcal{B}(s') \ge 0$

where, $\mathcal{S}'_s$ is a set of next-states for state s.

Let $C_{\mathcal{B}} = \{s : \mathcal{B}(s) \ge 0\}$ be the superlevel set of the function $\mathcal{B}$. The conditions above guarantee that if $s \in C_{\mathcal{B}}$, then $s' \in C_{\mathcal{B}}$. We refer to the region of state space covered under $C_{\mathcal{B}}$ as a *barrier region* (BR) [7]. The region exhibits a vital property known as *forward invariant property* [36], i.e., any trajectory originating from this region stays within the same region. One of the primary objectives of this work is to find the barrier region of the test policy, and we obtain it by learning the barrier certificate function using the trajectory samples generated from the test policy.

4 METHODOLOGY

In this section, we present our sampling-based approach for *probably approximate safety* (PAS) verification of a test RL policy π^{test}. Here, we refer to a trajectory τ from the distribution $P_{\pi^{\text{test}}}(\cdot)$ as a sample. Since we do not have access to the transition model $\mathcal{T}$, we get the trajectory sample from a simulator. Along with the trajectory samples, we also receive the cost value c_t from the simulator at every time step. We then use it to compute $V^{\text{unsafe}}(\cdot)$, the safety value of the trajectory sample. The complete system diagram is shown in Fig.2.

As discussed in the previous section, the notion of barrier region (BR) is mainly used in the classical safety verification problem of a dynamical system, where the goal is to find the BR of the system. It ensures that any state trajectory originating within the BR will not leave the region. However, as highlighted in the previous section, our goal in this work is to provide probabilistic safety guarantees for a given policy. Formally, we want to determine the probability of unsafe behavior of the policy staying below a safety target with high confidence. We now redefine our original problem in Problem 1 using barrier region.

Problem 2: Given a parameter $\beta \in (0, 1)$, the goal is to find a barrier region ($C_{\mathcal{B}}$) of the given policy over the state space of the MDP, such that with a degree of confidence $(1 - \beta)$ any trajectory $\tau \sim P_{\pi^{\text{test}}}$ generated within the barrier region satisfies the safety constraint in (2) with a high probability $(1 - \epsilon)$.

Let $\mathcal{B}_\phi$ be a parameterized linear function representing a barrier certificate (BC) with parameter ϕ, defined as $\mathcal{B}_\phi(s) = \phi_w \cdot s + \phi_b$, where $\phi = \{\phi_w, \phi_b\}$. Let $C_{\mathcal{B}_\phi} = \{s : \mathcal{B}_\phi(s) \geq 0\}$ be the BR we are interested in finding, $\tau = (s_0, a_0, s_1, a_1, ..., s_{H-1})$ be a trajectory sample from the distribution $P_{\pi^{\text{test}}}(\cdot)$. $\mathcal{S}_\tau = \{s_0, s_1, \ldots, s_{H-1}\}$ denote a set containing states from the trajectory τ and $\mathcal{S}^\phi_\tau = \{s : s \in \mathcal{S}_\tau \wedge s \in C_{\mathcal{B}_\phi}\}$ denote a set containing states that belong to both state trajectory set $\mathcal{S}_\tau$ and $C_{\mathcal{B}_\phi}$. We know that any BC must satisfy the conditions **C1** and **C2**. Regarding the condition **C1**, we can always initialize the parameters of the BC to include the initial states. The condition **C2** basically states, for any state s in the barrier region, we want to make sure that the next state s' with a minimum value of $\mathcal{B}_\phi(s') \geq 0$. For a given value of ϕ, we have,

$$J(\phi) = \max_{s \in \mathcal{S}^\phi_\tau,\, s' \in \mathcal{SP}^\tau_s} -\mathcal{B}_\phi(s') \tag{3}$$

where, $\mathcal{SP}^\tau_s$ denotes the set of next-states for state s from the trajectory τ. Note that in the case of a discrete state space, there can be multiple next states for a state in a trajectory. Our goal is to find the BR that also satisfies the safety behavior constraint (2) with high probability. Thus, the complete optimization problem becomes,

$$\min_\phi J(\phi) = \min_\phi \max_{s \in \mathcal{S}^\phi_\tau,\, s' \in \mathcal{SP}^\tau_s} -\mathcal{B}_\phi(s') \tag{4}$$

$$\text{s.t. } P\Big(V^{\text{unsafe}}\left(\mathcal{S}^\phi_\tau\right) \leq \alpha\Big) \geq (1 - \epsilon) \tag{5}$$

where, a trajectory τ is a sample from the distribution $P_{\pi^{\text{test}}}(\cdot)$ and $V^{\text{unsafe}}(\mathcal{S}^\phi_\tau) = \sum_{s_t \in \mathcal{S}^\phi_\tau} \gamma^t \cdot c_t$.

