Marvel: Verification-based Safe Control Policy Synthesis

We present Marvel, a verification-based reinforcement learning framework that synthesizes safe control policies for environments with continuous state and action space. The key idea is the integration of program reasoning techniques into reinforcement learning training loops. Marvel performs abstraction based program verification to reason about a policy and its environment as a closed-loop system. Based on a novel counterexample guided inductive synthesis loop for training, Marvel reduces the amount of safety violation in the system abstraction, which approximates the safety loss of the *worst-case* inputs, using gradient-descent style optimization. When an environment is unknown or its complexity does not allow scalable verification, Marvel accurately approximates it with a lightweight model to support efficient verification and synthesis with statistical guarantees. Experimental results over state-of-the-art verified reinforcement learning benchmarks demonstrate the substantial benefits of leveraging verification feedback for control policy safe-by-construction.

ACM Reference Format:
. 2021. Marvel: Verification-based Safe Control Policy Synthesis. *Proc. ACM Program. Lang.* 1, Manuscript, Article 1 (April 2021), 33 pages.

1 INTRODUCTION

Policy search is commonly used to govern cyber-physical systems such as autonomous vehicles, where high assurance is particularly important. Reinforcement Learning (RL) is a promising approach for policy search [François-Lavet et al. 2018]. State-of-the-art RL algorithms can learn motor skills autonomously through trial and error in simulated or even unknown environments, thus avoiding tedious manual engineering. However, the most performant policies may still be unsafe since the RL algorithms do not provide any formal guarantees on safety. A learned policy may fail occasionally but catastrophically, and debugging these failures can be challenging [Uesato et al. 2019]. For example, a self-driving car navigates to a target region, while avoiding a trap zone, may still enter unsafe corner regions irregularly after training.

Guaranteeing the safety of a reinforcement learning policy is therefore important. Since a control policy is just a program, principally it can be verified or even synthesized using program verification and program synthesis. Indeed, the use of automated program reasoning techniques to aid the design of reliable machine learning systems has risen rapidly over the last few years. A notable example is the application of abstract interpretation [Cousot and Cousot 1977] to verify robustness of convolutional neural networks [Gehr et al. 2018]. For supervised learning, the robustness property of a neural network requires that its outputs be consistent for narrow input spaces surrounding individual data points. A natural extended question is that can we use program verification and synthesis to address safe reinforcement learning where a system includes both an environment and a policy? Moreover, in case verification fails, can we exploit verification counterexamples to synthesize a safe policy? These are challenging problems because safety properties must be enforced in a much wider space including all possible initial states of the system.

For reinforcement learning in an unknown environment, the primary barrier of safety verification is that we have no clue about the state transition probability distribution of the environment. The absence of environment models prevents verifying an environment and a learned policy as a combined closed-loop system. Even in a simulated environment with a known state transition model, safety verification is still challenging because the model typically involves complex nonlinear behaviors that are difficult to estimate or characterize. State space over-approximation may quickly

Author's address:

2021. 2475-1421/2021/4-ART1 $15.00
https://doi.org/

incur large error which results in abstraction that is too conservative and cannot be used for safety verification [Dutta et al. 2019; Fan et al. 2020; Tran et al. 2020]. When verification fails, these approaches provide little insight on how to effectively leverage counterexamples to improve safety. To address the large approximation error, one could use statistical model checking to sample rollouts and evaluate the safety property with statistical confidence [Zarei et al. 2020]. The question remains of how the verification results might help further optimize the policy if the confidence level is not desirable.

To overcome these difficulties, we present Marvel as a new verification-based policy synthesis framework, depicted in Fig. 1. It consists of three main components, namely *environment model inference*, *policy verification*, and *policy synthesis*.

- **Environment Model Inference**. For an unknown environment or a simulated environment whose state transition model is, however, out of the reach of state-of-the-art verification techniques, Marvel accurately approximates the environment using a lightweight *time-varying linear Gaussian* state transition model with statistical guarantees. The model is tractable for both learning and verification. This environment inference component is optional since Marvel also supports safety verification based on user-provided (nonlinear) environment models.
- **Policy Verification**. Given an environment model and a policy, Marvel verifies the safety of the policy by abstract interpretation over a closed-loop system that combines the environment model and the control policy.
- **Policy Synthesis**. A safety counterexample detected by Marvel is a *symbolic rollout* of abstract states of the closed-loop system because it is obtained by abstract interpretation. Marvel quantifies the safety properties violation by the abstract states. The goal of policy synthesis is to effectively reduce the amount of safety violation to zero.

Based on this discussion, the main technical contributions of this paper are twofold: First, when an environment is unknown or its complexity does not allow scalable verification of a policy, we introduce a model-based approach that automatically learns a lightweight environment model from observations sampled by running the policy in the environment. Marvel verifies the policy against the environment model, which can be used to generate synthetic but tight abstraction of actual system behaviors. We provide statistical learning and verification guarantees on the accuracy of the learned environment model with respect to the policy. As such, formal safety guarantees on the policy generalize to the real environment with a desired high probability. The learned model itself is valuable as it pinpoints part of the state space where the policy is trustworthy.

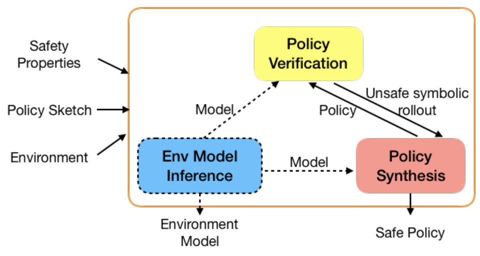


Fig. 1. The **Marvel** framework: **M**odel-**A**ided **R**easoning for **V**erification-**E**nabled **L**earning.

Second, the policy synthesis procedure is designed with formal verification in mind and is realized through a novel counterexample guided inductive synthesis loop (CEGIS) [Solar-Lezama 2008]. The most important feature of our CEGIS is that, instead of synthesizing a policy with *concrete* examples, we use *symbolic rollouts* with abstract states obtained by abstract interpretation. Marvel reduces the amount of safety properties violation by the abstraction states, which approximates the safety loss by the *worst-case* input, using a lightweight gradient-descent style optimization. Thus, Marvel can efficiently leverage verification feedback in a learning loop to enable policy safe-by-construction.

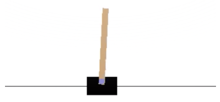
We present a detailed experimental study over a wide range of reinforcement learning systems. Our results manifest that Marvel can accurately approximate unknown environments or environments with complex dynamics using a lightweight model that facilitates scalable policy verification. Our experiments demonstrate the benefits of integrating formal verification as part of the training objective and leveraging verification feedback for safe policy synthesis.

Contributions. To summarize, this paper makes the following contributions:

- We propose a new model-based safe policy synthesis framework for challenging or even unknown reinforcement learning environments.
- We present an efficient policy safe-by-construction approach that allows the integration of formal verification within a reinforcement learning training loop.
- We implement the proposed ideas in a tool called Marvel, evaluate it in various application domains, and show that verification-based safe control policy synthesis is efficient and useful.

