Certifiably Robust Reinforcement Learning through Model-Based Abstract Interpretation

Chenxi Yang
UT Austin
cxyang@cs.utexas.edu

Greg Anderson*
Reed College
grega@reed.edu

Swarat Chaudhuri
UT Austin
swarat@cs.utexas.edu

***Abstract*—We present a reinforcement learning (RL) framework in which the learned policy comes with a machine-checkable *certificate of provable adversarial robustness*. Our approach, called CAROL, learns a model of the environment. In each learning iteration, it uses the current version of this model and an external *abstract interpreter* to construct a differentiable signal for provable robustness. This signal is used to guide learning, and the abstract interpretation used to construct it directly leads to the robustness certificate returned at convergence. We give a theoretical analysis that bounds the worst-case accumulative reward of CAROL. We also experimentally evaluate CAROL on four MuJoCo environments with continuous state and action spaces. On these tasks, CAROL learns policies that, when contrasted with policies from two state-of-the-art robust RL algorithms, exhibit: (i) markedly enhanced certified performance lower bounds; and (ii) comparable performance under empirical adversarial attacks.**

***Index Terms*—Certified Learning, Adversarial Robustness, Formal Verification, Abstract Interpretation, Reinforcement Learning**

I. INTRODUCTION

Reinforcement learning (RL) is an established approach to control tasks [1], [2], including safety-critical ones [3], [4]. However, state-of-the-art RL methods use neural networks as policy representations. This makes them vulnerable to adversarial attacks in which carefully crafted perturbations to a policy's inputs cause it to behave incorrectly. These problems are even more severe in RL than in supervised learning, as the effects of successive mistakes can cascade over a long time horizon.

These challenges have motivated research on RL algorithms that are robust to adversarial perturbations. In general, adversarial learning techniques can be divided into best-effort heuristic defenses and *certified* approaches that guarantee provable robustness. The latter are preferable as heuristic defenses are often defeated by counterattacks [5]. While many certified defenses are known for the supervised learning setting [6], [7], [8], extending these methods to RL has been difficult. The reason is that RL involves a black-box environment. To ensure the certified robustness of an RL policy, one needs to reason about repeated interactions between the policy, the environment, and the adversary, and there is no general approach to doing so. Existing approaches to deep certified RL typically sidestep the challenge through various simplifying assumptions, for example, that the perturbations are stochastic rather than adversarial [9], that the certificate only applies to one-shot interactions between the policy and the environment [10], [11], or that the action space is discrete [12].

In this paper, we develop a framework, called **CAROL** (**C**ertifi**A**bly **RO**bust Reinforcement **L**earning), that fills this gap in the literature. We reason about adversarial dynamics over entire episodes by learning a *model* of the environment and repeatedly composing it with the policy and the adversary. To this end, we consider a state-adversarial Markov Decision Process [11] in which the observed states are adversarially attacked states of the original environment. This threat model aligns with many existing efforts on robust RL [10], [11], [13], [12], [14] and is also important for real-world RL agents under unpredictable sensor noise. During exploration, our algorithm learns a model of the environment using an existing model-based reinforcement learning algorithm [15]. We perform *abstract interpretation* [16], [6] over compositions of the current policy and the learned environment model to estimate worst-case bounds on the agent's adversarial reward. The lower bound on the reward is then used to guide the learning.

A key benefit of our *model-based abstract interpretation* approach is that it not only computes bounds on a policy's worst-case reward but also offers a *proof* of this fact if it holds. A certificate of robustness in our framework consists of such a proof.

Our results include a theoretical analysis of our learning algorithm, which shows that our learned certificates give probabilistically sound lower bounds on the accumulative reward of any allowed adversary. We also empirically evaluate CAROL over four high-dimensional MuJoCo environments (Hopper, Walker2d, Halfcheetah, and Ant). We demonstrate that CAROL is able to successfully learn certified policies for these environments and that our strong certification requirements do not significantly compromise empirical performance.

To summarize, our main contributions are as follows:

- We offer CAROL, the first RL framework to guarantee episode-level certifiable adversarial robustness for continuous states and actions. The framework is based on a new combination of model-based learning and abstract interpretation that can be of independent interest.
- We give a rigorous theoretical analysis that establishes the (probabilistic) soundness of CAROL.

* Work completed while at UT Austin.

- We give experiments on four MuJoCo domains that establish CAROL as a new state-of-the-art for certifiably robust RL.

II. BACKGROUND

A. Markov Decision Processes (MDPs)

We start with the standard definition of an *Markov Decision Process* (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, P, \mathcal{S}_0)$. Here, $\mathcal{S}$ is a set of states, and $\mathcal{A}$ is a set of actions; for simplicity of presentation, we assume these sets to be $\mathbb{R}^k$ and $\mathbb{R}^m$ for suitable dimensionality k and m. $\mathcal{S}_0$ is a distribution of initial states; $P(s' \mid s, a)$, for $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$, is a probabilistic transition function; $r(s, a)$ for $s \in \mathcal{S}, a \in \mathcal{A}$ is a real-valued reward function. Our method assumes an additional property that is commonly satisfied in practice: that $P(s' \mid s, a)$ has the form $\mu_P(s, a) + f_P(s')$, where $f_P(s')$ is a distribution independent of (s, a) and μ_P is deterministic.

A *policy* in $\mathcal{M}$ is a distribution $\pi(a \mid s)$ with $s \in \mathcal{S}$ and $a \in \mathcal{A}$. A (finite) *trajectory* τ under π is a sequence $s_0, a_0, s_1, a_1, \ldots$ such that $s_0 \sim \mathcal{S}_0$, each $a_i \sim \pi(s_i)$, and each $s_{i+1} \sim P(s' \mid s_i, a_i)$. We denote by $R(\tau) = \sum_i r(s_i, a_i)$ the aggregate (undiscounted) reward along a trajectory τ, and by $R(\pi)$ the expected reward of trajectories unrolled under π.

B. State-Adversarial MDPs

We model adversarial dynamics using *state-adversarial MDPs* [11]. Such a structure is a pair $\mathcal{M}_\nu = (\mathcal{M}, B)$, where $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, P, \mathcal{S}_0)$ is an MDP, and $B : \mathcal{S} \to \mathcal{P}(\mathcal{S})$ is a *perturbation map*, where $\mathcal{P}(\mathcal{S})$ is the power set of $\mathcal{S}$. Intuitively, $B(s)$ is the set of all states that can result from adversarial perturbations of s.

Suppose we have a policy π in the underlying MDP $\mathcal{M}$. In an attack scenario, an adversary ν perturbs the *observations* of the agent at a state s. As a result, rather than choosing an action from $\pi(a \mid s)$, the agent now chooses an action from $\pi(a \mid \nu(s))$. However, the environment transition is still sampled from $P(s' \mid s, a)$ and *not* $P(s' \mid \nu(s), a)$, as the ground-truth state does not change under the attack. We denote by $\pi \circ \nu$ the state-action mapping that results when π is used under this attack scenario. In real world, the adversary could represent the state estimation error and the noise in the state measurement.

Naturally, if ν can arbitrarily perturb states, then adversarially robust learning is intractable. Consequently, we constrain ν using B, requiring $\nu(s) \in B(s)$ for all $s \in \mathcal{S}$. We denote the set of allowable adversaries in $\mathcal{M}_\nu$ as $\mathbb{A}_B = \{\nu : \mathcal{S} \to \mathcal{S} \mid \forall s \in \mathcal{S}.\ \nu(s) \in B(s)\}$.

C. Abstract Interpretation

We certify adversarial robustness using *abstract interpretation* [16], a classic framework for worst-case safety analysis of systems. Here, one represents sets of values — e.g., system states, actions, and reward values — using symbolic representations, or *abstractions*, in a predefined language (the *abstract domain*). For example, we can set our *abstract states* to be hyperintervals that maintain upper and lower bounds in each state space dimension. We denote abstract values with the superscript $\#$. For a set of concrete states, S, $\alpha(S)$ denotes the minimal-area abstract state which contains S. For an abstract state $s^\#$, $\beta(s^\#)$ is the set of concrete states represented by $s^\#$.

The core of abstract interpretation is the propagation of abstract states $s^\#$ through a function $f(s)$ that captures single-step system dynamics. For propagation, we assume that we have access to a map $f^\#(s^\#)$ that "lifts" f to abstract states. This function must satisfy the property $\beta(f^\#(s^\#)) \supseteq \{f(s) : s \in \beta(s^\#)\}$. Intuitively, $f^\#$ *overapproximates* the behavior of f: while the abstract state $f^\#(s^\#)$ may include some states that are not actually reachable through the application of f to states encoded by $s^\#$, it will *at least* include every state that is reachable this way. For simplicity of notation, we assume that functions over concrete states are lifted to their abstract analogs. For example, $\pi(s^\#)$ is short-hand for a function $\pi^\#(s^\#)$ satisfying $\beta(\pi^\#(s^\#)) \supseteq \{\pi(s) : s \in \beta(s^\#)\}$ for every abstract state $s^\#$. Similar functions are defined for abstract reward values, actions, and so on.

By starting with an abstraction $s_0^\#$ of the initial states and using abstract interpretation to propagate this abstract state through the transition function f, we can obtain an abstract state $s_i^\#$ which includes all states of the system that are reachable in i steps for increasing i. A sequence of abstract states $\tau^\# = s_0^\# s_1^\# s_2^\# \ldots$ is called an *abstract trace*.

D. Abstract States Propagation

Practically, we utilize the box domain [6] for all the abstract states, $s^\#$ in our work. For a program with m variables, each abstract state in the domain is represented by a m-dimensional box. Each abstract state is a pair $s^\# = (b_c, b_e)_\#$, where $b_c \in \mathbb{R}^m$ is the center of the box and $b_e \in \mathbb{R}^m_{\geq 0}$ represents the non-negative deviations. The i-th dimension of the concretization, $\beta(s^\#)$, is given by

$$[(b_c)_i - (b_e)_i, (b_c)_i + (b_e)_i].$$

Specifically, we showcase how the abstract state is propagated through programs or neural networks as below. Please refer to Appendix D for more details.

Add. For a concrete function f that replaces the i-th element in the input vector $x \in \mathbb{R}^m$ by the sum of the j-th and k-th element:

$$f(x) = (x_1, \ldots, x_{i-1}, x_j + x_k, x_{i+1}, \ldots x_m)^T.$$

The abstraction function of f is given by:

$$f^\#(s^\#) = (M \cdot b_c, M \cdot b_e)_\#,$$

where $M \in \mathbb{R}^{m \times m}$ can replace the i-th element of x by the sum of the j-th and k-th element by $M \cdot b_c$.

Matrix Multiplication. For a concrete function f that multiplies the input $x \in \mathbb{R}^m$ by a fixed matrix $M \in \mathbb{R}^{m' \times m}$:

$$f(x) = M \cdot x.$$

The abstraction function of f is given by:

$$f^\#(s^\#) = (M \cdot b_c, |M| \cdot b_e)_\#,$$

where $|M|$ is the element-wise absolute value of M. Convolutions follow the same approach, as they are also linear operations.

ReLU. For a concrete element-wise ReLU operation over $x \in \mathbb{R}^m$:

$$\mathrm{ReLU}(x) = (\max(x_1, 0), \ldots, \max(x_m, 0))^T,$$

the abstraction function of ReLU is given by:

$$\mathrm{ReLU}^{\#}(s^{\#}) = \Bigg(\frac{\mathrm{ReLU}(b_c + b_e) + \mathrm{ReLU}(b_c - b_e)}{2}, \frac{\mathrm{ReLU}(b_c + b_e) - \mathrm{ReLU}(b_c - b_e)}{2}\Bigg)_{\#}.$$

where $b_c + b_e$ and $b_c - b_e$ denotes the element-wise sum and element-wise subtraction between b_c and b_e.

III. Problem Formulation

We start by defining robustness. Assume an adversarial MDP $\mathcal{M}_\nu$, a policy π, and a threshold $\Delta > 0$. A *robustness property* is a constraint $\phi(\pi, \Delta)$ of the form $\forall \nu \in \mathbb{A}_B.\ R(\pi) - R(\pi \circ \nu) < \Delta$. Intuitively, ϕ states that no allowable adversary can reduce the expected reward of π by more than Δ.