Note that this formulation is an instance of a chance-constrained optimization problem (CCP) [11] with trajectory τ as the uncertain parameter. CCPs are known to be computationally hard to solve. In this work, we use the scenario optimization approach proposed by [9, 10] to address the CCP above. This approach offers a deterministic relaxation of the original stochastic problem with a confidence bound on the solution. The key idea is to substitute the probabilistic constraint with N i.i.d sample constraints. We substitute the chance-constraint (5) in the formulation above with N sample instances of the constraint as follows,

$$\min_\phi J^N(\phi) = \min_\phi \max_{s \in \mathcal{S}^\phi_N,\, s' \in \mathcal{SP}^N_s} -\mathcal{B}_\phi(s') \tag{6}$$

$$\text{s.t. } V^{\text{unsafe}}\left(\mathcal{S}^\phi_{\tau_i}\right) \leq \alpha, \ \ \forall i \in [1, N] \tag{7}$$

where $\mathcal{S}^\phi_N = \bigcup_{i=1}^N \mathcal{S}^\phi_{\tau_i}$ and $\mathcal{SP}^N_s = \bigcup_{i=1}^N \mathcal{SP}^{\tau_i}_s$.

A key limitation in finding the solution satisfying all the N constraints is that the quality of the resulting solution can be poor. The solution may not be a good approximation of the chance-constrained solution. Thus, we allow some violating constraints from the total sampled constraints to be removed. This essentially improves the robustness of the solution quality[1]. The number of violating constraints that are removed has a direct impact on the safety probability value $(1 - \epsilon)$. For a given value of a total number of samples N and safety threshold α, the actual number of violating constraints is fixed. We also perform an ablation study in the experimental section to analyze the effect on the safety probability value by increasing the number of violating constraints that are removed from a smaller value to the actual number of violating constraints.

Let k out-of N violating constraints be selected and removed. We now rewrite the above optimization with the remaining constraints,

$$\min_\phi J^{N,k}(\phi) = \min_\phi \max_{s \in \mathcal{S}^\phi_{N-k},\, s' \in \mathcal{SP}^{N-k}_s} -\mathcal{B}_\phi(s') \tag{8}$$

$$\text{s.t. } V^{\text{unsafe}}\left(\mathcal{S}^\phi_{\tau_i}\right) \leq \alpha, \ \ \forall i \in [1, N-k] \tag{9}$$

where $\mathcal{S}^\phi_{N-k} = \bigcup_{i=1}^{N-k} \mathcal{S}^\phi_{\tau_i}$ and $\mathcal{SP}^{N-k}_s = \bigcup_{i=1}^{N-k} \mathcal{SP}^{\tau_i}_s$.

The complete algorithm for solving the optimization problem above is presented in Algorithm 1.

Algorithm sketch: We first initialize the barrier region set $C_{\mathcal{B}_\phi} = S^{\text{ini}}$ to the set of initial states. Next, we get N trajectory samples from the simulator by executing the test policy π^{test}. Then, we identify the trajectories out of N trajectories that violate the safety constraint and denote it as k. From line 8 to line 22, we are essentially computing the inner maximization in (8), which is to find s' for a given ϕ. In lines 23 and 24, we obtain the gradient w.r.t ϕ and update the parameter using gradient descent. We repeat this process for M number of iterations. In line 26, we obtain the barrier region $C_{\mathcal{B}_\phi}$.

4.1 Computation of ϵ

Let $\mathbb{P}(\cdot)$ be a violation probability function of the safety constraint for any given value of ϕ and τ, and p^ϕ_{vio} denote the probability value.

$$p^\phi_{\text{vio}} = \mathbb{P}\left(V^{\text{unsafe}}\left(\mathcal{S}^\phi_\tau\right) > \alpha\right) \tag{10}$$

Let $\phi_{N,k}$ be the solution of the sampling based optimisation in Eq. (8). Note that $\phi_{N,k}$ is a random variable because it depends on random multi-samples of τ as $(\tau_1, \ldots, \tau_N)$. As a result, the violation probability with $\phi_{N,k}$, $p^{\phi_{N,k}}_{\text{vio}}$ is also a random variable. Let $\mathbb{P}^N = \mathbb{P} \times \cdots \times \mathbb{P}$ be a probability measure which is a product probability because each trajectory τ_i is an independent sample of the

[1]Appendix A, part A.1 in [10] provides a more detailed description regarding constraint removal for interested readers to gain more insights.