2 MOTIVATION AND OVERVIEW

To motivate the problem and to provide an overview of our approach, consider the CartPole environment taken from OpenAI Gym [Brockman et al. 2016]. A pictorial view of the environment is given on the right. A pole is attached by an unactuated joint to a cart, which moves along a frictionless track. The pendulum starts close upright, and the goal is to prevent it from falling over more than 12 degrees from vertical or moving more than 2.4 units from the center. A control policy should do so by taking actions to increase or reduce the cart's velocity. To make the example more interesting and challenging, we modify the original Gym environment: (1) we strengthen the safety constraint by disallowing the cart to move more than 0.5 units from the center, (2) we aim to synthesize a policy that applies continuous actions as opposed to discrete forces to control the cart. A system as such is representative of a number of practical Markov Decision Processes (MDPs), such as autonomous drones, that have thus far proven difficult for safety verification, but for which high assurance is extremely important. We now explain our key ideas to address safe control policy synthesis using the CartPole environment.



2.1 Safety Specification

According to the OpenAI Gym test criteria, the CartPole problem is considered solved when the learned policy can balance the pole well starting from a random initial state, while ensuring the cart not moving out of the safety range, within 200 timesteps over 100 consecutive rollouts. We specify the safety property formally. The CartPole problem is modeled as an MDP (Sec. 3) of four variables $x, \dot{x}, \theta, \dot{\theta}$, representing cart position, velocity, pole angle, and angular velocity. According to the test criteria, we specify the initial states S_0 as

$$S_0 \equiv \{(x, \dot{x}, \theta, \dot{\theta}) \mid -0.05 \leq x, \dot{x}, \theta, \dot{\theta} \leq 0.05\}$$

The safety property φ_{safe} of the system is specified as

$$\varphi_{safe}((x, \dot{x}, \theta, \dot{\theta})) \equiv -0.5 \leq x \leq 0.5 \land -0.21 \leq \theta \leq 0.21$$

(12 degrees ≈ 0.21 radians). Given the time horizon $T = 200$ timesteps, a safe policy for the CartPole problem should *at least* ensure that, for any environment state s in a T-timestep episode rollout from an initial state in S_0, $\varphi_{safe}(s)$ is true.

2.2 Inductive Invariant

However, φ_{safe} is a weak safety property. For various policies trained by reinforcement learning algorithms [Lillicrap et al. 2016; Rajeswaran et al. 2017; Schulman et al. 2015] in the CartPole environment, even though they can pass the test criteria, we observed that sometimes the RL agents tend to lose balance approaching 200 timesteps. We also lack a formal guarantee that the policy works well for unsampled rollouts. Our goal is to learn a policy that can be shown safe in *infinite timesteps* for *all possible initial states*. To this end, we introduce another safety property φ_{reach}, requiring that certain goal condition be met eventually. For the CartPole problem, we set φ_{reach} *equivalent to the specification of the initial states*:

$$\varphi_{reach}((x,\dot{x},\theta,\dot{\theta})) \equiv (x,\dot{x},\theta,\dot{\theta}) \in S_0 \equiv -0.05 \le x,\dot{x},\theta,\dot{\theta} \le 0.05$$

An episode rollout of the CartPole environment is $\tau = s_0, a_0, s_1, \ldots, a_{T-1}, s_T$ ($T = 200$) that starts from an initial state $s_0 \in S_0$. In τ, executing a control action a_t at an environment state s_t yields a new state s_{t+1}. We define that τ is *safe* iff there exists $T' \le T$ such that:

$$\underbrace{s_0}_{\varphi_{reach}(s_0)}, \quad \underbrace{s_1, \ldots, s_{T'-1}}_{\forall i \in 1 \ldots T'-1,\ \varphi_{safe}(s_i)} \quad \underbrace{s_{T'}}_{\varphi_{reach}(s_{T'})} \tag{1}$$

Essentially, we specified φ_{reach} as an *inductive invariant* of the CartPole problem: (1) φ_{reach} is equivalent to the initial state set, (2) any rollout starting from φ_{reach} eventually turns back to it, (3) any states in a rollout (including those that temporarily leave the inductive invariant set) are safe.

2.3 Verification-based Policy Optimization

To control an environment, at each timestep t, a state-feedback policy π takes as input the current environment state s_t and outputs a control action $a_t \sim \pi(\cdot|s_t)$. Reinforcement learning algorithms seek to train a policy π that is expected to maximize the long-term cumulative reward $J^R(\pi)$, evaluated on the episode rollouts of π:

$$J^R(\pi) = \mathbb{E}_{(\tau=s_0,a_0,\cdots,s_T)\sim\pi}\left[\sum_{t=0}^{T} \beta^t R(s_t, a_t)\right]$$

where β is the discount factor, τ is a rollout, and $\tau \sim \pi$ is shorthand for showing that the distribution over rollouts depends on π. $R(s_t, a_t)$ is the performance reward received after taking action a_t at state s_t in the environment.

Ideally, synthesizing a CartPole policy that satisfies the safety property φ_{safe} and the inductive invariant φ_{reach} can be achieved by shaping the reward function consistent with φ_{safe} and φ_{reach}:

$$R(s,a) = \begin{cases} 2 & \text{if } \varphi_{reach}(s') \\ 1 & \text{if } \varphi_{safe}(s') \land \neg\varphi_{reach}(s') \\ -1 & \text{if } \neg\varphi_{safe}(s') \end{cases} \tag{2}$$

where s' is the resulting state of taking action a at s. Compared with the reward function in the Gym CartPole environment, R additionally encourages the agent to stay within φ_{reach}. However, in our experience, both the unmodified Gym reward function and $R(s,a)$ above do not lead to a safe policy after RL training.[1]

An alternative approach to reward shaping is constrained safe reinforcement learning [Moldovan and Abbeel 2012; Turchetta et al. 2016]. Most safe-RL algorithms specify safety constraints as a cost function in addition to an objective reward function [Achiam et al. 2017; Berkenkamp et al. 2017; Dalal et al. 2018; Le et al. 2019; Li and Belta 2019]. For the CartPole problem, our goal would be to train a policy π that maximizes the cumulative object reward $J^R(\pi)$ and bounds the amount of safety violation to φ_{safe} during an episode and φ_{reach} by the end of the episode under a threshold (e.g. zero) on some sampled rollouts. However, there is no formal safety guarantee on a learned

[1]We also tried another reward function $R(s,a) = -||s||_2$ and achieved an even worse result.

policy. In our experience, applying state-of-the-art safe-RL algorithms [Achiam et al. 2017] in the CartPole environment does not lead to a policy that satisfies φ_{reach} on sampled episode rollouts.

In this paper, we instead formalize safe policy synthesis as a verification-based policy optimization problem. A synthesized policy π is certified by a novel program verifier $\mathcal{V}(\pi, \varphi_{safe}, \varphi_{reach})$ against the safety constraints. The verifier returns true if π is provably safe. Otherwise, π is optimized to eliminate verification counterexamples directly in the proof space. Formally, we aim to synthesize a policy π such that π balances the CartPole system quickly by maximizing the objective reward $J^R(\pi)$ while being verified safe (inductively) for infinite timesteps:

$$\begin{aligned} &\max_{\pi} J^R(\pi) \\ &\textit{subject to } \mathcal{V}(\pi, \varphi_{safe}, \varphi_{reach}) \textit{ is true} \end{aligned} \tag{3}$$

Next, Sec. 2.4, 2.5, and 2.6 illustrate how we solve the optimization problem in Equation (3) via the Marvel framework depicted in Fig. 1.