Our goal in this paper is to learn policies that are *provably* robust. Accordingly, we expect our learning algorithm to produce, in addition to a policy π, a *certificate*, or proof, c of robustness. Formally, let Π be the universe of all policies in a given state-adversarial MDP $\mathcal{M}_\nu = (\mathcal{M}, B)$. For a policy π and a robustness property ϕ, we write $\pi \vdash_c \phi$ if π *provably* satisfies ϕ, and c is a proof of this fact.

The problem of *reinforcement learning with robustness certificates* is now defined as:

$$(\pi^*, c) = \arg\max_{\pi \in \Pi} \mathop{\mathbb{E}}_{\tau \sim (\mathcal{M}, \pi)} [R(\tau)], \text{s.t. } \ \pi^* \vdash_c \phi. \quad (1)$$

That is, we want to find a policy that maximizes the standard expected reward in RL but also ensures that the expected worst-case *adversarial* reward is provably above a threshold. We assume that a policy satisfying the constraint ϕ exists.

Our certificates can be constructed using a variety of symbolic or statistical techniques. In Carol, certificates are constructed using an abstract interpreter. Suppose we have a policy π and an abstract trace $\tau^{\#} = s_0^{\#} s_1^{\#} \ldots s_n^{\#}$ such that for all length-n trajectories $\tau = s_0 \ldots s_n$ and for all i, $s_i \in \beta(s_i^{\#})$. The abstract trace allows us to compute a lower bound on the expected reward for π and also serves as a proof of this bound. We give an example of such certification in a simple state-adversarial MDP, assumed to be available in white-box form, in Table I.

A challenge here is that abstract interpretation requires a white-box transition function, which is not available in RL. We overcome this challenge by learning a model of the environment during exploration. Model learning is a source of error, so our certificates are probabilistically sound, i.e., they guarantee robustness with high probability. However, this error only depends on the underlying model-based RL algorithm and does not restrict the adversary.

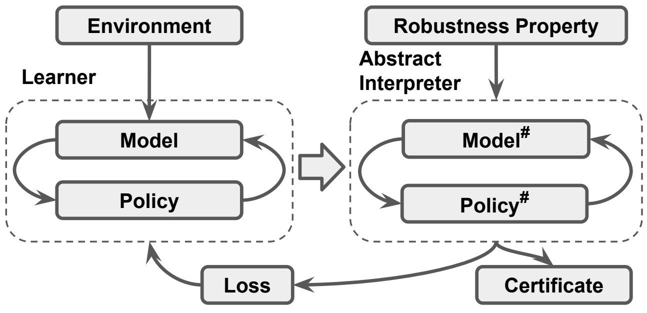


Fig. 1: Schematic of Carol

IV. Learning Algorithm

Now we present the Carol framework. The framework (Figure 1) has two key components: a model-based learner and an abstract interpreter. During each training round, the learner maintains a model of the environment dynamics and a policy. These are sent to the abstract interpreter, which calculates a lower bound on the abstract reward. The lower bound is used to compute a differentiable loss the learner uses in the next iteration of learning. At convergence, the abstract trace computed during abstract interpretation is returned as a certificate of robustness.

A. Abstract Interpretation in Carol

Now we describe the abstract interpreter in Carol in more detail. Recall that our definition of robustness compares the *expected* reward of the original policy to the *expected* reward of the policy under an adversarial perturbation. As a result, our verifier is designed to reason about the worst-case reward under adversarial perturbations, while considering average-case behavior for stochastic policies and environments. Algorithm 1 finds a lower bound on this worst-case expected reward using abstract interpretation to overapproximate the adversary's possible behaviors along with sampling to approximate the average-case behavior of the policy and environment. We denote this lower bound from Algorithm 1 as *worst-case accumulative reward* (**Wcar**), which is also used to measure the certified performance in our evaluation.

In more detail, Algorithm 1 proceeds by sampling a starting state $s_0 \sim \mathcal{S}_0$. Then in Algorithm 2 for each time step, we find an overapproximation $s^{\#}_{\mathrm{obs}_i}$ which includes all of the possible ways the adversary may perturb s_i. Based on this approximation, we *sample* a new approximation from the policy π. Intuitively, this may be done by using a policy π whose randomness does *not* depend on the current state of the system. More formally, $\pi(a \mid s) = \mu_\pi(s) + f_\pi(a)$ where $f_\pi(a)$ is a distribution with zero mean which is independent of s. Then $a_i^{\#}$ may be computed as $\mu_\pi(s^{\#}_{\mathrm{obs}_i}) + \alpha(\{e\})$ where $e \sim f_\pi(a)$. Once the abstract action is computed, we may find the new (abstract) state and reward using the environment model E. The model is assumed to satisfy a PAC-style bound, i.e., there exist δ_E and ε_E such that with probability at least $1 - \delta_E$,

$s_0 = 1.0$	$\nu(s) = \langle s_0 + \epsilon_0, s_1 + \epsilon_1 \rangle$	s_0	s_{obs0}	a_0	s_1	s_{obs1}	a_1	$R(\pi \circ \nu)$	$\mathbb{E}_{\tau \sim \pi \circ \nu}[R]$
No-Adv	$\epsilon_0 = \epsilon_1 = 0.0$	1	1	1	$2 + e$	$2 + e$	$2 + e$	$6 + 2e$	6
Adv-1	$\epsilon_0 = 0.1,$ $\epsilon_1 = -0.4$	1	1.1	1.1	$2.1 + e$	$1.7 + e$	$1.7 + e$	$5.9 + 2e$	5.9
Adv-2	$\epsilon_0 = -0.2,$ $\epsilon_1 = -0.3$	1	0.8	0.8	$1.8 + e$	$1.5 + e$	$1.5 + e$	$5.1 + 2e$	5.1
Reward Bound ($R^{\#}$)	$\epsilon_t \in [-0.5, 0.5],$ $\epsilon_t^{\#} = [-0.5, 0.5]$	1	$1 + [-0.5, 0.5]$	$[0.5, 1.5]$	$[1.5, 2.5] + e$	$[1, 3] + e$	$[1 + e,$ $3 + e]$	$[4 + 2e,$ $8 + 2e]$	$[4, 8]$

TABLE I: Example of reward bound calculation. The MDP in this example has initial state set $S_0 = [1.0, 1.0]$, white-box transition function $P(s'|s,a) = s + a + \mathcal{N}(0,1)$, reward function $r(s,a) = s + a$, and adversary $\nu(s) \in [s - 0.5, s + 0.5]$. ϵ_t denotes the disturbance added on step t. e represents the stochasticity from the transition, where $e \sim \mathcal{N}(0,1)$. We aim to certify over the worst-case accumulative reward of a deterministic policy π defined as $\pi(s) = s$. We define the *worst-case* by considering all potential adversaries while still considering the expected behavior over the stochastic environment, P. As shown in the above table, we first demonstrate three traces from fixed adversaries. In the last row, we demonstrate how we consider all the adversary behaviors through an abstract trace via abstract interpretation with intervals. The worst-case accumulative reward in this example is 4 as $\mathbb{E}_{\mathcal{N}}[e] = 0$. The abstract trace over all the adversaries in the last row is our certificate which serves as a proof that the policy satisfies our property. We want to ensure that the lower bound of the $R^{\#}$ should not be lower than a threshold. In training, we use the abstract trace to compute a loss to guide the learning process.

$\|E(s,a) - P(s,a)\| \le \varepsilon_E$. The values of δ_E and ε_E can be measured during model construction.

One way to understand Algorithm 1 is to consider pairs of abstract and concrete trajectories in which the randomness is resolved in the same way. Specifically, if $\pi(a \mid s) = \mu_\pi(s) + f_\pi(a)$ and $E(s' \mid s,a) = \mu(s,a) + f_E(s')$, the initial state s_0 combined with the sequence of values $e_i \sim f_\pi(a)$ and $e'_i \sim f_E(s')$ for $0 \le i \le T$ uniquely determine a trajectory. For a given set of values, the reward bound $\inf \beta(R^{\#}_{\min_t})$ represents the worst-case reward under any adversary *for a particular resolution of the randomness* in the environment and the policy. The outer loop of Algorithm 1 approximates the expectation over these different random values by sampling. Theorem 1 in Section V shows formally that with high probability, Algorithm 1 gives a lower bound on the true adversarial reward.

B. Learning in CAROL

Now we discuss how to learn a policy and environment model that may be proven robust by Algorithm 1. At a high level, Algorithm 3 works by introducing a symbolic loss term L_ψ^{symbolic} which measures the robustness of the policy. Because robustness is a constrained optimization problem, we use this symbolic loss with a Lagrange multiplier in an alternating gradient descent scheme to find the optimal robust policy. Formally, for a given environment model E, the inner loop in Algorithm 3 solves the optimization problem

$$\arg\min_\psi L^{\text{normal}}(\pi_\psi, \mathcal{D}_{\text{model}}) \quad \text{s.t.} \quad L^{\text{symbolic}}(\pi, E) \le \Delta$$

via the Lagrangian

$$\arg\min_\psi \max_{\lambda \ge 0} L^{\text{normal}}(\pi_\psi, \mathcal{D}_{\text{model}}) + \lambda(L^{\text{symbolic}}(\pi, E) - \Delta). \quad (2)$$

We ensure that solving this problem solves the certifiable robustness problem by enforcing the following conditions: (i) E accurately models the environment and (ii) $L^{\text{symbolic}}(\pi, E)$ measures the "provable robustness" of π. Condition (i) is handled by alternating model updates with policy updates, in the style of Dyna [17], so we will focus on condition (ii).

The computation of L^{symbolic} uses the same underlying abstract rollouts (Algorithm 2) as the verifier described in Algorithm 1. Once again, this algorithm estimates the reward achieved by a policy under worst-case adversarial perturbations but average-case policy actions and environment transitions. We then define the robustness loss as the difference between the nominal loss R^o and the *provable* lower bound on the worst-case loss $R_{\min}$. Now as long as $L^{\text{symbolic}} < \Delta$, we satisfy the definition of robustness given in Section III for that specific trace. Repeating these gradient updates gives an approximation of the average-case behavior which is considered in Algorithm 1.

V. THEORETICAL ANALYSIS

Now we explore some key theoretical properties of CAROL. Proofs are deferred to Appendix B.

Theorem 1. *Assume the environment transition distribution is $P(s' \mid s,a) = \mathcal{N}(\mu_P(s,a), \mathbf{\Sigma}_P)$ and the environment model is $E(s' \mid s,a) = \mathcal{N}(\mu_E(s,a), \mathbf{\Sigma}_E)$ with $\mathbf{\Sigma}_P, \mathbf{\Sigma}_E$ diagonal, which is standard under the model-based RL setting [15]. Further, we assume that the model satisfies a PAC-style guarantee: for any state s, action a, and $\epsilon \in \mathcal{S}$, $|(\mu_P(s,a) + \mathbf{\Sigma}_P^{1/2}\epsilon) - (\mu_E(s,a) + \mathbf{\Sigma}_E^{1/2}\epsilon)| \le \varepsilon_E$ with probability at least $1 - \delta_E$. For any policy π, let the result of Algorithm 1 be $\hat{R}^{\#}$ and let the reward of π under the optimal adversary ν^* be R. Then for any $\delta > 0$ with probability at least $1 - \delta$, we have*

$$R \ge \hat{R}^{\#} - \frac{1}{\sqrt{\delta}}\sqrt{\frac{\text{Var}\left[R^{\#}\right]}{N}} - \left(1 - (1 - \delta_E)^T\right) C.$$

where C is a remainder depending on time horizon T (see Appendix B for details of C).