Algorithm 1:

```
1  Initialize barrier certificate parameter $\phi$, $C_{\mathcal{B}_\phi} = S^{\text{ini}}$
2  Collect $N$ trajectories $\{\tau_1, \dots, \tau_N\}$ from the simulator by running
     test policy $\pi^{\text{test}}$
3  Identify all trajectories that violate the safety constraints and
     remove them. Let this number denote $k$.
4  For the remaining $(N-k)$ trajectories,
5      Compute $\mathcal{S}_{N-k} = \bigcup_{i=1}^{N-k} \mathcal{S}_{\tau_i}$
6      Compute $\mathcal{SP}_s^{N-k} = \bigcup_{i=1}^{N-k} \mathcal{SP}_s^{\tau_i}$  $\forall s \in \mathcal{S}_{N-k}$,
7  for $M$ iterations do
8      $\mathcal{S}^{\phi}_{N-k} = \{\}$
9      for each $s \in \mathcal{S}_{N-k}$ do
10         if $\mathcal{B}_\phi(s) \geq 0$ then
11             $\mathcal{S}^{\phi}_{N-k} = \mathcal{S}^{\phi}_{N-k} \cup \{s\}$  /*Only the states within BR */
12             $C_{\mathcal{B}_\phi} = C_{\mathcal{B}_\phi} \cup \{s\}$
13         end
14     end
15     for each $s \in \mathcal{S}^{\phi}_{N-k}$ do
16         for each $s' \in \mathcal{SP}_s^{N-k}$ do
17             if $-\mathcal{B}_\phi(s') \geq 0$ and $-\mathcal{B}_\phi(s') \geq -\mathcal{B}_\phi(\text{max\_}s')$ then
18                 max_$s' = s'$
19             end
20         end
21     end
22     $s'^{*} = \text{max\_}s'$
23     Compute gradient, $\nabla_\phi J(\phi) = -\nabla_\phi \mathcal{B}_\phi(s'^{*})$
24     Update, $\phi \leftarrow \phi - \delta \cdot \nabla_\phi J(\phi)$
25 end
26 return $C_{\mathcal{B}_\phi}$
```

distribution $P_{\pi^{\text{test}}}(\cdot)$. Thus, $p^{\phi_{N,k}}_{\text{vio}}$ can be less than ϵ for some multi-samples $(\tau_1, \dots, \tau_N)$ and greater for others. The theorem below establishes the condition under which $p^{\phi_{N,k}}_{\text{vio}} > \epsilon$ has any arbitrary small probability β.

Theorem 4.1. *[10] Let $\beta \in (0,1)$ be any small confidence parameter. If N and k are such that*

$$\binom{k+m-1}{k} \sum_{i=0}^{k+m-1} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i} \leq \beta \tag{11}$$

where $m = |\phi|$ is the number of optimization variables, then with at least $(1-\beta)$ confidence, we have,

$$\mathbb{P}^N \left(p^{\phi_{N,k}}_{\text{vio}} \leq \epsilon \right) \geq (1-\beta) \tag{12}$$

Substituting $p^{\phi_{N,k}}_{\text{vio}}$ from (10) in the above inequality, we get,

$$\mathbb{P}^N \left(\mathbb{P} \left(V^{\text{unsafe}} \left(\mathcal{S}^{\phi_{N,k}}_{\tau_i} \right) > \alpha \right) \leq \epsilon \right) \geq (1-\beta) \tag{13}$$

where, $i \in [1, N-k]$. The above inequality can also be shown in terms of the probability of constraint satisfaction,

$$\mathbb{P}^N \left(\mathbb{P} \left(V^{\text{unsafe}} \left(\mathcal{S}^{\phi_{N,k}}_{\tau_i} \right) < \alpha \right) \leq (1-\epsilon) \right) \geq (1-\beta) \tag{14}$$

This theorem establishes the relation between the key parameters—the number of trajectory samples (N), the number of trajectories violating the safety constraint (k), the probability of satisfying the safety constraint $(1-\epsilon)$, and confidence level $(1-\beta)$. From Eq. (8) in [10], and assuming $m \leq 5$, we get an expression to compute a lower bound on ϵ given other parameters,

$$\epsilon \geq \min \left\{ 1, \; \frac{1}{N} \left[k + \ln \frac{1}{\beta} + \sqrt{\ln^2 \frac{1}{\beta} + 2k \ln \frac{1}{\beta}} \; \right] \right\} \tag{15}$$