2.4 Environment Model Inference

Verifying a CartPole policy against the safety property φ_{safe} and the inductive invariant φ_{reach} is challenging because we require an environment model to verify the policy's behavior in the environment. However, in general, an accurate RL environment model may not exist. For the CartPole example, even if its environment model is explicitly provided[2], safety verification is still infeasible as the model's state transition function involves complex system dynamics and nonlinear computations which are out of the reach of state-of-the-art verification techniques.

Our solution is based on model-based learning. The high level idea is to sample the true environment and learn a verification-friendly probabilistic state transition model from the samples with theoretical guarantees. We learn an environment model $\mathcal{G}$ of the CartPole system as *time-varying linear-Gaussian* (TVLG), a kind of piecewise linear models. It is well-known that any deterministic or low-entropy stochastic policy operating in a physical environment with smooth system dynamics can be modeled with TVLG [Boyd and Vandenberghe 2004]. This representation was also found well suited for modeling noisy real-world tasks with stochastic dynamics [Levine and Abbeel 2014]. The state transition of TVLG at a timestep depends on a *Gaussian* distribution:

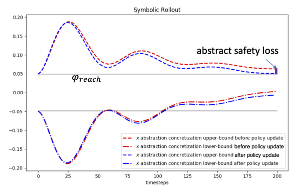
$$s_{t+1} \sim \mathcal{N}(A_t \cdot s_t + B_t \cdot a_t + c_t, \Sigma_t)$$

where s_{t+1} (resp. s_t) is a sampled state at timestep $t+1$ (resp. t) and a_t is an action taken at t. The mean of the Gaussian distribution is *linear* to s_t and a_t weighted by A_t and B_t (added with a bias vector c_t). State transition is *time-varying* as A_t, B_t, c_t, and covariance Σ_t vary at each timestep, allowing the model $\mathcal{G}$ to capture highly non-linear state-transition functions.

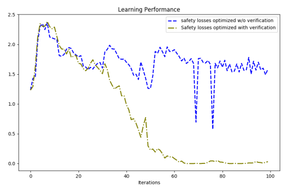
The main reason that we use time-vary linear-Gaussian to approximate highly-nonlinear or unknown environments is that such a model is tractable for both *learning* and *verification*. For learning, the particular form of a TVLG model implies that one can simply use linear regression to determine A_t, B_t, c_t, and then fit Σ_t based on the errors at each timestep t, using sampled episode rollouts of a policy π. We provide theoretical guarantees on the accuracy of a learned environment model $\mathcal{G}$ with respect to π and a practical algorithm to verify $\mathcal{G}$'s accuracy in Sec. 5.5.

The main challenge is that the sample complexity of linear regression scales with the dimensionality of a state space. For a much higher-dimensional robotic system than CartPole, the need of a large number of samples for a good dynamics fit at each timestep can be reduced by bringing in information from a global model fitted to all of the transitions from other timesteps (Sec. 5.1), or even prior models [Levine and Abbeel 2014].

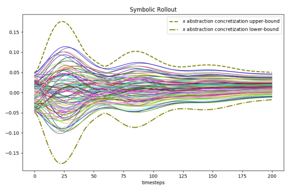
[2]https://github.com/openai/gym/blob/master/gym/envs/classic_control/cartpole.py



(a) Verification for Policy Update



(b) Marvel/NG-LIN Safety losses



(c) Environment Model Accuracy

Fig. 2. Verification-based policy synthesis (a), Learning performance (b), and Model accuracy (c) of CartPole

2.5 Policy Verification

For action planning in the CartPole environment, we consider a linear-Gaussian state-feedback policy:

$$a_t \sim \pi(\cdot \mid s_t) = \mathcal{N}(K \cdot s_t + k, \Gamma)$$

The policy π at each timestep t samples an action a_t from a normal distribution whose mean is *linear* to states s_t weighted by policy parameters K and k. The normal distribution has a learnable diagonal covariance matrix Γ (which is state-independent).

Since both the CartPole environment model $\mathcal{G}$ and the policy π are linear-Gaussian, the combined TVLG system $\mathcal{G}[\pi]$ of the model and the policy, as a Markov chain, is still linear-Gaussian. This is because the sum of Gaussian variables is again a Gaussian variable. From an initial state, (*i*) the mean μ_{s_t} of all possible states sampled at a timestep t can be computed by a *linear* system whose state-transition model is linear to the dynamics matrices A_t, B_t, c_t, and policy parameters K and k:

$$\begin{aligned} \mu_{a_t} &= K \cdot \mu_{s_t} + k, && \forall t \geq 0 \\ \mu_{s_{t+1}} &= A_t \cdot \mu_{s_t} + B_t \cdot \mu_{a_t} + c_t, && \forall t \geq 0 \end{aligned} \tag{4}$$

(*ii*) the deviation of any state at timestep t from the mean μ_{s_t} is constrained by a *Gaussian* covariance Ω_t that is obtained as the sum of the covariance Γ from the policy and the covariances from the TVLG model $\Sigma_1, \ldots, \Sigma_t$, weighted by policy and model parameters (Sec. 5). We exploit these two nice features (*i*) and (*ii*) to build a verification procedure for the combined TVLG system $\mathcal{G}[\pi]$ based on *abstract interpretation*.

At each timestep t, a tight abstraction of the *linear* system for mean μ_{s_t} in equation (4) combined with a box estimation of the confidence interval of the Gaussian distribution at t, parameterized by the covariance matrix Ω_t, accurately approximates possible states that can be sampled from the TVLG system $\mathcal{G}[\pi]$ at t (Sec. 5.2). With an abstract domain, the computation of the abstract semantics of TVLG yields a *symbolic rollout* of abstract states. We use the DeepPoly abstract domain [Singh et al. 2019] for this example. An abstract state in the DeepPoly domain is a tuple $\langle a^{\leq}(X), a^{\geq}(X), l, u \rangle$ where $a^{\leq}$ and $a^{\geq}$ are the (symbolic) lower and upper polyhedral constraints over the system variables X. l and u are the concrete lower and upper bounds that over-approximate the two symbolic bounds. One simple concertization of the abstract state is the interval $[l, u]$ as it gives the state value lower- and upper-bounds.