Algorithm 1 Worst-Case Accumulative Reward (WCAR)

```
1: Input: policy $\pi$, model $E$
2: Output: Estimated worst case reward of $\pi$ under any adversary
3: for $t$ from 1 to $N$ do
4:     Sample an initial state $s_0 \sim \mathcal{S}_0$
5:     Get the worst case reward $R_{\min_t}$ using Algorithm 2 over horizon $T$ starting from $s_0$
6: end for
7: return $\frac{1}{N}\sum_{t=1}^{N} R_{\min_t}$
```

Algorithm 2 Worst-case rollout under adversarial perturbation

```
1: Input: Initial state $s_0$, rollout horizon $T$
2: Output: Worst-case reward of $\pi$ starting from $s_0$ over one random trajectory
3: Abstract the initial state and reward: ${s_{\text{original}}}_0^{\#} \leftarrow \alpha(\{s_0\})$,   $R^{\#}_{{\min_t}_i} \leftarrow \alpha(\{0\})$
4: for $i$ from 1 to $T$ do
5:     Abstract over possible perturbations: $s^{\#}_{\text{obs}i} \leftarrow B(s^{\#}_{\text{original}_i})$
6:     Calculate symbolic predicted actions: $a_i^{\#} \leftarrow \pi(s^{\#}_{\text{obs}i})$
7:     Calculate symbolic next-step states and rewards:
       $s^{\#}_{\text{original}_{i+1}}, r_i^{\#} \leftarrow E_\theta(s^{\#}_{\text{original}_i}, a_i^{\#}) + \alpha(\{x \mid \|x\| \leq \varepsilon_E\})$
8:     Update the estimated worst-case reward: $R^{\#}_{\min t} \leftarrow R^{\#}_{\min t} + r_i^{\#}$
9: end for
10: return $\inf \beta(R^{\#}_{\min t})$
```

Algorithm 3 Certifiably Robust Reinforcement Learning

```
1: Initialize a random policy $\pi_\psi$, random environment model $E_\theta$, and empty model dataset $\mathcal{D}_{\text{model}}$.
2: Initialize an environment dataset $\mathcal{D}_{\text{env}}$ by unrolling trajectories under a random policy.
3: for $N$ epochs do
4:     Train model $E_\theta$ on $\mathcal{D}_{\text{env}}$ via maximum likelihood
5:     Unroll $M$ trajectories in the model under $\pi_\psi$; add to $\mathcal{D}_{\text{model}}$
6:     Take action in environment according to $\pi_\psi$; add to $\mathcal{D}_{\text{env}}$
7:     for $G$ gradient updates do
8:         Calculate normal policy loss $L^{\text{normal}}(\pi_\psi, \mathcal{D}_{\text{model}})$ as in MBPO [15]
9:         Sample $\langle s_t, a_t, s_{t+1}, r_t \rangle$ uniformly from $\mathcal{D}_{\text{model}}$
10:        Rollout $\pi$ starting from $s_t$ under $E_\theta$ for $T_{\text{train}}$ steps and compute the total reward $R^o$
11:        Compute the worst-case reward $R_{\min}$ using Algorithm 2 over horizon $T_{\text{train}}$.
12:        Compute the robustness loss $L^{\text{symbolic}}(\pi_\psi, E_\theta) \leftarrow R^o - R_{\min}$
13:        Update policy parameters: $\psi \leftarrow \psi - \alpha\nabla_\psi(L^{\text{normal}}(\pi_\psi, \mathcal{D}_{\text{model}}) + \lambda(L^{\text{symbolic}}(\pi_\psi, E_\theta) - \Delta))$
14:        Update Lagrange multiplier: $\lambda \leftarrow \max(0, \lambda + \alpha'(L^{\text{symbolic}}(\pi_\psi, E_\theta) - \Delta))$
15:    end for
16:    Unroll $n$ trajectories in the true environment under $\pi_\psi$; add to $\mathcal{D}_{\text{env}}$
17: end for
```

We divide the proof into two main parts: (1) Algorithm 2 returns the lower bound of the reward. (2) In cases where Algorithm 2 does not return the lower bound of the reward, we bound the distribution shifts between the returned lower bound and the ground truth lower bound. The proof strategy involves bounding the per-step distribution shifts over the states under the assumption of the stochaticity limit, PAC-style guarantee of the learnt environment model, and the Lipschitz continuity of the environment. Subsequently, we leverage the bounded shifts over step-wise states to bound the expected reward of the abstract rollouts in Algorithm 2. Please see Appendix B for a detailed proof.

Theorem 1 shows that our checker is a valid (probabilistic) proof strategy for determining if a policy is robust. That is, if we use Algorithm 1 to measure the reward of a policy under perturbation, the result is a lower bound of the true worst-case reward (minus a constant) with high probability, assuming an accurate environment model. The bound in Theorem 1 gives some interesting insights. First, the bound grows as δ shrinks, so we pay the price of a looser bound as we consider higher confidence levels. Second, the bound depends on the variance of the abstract reward and the number of samples in an intuitive way — higher variance makes it harder to measure the true reward, and more samples make the bound tighter. Third, as

δ_E increases, the last term of the bound grows, indicating that a less accurate environment model leads to a looser bound. Finally, the bound grows with T, indicating that over longer time horizons, our reward measurement gets less accurate. This is consistent with the intuition that the environment model may drift away from the true environment over long rollouts.

Theorem 2. *Assume there exists a provably robust policy. Let π_ψ be a solution to the optimization problem defined in Equation 2. Let π be any policy that is provably robust in the sense that $L^{symbolic}(\pi, E) \leq \Delta$. Then π_ψ is provably robust and has a policy loss that does not exceed that of π (that is, $L^{normal}(\pi_\psi, \mathcal{D}_{model}) \leq L^{normal}(\pi, \mathcal{D}_{model})$.*

Proof. Note that if π_ψ is not provably robust then $L^{\text{symbolic}}(\pi_\psi, E) > \Delta$ by the soundness of abstract interpretation. By assumption there exists some policy π^* which is provably robust so that $L^{\text{symbolic}}(\pi^*, E) \leq \Delta$. Let $\ell = L^{\text{normal}}(\pi^*, E)$ be the value of the objective function in Equation 2 for π^*. Then for any

$$\lambda > \frac{\ell - L^{\text{normal}}(\pi_\psi, \mathcal{D}_{\text{model}})}{L^{\text{symbolic}}(\pi_\psi, E) - \Delta}$$

we have

$$L^{\text{normal}}(\pi_\psi, \mathcal{D}_{\text{model}}) + \lambda \left(L^{\text{symbolic}}(\pi_\psi, E) - \Delta \right) > \ell$$

so that in particular π_ψ would not be an optimum of the problem defined in Equation 2. Therefore we have that π_ψ is provably robust.

Now consider any policy π, which provably satisfies the robustness property. In this case, the optimal λ for π is 0, so the objective of the saddle point problem is just $L^{\text{normal}}(\pi, \mathcal{D}_{\text{model}})$. By assumption, $L^{\text{normal}}(\pi_\psi, \mathcal{D}_{\text{model}}) \leq L^{\text{normal}}(\pi, \mathcal{D}_{\text{model}})$ because π_ψ is a minimizer for Equation 2. □

Intuitively, Theorem 2 shows that the saddle point problem solved by Algorithm 3 also solves the certifiable robustness problem, i.e., it converges to a policy that passes the check by Algorithm 1. The primal-dual gradient descent approach outlined in Algorithm 3 is a standard technique for solving such saddle point problems [18].

VI. Evaluation

We study the following research questions for evaluation:

RQ1: Can Carol learn policies with nontrivial certified reward bounds? We assess Wcar with the associated environment model to demonstrate the certified performance.

RQ2: Do the certified bounds for Carol beat those for other (non-certified) robust RL methods? By truncating Carol and extracting the its training policies, we compare Carol against baselines in terms of Wcar.

RQ3: How does the model error, ε_E, distribute? We incorporate the model error, ε_E in Algorithm 2 and assume the PAC-style guarantee in Section V. To better exhibit the results from our main algorithms in Section IV, we empirically measure how the model error distributes.

RQ4: How tight is the certified bound? To gauge the tightness of our certified bound within limited time horizons, we compare the certified reward bound with the reward trace under empirical attacks.

RQ5: What is Carol's performance against empirical adversarial inputs? Certified defenses and heuristic defenses are distinct well-studied categories in adversarial ML. A good certified performance may sacrifice the empirical performance (reward under empirical attacks). We evaluate the reward under empirical attacks on state observations to ascertain that our certified defenses maintain high reward.

RQ6: How does the estimation of the model error, ε_E, impact the certified reward bounds? We present the certified reward bounds under the assumption of a flawless model, where $\varepsilon_E = 0.0$.

RQ7: How does the model-based training strategy of Carol influence its performance? We conduct an ablation study of the core algorithm by separating data sampling for training loss from the process of environment model learning.

A. Experiments Setup and Training Details

a) Environments and Setup: Our experiments consider l_∞-norms perturbation of the state with radius ϵ: $B_p(s, \epsilon) := \{s' \| \|s' - s\| \leq \epsilon\}$. We implement Carol on top of the MBPO [15] model-based RL algorithm using the implementation from [19]. For training, we use Interval Bound Propagation (IBP) [20] as a scalable abstract interpretation mechanism to compute the layer-wise bounds for the neural networks, where all the abstract states are represented as intervals per dimension. More details of the abstract transition are omitted to Appendix D. During the evaluation, we use CROWN [21], a more computationally expensive but tighter bound propagation method based on IBP. During training, we use a smoothed linear ϵ-schedule [20], [11] to slowly increase the ϵ_t at each epoch within the perturbation budget until reaching ϵ. Note that the policies take action stochastically during training, but we set them to be deterministic during evaluation.

We experiment on four MuJoCo environments in OpenAI Gym [22]. For Carol, we use the same hyperparameters for the base RL algorithms as in [19] without further tuning. Specifically, we do not use an ensemble of dynamics models. Instead, we use a single dynamic model, which is the case when the ensemble is of size 1. We use Gaussian distribution as the independent noise distribution, $f_\pi(a), f_E(s')$ for both policy and model in the experiments. Concretely, the output of our policies are the parameters μ_π, $\mathbf{\Sigma}_\pi$ of a Gaussian, with $\mathbf{\Sigma}_\pi$ being diagonal and independent of input state s. For the model, the output are the parameters μ_E, $\mathbf{\Sigma}_E$ of a Gaussian, with $\mathbf{\Sigma}_E$ being diagonal and independent of input s, a.

We compare Carol with the following methods:

- MBPO [15], our base RL algorithm.
- SA-PPO [11], a robust RL algorithm bounding per-step action distance.
- RADIAL-PPO [10], a robust RL algorithm using lower bound PPO loss to update the policy. In Carol, we update the policy loss L^{normal} with the data sampled from the

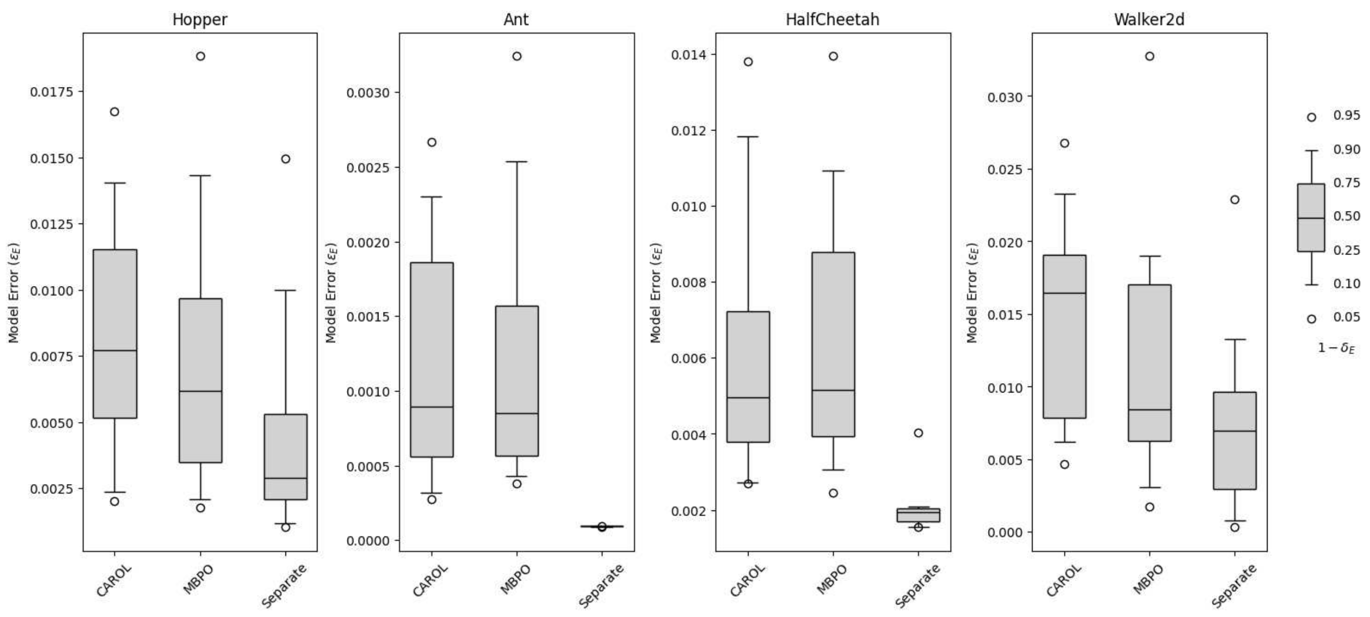


Fig. 2: Demonstration of the model error distribution. *CAROL*, *MBPO*, and *Separate* represent the distribution from the models trained with CAROL, models trained with MBPO, and the models trained from datasets from rollout with a set of random policies, respectively.

rollout between the learned model and the policy. While in CAROL-SS, the data for L^{normal} is sampled from the rollout between the environment and the policy.