5 EXPERIMENTS

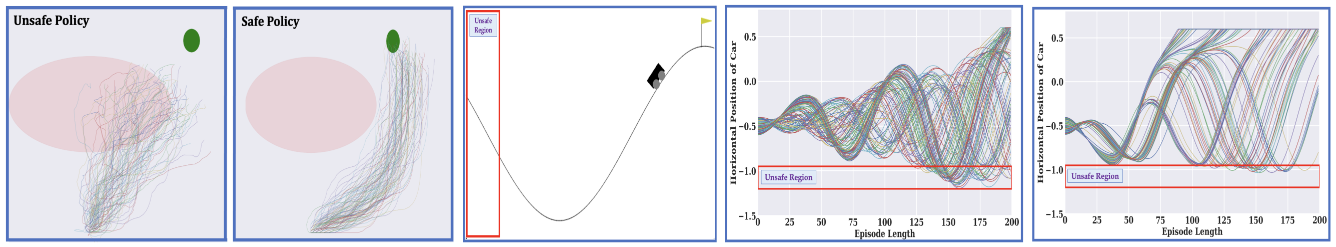
We evaluate our proposed *probably approximate safety* (PAS) verification approach on three RL domains: i) safe navigation as shown in Fig.1, ii) safe mountain car as shown in Fig.3c and iii) safe cartpole as shown in Fig.4a. As mentioned above, our objective in this paper is not to find an optimal policy but to provide a probability of safe behavior for a given policy. For our analysis, we use two kinds of policies— *unsafe* and *safe* policies. The motivation behind using the two policies is to compare the probability of safe behavior between the two policies and verify empirically the safe policy has a higher safety probability than the unsafe one. We obtained the unsafe policy by training for fewer training steps than the safe policy.

5.1 Test environments

Safe navigation: We modified the popular multiagent-RL domain of cooperative navigation [21] for a single-agent setting and added an unsafe region in the state space as shown in Fig.1. In this environment, the agent starts from an initial state below the unsafe region, and the goal is to reach the green circle by avoiding the red ellipse region (unsafe region)[2]. The agent's reward is a combination of a negative of the Euclidian distance from the goal and a cost of -10 if the agent enters the unsafe region. We train both policies using a popular off-the-shelf RL algorithm called Proximal Policy Optimization (PPO) [32]. We use the stable-baseline3 [30] implementation of the PPO algorithm. We train the safe policy for 500k steps, while the unsafe policy is trained only for 100k steps. In Fig.3a and Fig.3b, we present the trajectory samples generated by unsafe and safe policies, respectively. For the unsafe policy in Fig.3a, most of the trajectories are entering the unsafe region. In contrast, for the safe policy in Fig.3b, almost all the trajectories can avoid the unsafe region while also reaching close to the goal. For our PAS algorithm, we use the cost value $c_t = 10$ to compute the unsafe behavior value $(V^{\text{unsafe}}(\tau))$ of a trajectory (τ) and the number of parameters of the barrier certificate $m = 5$.

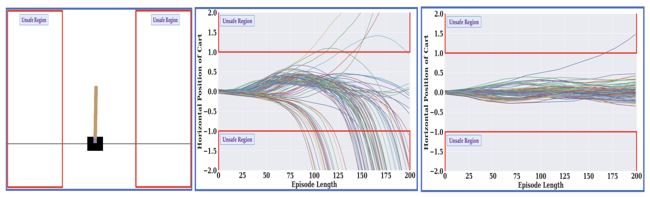
Safe mountain car: We modify the original mountain car domain [8, 25] to include unsafe region (red box) as shown in Fig.3c. The unsafe region is based on the car's position along the x-axis. If the car's position is less than -0.95, the car enters the unsafe region. In the original mountain car environment, the car receives a reward of −1 at every time step. The goal is to reach the flag as quickly as possible. We also include an additional cost of −10 to the reward if the agent enters the unsafe region. We use the PPO algorithm to train both unsafe and safe policies. The unsafe and safe policies are trained for 1M and 5M steps, respectively. In Fig.3d and Fig.3e, we show the trajectory samples generated from the unsafe and

[2]Note, this unsafe region is unknown to our PAS algorithm but is only used in the trajectory simulator to generate cost values.



(a) Unsafe policy **(b)** Safe policy **(c)** Safe mountain car **(d)** Unsafe policy **(e)** Safe policy

Figure 3: (a) and (b) show the trajectory samples generated from the unsafe and safe policies in a safe navigation environment, respectively. (c) shows the safe mountain car environment. The goal of the car agent is to reach the yellow flag on the top of the mountain by avoiding the unsafe region in the red box. (d) and (e) show the trajectory samples generated from unsafe and safe policies. The x-axis shows the episode length, and the y-axis shows the horizontal position of the car. The various curves denote the trajectory samples.



(a) Safe cartpole **(b)** Unsafe policy **(c)** Safe policy

Figure 4: (a) shows the safe cartpole environment. The goal is to balance the pole by applying forces in the left and right direction of the cart and also by avoiding entering into unsafe regions. (b) and (c) show the trajectory samples generated from the unsafe and safe policies. The x-axis shows the episode length, and the y-axis shows the horizontal position of the cart. The various curves denote the trajectory samples.

safe policies, respectively. In Fig.3d, many of the trajectories are entering the unsafe region (red box), and very few reach the flag at the horizontal position of 0.6. However, for the safe policy in Fig.3e, we see very few enter the unsafe region for a short number of time steps, and most of them reach the flag position. For our PAS algorithm, we use the cost value $c_t = 10$ to compute the unsafe behavior value ($V^{\text{unsafe}}(\tau)$) of a trajectory (τ) and the number of parameters of the barrier certificate $m = 2$.