We depict the interval concretization of each abstract state within the symbolic rollout of a TVLG-approximated CartPole system $\mathcal{G}[\pi]$ for a trained policy π in Fig. 2a in red and we only show the abstraction for cart positions x. It can be seen that all states within the upper- and lower-bounds satisfy φ_{safe}. However, φ_{reach} is not satisfied as the upper bounds of cart positions are above 0.05, which is informally the upper-bound of cart position in φ_{reach}. Clearly, φ_{reach} is not an inductive invariant as per equation (1) for the combined system $\mathcal{G}[\pi]$ because there is a gap between the red upper-bound and that of φ_{reach} at the last timestep, which we refer to as *abstract safety loss*, denoted as $\mathcal{L}(\mathcal{G}[\pi], \varphi_{safe}, \varphi_{reach})$. The loss is computed based on the abstraction of $\mathcal{G}[\pi]$ against the safety properties φ_{safe} and φ_{reach}. As depicted in Fig. 2a, the abstract safety loss approximates the safety loss of the *worst-case* input. The combined system $\mathcal{G}[\pi]$ is verified safe when its abstract safety loss is zero. We formally characterize the abstract safety loss between abstract states and safety properties in Sec. 5.3. Based on this abstract interpretation mechanism, our safety verifier in Sec. 2.3 can be implemented as a formal safety check:

$$\mathcal{V}(\pi, \varphi_{safe}, \varphi_{reach}) = \big(\mathcal{L}(\mathcal{G}[\pi], \varphi_{safe}, \varphi_{reach}) = 0\big)$$

where $\mathcal{G}$ is a fitted TVLG model with respect to the policy π. The soundness of our safety verifier with respect to a learned environment model is given in Sec. 5.5.

A simple verification strategy is to iteratively run abstract interpretation to verify the learned policy π at each RL iteration with a fitted TVLG model $\mathcal{G}$ of the environment. A safe policy is found if $\mathcal{V}(\pi, \varphi_{safe}, \varphi_{reach})$ is true. However, the simple strategy does not work as shown in Fig. 2b. The blue curve records the abstract safety losses of the RL policies learned by an RL algorithm NG-LIN [Rajeswaran et al. 2017] at each training iteration, with the shaped reward function in Equation (2). We choose this algorithm because it performed better than other RL algorithms (including a safe-RL algorithm considered in Sec. 2.3) for this environment, e.g. [Achiam et al. 2017; Mania et al. 2018; Schulman et al. 2015]. A previous TVLG-model-based RL algorithm [Levine and Abbeel 2014] also fails this task. This result is not very surprising as the RL algorithms do not attempt to use verification feedback for safety improvement and there is no safety guarantee provided by these algorithms.

2.6 Policy Synthesis

One unique feature of our safe policy synthesis algorithm is that it consists in a symbolic counterexample guided inductive synthesis (**CEGIS**) loop. When verification fails we use symbolic rollouts as counterexamples to improve policy safety. In Fig. 2a, the abstract safety loss of the red unsafe symbolic rollout is based on an abstract state. Abstract states are nonetheless parameterized by policy parameters (K, k and Γ) as shown in equation (4). Thus, we can leverage a gradient-style optimization to update policy parameters by taking steps proportional to the negative of the gradient of the abstract safety loss. As opposed to standard gradient descent, our CEGIS algorithm optimizes policies based on symbolic rollouts in an abstract domain, favouring the safety verifier directly for provable guarantees. Fig. 2a depicts the symbolic rollout of the updated policy in blue where the abstract safety loss is reduced.

We integrate our synthesis algorithm within reinforcement learning loops to enhance both verified safety and reward performance. Formally, we solve the verification-based policy optimization problem in Equation (3) by reducing it to an equivalent unconstrained optimization problem:

$$\max_{\pi} \min_{\lambda \geq 0} J^R(\pi) - \lambda \mathcal{L}(\mathcal{G}[\pi], \varphi_{safe}, \varphi_{reach}) \tag{5}$$

where an environment model $\mathcal{G}$ is fitted with respect to a policy π. To this end, we could perform gradient descent on the adaptive penalty coefficient λ and gradient ascent on π, leveraging policy gradients [Sutton et al. 1999] and our synthesis algorithm to maximize the cumulative reward $J^R(\pi)$ and the negative abstract safety loss $\mathcal{L}(\mathcal{G}(\pi), \varphi_{safe}, \varphi_{reach})$. In this paper, we simplify this optimization problem by fixing $\lambda = 1$. Our approach can be deemed as seeking a high cumulative reward while enforcing the safety constraints.

Fig. 2b compares the performance of our approach and NG-LIN for the CartPole problem. Our algorithm discovered a verified policy with zero abstract safety loss at 65 iteration (olive curve) while the RL algorithm struggled to find a safe policy (blue curve). The result demonstrates that optimizing policies with the safety verifier providing policy gradients directly in the proof space outperforms policy updates solely using simulated rollouts.

To show the accuracy of the fitted TVLG model, we simulated a verified safe policy in the CartPole environment. A model is *accurate* if the simulated rollouts lie within the interval conretization of abstract states within the system's symbolic rollout. As depicted in Fig. 2c, the learned model is accurate for cart position x. It is accurate for the other dimensions as well (see Appendix. A.5).

3 PROBLEM SETUP

We formulate safe policy synthesis in the context of Markov Decision Process (MDP) with probabilistic dynamics.

MDP. Formally, an MDP is a structure $M = (X, S, A, F : \{S \times A \longrightarrow S\}, S_0, \varphi_{safe}, \varphi_{reach}, R : \{S \times A \longrightarrow \mathbb{R}\}, \beta)$ where X is a finite set of variables interpreted over the reals $\mathbb{R}$; S is an infinite set of *continuous real-vector* environment states which are valuations of the variables X ($S \subseteq \mathbb{R}^{|X|}$); A is a set of *continuous real-vector* agent actions; F is a stochastic state transition function that emits the next environment state given a current state s and an agent action a; $F(s, a)$ is defined as some probabilistic distribution $P(\cdot|s, a)$; we assume that the initial states of M are uniformly sampled from a set of environment states $S_0 \subseteq S$. $R(s, a)$ is the immediate reward after transition from an environment state s with action a and $0 < \beta \leq 1$ is a reward discount factor.

Policy. An agent of an MDP M can interact with the environment by taking actions via a policy conditioned on environment states. Formally, a policy is a (stochastic) map $\pi : \{S \longrightarrow A\}$ that determines which action the agent ought to take in a given state. Popular policy structures include neural networks and linear functions.

MDP Rollout. Given a time horizon T, a T-timestep MDP rollout of a policy π is denoted as $s_0, a_0, s_1, \ldots, s_{T'} \sim \pi$ where $T' \leq T$ and s_t and a_t are the environment state and the action taken at timestep t such that $s_0 \in S_0$, $s_{t+1} \sim F(s_t, a_t)$, and $a_t \sim \pi(a_t|s_t)$. The aggregate reward of π is

$$J^R(\pi) = \mathbb{E}_{s_0, a_0, \ldots, s_{T'} \sim \pi}[\sum_{t=0}^{T'} \beta^t R(s_i, a_i)]$$

Policy search, such as reinforcement learning, aims to produce a policy π that maximizes $J^R(\pi)$.

Safe MDP Rollout. In this work, an MDP M includes two safety properties φ_{safe} and φ_{reach} as logical formulae over environment states. They specify the intended behavior of any agents of M. φ_{safe} enforces that agents should only visit safe environment states evaluated true by φ_{safe}. For example, an agent should remain within a safety boundary or avoid any obstacles. Additionally, M enforces that agents should eventually reach some environment states evaluated true by φ_{reach}. For instance, an agent should meet some goals.