- CAROL-Separate Sampler(CAROL-SS), an ablation of CAROL.

The ϵ_{train} is 0.075, 0.05, 0.075, 0.05 for Hopper, Walker2d, HalfCheetah, and Ant for CAROL, CAROL-SS, SA-PPO, and RADIAL-PPO in this section for consistency with baselines.

For both the policy networks and the model networks, we use the same network as in [19]. For both MBPO and CAROL, we use the optimal hyperparameters in [19]. We set $T_{\text{train}} = 1$ for all the training of CAROL. We mainly set two additional parameters, regularization parameters and the ϵ-schedule [11], [10], [20] parameters for CAROL. The additional regularization parameter λ to start with for regularizing L^{symbolic} is chosen in $\{0.1, 0.3, 0.5, 0.7, 1.0\}$. The ϵ-schedule starts as an exponential growth from $\epsilon = 10^{-12}$ and transitions smoothly into a linear schedule until reaching ϵ_{train}. Then the schedule keeps $\epsilon_t = \epsilon_{\text{train}}$ for the rest of iterations. We set the temperature parameter controlling the exponential growth with 4.0 for all experiments. We have two other parameters to control the ϵ-schedule: *endStep*, and *finalStep*, where *endStep* is the step where ϵ_t reaches ϵ_{train} and *finalStep* is the steps for the total training. The *midStep* $= 0.25 *$ *endStep* is the turning point from exponential growth to linear growth. Table II shows the details of each parameter.

b) Evaluation Metrics: We evaluate the performance of policies with two metrics: (i). WCAR, which was formally defined in Algorithm 1 for certified performance. (ii). total reward under MAD attacks [11] for empirical performance.

Environments	Methods	*endStep*	*finalStep*
Hopper	CAROL	4×10^5	5×10^5
	CAROL-SS	4×10^5	5×10^5
Ant	CAROL	8×10^5	9×10^5
	CAROL-SS	4×10^6	5×10^6
Walker2d	CAROL	7×10^5	7.5×10^5
	CAROL-SS	1.5×10^6	2×10^6
HalfCheetah	CAROL	7.5×10^5	8.5×10^5
	CAROL-SS	7.5×10^5	8.5×10^5

TABLE II: Parameters for ϵ-schedule.

B. RQ1: Certified Performance with Learned-together Certificate

Upon completion of training, we obtain a policy, π, and an associated environment model, E, which is trained in tandem with the policy. Then, we evaluate the WCAR following Algorithm 1 with π and E. Note that we set an $\epsilon_{\text{test}} = \frac{1}{255}$ for the evaluation of provability as certifying over long-horizon traces of neural network models tightly is a challenging task for abstract interpreters due to accumulated approximation error. The proof becomes more challenging as the horizon increases, primarily arising from the need to account for the potential adversary's behavior in the most unfavorable scenarios at each step, and the step-wise impact from the worst-case adversary accumulates. We vary the certified horizon under the ϵ_{test} to exhibit the certified performance.

Figure 3 exhibits the certified performance of CAROL. Both CAROL and MBPO are evaluated with the model trained together. We are able to train a policy with better certified accumulative reward under the worst attacks compared to the base algorithm, MBPO, which does not use the regularization L^{symbolic}. As the time horizon increases, it becomes harder to certify the accumulative reward. For example, in Ant and HalfCheetah, CAROL is not able to give a good certified performance when the horizon reaches 10 and 20 respectively because of the accumulative influence from the worst-case attack and the overapproximation from the abstract interpreter. We also highlight that Ant is a challenging task for certification due to the high-dimensional state space.

C. RQ2: Comparison of Certified Performance with Other Methods

We compare CAROL with two robust RL methods, SA-PPO [11] and RADIAL-PPO [10], which both bound the per-step performance of the policy during training. SA-PPO bounds the per-step action deviation under perturbation, and RADIAL-PPO bounds the one-step loss under perturbation. *As described in Section VI-D, to have a fair comparison of the certified performance of policies and alleviate the impact from model error bias across methods, we separately train 5 additional environment models, $\{E_i\}$, with the trajectory datasets unrolled from 5 additional random policies and the environment.* We truncate CAROL by extracting the policies from training and certifying them with these separately trained environment models. This setting is not completely in line with CAROL's learned certificate and verification (see Section VI-D) but is designed for a fair comparison across policies.

As shown in Figure 4, the CAROL's certified performance with separately trained models is slightly worse yet comparable to its performance when using learned-together certificates. Compared with non-certified RL policies, CAROL consistently exhibits better certifiable performance. It is worth noting that CAROL is able to provide worst-case rewards over time for benchmarks aligning with the reward mechanisms used in these environments. We show the abstract trace lower bound (R_{min}) sampled from trajectories in Appendix C. These results demonstrate that CAROL is able to provide reasonable certified performance, while the other methods, which are not specifically designed for worst-case accumulative reward certification, struggle to attain the same goal.

D. RQ3: Model Error in Practice

We incorporate the model error, denoted as ε_E, in Algorithm 2. Our approach adheres to Assumption 5 detailed in Appendix B for ε and δ_E. The ideal scenario involves accurately quantifying $\max_{\forall(s,a)} ||P(s,a) - E(s,a)||$. However, gauging this error across the entire (s,a) space poses significant challenges. To mitigate the gap between the theoretical maximum error and the empirical maximum error, we propose a way to estimate the model error using $\max_{(s,a=\pi(s))} ||P(s,a)-E(s,a)||$, where π represents a collection of random policies.

In our practical experiments, we assess the model error associated with three training methodologies: *CAROL*, *MBPO*, and *Separate*. Both *CAROL* and *MBPO* encapsulate environment models that are concurrently trained within their respective algorithms. In contrast, *Separate* models are derived from supervised learning, leveraging a rollout dataset from the original environment and a suite of random policies. Specifically, *Separate* is mainly used in Section VI-C for a fair comparison between model-based algorithms and model-free algorithms. All the underlying environment model architectures remain consistent with those described in [19]. Figure 2 illustrates the model error distribution across methods. In subsequent sections, Section VI-B and Section VI-C, we evaluate the certified performance of policies originating from various algorithms, utilizing the ε_E with the $1-\delta_E$ of 0.90.

E. RQ4: Qualitative Evaluation of the Abstract Trace Lower Bound

We evaluate and demonstrate the lower bound of the abstract traces over CAROL with examples in Figure 5. Specifically, we show the reward under one empirical attack (MAD) and our WCAR (incorporating the model error) over horizons starting from the same initial state. WCAR being always smaller than the reward under empirical attack indicates soundness. The reasonably small gap between the two lines indicates tightness. One interesting observation is that as the horizon increases, the gap increases. We give two possible explanations for this:

- The empirical attack is not strong enough to reveal the agents' performance under the worst-case attack.
- The overapproximation error and the model error from CAROL accumulate as the horizon increases.

Environment	Model	Nominal $\epsilon = 0$	Attack (MAD) $\epsilon = \epsilon_{\text{train}}$
Hopper ($\epsilon_{\text{train}} = 0.075$)	MBPO	3246.0±76.1	2874.2±203.4
	SA-PPO	3423.9±164.2	**3213.8±284.8**
	RADIAL-PPO	**3547.0±166.9**	3100.3±368.3
	CAROL	3290.1±104.9	3201.4±100.5
Ant ($\epsilon_{\text{train}} = 0.05$)	MBPO	4051.9±526.2	406.2±83.5
	SA-PPO	5368.8±96.4	5327.4±112.7
	RADIAL-PPO	4694.1±219.5	4478.9±232.8
	CAROL	**5696.6±277.9**	**5362.2±242.8**
HalfCheetah ($\epsilon_{\text{train}} = 0.075$)	MBPO	**7706.3±710.1**	2314.6±566.7
	SA-PPO	3193.9±650.7	3231.6±659.9
	RADIAL-PPO	3686.5±439.2	3409.6±683.9
	CAROL	5821.5±2401.9	**3961.6±899.5**
Walker2d ($\epsilon_{\text{train}} = 0.05$)	MBPO	3815.6±211.9	3616.5±228.2
	SA-PPO	**4271.7±222.2**	**4444.4±286.0**
	RADIAL-PPO	2935.1±272.1	3022.6±381.7
	CAROL	3784.4±329.1	3774.3±260.3

TABLE III: Average episodic reward ± standard deviation over 100 episodes on three baselines and CAROL. We show natural rewards (under no attack) and rewards under adversarial attacks. The best results over all methods are in bold.

F. RQ5: Comparison of Empirical Performance with Other Methods

Usually, there is a trade-off between certified robustness and empirical robustness. One can get good provability but

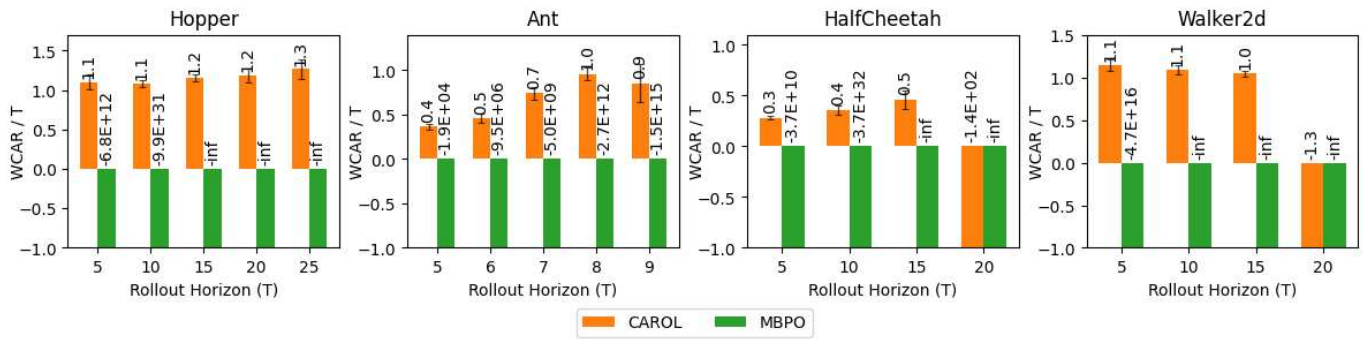


Fig. 3: Certified performance of policies π with the *learned-together* model, E. To have a fair comparison across different horizons, we quantify the certified performance by WCAR$/T$, where WCAR is formally defined in Algorithm 1 and T is the rollout horizon in Algorithm 2. Each bar is an average of 25 starting states. We use negative infinity, *-inf*, to exhibit that (π, E) is not certifiable by a third-party verifier [21]. A higher value indicates a better certified *worst-case* performance. The results are based on ε_E with a $1-\delta_E$ of 0.9.