Safe cartpole: We modify the original cartpole environment [3, 8] to include unsafe regions (red boxes) as shown in Fig.4a. The unsafe regions are based on the cart's position along the x-axis. If the cart's position is less than -1 or more than 1, then the cart enters the unsafe region. In the original cartpole environment, the RL agent receives a reward of +1 at every time step for keeping the pole upright and 0 otherwise. The goal is to keep the pole upright as long as possible. We also include an additional cost of -5 to the reward if the agent enters the unsafe regions. We use the PPO algorithm to train both unsafe and safe policies. The unsafe and safe policies are trained for 300k and 1M steps, respectively. In Fig.4b and Fig.4c, we show the trajectory samples generated from the unsafe and safe policies, respectively. In Fig.4b, many trajectories enter the unsafe region (red box). However, for the safe policy in Fig.4c, we see very few enter the unsafe region, thus resulting in a high total reward. For our PAS algorithm, we use the cost value $c_t = 5$ to compute the unsafe behavior value ($V^{\text{unsafe}}(\tau)$) of a trajectory (τ) and the number of parameters of the barrier certificate $m = 2$.

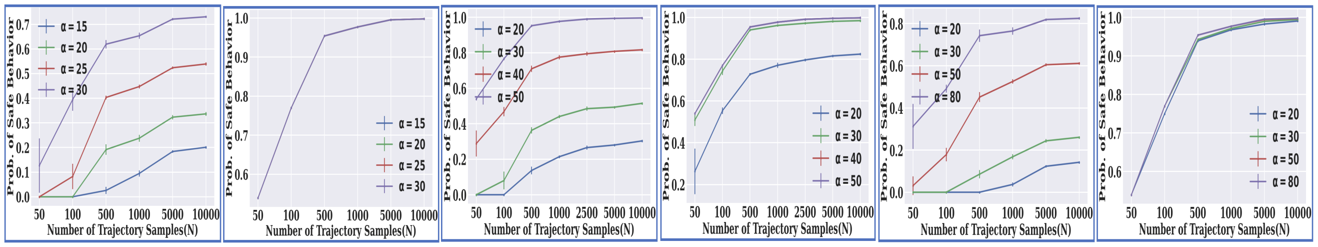
The complete description of the experimental setup and different hyperparameters used are provided in the supplementary material[3].

5.2 Probability of safe behavior with varying number of trajectory samples

Since our proposed method is sampling-based, this experiment shows how the probability of safe behavior changes with varying numbers of trajectory samples. We compute the safety probability $(1 - \epsilon)$ from the expression for ϵ in (15). In all experiments, we use five different seeds because there can be multiple sample trajectories for a fixed value of N. We set a high confidence level $(1 - \beta) = 0.99999$, and the total number of constraint violations k is obtained by evaluating whether each trajectory sample is violating the safety constraint (2) for the given threshold α. Also, note that, for a given value of α and N, we may have different values of k for different seeds depending on the quality of the trajectory samples. This will lead to different values of the safety probability $(1 - \epsilon)$ since ϵ depends on N, β, and k as per Eq. (15). Thus, we report the average probability value and standard deviations over the different seeds.

Results: In Fig.5, we show the probability of safe behavior for the unsafe and safe policies for the three test environments. We evaluated the results for varying values of safety constraint threshold α. Typically, the safety constraint threshold values are domain-specific. For our evaluation, the main motivation is to analyze how the safety probability values change with varying restrictions on the safety constraints, i.e., from tighter to looser constraints. The lower and higher values of α denote the tighter and looser constraints, respectively. For all the figures Fig.5a to Fig.5f, we observe a common trend for both policies: the probability value increases with increasing number of trajectory samples and converges around $N = 5000$ as the probability values do not change much beyond 5000. For the unsafe policies shown in Fig.5a, Fig.5c and Fig.5e, we observe a high variation in the probability values for different values of α. The safety probability value increases with increasing value of the safety threshold. This is an expected behavior for the unsafe policy because as the safety constraint loosens, more trajectories satisfy the safety constraint, thus increasing the probability value.

[3]http://jamesarambam.github.io/files/aamas24_sup.pdf



(a) Safe Nav.: Unsafe policy **(b)** Safe Nav.: Safe policy **(c)** Safe Mcar: Unsafe policy **(d)** Safe Mcar: Safe policy **(e)** Safe Cpole: Unsafe policy **(f)** Safe Cpole: Safe policy

Figure 5: The figures show the probability of safe behavior for the unsafe and safe policies for the safe navigation environment in (a) and (b), the safe mountain car environment in (c) and (d), and the safe cartpole environment in (e) and (f) with varying numbers of trajectory samples (N) for different safety thresholds (α). The x-axis shows the number of trajectory samples (N), and the y-axis shows the probability of safe behavior $(1 - \epsilon)$.