Definition 3.1 (Safety Property). Given an MDP, S_0 defines a bounded domain in the form of an interval $[\underline{x}, \overline{x}]$ where $\underline{x}, \overline{x} \in \mathbb{R}^{|X|}$ are lower and upper bounds of the initial states. Both φ_{safe} and φ_{reach} are quantifier-free Boolean combinations of linear inequalities over the MDP's variables X:

$$\langle \varphi_{safe}, \ \varphi_{reach} \rangle ::= \langle P \rangle \mid \langle P \rangle \wedge \langle P \rangle \mid \langle P \rangle \vee \langle P \rangle;$$

$$\langle P \rangle ::= \mathcal{A} \cdot x \leq b \ \textit{where} \ \mathcal{A} \in \mathbb{R}^{|X|}, \ b \in \mathbb{R};$$

An MDP state $s \in S$ satisfies φ_{safe} or φ_{reach}, denoted as $s \models \varphi_{safe}$ or $s \models \varphi_{reach}$, iff $\varphi_{safe}(s)$ or $\varphi_{reach}(s)$ is true.

A T-timstep MDP rollout $s_0, a_0, s_1, \ldots, s_{T'}$ ($T' \leq T$) is *safe* with respect to φ_{safe} and φ_{reach} iff:

$$\underbrace{s_0, a_0, s_1, \ldots a_{T'-2}, s_{T'-1}}_{\forall i \in 1 \ldots T'-1, \ s_i \models \varphi_{safe}} \ \underbrace{a_{T'-1}, s_{T'}}_{s_{T'} \models \varphi_{reach}} .$$

We require that the reward function R of an MDP be consistent with φ_{safe} and φ_{reach}. In fact, one can use reward shaping to design reward signals encouraging a policy to satisfy φ_{safe} and φ_{reach} by learning, e.g. Equation (2).

Safe Policy. Other than maximizing the accumulative reward $J^R(\pi)$, we aim to develop safe policy synthesis producing MDP policies that can be formally verified safe. Given an MDP M and a time horizon T, a policy π is *safe*, denoted as $M[\pi] \models \varphi_{safe}, \varphi_{reach}$, iff any T-timestep rollout of M is safe.

4 SAFE POLICY SYNTHESIS FOR DETERMINISTIC ENVIRONMENTS

We firstly focus on safe policy synthesis for a kind of MDPs with a *known* and *deterministic* state transition dynamics function in Sec. 4. In this case, at each timestep t, an MDP's state transition function $F(s_t, a_t)$ produces a deterministic next state s_{t+1}. We defer safe policy synthesis for general stochastic environments with unknown state transition dynamics to Sec. 5, which further extends our solution for the deterministic case in Sec. 4.

It suffices to learn a deterministic policy $a_t = \pi(s_t)$ to control an MDP M with deterministic dynamic.

*Definition 4.1 (**DSTS** – Deterministic State Transition Systems).* If the state transition function $F : \{S \times A \longrightarrow S\}$ of an MDP $M = (X, S, A, F, S_0, \varphi_{safe}, \varphi_{reach}, R, \beta)$ is deterministic, M can be redefined as a deterministic state transition system $\mathcal{F}[\cdot] = (F, \cdot, S_0)$, parameterized by an unknown deterministic policy $\pi : S \longrightarrow A$, where the state transition function F receives control actions given by the policy and S_0 is the initial state space. We explicitly model policy deployment as $\mathcal{F}[\pi] = (F, \pi, S_0)$. Structurally, $\mathcal{F}[\pi]$ consists of T layers. Each layer at t ($t \geq 0$) is a function F^t_π: $\lambda s_t.F(s_t, \pi(s_t))$, which takes as input a state s_t at timestep t and outputs its next state s_{t+1} at timestep $t + 1$. Formally, layer t is a compositional function of an initial state: $\mathcal{F}_t[\pi] = F^{t-1}_\pi \circ \ldots \circ F^1_\pi \circ F^0_\pi$.

Similar to the general MDP case, given a time horizon T, a deterministic policy π is safe for a deterministic state transition system $\mathcal{F}[\cdot]$ with respect to φ_{safe} and φ_{reach}, denoted as $\mathcal{F}[\pi] \models \varphi_{safe}, \varphi_{reach}$, iff, for any rollout of $\mathcal{F}[\pi]$,

$$\underbrace{s_0, s_1, \ldots s_{T'-2}, s_{T'-1}}_{\forall t \in 1\ldots T'-1,\ s_i \models \varphi_{safe}} \quad \underbrace{s_{T'}}_{s_{T'} \models \varphi_{reach}}$$

where $T' \leq T$, $s_0 \in S_0$ and $\forall t \geq 0, s_{t+1} = \mathcal{F}_t[\pi](s_0)$.

4.1 DSTS Policy Verification

To verify a DSTS over an infinite set of initial states, we apply abstract interpretation to approximate the infinite set of system behaviors.

Abstract Interpretation. We verify the safety of a deterministic state transition system (**DSTS**) using abstract interpretation. The abstract interpretation framework [Lin et al. 2019; Singh et al. 2019] is defined as a tuple: $\langle \mathcal{D}_c, \mathcal{D}_a, \alpha, \gamma, f \rangle$ where

- $\mathcal{D}_c : \{s \mid s \in \mathbb{R}^d\}$ is the concrete domain;
- $\mathcal{D}_a$ is the abstract domain of interest;
- $\alpha(\cdot)$ is an *abstraction* function that maps a set of concrete elements to an abstract element;
- $\gamma(\cdot)$ is a *concretization* function that maps an abstract element to a set of concrete elements;
- $f = \{(f_c, f_a) \mid f_c(\cdot) : \mathcal{D}_c \longrightarrow \mathcal{D}_c, f_a(\cdot) : \mathcal{D}_a \longrightarrow \mathcal{D}_a\}$ is a set of transformer pairs over $\mathcal{D}_c$ and $\mathcal{D}_a$.

The abstract interpretation framework is sound [Singh et al. 2019] if for all $S \subseteq \mathcal{D}_c$, $S \subseteq \gamma(\alpha(S))$ holds and given $(f_c, f_a) \in f$,

$$\forall c \in \mathcal{D}_c, a \in \mathcal{D}_a,\ c \in \gamma(a) \implies f_c(c) \in \gamma(f_a(a)).$$

Definition 4.2 (DSTS Abstract Interpretation). Given a deterministic state transition system $\mathcal{F}[\pi] = (F, \pi, S_0)$, let $\mathcal{D} = \langle \mathcal{D}_c \equiv S, \mathcal{D}_a, \alpha, \gamma, f\} \rangle$ be an instantiated abstract interpreter with a chosen abstract domain $\mathcal{D}_a$, abstraction function α, and concretization function γ. For every operation $\iota_c(\cdot)$ in F and π, there exists a suitable abstract transformer ι_a such that $(\iota_c, \iota_a) \in f$.