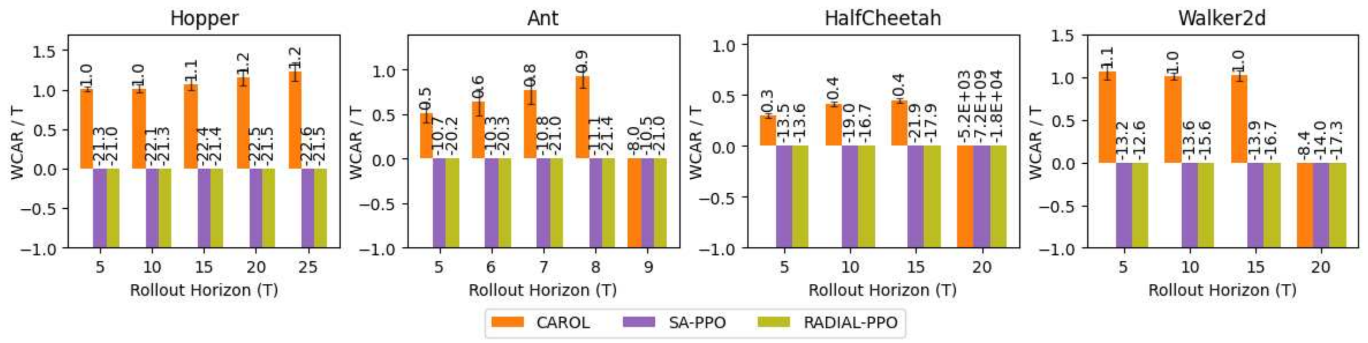


Fig. 4: Certified performance of policies π under a set of *separately learned* models, $\{E_i\}$. Each bar averages the learned policies on each E_i of 25 starting states. The results are based on ε_E with a $1-\delta_E$ of 0.9.

may sacrifice empirical rewards. We show that policy from our algorithm shows comparable natural rewards (without attack) and adversarial rewards compared with other methods. In Table III, we show results on 4 environments and comparison with MBPO, SA-PPO, and RADIAL-PPO. The policies are the same ones evaluated for Section VI-D and Section VI-B. For each environment, we compare the performance under MAD attacks [11]. CAROL outperforms other methods on Ant and HalfCheetah under attacks when the base algorithm, MBPO, is extremely not robust. For Hopper, CAROL has comparable adversarial rewards with the best methods. CAROL's reward is worse on Walker2d though still reasonable.

G. RQ6: Impact of Model Certification

We showcase the certified performance under the assumption that the ε_E is zero in Figure 6 and Figure 7. The general trend of the certified performance does not change much, while the exact WCAR $/T$ increases. Specifically, Walker2d could give reasonable certification over longer horizons.

The results exhibit that whether the environment model is trained accurately enough does not influence the gap between our certified performance with other methods.

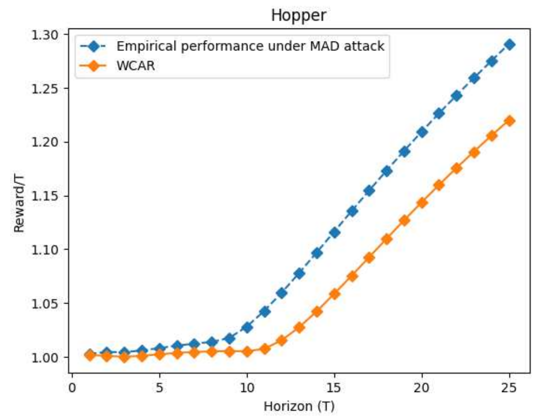
H. RQ7: Impact of Model-Based Training

In this part, we investigate the impact of our design choices for L^{normal} on performance. We compare our framework, CAROL, with an ablation of it, CAROL-SS, to understand how rollout with the learned model for L^{normal} matters in CAROL. We present a comparison of performance during training, as shown in Figure 8. In the implementation, we set a smoother ϵ-schedule for CAROL-SS by allowing CAROL-SS to take longer steps from $\epsilon = 0$ to the target ϵ. These results show that CAROL converges much faster while achieving a comparable or better final performance due to the benefits of the sample efficiency of MBRL. Additionally, the consistency between the rollout datasets for L^{normal} and the ones for L^{symbolic} also leads to a better natural reward at convergence in training.

VII. Related Work

A. Adversarial RL

Adversarial attacks on RL systems have been extensively studied. Specific attacks include adversarial perturbations on agents' observations or actions [23], [13], [24], adversarial disturbance forces to the system [25], and other adversarial policies in a multiagent setting [26]. Most recently, [27] and



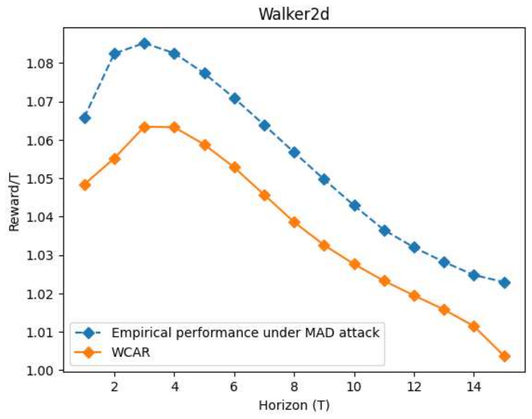


Fig. 5: Examples of robustness certification of CAROL. We show the reward under one empirical attack (MAD) and WCAR (incorporating the model error with $1 - \delta_E$ being 0.90) over horizons starting from the same initial state.

[28] consider an optimal adversary and propose methods to train agents together with a learned adversary in an online way to achieve a better adversarial reward.

B. Robust RL and Certifiable Robustness in RL

Multiple robust training methods have been applied to deep RL. Mankowitz et al. [29] explore a broader adversarial setting related to model disturbances and model uncertainty. Fischer et al. [14] leverage additional student networks to help the robust Q learning, and Everett et al. [30] enhance an agent's robustness during testing time by computing the lower bound of each action's Q value at each step. Zhang et al. [11] and Oikarinen et al. [10] leverage a bound propagation technique in a loss regularizer to encourage the agent to either follow its original actions or optimize over a loss lower bound. While these efforts achieve robustness by deterministic certification techniques for neural networks [20], [31], they mainly focus on the step-wise certification and are not able to give robustness certification if the impact from attacks accumulates across multiple steps. CAROL differs from these papers by offering certified robustness for the aggregate reward in an episode. We know of only two recent efforts that study robustness certification for cumulative rewards. The first, by Wu et al. [32], gives a framework for certification rather than certified learning. The second, by Kumar et al. [9], proposes a certified learning algorithm under the assumption that the adversarial perturbation is smoothed using random noise. The attack model here is weaker than the adversarial model assumed by CAROL and most other work on adversarial learning.

C. Certified RL

Safe control with learned certificates is an active field [33]. A few efforts in this space have considered controllers discovered through RL. Many works use a given certificate with strong control-theoretic priors to constrain the actions of an RL agent [3], [34], [35] or assume the full knowledge of the environment to yield the certificate during the training of an agent [36]. Chow et al. [37], [38] attempt to derive certificates from the structure of the constrained Markov decision process [39] for the safe control problems. Chang et al. [40] incorporate Lyapunov methods in deep RL to learn a neural Lyapunov critic function to improve the stability of an RL agent. We differ from this work by focusing on adversarial robustness rather than stability.

VIII. DISCUSSION

We have presented CAROL, the first RL framework with certifiable episode-level robustness guarantees. Our approach is based on a new combination of model-based RL and abstract interpretation. We have given a theoretical analysis to justify the approach and validated it empirically in four challenging continuous control tasks.

We present a detailed discussion about the limitations and future directions of our work below.

a) Precision of Abstract Interpretation: A key challenge in CAROL is that our abstract interpreter may not be sufficiently precise, and attempts to increase precision may compromise scalability. Future research should work to address this issue with more accurate and scalable verification techniques.

b) Adversarial Setting: We focus on the state-adversarial setting. There are broader adversarial settings related to model disturbances and model uncertainty [29]. Exploring the certified learning over environment dynamics perturbation is of interest. Specifically, we do not require a predefined model. We learn a model where the model misspecification amounts to supervised learning error. Future works would incorporate the potential disturbance of the environment in training to learn the dynamics model under CAROL framework.

c) Dimensionality: We focus on control benchmarks in this work. Related works about certification [32] use higher dimensional environments (e.g., Atari). However, the methods in CROP [32] primarily work on discrete state/action space and assume a deterministic environment. CAROL is more general; while CROP does post-hoc verification, we focus on certified learning, where verification is integrated with learning. In addition, to the best of our knowledge, the environments we evaluate over have the largest dimensionality in certified RL papers [41], [40].

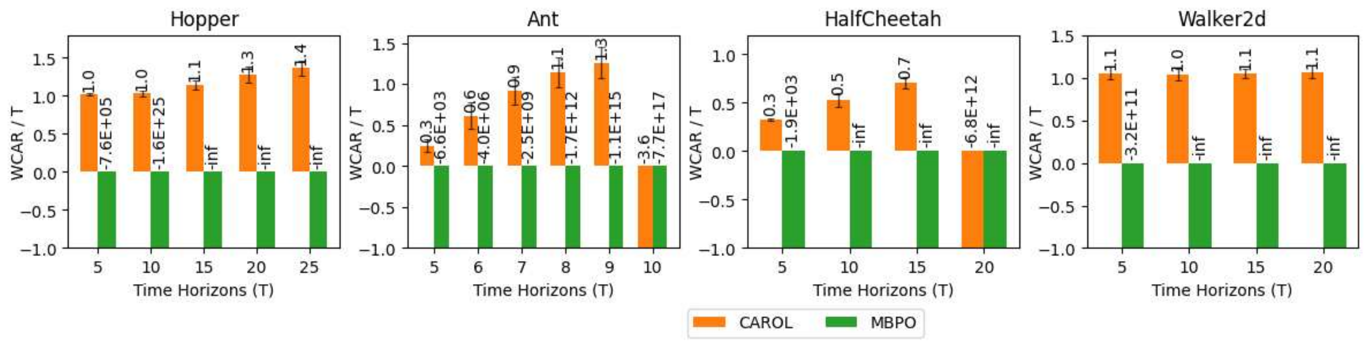


Fig. 6: Certified performance of policies π with the learned-together model, E. Each bar is an average of 25 starting states. The results are based on the assumption of ε_E being 0.0.

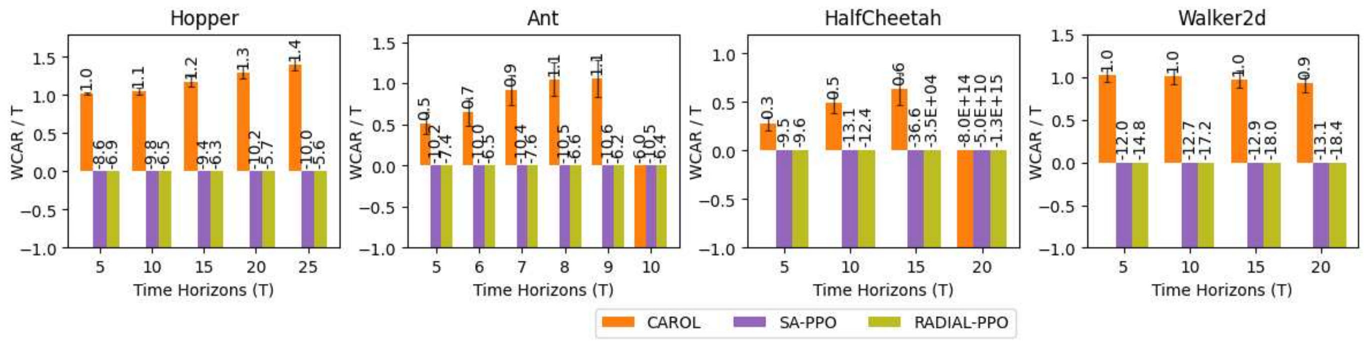


Fig. 7: Certified performance of policies π under a set of separately learned models, $\{E_i\}$. Each bar averages the learned policies on each E_i of 25 starting states. The results are based on the assumption of ε_E being 0.0.

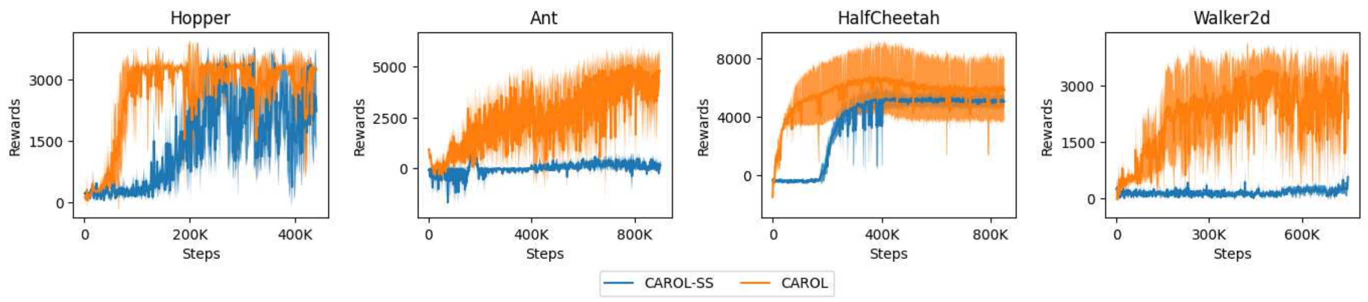


Fig. 8: Training Curves of CAROL and CAROL-SS. The solid lines in the graph show the average natural rewards of five training trials, and the shaded areas represent the standard deviation among those trials.

d) Stochasticity: Because abstract interpretation of probabilistic systems is difficult, our approach assumes that the randomness in the environment transitions is state-independent. Future work should try to eliminate this assumption through abstract interpreters tailored to probabilistic systems. In addition, we have challenges handling highly random environments, which is a fundamental limitation of all certified learning techniques. We consider the stochasticity in the theoretical analysis by incorporating the variance of $R^{\#}$in the soundness bound. When stochasticity is large, the tightness of our bound may be affected.

e) Complexity: Certified learning is more expensive than regular learning due to certifiability and soundness guarantee requirements. In training, the symbolic state is represented by the center and the width of a box (2x information representation per state). Propagating over a box needs an additional 2x computation compared to computation without symbolic states. Consequently, CAROL is approximately two times slower than the base RL algorithm per step.