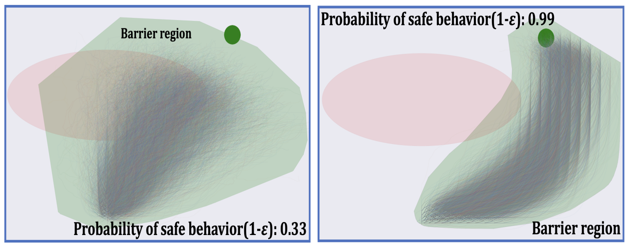
At $N = 10000$ for the tightest constraint (lowest α), we get very low probabilities < 0.3 in all three domains.

However, in the case of the safe policies in Fig.5b, Fig.5d and Fig.5f, we observe a high value of the safety probability at $N = 10000$ even for the tighter constraints (lowest α) in all three domains. This is because of the good quality trajectory samples from the policy; only a small fraction of the trajectory samples enter the unsafe region for a small number of time steps. Thus resulting in a high probability of safe behavior. Hence, with these experiments, we empirically verified that the probability of safe behavior for the safe policy is higher than that of unsafe policy.

5.3 Barrier region of policies

In Fig.6a and Fig.6b, we show the barrier region $C_{\mathcal{B}_\phi}$ of the unsafe and safe policies, respectively, for the safe navigation environment. We construct an approximate graphical representation of the barrier region (in green color polygon) by joining the outermost points of the point cloud (set of states) belonging to the barrier region set $C_{\mathcal{B}_\phi}$. We plot the barrier regions for the safe navigation experiment in Fig.5a and Fig.5b at $N = 10000$ and $\alpha = 20$. We use different 10000 trajectory samples for the trajectories inside the barrier region to show the robustness of the obtained barrier region. For the unsafe policy case in Fig.6a, we observe the obtained barrier region overlaps significantly with the unsafe region. Most trajectories enter the unsafe region for many time steps, thus violating the safety constraint, resulting in a very low safety probability value of 0.333. On the other hand, in the safe policy case, we observe a more concise graphical representation of the barrier region, as shown in Fig.6b. This is due to the more structured behavior of the trajectory samples, which is to avoid the unsafe region, thus giving a high safety probability value of 0.997. In Table 1, we provide a numerical value of the overlapping region. We compute it by counting the number of states in the barrier region that are also inside the unsafe region. The exact count of the overlap region (OR) is provided in Table 1 safe navigation environment column. We observe an OR value of 54641 for the unsafe policy with a low safety probability value of 0.333 while the safe policy has a very low OR value = 2431 with a very high safety probability value of 0.997 of the barrier region.

In Fig.7, we show the barrier regions of the policies for the safe mountain car and safe cartpole environment. We plot the barrier regions for the safe mountain car and safe cartpole experiments in Fig.5(c-d) and Fig.5(e-f), respectively, for both the policies at $N = 10000$ and $\alpha = 20$. In both environments, since the unsafe regions are based on the horizontal positions of the car (in safe mountain car) and the cart (in safe cartpole), which is a one-dimensional scalar value, the barrier regions are a line segment. We obtain the two ends of the line segment as the minimum and maximum values from the barrier region $C_{\mathcal{B}_\phi}$ set. For the unsafe policy case in Fig.7a and Fig.7c, the barrier regions cover the whole possible range of the horizontal position, which also include the unsafe regions (in red color). Thus, we observe a high overlapping region (OR) value of 11218, resulting in a low safety probability value of 0.303 from Table1 for the safe mountain car environment. Similarly, in safe

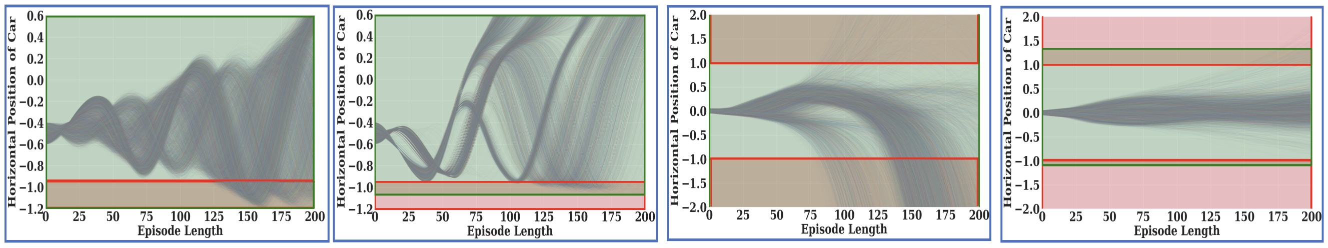


(a) Unsafe policy **(b)** Safe policy

Figure 6: The figures show an approximate graphical representation of the barrier region of the unsafe and safe policies for the safe navigation environment. The green polygon denotes the barrier region with trajectory samples inside it.