For a DSTS $\mathcal{F}[\pi]$ and an abstract interpreter $\mathcal{D}$, we denote by $\mathcal{F}_t^{\mathcal{D}}[\pi] : \mathcal{D}_a \to \mathcal{D}_a$ the over-approximation of $\mathcal{F}_t[\pi]$ (Definition 4.1) using $\mathcal{D}$ where any operations $\iota_c(\cdot)$ in $F(\cdot)$ and policy $\pi(\cdot)$ in $\mathcal{F}_t[\pi]$ are replaced in $\mathcal{F}_t^{\mathcal{D}}[\pi]$ by their corresponding abstract transformers ι_a in $\mathcal{D}$.

Definition 4.3 (DSTS Symbolic Rollout). Given a deterministic state transition system $\mathcal{F}[\pi] = (F, \pi, S_0)$ and an abstract interpreter $\mathcal{D}$, a symbolic rollout of $\mathcal{F}$ over $\mathcal{D}$ is $S_0^{\mathcal{D}}, S_1^{\mathcal{D}}, \ldots$ where $S_0^{\mathcal{D}} = \alpha(S_0)$ is the abstraction of the initial states S_0. And $S_t^{\mathcal{D}} = \mathcal{F}_t^{\mathcal{D}}[\pi](\alpha(S_0))$ over-approximates all possible states in the concrete domain corresponding to any initial state $s_0 \in S_0$ at timestep t.

Theorem 4.4 (DSTS Over-approximation Soundness). *For a deterministic state transition system $\mathcal{F}[\pi]$, given an abstract interpreter $\mathcal{D}$, at arbitrary timestep t, we have:*

$$\forall s_0.\ s_0 \in S_0 \implies \mathcal{F}_t[\pi](s_0) \in \gamma\Big(\mathcal{F}_t^{\mathcal{D}}[\pi]\big(\alpha(S_0)\big)\Big).$$

4.2 DSTS Policy Safety Loss

Any unsafe rollouts of a policy must have violated safety properties at some states. We define a safety loss function to quantify the safety violation of a state.

Definition 4.5 (Safety Loss Function). For a safety property φ over states $s \in S$, we define a non-negative loss function $\mathcal{L}(s, \varphi)$ such that $\mathcal{L}(s, \varphi) = 0$ iff s satisfies φ, i.e. $s \models \varphi$. We define $\mathcal{L}(s, \varphi)$ recursively, based on the possible shapes of φ (Definition 3.1):

- $\mathcal{L}(s, \mathcal{A} \cdot x \leq b) := \max(\mathcal{A} \cdot s - b, 0)$
- $\mathcal{L}(s, \varphi_1 \wedge \varphi_2) := \max(\mathcal{L}(s, \varphi_1), \mathcal{L}(s, \varphi_2))$
- $\mathcal{L}(s, \varphi_1 \vee \varphi_2) := \min(\mathcal{L}(s, \varphi_1), \mathcal{L}(s, \varphi_2))$

Note that $\mathcal{L}(s, \varphi_1 \wedge \varphi_2) = 0$ iff $\mathcal{L}(s, \varphi_1) = 0$ and $\mathcal{L}(s, \varphi_2) = 0$, which by construction is true if both φ_1 and φ_2 are satisfied by s, and similarly $\mathcal{L}(\varphi_1 \vee \varphi_2) = 0$ iff $\mathcal{L}(\varphi_1) = 0$ or $L(\varphi_2) = 0$.

Although it is possible to learn DSTS policy parameters solely via reinforcement learning using the safety loss function as a negative reward function, we do not have any formal safety guarantee of a learned policy. Instead, we aim to use verification feedback to improve policy safety. To this end, we lift the safety loss function over concrete states (Definition 4.5) to an *abstract safety loss* function for over-approximated abstract states.

Definition 4.6 (Abstract Safety Loss Function). Given an abstract state $S^{\mathcal{D}}$ and a safety property φ, we define an abstract safety loss function as

$$\mathcal{L}_{\mathcal{D}}(S^{\mathcal{D}}, \varphi) = \max_{s \in \gamma(S^{\mathcal{D}})} \mathcal{L}(s, \varphi)$$

The abstract safety loss function applies γ to obtain all concrete states represented by an abstract state $S^{\mathcal{D}}$. It measures the worst-case safety loss of φ among all concrete states subsumed by $S^{\mathcal{D}}$. Observe that, given an abstract domain $\mathcal{D}_a$, we can usually approximate $\gamma(S^{\mathcal{D}})$ with a tight interval $\gamma_I(S^{\mathcal{D}})$. For example, given $\mathcal{D}_a$ as a Zonotope domain, $S^{\mathcal{D}} \equiv \langle z_C, z_E \rangle$ where $z_C \in \mathbb{R}^{|X|}$ is the center of the zonotope, and $z_E \in \mathbb{R}^{|X| \times m}$ for some m describes a linear relationship between the error vector $e \in [-1, 1]^m$ and the output. The approximated interval concretization of $S^{\mathcal{D}}$ is [Mirman et al. 2018]:

$$\gamma_I(S^{\mathcal{D}}) = ([(z_C)_i - \Sigma_{j=1}^m |(z_E)_{i,j}|]_i, [(z_C)_i + \Sigma_{j=1}^m |(z_E)_{i,j}|]_i)$$

Hence, based on the possible shape of φ, we can more efficiently compute $\mathcal{L}_{\mathcal{D}}(S^{\mathcal{D}}, \varphi)$ as:

- $\mathcal{L}_{\mathcal{D}}(S^{\mathcal{D}}, \mathcal{A} \cdot x \leq b) := \max_{s \in \gamma_I(S^{\mathcal{D}})} \big(\max(\mathcal{A} \cdot s - b, 0)\big)$
- $\mathcal{L}_{\mathcal{D}}(S^{\mathcal{D}}, \varphi_1 \wedge \varphi_2) := \max(\mathcal{L}_{\mathcal{D}}(S^{\mathcal{D}}, \varphi_1), \mathcal{L}_{\mathcal{D}}(S^{\mathcal{D}}, \varphi_2))$
- $\mathcal{L}_{\mathcal{D}}(S^{\mathcal{D}}, \varphi_1 \vee \varphi_2) := \min(\mathcal{L}_{\mathcal{D}}(S^{\mathcal{D}}, \varphi_1), \mathcal{L}_{\mathcal{D}}(S^{\mathcal{D}}, \varphi_2))$

Theorem 4.7 (Abstract Safety Loss Function Soundness). *Given an abstract state $S^{\mathcal{D}}$ and a safety property φ, we have:*

$$\mathcal{L}_{\mathcal{D}}(S^{\mathcal{D}}, \varphi) = 0 \implies s \models \varphi \ \forall s \in \gamma_I(S^{\mathcal{D}}).$$

We further extend the definition for the safety of an abstract state to the safety of a symbolic rollout.