In summary, CAROL 's performance is limited in the very large-scale MDPs over long rollout horizons mainly due to two reasons:

- Efficient abstract interpretation domains (e.g., Inter-

val/Box) can give accumulated over-approximation error.
- Tighter abstract interpretation domains are expensive in training and may not be differentiable. (e.g. bounded Zonotopes [42] or Polyhedra [43]).

We believe that future works on tighter, more efficient, and differentiable abstract interpretation techniques would benefit CAROL as our framework is not built on top of one particular abstract interpretation method. Additionally, a broader setting of adversarial perturbations would be an interesting future direction to extend our work.

ACKNOWLEDGEMENTS

This work was supported in part by ONR under Award No. N00014-20-1-2115, and by NSF under grants CCF-1901376 and CCF-2033851.

REFERENCES

[1] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "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, 2019, pp. 3387–3395.

[4] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[5] A. Russo and A. Proutiere, "Optimal attacks on reinforcement learning policies," *arXiv preprint arXiv:1907.13548*, 2019.

[6] M. Mirman, T. Gehr, and M. Vechev, "Differentiable abstract interpretation for provably robust neural networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3578–3586.

[7] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1310–1320.

[8] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5286–5295.

[9] A. Kumar, A. Levine, and S. Feizi, "Policy smoothing for provably robust reinforcement learning," in *International Conference on Learning Representations*, 2021.

[10] T. Oikarinen, W. Zhang, A. Megretski, L. Daniel, and T.-W. Weng, "Robust deep reinforcement learning through adversarial loss," *Advances in Neural Information Processing Systems*, vol. 34, pp. 26 156–26 167, 2021.

[11] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. Boning, and C.-J. Hsieh, "Robust deep reinforcement learning against adversarial perturbations on state observations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 024–21 037, 2020.

[12] B. Lütjens, M. Everett, and J. P. How, "Certified adversarial robustness for deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2020, pp. 1328–1337.

[13] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," *arXiv preprint arXiv:1703.06748*, 2017.

[14] M. Fischer, M. Mirman, S. Stalder, and M. Vechev, "Online robustness training for deep reinforcement learning," *arXiv preprint arXiv:1911.00887*, 2019.

[15] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[16] P. Cousot and R. Cousot, "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. Los Angeles, California: ACM Press, New York, NY, 1977, pp. 238–252.

[17] R. S. Sutton, "Integrated architecture for learning, planning, and reacting based on approximating dynamic programming," in *Proceedings of the Seventh International Conference (1990) on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, p. 216–224.

[18] Y. Nandwani, A. Pathak, Mausam, and P. Singla, "A primal dual formulation for deep learning with constraints," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/cf708fc1decf0337aded484f8f4519ae-Paper.pdf

[19] L. Pineda, B. Amos, A. Zhang, N. O. Lambert, and R. Calandra, "Mbrl-lib: A modular library for model-based reinforcement learning," *Arxiv*, 2021. [Online]. Available: https://arxiv.org/abs/2104.10159

[20] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. Mann, and P. Kohli, "On the effectiveness of interval bound propagation for training verifiably robust models," *arXiv preprint arXiv:1810.12715*, 2018.

[21] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," *Advances in neural information processing systems*, vol. 31, 2018.

[22] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[23] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[24] T.-W. Weng, K. D. Dvijotham, J. Uesato, K. Xiao, S. Gowal, R. Stanforth, and P. Kohli, "Toward evaluating robustness of deep reinforcement learning with continuous control," in *International Conference on Learning Representations*, 2019.

[25] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2817–2826.

[26] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," *arXiv preprint arXiv:1905.10615*, 2019.

[27] H. Zhang, H. Chen, D. Boning, and C.-J. Hsieh, "Robust reinforcement learning on state observations with learned optimal adversary," *arXiv preprint arXiv:2101.08452*, 2021.

[28] Y. Sun, R. Zheng, Y. Liang, and F. Huang, "Who is the strongest enemy? towards optimal and efficient evasion attacks in deep rl," *arXiv preprint arXiv:2106.05087*, 2021.

[29] D. J. Mankowitz, N. Levine, R. Jeong, Y. Shi, J. Kay, A. Abdolmaleki, J. T. Springenberg, T. Mann, T. Hester, and M. Riedmiller, "Robust reinforcement learning for continuous control with model misspecification," *arXiv preprint arXiv:1906.07516*, 2019.

[30] M. Everett, B. Lütjens, and J. P. How, "Certifiable robustness to adversarial state uncertainty in deep reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[31] K. Xu, Z. Shi, H. Zhang, M. Huang, K. Chang, B. Kailkhura, X. Lin, and C. Hsieh, "Automatic perturbation analysis on general computational graphs," Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), Tech. Rep., 2020.

[32] F. Wu, L. Li, Z. Huang, Y. Vorobeychik, D. Zhao, and B. Li, "Crop: Certifying robust policies for reinforcement learning through functional smoothing," in *International Conference on Learning Representations*, 2021.

[33] C. Dawson, S. Gao, and C. Fan, "Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods," *arXiv preprint arXiv:2202.11762*, 2022.

[34] X. Li and C. Belta, "Temporal logic guided safe reinforcement learning using control barrier functions," *arXiv preprint arXiv:1903.09885*, 2019.

[35] R. Cheng, A. Verma, G. Orosz, S. Chaudhuri, Y. Yue, and J. Burdick, "Control regularization for reduced variance reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1141–1150.

[36] C. Yang and S. Chaudhuri, “Safe neurosymbolic learning with differentiable symbolic execution,” in *International Conference on Learning Representations*, 2021.

[37] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, and M. Ghavamzadeh, “Lyapunov-based safe policy optimization for continuous control,” *arXiv preprint arXiv:1901.10031*, 2019.

[38] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, “A lyapunov-based approach to safe reinforcement learning,” *Advances in neural information processing systems*, vol. 31, 2018.

[39] E. Altman, *Constrained Markov decision processes: stochastic modeling*. Routledge, 1999.

[40] Y.-C. Chang and S. Gao, “Stabilizing neural control using self-learned almost lyapunov critics,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1803–1809.

[41] C. Dawson, Z. Qin, S. Gao, and C. Fan, “Safe nonlinear control using robust neural lyapunov-barrier functions,” in *Conference on Robot Learning*. PMLR, 2022, pp. 1724–1735.

[42] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz, “Constrained zonotopes: A new tool for set-based estimation and fault detection,” *Automatica*, vol. 69, pp. 126–136, 2016.

[43] G. Singh, M. Püschel, and M. Vechev, “Fast polyhedra abstract domain,” in *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, 2017, pp. 46–59.

APPENDIX

A. Symbols

We give a summary of the symbols used in this paper below.

Definition	Symbol/Notation
Policy	π_ψ, π
Model of the environment	E_θ, E
Environment transition	P
Parameters (mean, covariance) for Gaussian distribution for environment, model, and policy	$\mu_P, \mathbf{\Sigma}_P, \mu_E, \mathbf{\Sigma}_E, \mu_\pi, \mathbf{\Sigma}_\pi$
Distribution representing noise for environment, model, and policy	f_P, f_E, f_π
Adversary	ν
Regular policy loss	L^{normal}
Robustness loss	L^{symbolic}
Lipschitz constants for the environment model and policy mean	L_E, L_π
Model error	ε_E
PAC bound probability	δ_E
Abstract lifts	$.^{\#}$
Abstraction	α
Concretization	β
Horizon for training and testing	T, T_{train}, T_{test}
Disturbance for training and testing	ϵ, ϵ_{train}, ϵ_{test}

B. Proofs

In this section, we present proofs of the theorems from Section V.

Assumption 3. The horizon of the MDP is bounded by T.

Assumption 4. The environment transition distribution has the form $P(s' \mid s, a) = \mathcal{N}(\mu_P(s,a), \mathbf{\Sigma}_P)$ with $\mathbf{\Sigma}_P$ diagonal and the environment model E has the form $E(s' \mid s, a) = \mathcal{N}(\mu_E(s,a), \mathbf{\Sigma}_E)$ with $\mathbf{\Sigma}_E$ diagonal.

Assumption 5. There exist values ε_E and δ_E such that for all s_i, a_i from step i and for any fixed e with probability at least $1-\delta_E$, $\max_{0\le i\le T} \left\| \left(\mu_P(s_i,a_i) + \mathbf{\Sigma}_P^{1/2} e\right) - \left(\mu_E(s_i,a_i) + \mathbf{\Sigma}_E^{1/2} e\right) \right\| \le \varepsilon_E$. Further, there exists some d_E such that for all s, a, $\left\| \left(\mu_P(s,a) + \mathbf{\Sigma}_P^{1/2} e\right) - \left(\mu_E(s,a) + \mathbf{\Sigma}_E^{1/2} e\right) \right\| \le d_E$.

Assumption 6. The environment model mean function $\mu_E(s,a)$ is L_E-Lipschitz continuous, the immediate reward function $r(s,a)$ is L_r-Lipschitz continuous, and the policy mean $\mu_\pi(s)$ is L_π-Lipschitz continuous.

Assumption 7. For all $s \in \mathcal{S}$, we have $s \in B(s)$. That is, the adversary may choose not to perturb any state.

Theorem 1. *For any policy π, let the result of Algorithm 1 be $\hat{R}^{\#}$, let ν^* be the optimal adversary (i.e., for all $\nu \in \mathbb{A}_B$, $R(\pi\circ\nu^*) \le R(\pi\circ\nu)$), and let the reward of $\pi\circ\nu^*$ be R. Then for any $\delta > 0$ with probability at least $1-\delta$, we have*

$$R \ge \hat{R}^{\#}(\tau) - \frac{1}{\sqrt{\delta}}\sqrt{\frac{\operatorname{Var}\left[R^{\#}\right]}{N}} - \left(1 - (1-\delta_E)^T\right) L_r(1+L_\pi) d_E \frac{(L_E L_\pi)^T + (1 - L_E L_\pi) T - 1}{(1 - L_E L_\pi)^2}.$$

Proof. Recall that the environment transition P and policy π are assumed to be separable, i.e., $P(s' \mid s,a) = \mu_P(s,a) + f_P(s')$ and $\pi(a \mid s) = \mu_\pi(s) + f_\pi(a)$ with μ_P and μ_π deterministic. As a result, a trajectory under policy $\pi\circ\nu^*$ may be written $\tau = s_0, a_0, s_1, a_1, \ldots, s_n, a_n$ where $s_0 \sim \mathcal{S}_0$, each $a_i = \mu_\pi(\nu^*(s_i)) + e_i^\pi$ for $e_i^\pi \sim f_\pi(a)$, and each $s_i = \mu_P(s_{i-1}, a_{i-1}) + e_i^P$ for $e_i^P \sim f_P(s')$. By Assumption 4, we know that $e_i^P \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_P)$ so that $e_i^P = \mathbf{\Sigma}_P^{1/2} e_i$ where $e_i \sim \mathcal{N}(0, \boldsymbol{I})$. In particular, because each trajectory τ is uniquely determined by $s_0, \{e_i^\pi\}_{i=0}^n, \{e_i\}_{i=1}^n$, we can write the reward of $\pi\circ\nu^*$ as

$$R(\pi\circ\nu^*) = \mathop{\mathbb{E}}_{s_0\sim\mathcal{S}_0, \left\{e_i^\pi \sim f_\pi(a)\right\}_{i=0}^n, \left\{e_i \sim \mathcal{N}(\mathbf{0},\boldsymbol{I})\right\}_{i=1}^n} R(\tau)$$

Because this expectation ranges over the values of $s_0, \{e_i^\pi\}_{i=0}^n, \{e_i\}_{i=1}^n$, we will proceed by considering pairs of abstract and concrete trajectories unrolled with the same starting state and noise terms.