Envs	Safe navigation		Safe mountaincar		Safe cartpole	
Pol.	$\alpha = 20$		$\alpha = 20$		$\alpha = 20$	
	OR	Prob.	OR	Prob.	OR	Prob.
Unsafe	54641	0.333	11218	0.303	16287	0.141
Safe	**2431**	**0.997**	**9433**	**0.824**	**783**	**0.984**

Table 1: This table shows the quantitative results on overlapping regions (OR) of the barrier region and unsafe region for both the unsafe and safe policies for the three domains along with the corresponding safety probabilities (Prob.) associated with the barrier regions of the policies



(a) Safe Mcar: Unsafe policy (b) Safe Mcar: Safe policy (c) Safe Cpole: Unsafe policy (d) Safe Cpole: Safe policy

Figure 7: (a) and (b) show a graphical representation of the barrier regions of the unsafe and safe policies for the safe mountain car environment. (c) and (d) show the barrier regions of the unsafe and safe policies for the safe cartpole environment. The green region denotes the barrier region with trajectory samples inside it, and the red region denotes the unsafe region.

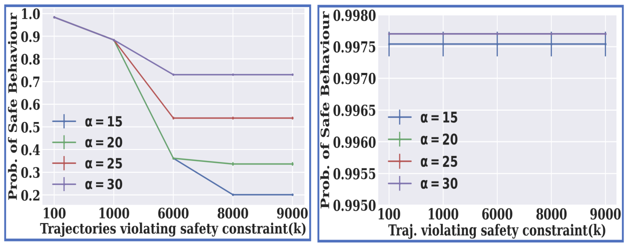
cartpole environment, we observe a high OR value of 16287 and a low safety probability value of 0.141.

However, for the safe policies, as shown in Fig.7b for the safe mountain car environment, the barrier region only slightly overlaps with the unsafe region. In the safe cartpole environment in Fig.7d, one end of the barrier region is at the boundary of the unsafe region at the bottom, and the other end slightly overlaps with the unsafe region at the top. Thus, we get a low OR value of 783 and a high safety probability value of 0.987 for the safe cartpole environment. Similarly, in the safe mountain car, we observe a comparatively low OR value of 9433 and a high safety probability value of 0.824.

5.4 Ablation

We also performed an ablation study in the safe navigation environment to analyze the impact on the safety probability values $(1-\epsilon)$ as we remove more trajectories that violate the safety constraints (k) from a smaller value to the actual number of violating trajectories. Note that the actual number of trajectories that violate the safety constraint is fixed for a given number of trajectory samples (N) and safety threshold (α). We also highlight that the probability values computed with k being smaller than the actual number of violating trajectories are incorrect and do not represent the true probabilistic guarantee. In this experiment, we set the total number of trajectory samples $N = 10000$ and a high confidence level $(1-\beta) = 0.99999$. We evaluated for varying safety thresholds $\alpha = [15, 20, 25, 30]$ and ran the experiment for five seeds, resulting in five different 10000 trajectory samples.

In the case of unsafe policy, as shown in Fig.8a, for a low number of violating trajectories $k = 100$, we get a high safety probability value for all α. $k = 100$ is a small fraction of the total trajectory samples, and given that the trajectory samples are from an unsafe policy, there is a high likelihood that all 100 trajectories violate the safety constraints for all α values. Thus, we get the same safety probability value for all α at $k = 100$. However, as we remove more violating trajectories i.e., increasing the value of k, the safety probability values for all α drop and converge to corresponding fixed values (true guarantees). For loosened constraint $\alpha = 30$ (purple color), $k = 6000$ is the maximum number of violating trajectories because the safety probability value of around 0.72 does not change beyond 6000. But, for a much tighter constraint $\alpha = 15$ (blue color), the total number of violating trajectories goes up to $k = 8000$, thus resulting in a much lower safety probability of around 0.2.



(a) Unsafe policy (b) Safe policy

Figure 8: The figures show the probability of safe behavior for the unsafe and safe policies in the safe navigation environment with varying numbers of trajectories violating the safety constraint (k) for different safety thresholds (α). The x-axis shows the number of safety constraint violations (k), and the y-axis shows the probability of safe behavior ($1-\epsilon$).

However, for the safe policy in Fig.8b, we observe a more consistent and high safety probability value for all α. This is because the trajectory samples are from the safe policy; even for a tighter constraint ($\alpha = 15$), the number of violating trajectories is 100 or less, thus resulting in a high probability of safe behavior.

6 CONCLUSION

In this paper, we addressed the safety verification problem of a given RL policy. Our key objective is to provide a probabilistic guarantee of safe behavior for the policy, and for this purpose, we developed a scenario optimization based Probably Approximate Safety (PAS) verification algorithm. Unlike previous methods, our sampling-based approach does not require any prior knowledge about the system, e.g., the transition model or set of unsafe states. Our approach is also scalable because the sampling process can be easily parallelized. The extensive empirical evaluations on different RL domains demonstrate the validity and robustness of our proposed method.