Definition 4.8 (Symbolic Rollout Safety Loss). Given a DSTS $\mathcal{F}[\pi]$, its T-step symbolic rollout $S_0^{\mathcal{D}}, S_1^{\mathcal{D}}, S_2^{\mathcal{D}}, \ldots, S_{T-1}^{\mathcal{D}}, S_T^{\mathcal{D}}$ (Definition 4.3) satisfies the safety properties φ_{safe} and φ_{reach} iff there exists $T' \leq T$ such that:

$$\underbrace{S_0^{\mathcal{D}}, S_1^{\mathcal{D}}, S_2^{\mathcal{D}}, \ldots, S_{T'-1}^{\mathcal{D}}}_{\forall t \in 1 \ldots T'-1,\ \mathcal{L}_{\mathcal{D}}(S_t^{\mathcal{D}}, \varphi_{safe})=0} \quad \underbrace{S_{T'}^{\mathcal{D}}}_{\mathcal{L}_{\mathcal{D}}(S_{T'}^{\mathcal{D}}, \varphi_{reach})=0}$$

If the symbolic rollout is unsafe, the safety loss of $\mathcal{F}[\pi]$ is:

$$\mathcal{L}_{\mathcal{D}}(\mathcal{F}[\pi], \varphi_{safe}, \varphi_{reach}) = \mathcal{L}_{\mathcal{D}}(S_T^{\mathcal{D}}, \varphi_{reach}) + \sum_{t=1}^{T} \mathcal{L}_{\mathcal{D}}(S_t^{\mathcal{D}}, \varphi_{safe})$$

Given a DTST system $\mathcal{F}[\pi] = (F, \pi, S_0)$, if the deployed policy π is complex (e.g. neural networks) or the state transition function F is nonlinear, abstract interpretation over the entire initial state space S_0 could incur very coarse abstractions, thereby giving false alarms. Initial state space partitioning is a common strategy to defeat large approximation error. Using a similar strategy to [Wang et al. 2018b], we pick state dimensions that significantly contribute to the unsafety of the $\mathcal{F}[\pi]$ symbolic rollout (Definition 4.8) based on their cumulative gradient of the abstract safety loss function (by calculation or estimation see below). A larger gradient suggests greater potential of decreasing the abstract safety loss if we bisect the initial state space along the picked dimensions. Formally, an input space partition of $\mathcal{F}$ is $\mathcal{F}^1, \mathcal{F}^2, \ldots, \mathcal{F}^N$ where $\mathcal{F}^i[\pi] = (F, \pi, S_0^i)$ and $S_0 = \bigcup_{i=1}^{N} S_0^i$. The safety loss of $\mathcal{F}$ can be redefined as:

$$\mathcal{L}_{\mathcal{D}}(\mathcal{F}[\pi], \varphi_{safe}, \varphi_{reach}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\mathcal{D}}(\mathcal{F}^i[\pi], \varphi_{safe}, \varphi_{reach})$$

The above and Definition 4.8 specifies a sound verification procedure for DSTS, formalized below.

Theorem 4.9 (Safety Verification Soundness). *For a DTST system $\mathcal{F}[\pi]$ deployed with a deterministic policy π, $\mathcal{F}[\pi] \models \varphi_{safe}, \varphi_{reach}$ denotes that the deployed system satisfies safety properties φ_{safe} and φ_{reach}.*

$$\mathcal{L}_{\mathcal{D}}(\mathcal{F}[\pi], \varphi_{safe}, \varphi_{reach}) = 0 \implies \mathcal{F}[\pi] \models \varphi_{safe}, \varphi_{reach}.$$

4.3 DSTS Policy Synthesis

In the following, we deem a policy as a function $\pi(\theta)$ of its parameters θ (e.g. the weights of a neural network). We abbreviate $\pi(\theta)$ as π_θ for simplicity. Given a DSTS $\mathcal{F}[\pi_\theta]$, the abstract safety loss function $\mathcal{L}_{\mathcal{D}}$ of $\mathcal{F}[\pi_\theta]$ is essentially a function of the policy π_θ, or more specifically, a function of π_θ's parameters θ. To reduce the abstract safety loss of π_θ, we can leverage a gradient-descent style optimization to update θ by taking steps proportional to the negative of the gradient of $\mathcal{L}_{\mathcal{D}}$

at θ. As opposed to standard gradient descent, we optimize π_θ based on the symbolic rollouts produced by the policy in an abstract domain for $\mathcal{F}[\pi_\theta]$, favouring the abstract interpreter directly for verification-based safe policy synthesis.

We propose a white-box approach for calculating the gradient of the safety loss function $\mathcal{L}_\mathcal{D}$ and a black-box approach for estimating the gradient if $\mathcal{L}_\mathcal{D}$ is not differentiable.

White-box policy gradient. Our verifier is parametric over an abstract interpreter $\mathcal{D}$, if abstract transformers associated with $\mathcal{D}$ for the policy π_θ and for the state transition function are differentiable, we compute the gradients [Werbos 1990] of the abstract safety loss function $\mathcal{L}_\mathcal{D}(\mathcal{F}[\pi_\theta], \varphi_{safe}, \varphi_{reach})$ using off-the-shelf automatic differentiation libraries such as PyTorch [Paszke et al. 2019].

$$\nabla_\theta \mathcal{L}_\mathcal{D} \leftarrow \frac{\partial \mathcal{L}_\mathcal{D}(\mathcal{F}[\pi_\theta], \varphi_{safe}, \varphi_{reach})}{\partial \theta}$$

Black-box policy gradient. Even in cases when abstract transformers are not differentiable or third-party implementations do not allow differentiation, we can effectively estimate gradients based on random search [Mania et al. 2018]. Given a DSTS $\mathcal{F}[\pi_\theta]$, at each training iteration, we obtain perturbed DSTSs $\mathcal{F}[\pi_{\theta+\nu\omega}]$ and $\mathcal{F}[\pi_{\theta-\nu\omega}]$ where we add sampled Gaussian noise ω to the current policy π_θ's parameters θ in both directions and ν is a small positive real number. By evaluating the abstract safety losses of the symbolic rollouts of $\mathcal{F}[\pi_{\theta+\nu\omega}]$ and $\mathcal{F}[\pi_{\theta-\nu\omega}]$, we update the policy parameters θ with a finite difference approximation along an unbiased estimator of the policy gradient:

$$\nabla_\theta \mathcal{L}_\mathcal{D} \leftarrow \frac{1}{N} \sum_{k=1}^{N} \frac{\mathcal{L}_\mathcal{D}(\mathcal{F}[\pi_{\theta+\nu\omega_k}],\ \varphi_{safe},\ \varphi_{reach}) - \mathcal{L}_\mathcal{D}(\mathcal{F}[\pi_{\theta-\nu\omega_k}], \varphi_{safe}, \varphi_{reach})}{\nu} \omega_k$$

To achieve verified safety, with a verification-based policy gradient $\nabla_\theta \mathcal{L}_\mathcal{D}$, we update policy parameters θ as below where η is a learning rate:

$$\theta \leftarrow \theta - \eta \cdot \nabla_\theta \mathcal{L}_\mathcal{D}$$

We form a counterexample guided inductive synthesis (CEGIS) loop to repeatedly apply policy verification and synthesis until a verifiably safe DSTS policy is synthesized.

5 SAFE POLICY SYNTHESIS FOR UNKNOWN ENVIRONMENTS

As explained in Sec. 2, for unknown or complex nonlinear environments, our approach extends the solution for the deterministic case in Sec. 4. The core idea is to learn a time-vary linear Gaussian (TVLG) to approximate unknown environments, based on model-based learning.