To do this, we analyze Algorithm 2 for some fixed $s_0, \{e_i^\pi\}_{i=0}^n, \{e_i\}_{i=1}^n$. Let $e_i^E = \mathbf{\Sigma}_E^{1/2} e_i$. That is, given the same underlying sample from $\mathcal{N}(0, \boldsymbol{I})$, e_i^P is the noise in the true environment while e_i^E is the noise in the modeled environment. We show by induction that for all i, $s_i \in \beta\left(s^{\#}_{\text{original}_i}\right)$ with probability at least $(1-\delta_E)^i$. Note that, because abstract interpretation is sound, $s_0 \in \beta\left(s^{\#}_{\text{original}_0}\right)$. Additionally, for all i if $s_i \in \beta\left(s^{\#}_{\text{original}_i}\right)$ then $\nu^*(s_i) \in \beta\left(s^{\#}_{\text{obs}_i}\right)$. Moreover, since e_i^π is fixed, we have

$$\pi\left(s^{\#}_{\text{obs}_i}\right) = \mu_\pi\left(s^{\#}_{\text{obs}_i}\right) + e_i^\pi$$

so that $\pi\left(\nu^*(s)\right) \in \beta\left(a_i^{\#}\right)$. Similarly, because e_i^E is fixed, let $\Delta_E = \alpha\left(\{x \mid \|x\| \leq \varepsilon_E\}\right)$ and we have

$$E\left(s^{\#}_{\text{original}_i}, a_i^{\#}\right) + \Delta_E = \mu_E\left(s^{\#}_{\text{original}_i}, a_i^{\#}\right) + e_i^E + \Delta_E.$$

By the induction hypothesis, we know that $s_{i-1} \in \beta\left(s^{\#}_{\text{original}_{i-1}}\right)$ with probability at least $(1-\delta_E)^{i-1}$ and therefore $a_{i-1} \in \beta\left(a_{i-1}^{\#}\right)$. By Assumption 5, we have that $\left\|\left(\mu_P(s,a) + e_i^P\right) - \left(\mu_E(s,a) + e_i^E\right)\right\| < \varepsilon_E$ with probability at least $1-\delta_E$. In particular, $\left(\mu_P(s,a) + \varepsilon_i^P\right) - \left(\mu_E(s,a) + \varepsilon_i^E\right) \in \Delta_E$, so that $\mu_P(s,a) + e_i^P \in \beta\left(E\left(s^{\#}_{\text{original}_i}, a_i^{\#}\right) + \Delta_E\right)$. Then with probability at least $1-\delta_E$, if $s_{i-1} \in \beta\left(s^{\#}_{\text{original}_{i-1}}\right)$ then $s_i \in \beta\left(s^{\#}_{\text{original}_i}\right)$. As a result, $s_i \in \beta\left(s^{\#}_{\text{original}_i}\right)$ with probability at least $(1-\delta_E)^i$. In particular, by Assumption 3, $n \leq T$ so that for a fixed τ defined by $s_0, \{e_i^\pi\}_{i=0}^n, \{e_i\}_{i=1}^n$, we have that with probability at least $(1-\delta_E)^T$, Algorithm 2 returns a lower bound on $R(\tau)$.

Now we consider the case where Algorithm 2 does *not* return a lower bound of $R(\tau)$. In this case, we show (again by induction) that for all $0 \leq i \leq T$, there exists a point $s_i' \in \beta\left(s^{\#}_{\text{original}_i}\right)$ such that

$$\|s_i - s_i'\| \leq \sum_{j=0}^{i-1} (L_E L_\pi)^j d_E = d_E\left(\frac{1-(L_E L_\pi)^{i-1}}{1-L_E L_\pi}\right)$$

(when $\sum_{j=0}^{-1}(L_E L_\pi)^j d_E$ is taken to be zero). First, note that $s_0 \in \beta\left(s^{\#}_{\text{original}_0}\right)$, so the base case is trivially true. Now by the induction hypothesis we have that there exists some $s_{i-1}' \in \beta\left(s^{\#}_{\text{original}_{i-1}}\right)$ with $\left\|s_{i-1} - s_{i-1}'\right\| \leq \sum_{j=0}^{i-2}(L_E L_\pi)^j d_E$. Notice that by Assumption 7, we also have $s_{i-1}' \in \beta\left(s^{\#}_{\text{obs}_{i-1}}\right)$. Now because abstract interpretation is sound, we have that $\mu_\pi\left(s_{i-1}'\right) + e_{i-1}^\pi \in \beta\left(a_{i-1}^{\#}\right)$ and by Assumption 6, $\|\mu_\pi(s_{i-1}) - \mu_\pi\left(s_{i-1}'\right)\| \leq L_\pi \sum_{j=0}^{i-2}(L_E L_\pi)^j d_E$. Similarly, we have $\mu_E\left(s_{i-1}', \mu_\pi\left(s_{i-1}'\right) + e_{i-1}^\pi\right) + e_i^E \in \beta\left(s^{\#}_{\text{original}_i}\right)$, and $\left\|\mu_E\left(s_{i-1}, \mu_\pi(s_{i-1}) + e_{i-1}^\pi\right) - \mu_E\left(s_{i-1}', \mu_\pi\left(s_{i-1}'\right) + e_{i-1}^\pi\right)\right\| \leq L_E L_\pi \sum_{j=0}^{i-2}(L_E L_\pi)^j d_E$. Let $\hat{s}_i = \mu_E\left(s_{i-1}, \mu_\pi(s_{i-1}) + \varepsilon_{i-1}^\pi\right) + \varepsilon_i^E$. Then by Assumption 5, we have $\|\hat{s}_i - s_i\| \leq d_E$, so that in particular $\left\|s_i - \mu_E\left(s_{i-1}', \mu_\pi\left(s_{i-1}'\right) + e_{i-1}^\pi\right) + \varepsilon_i^E\right\| \leq d_E + L_E L_\pi \sum_{j=0}^{i-2}(L_E L_\pi)^j d_E$. Letting $s_i' = \mu_E\left(s_{i-1}', \mu_\pi\left(s_{i-1}'\right) + e_{i-1}^\pi\right) + \varepsilon_i^P$, we have the desired result.

We use this result to bound the difference in reward between the abstract and concrete rollouts when Algorithm 2 does not return a lower bound. For each i, because $s_i' \in \beta\left(s^{\#}_{\text{original}_i}\right)$ and $\mu_\pi(s_i') + e_i^\pi \in a_i^{\#}$, we define $r_i' = r\left(s_i', \mu_\pi(s_i') + e_i^\pi\right)$ and we know that $r' \in r_i^{\#}$. Because $\|s_i - s_i'\| \leq d_E\left(\frac{1-(L_E L_\pi)^{i-1}}{1-L_E L_\pi}\right)$ we have $\|a_i - a_i'\| \leq L_\pi d_E\left(\frac{1-(L_E L_\pi)^{i-1}}{1-L_E L_\pi}\right)$ and $|r(s_i, a_i) - r_i'| \leq L_r(1+L_\pi)d_E\left(\frac{1-(L_E L_\pi)^{i-1}}{1-L_E L_\pi}\right)$. In particular, let $R' = \sum_i r_i'$ and then

$$\begin{aligned}|R(\tau) - R'| &\leq \sum_{i=1}^{T} L_r(1+L_\pi)d_E\left(\frac{1-(L_E L_\pi)^{i-1}}{1-L_E L_\pi}\right)\\ &= L_r(1+L_\pi)d_E\frac{(L_E L_\pi)^T + (1-L_E L_\pi)T - 1}{(1-L_E L_\pi)^2}.\end{aligned}$$

We now combine these two cases to bound the expected difference between the reward returned by Algorithm 2, denoted $R^{\#}(\tau)$, and the reward of τ. Let $D = R^{\#}(\tau) - R(\tau)$ be a random variable representing this difference. Then with probability at least $(1-\delta_E)^T$, $D \leq 0$ and in all other cases (i.e., with probability no greater than $1-(1-\delta_E)^T$), $D \leq L_R(1+L_\pi)d_E\frac{(L_E L_\pi)^T + (1-L_E L_\pi)T - 1}{(1-L_E L_\pi)^2}$. In particular then,

$$\mathbb{E}[D] \leq \left(1-(1-\delta_E)^T\right) L_r(1+L_\pi)d_E\frac{(L_E L_\pi)^T + (1-L_E L_\pi)T - 1}{(1-L_E L_\pi)^2}.$$

By definition $\mathbb{E}\left[R^{\#}(\tau)\right] = \mathbb{E}[R(\tau)] + \mathbb{E}[D]$. Therefore, we have

$$\begin{aligned}\mathbb{E}[R(\tau)] = \mathbb{E}\left[R^{\#}(\tau)\right] - \mathbb{E}[D] &\geq \mathbb{E}\left[R^{\#}(\tau)\right]\\ &\quad - \left(1-(1-\delta_E)^T\right) L_r(1+L_\pi)d_E\frac{(L_E L_\pi)^T + (1-L_E L_\pi)T - 1}{(1-L_E L_\pi)^2}.\end{aligned} \tag{3}$$

Algorithm 3 approximates $\mathbb{E}\left[R^{\#}\left(\tau\right)\right]$ by sampling N values. Let $\hat{R}^{\#}\left(\tau\right)$ be the measured mean and recall $\mathbb{E}\left[\hat{R}^{\#}\left(\tau\right)\right]=\mathbb{E}\left[R^{\#}\left(\tau\right)\right]$ and $\mathrm{Var}\left[\hat{R}^{\#}\left(\tau\right)\right]=\mathrm{Var}\left[R^{\#}\left(\tau\right)\right]/N$. Then by Chebyshev's inequality we have the for all $k>0$, $\Pr\left[\left|\hat{R}^{\#}\left(\tau\right)-\mathbb{E}\left[R^{\#}\left(\tau\right)\right]\right|\geq k\sqrt{\mathrm{Var}\left[\hat{R}^{\#}\left(\tau\right)\right]}\right]\leq 1/k^2$. Then in particular, with probability at least $1-1/k^2$,

$$\hat{R}^{\#}\left(\tau\right)-k\sqrt{\frac{\mathrm{Var}\left[R^{\#}\left(\tau\right)\right]}{N}}\leq R^{\#}\left(\tau\right).$$

Combining this with Equation 3 above and letting $k=1/\sqrt{\delta}$, we have with probability at least $1-\delta$,

$$\mathbb{E}\left[R\left(\tau\right)\right]\geq\hat{R}^{\#}\left(\tau\right)-\frac{1}{\sqrt{\delta}}\sqrt{\frac{\mathrm{Var}\left[R^{\#}\left(\tau\right)\right]}{N}}$$
$$-\left(1-\left(1-\delta_E\right)^T\right)L_r(1+L_\pi)d_E\frac{\left(L_EL_\pi\right)^T+\left(1-L_EL_\pi\right)T-1}{\left(1-L_EL_\pi\right)^2}.$$

□

While this paper focuses on continuous state and action spaces, we can extend our main theoretical result to discrete state and action spaces if the environment is deterministic. For this analysis, we maintain Assumptions 3 and 7, but we add a few new assumptions for the discrete setting.

Assumption 8. The environment model E is deterministic and $E(s,a)=P(s,a)$ with probability at least $1-\delta_E$.

Assumption 9. The single-step reward for any state s and action a is bounded by $r_{\min}\leq r(s,a)\leq r_{\max}$.

Theorem 10. *For a deterministic policy π, let the result of Algorithm 1 be $\hat{R}^{\#}$, let ν^* be the optimal adversary, and let the reward of $\pi\circ\nu^*$ be R. Then for any δ, with probability at least $1-\delta$,*

$$R\geq\hat{R}^{\#}-\frac{1}{\sqrt{\delta}}\sqrt{\frac{\mathrm{Var}[R^{\#}]}{N}}-\left(1-\left(1-\delta_E\right)^T\right)T(r_{\max}-r_{\min}).$$

Proof. Consider a trajectory $\tau=s_0,a_0,s_1,a_1,\ldots,s_n,a_n$ where $s_0\sim\mathcal{S}_0$, each $a_i=\pi(s_i)$, and each $s_{i+1}=P(s_i,a_i)$. Note that because the dynamics of the environment and the policy are deterministic, the only randomness in the trajectory comes from sampling the initial state. Then $R(\pi\circ\nu^*)=\mathbb{E}_{s_0\sim\mathcal{S}_0}R(\tau)$. Similar to the proof of Theorem 1, we proceed by considering pairs of abstract and concrete trajectories unrolled from the same starting state.