ACKNOWLEDGMENTS

This research is supported by AI Singapore as part of the *Trust to Train and Train to Trust: Agent Training Programs for Safety-Critical Environments* project in collaboration with Singapore Management University (SMU), Nanyang Technological University (NTU), and National University of Singapore (NUS).

REFERENCES

[1] Edoardo Bacci and David Parker. 2020. Probabilistic guarantees for safe deep reinforcement learning. In *Formal Modeling and Analysis of Timed Systems: 18th International Conference, FORMATS 2020, Vienna, Austria, September 1–3, 2020, Proceedings 18*. Springer, 231–248.

[2] Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy P. Lillicrap. 2018. Distributed Distributional Deterministic Policy Gradients. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[3] Andrew G Barto, Richard S Sutton, and Charles W Anderson. 1983. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics* 5 (1983), 834–846.

[4] Osbert Bastani, Shuo Li, and Anton Xu. 2021. Safe Reinforcement Learning via Statistical Model Predictive Shielding.. In *Robotics: Science and Systems*. 1–13.

[5] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. 2018. Verifiable reinforcement learning via policy extraction. *Advances in neural information processing systems* 31 (2018).

[6] Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. 2017. *Formal methods for discrete-time dynamical systems*. Vol. 89. Springer.

[7] Felix Berkenkamp, Matteo Turchetta, Angela P. Schoellig, and Andreas Krause. 2017. Safe Model-based Reinforcement Learning with Stability Guarantees. In *Advances in Neural Information Processing Systems, December 4-9, 2017, Long Beach, CA, USA*. 908–918.

[8] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv:arXiv:1606.01540

[9] Giuseppe Carlo Calafiore and Marco C Campi. 2006. The scenario approach to robust control design. *IEEE Transactions on automatic control* 51, 5 (2006), 742–753.

[10] Marco C. Campi and Simone Garatti. 2011. A Sampling-and-Discarding Approach to Chance-Constrained Optimization: Feasibility and Optimality. *J. Optim. Theory Appl.* 148, 2 (2011), 257–280.

[11] Abraham Charnes and William W Cooper. 1959. Chance-constrained programming. *Management science* 6, 1 (1959), 73–79.

[12] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. 2019. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 3387–3395.

[13] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning* 110, 9 (2021), 2419–2468.

[14] Nathan Fulton and André Platzer. 2018. Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[15] Akshita Gupta and Inseok Hwang. 2020. Safety Verification of Model Based Reinforcement Learning Controllers. *arXiv preprint arXiv:2010.10740* (2020).

[16] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden*. PMLR, 1856–1865.

[17] John Jackson, Luca Laurenti, Eric Frew, and Morteza Lahijanian. 2020. Safety verification of unknown dynamical systems via gaussian process regression. In *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 860–866.

[18] Morteza Lahijanian, Sean B Andersson, and Calin Belta. 2015. Formal verification and synthesis for discrete-time stochastic systems. *IEEE Trans. Automat. Control* 60, 8 (2015), 2031–2045.

[19] Matthew Landers and Afsaneh Doryab. 2023. Deep Reinforcement Learning Verification: A Survey. *Comput. Surveys* (2023).

[20] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

[21] Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 6379–6390.

[22] Yuping Luo and Tengyu Ma. 2021. Learning Barrier Certificates: Towards Safe Reinforcement Learning with Zero Training-time Violations. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. 25621–25632.

[23] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1928–1937.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[25] Andrew William Moore. 1990. *Efficient memory-based learning for robot control*. Technical Report. University of Cambridge, Computer Laboratory.

[26] Stephen Prajna, Ali Jadbabaie, and George J. Pappas. 2004. Stochastic safety verification using barrier certificates. In *43rd IEEE Conference on Decision and Control, CDC 2004, Nassau, Bahamas, December 14-17, 2004*. IEEE, 929–934.

[27] Stephen Prajna and Anders Rantzer. 2005. On the necessity of barrier certificates. *IFAC Proceedings Volumes* 38, 1 (2005), 526–531.

[28] Martin L Puterman. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[29] Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, and Chuchu Fan. 2021. 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.

[30] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22, 268 (2021), 1–8. http://jmlr.org/papers/v22/20-1364.html

[31] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. PMLR, 1889–1897.

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR* abs/1707.06347 (2017).

[33] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *Nature* 550, 7676 (2017), 354.

[34] Leslie G Valiant. 1984. A theory of the learnable. *Commun. ACM* 27, 11 (1984), 1134–1142.

[35] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. 2018. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*. PMLR, 5045–5054.

[36] Jun Zeng, Bike Zhang, and Koushil Sreenath. 2020. Safety-Critical Model Predictive Control with Discrete-Time Control Barrier Function. *CoRR* abs/2007.11718 (2020).