We show by induction that for all i, $s_i\in\beta\left(s^{\#}_{\text{original}_i}\right)$ with probability at least $\left(1-\delta_E\right)^i$. For the base case, note that because abstract interpretation is sound, $s_0\in\beta\left(s^{\#}_{\text{original}_0}\right)$. Additionally, for all i, if $s_i\in\beta\left(s^{\#}_{\text{original}_i}\right)$ then $\nu^*(s_i)\in\beta\left(s^{\#}_{\text{obs}_i}\right)$ and $\pi(\nu^*(s_i))\in\beta\left(a^{\#}_i\right)$. From the induction hypothesis, we have $s_{i-1}\in\beta\left(s^{\#}_{\text{original}_{i-1}}\right)$ with probability at least $\left(1-\delta_E\right)^{i-1}$. By Assumption 8, we have $s_{i+1}=E(s_i,a_i)$ with probability at least $1-\delta_E$. Thus if $s_{i-1}\in\beta\left(s^{\#}_{\text{original}_{i-1}}\right)$ then with probability at least $1-\delta_E$ we know $s_i\in\beta\left(s^{\#}_{\text{original}_i}\right)$. Combined with the induction hypothesis, this implies that $s_i\in\beta\left(s^{\#}_{\text{original}_i}\right)$ with probability at least $\left(1-\delta_E\right)^i$. Now by Assumption 3, $n\leq T$ so that for a fixed τ from a starting state s_0, we have that with probability at least $\left(1-\delta_E\right)^T$, Algorithm 1 returns a lower bound on $R(\tau)$.

As in the proof of Theorem 1, we now turn to the case where Algorithm 1 does not return a lower bound of $R(\tau)$. In this case, let r_i be the true adversarial reward at time step i. Then by Assumption 9, we have $r_i\geq r_{\min}$, and $\inf\beta\left(r^{\#}_i\right)\leq r_{\max}$. Thus in particular, letting $R^{\#}(\tau)$ represent the bound returned by Algorithm 1, we have $R^{\#}(\tau)-R(\tau)\leq T(r_{\max}-r_{\min})$.

Now letting $D=R^{\#}(\tau)-R(\tau)$, we have that with probability at least $\left(1-\delta_E\right)^T$, $D\leq 0$ and in all other cases $D\leq T(r_{\max}-r_{\min})$. In particular,

$$\mathbb{E}[D]\leq\left(1-\left(1-\delta_E\right)^T\right)T\left(r_{\max}-r_{\min}\right).$$

By definition, $\mathbb{E}[R^{\#}(\tau)]=\mathbb{E}[R(\tau)]+\mathbb{E}[D]$ so that

$$\mathbb{E}[R(\tau)]=\mathbb{E}[R^{\#}(\tau)]-\left(1-\left(1-\delta_E\right)^T\right)T(r_{\max}-r_{\min}).$$

Following the same sampling argument we make in the proof of Theorem 1, we have that for any δ, with probability at least $1-\delta$,

$$\mathbb{E}[R(\tau)]\geq\hat{R}^{\#}(\tau)-\frac{1}{\sqrt{\delta}}\sqrt{\frac{\mathrm{Var}[R^{\#}(\tau)]}{N}}-\left(1-\left(1-\delta_E\right)^T\right)T(r_{\max}-r_{\min}).$$

□

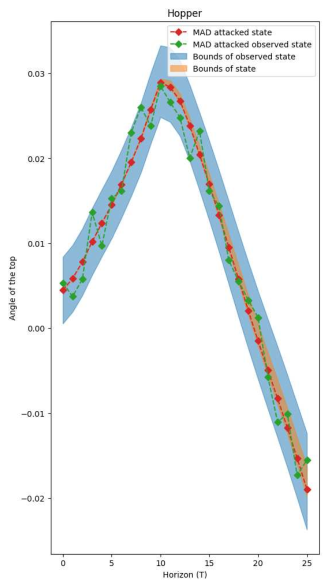


Fig. 9: Physical Meaning Demonstration. Example of the bound of the state and the observation of the state. We select the second dimension of the Hopper states, representing *the angle of the top* of the Hopper agent. We extract the lower bound and upper bound of the state value and the observed state value during the reasoning of WCAR of Hopper. For one time step, i, the orange area represents the interval bound of s_i and the blue area represents s_{obs_i}. The red trajectory shows one example of the state after the MAD attack. The green trajectory exhibits the observed state trajectory. Our certificate aims to bound the state trajectories from all attacks within our allowed adversary set.

C. Physical Meaning of the Attack and Certification bound

We demonstrate the physical meaning of the attack and certification bound in Figure 9.

D. Abstract Bound Propagation

Now, we give an explanation of how interval bound propagation (IBP) works. CROWN [21] optimizes over IBP for tighter bound (specifically for Relu and sigmoid, etc.). IBP considers the box domain in the implementation. For a program with m variables, each component in the domain represents a m-dimensional box. Each component of the domain is a pair $s^{\#} = \langle b_c, b_e \rangle$, where $b_c \in \mathbb{R}^m$ is the center of the box and $b_e \in \mathbb{R}^m_{\geq 0}$ represents the non-negative deviations. The interval concretization of the i-th dimension variable of $s^{\#}$ is given by

$$[(b_c)_i - (b_e)_i, (b_c)_i + (b_e)_i].$$

Now we give the abstract update for the box domain following [6].

a) Add.: For a concrete function f that replaces the i-th element in the input vector $x \in \mathbb{R}^m$ by the sum of the j-th and k-th element:

$$f(x) = (x_1, \ldots, x_{i-1}, x_j + x_k, x_{i+1}, \ldots x_m)^T.$$

The abstraction function of f is given by:

$$f^{\#}(s^{\#}) = \langle M \cdot b_c, M \cdot b_e \rangle,$$

where $M \in \mathbb{R}^{m \times m}$ can replace the i-th element of x by the sum of the j-th and k-th element by $M \cdot b_c$.

b) Multiplication.: For a concrete function f that multiplies the i-th element in the input vector $x \in \mathbb{R}^m$ by a constant w:

$$f(x) = (x_1, \ldots, x_{i-1}, w \cdot x_i, x_{i+1}, \ldots, x_m)^T.$$

The abstraction function of f is given by:

$$f^{\#}(s^{\#}) = \langle M_w \cdot b_c, M_{|w|} \cdot b_e \rangle,$$

where $M_w \cdot b_c$ multiplies the i-th element of b_c by w and $M_{|w|} \cdot b_e$ multiplies the i-th element of b_e with $|w|$.

c) Matrix Multiplication.: For a concrete function f that multiplies the input $x \in \mathbb{R}^m$ by a fixed matrix $M \in \mathbb{R}^{m' \times m}$:

$$f(x) = M \cdot x.$$

The abstraction function of f is given by:

$$f^{\#}(s^{\#}) = \langle M \cdot b_c, |M| \cdot b_e \rangle,$$

where M is an element-wise absolute value operation. Convolutions follow the same approach, as they are also linear operations.

d) ReLU.: For a concrete element-wise ReLU operation over $x \in \mathbb{R}^m$:

$$\text{ReLU}(x) = (\max(x_1, 0), \ldots, \max(x_m, 0))^T,$$

the abstraction function of ReLU is given by:

$$\text{ReLU}^{\#}(s^{\#}) = \langle \frac{\text{ReLU}(b_c + b_e) + \text{ReLU}(b_c - b_e)}{2}, \frac{\text{ReLU}(b_c + b_e) - \text{ReLU}(b_c - b_e)}{2} \rangle.$$

where $b_c + b_e$ and $b_c - b_e$ denotes the element-wise sum and element-wise subtraction between b_c and b_e.

e) Sigmoid.: As Sigmoid and ReLU are both monotonic functions, the abstraction functions follow the same approach. For a concrete element-wise Sigmoid operation over $x \in \mathbb{R}^m$:

$$\text{Sigmoid}(x) = (\frac{1}{1 + \exp(-x_1)}, \ldots, \frac{1}{1 + \exp(-x_m)})^T,$$

the abstraction function of Sigmoid is given by:

$$\text{Sigmoid}^{\#}(s^{\#}) = \langle \frac{\text{Sigmoid}(b_c + b_e) + \text{Sigmoid}(b_c - b_e)}{2}, \frac{\text{Sigmoid}(b_c + b_e) - \text{Sigmoid}(b_c - b_e)}{2} \rangle.$$

where $b_c + b_e$ and $b_c - b_e$ denotes the element-wise sum and element-wise subtraction between b_c and b_e. All the above abstract updates can be easily differentiable and parallelized on the GPU.

E. Certified Performance When Evaluating against Various Perturbation Range

We show the provability results with ϵ_{test} being 0.075 in Figure 10 and Figure 11.

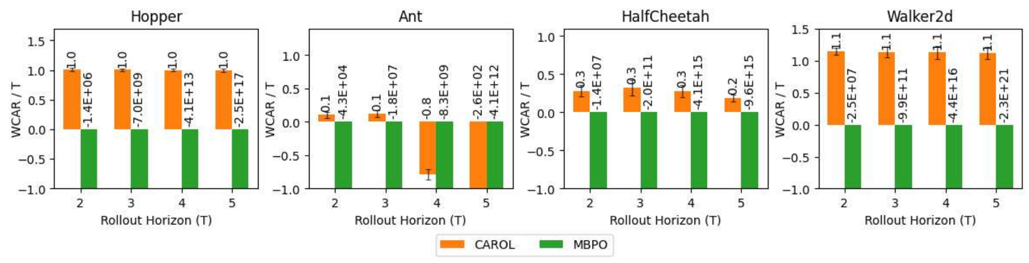


Fig. 10: Certified performance of policies π with the learned-together model, E. The results are based on ε_E with a $1 - \delta_E$ of 0.9.

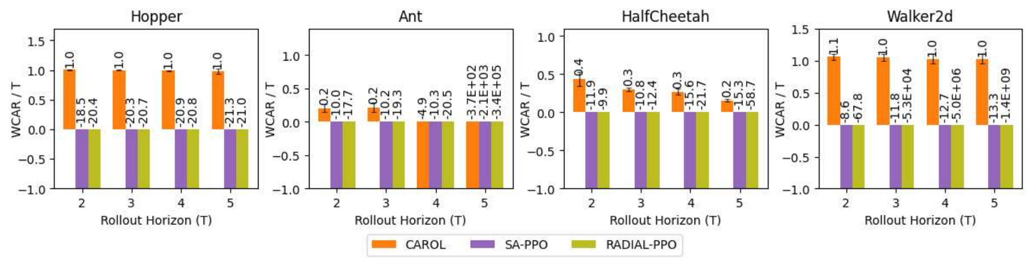


Fig. 11: Certified performance of policies π under a set of separately learned models, $\{E_i\}$. The results are based on ε_E with a $1 - \delta_E$ of 0.9.

We show the provability results with ϵ_{test} being 0.001 in Figure 12.

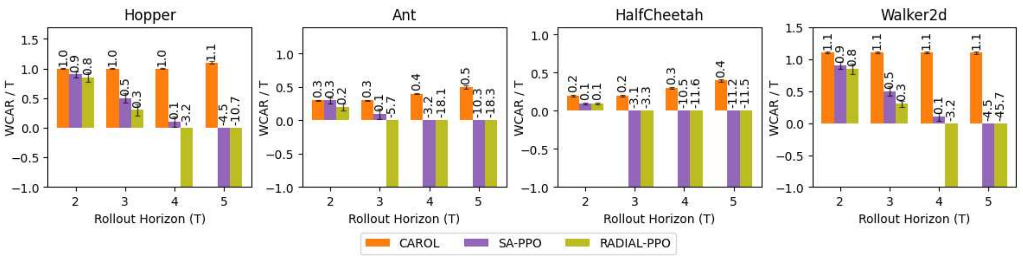


Fig. 12: Certified performance of policies π under a set of separately learned models, $\{E_i\}$. The results are based on ε_E with a $1-\delta_E$ of 0.9.