In this section, we limit policy synthesis to linear Guassian $\pi(\cdot|s_t) \sim \mathcal{N}(K \cdot s_t + k, \Gamma)$ where K and k are parameters and Γ is a diagonal covariance matrix. State-of-the-art results [Kakade 2001; Mania et al. 2018] demonstrate that such simple policies are comparable to deep neural network policies even in challenging robotic environments. It is also straightforward to extend our approach to support time-vary linear-Gaussian policies for increased expressivity.

Since both environment models and policies are linear-Gaussian, an MDP system combined of such a model and a policy is still linear-Gaussian. Therefore, a state of a TVLG system at timestep t is essentially sampled from a Gaussian distribution. In the following, we directly interpret a concrete TVLG state at timestep t as a Gaussian distribution (μ_{s_t}, Ω_t) where μ_{s_t} denotes the mean of all possible state samples at t and Ω_t denotes the covariance of all such samples.

Definition 5.1 (**TVLG** – *Time-varying Linear-Gaussian).* If the state transition function F of an MDP $M = (X, S, A, F, S_0, \varphi_{safe}, \varphi_{reach}, R, \beta)$ is unknown, we approximate M as a TVLG $\mathcal{G}[\cdot] = ((A_t, B_t, c_t, \Sigma_t),\ \cdot,\ (S_0, \Omega_0))$, parameterized by an unknown stochastic policy $\pi(\cdot|s_t) \sim \mathcal{N}(K \cdot s_t +$

$k, \Gamma)$, where A_t, B_t, c_t, Σ_t are the time-varying state transition matrices and (S_0, Ω_0) defines the initial states (see below). We explicitly model policy deployment as $\mathcal{G}[\pi] = ((A_t, B_t, c_t, \Sigma_t),\ \pi,\ (S_0, \Omega_0))$. A concrete state at timestep t in a rollout of $\mathcal{G}[\pi]$ is (μ_t, Ω_t), which expresses a multivariate normal distribution $\mathcal{N}(\mu_t, \Omega_t)$, and (μ_{s_0}, Ω_0) is the concrete initial state:

$$\begin{aligned} \mu_{s_0} &= s_0 \wedge s_0 \in S_0 \\ \mu_{a_t} &= K \cdot \mu_{s_t} + k, && \forall t \geq 0 \\ \mu_{s_{t+1}} &= A_t \cdot \mu_{s_t} + B_t \cdot (\mu_{a_t}) + c_t, && \forall t \geq 0 \end{aligned} \tag{6}$$

and

$$\begin{aligned} \Omega_0 &= \left[0\right]_{|S|\times|S|} \\ \Delta_t &= \begin{bmatrix} \Omega_t & \Omega_t \cdot K^T \\ K \cdot \Omega_t & K \cdot \Omega_t \cdot K^T + \Gamma \end{bmatrix}, && \forall t \geq 0 \\ \Omega_{t+1} &= \Sigma_t + \left[A_t, B_t\right] \cdot \Delta_t \cdot \left[A_t, B_t\right]^T, && \forall t \geq 0 \end{aligned} \tag{7}$$

The intuition behind Equation (7) is that the model uncertainty captured in the covariance Ω_{t+1} at timestep $t+1$ is the sum of the model uncertainty Σ_t at timestep t and the uncertainty propgated from all previous timesteps.

5.1 Environment Model Inference

The shape of a TVLG model implies that one can simply use linear regression to fit the model with N rollouts sampled from the actual environment to determine A_t, B_t, c_t, and then fit Σ_t based on the errors. If we take N to be sufficiently large, then any new rollout lies within the confidence interval of the linear-Gaussian model at each timestep t with high probability. However, in practice, we are more interested in learning an accurate model with only a small number of rollouts so that model-based verification is inexpensive. One observation is that environment dynamics at nearby timesteps are often strongly correlated. This means that sample complexity for the local model at a timestep can be reduced if we incorporate information from other timesteps or even models learned from previous training iterations. To this end, following [Levine and Abbeel 2014], we learn the transition matrices by linear regression with a learned Gaussian mixture model (GMM) prior. We fit a GMM prior to all of the samples (s_t, a_t, s_{t+1}) at all timesteps from all N rollouts. We then estimate the TVLG dynamics by fitting the linear-Gaussian dynamics for a particular timestep t to the samples at t, conditioned on the GMM prior.

Previous RL work based on TVLG (e.g. [Levine and Abbeel 2014]) does not provide theoretical guarantees on learned models. However, since our learning approach is based on verification, it is important to guarantee that verification is conditioned on an accurate environment model. We provide an analysis of the theoretical guarantees on the accuracy of TVLG modeling in Sec. 5.5.

5.2 TVLG Policy Verification

Given a TVLG dynmaics system $\mathcal{G}[\pi] = ((A_t, B_t, c_t, \Sigma_t), \pi, (S_0, \Omega_0))$, we verify the safety of π using abstract interpretation. Each TVLG state (μ_{s_t}, Ω_t) is interpreted as the mean and covariance of all possible sampled states at timestep t. As defined in Equation (7), since the covariances do not depend on states, we can leave Ω_t *unabstraced* and only choose a suitable abstract domain for the means μ_{s_t}, which are state-dependent. The transition matrices (A_t, B_t, c_t) define a deterministic state transition system for μ_{s_t} in Equation (6), which means that we can reduce TVLG verification to DSTS abstract interpretation (Definition. 4.2).

TVLG Symbolic Rollout. Given a TVLG $\mathcal{G}[\pi]$, let $\mathcal{D} = \langle \mathcal{D}_c \equiv S, \mathcal{D}_a, \alpha, \gamma, f \equiv \{(+, \oplus), (\cdot, \odot)\}\rangle$ be an instantiated abstract interpreter for the state transition system in Equation (6) for μ_{s_t}, the means of all possible sampled states at a timestep t. Here $\oplus$ and $\odot$ are suitable abstract transformers for matrix sum $+$ and multiplication $\cdot$ in the chosen abstract domain $\mathcal{D}_a$. $\mathcal{G}[\pi]$'s symbolic rollout is a sequence of abstract states $S_0^{\mathcal{D}} = (\mu_{S_0}^{\mathcal{D}}, \Omega_0), S_1^{\mathcal{D}} = (\mu_{S_1}^{\mathcal{D}}, \Omega_1), \dots, S_T^{\mathcal{D}} = (\mu_{S_T}^{\mathcal{D}}, \Omega_T)$, where $\Omega_0, \Omega_1, \dots, \Omega_T$ are covariances from equations (7) and $\mu_{S_t}^{\mathcal{D}}$ is the abstraction of μ_{s_t}:

$$\begin{aligned}
\mu_{S_0}^{\mathcal{D}} &= \alpha(S_0) \\
\mu_{A_t}^{\mathcal{D}} &= (K \odot \mu_{S_t}^{\mathcal{D}}) \oplus k, && \forall t \geq 0 \\
\mu_{S_{t+1}}^{\mathcal{D}} &= \left(A_t \odot \mu_{S_t}^{\mathcal{D}}\right) \oplus \left(B_t \odot (\mu_{A_t}^{\mathcal{D}})\right) \oplus c_t, && \forall t \geq 0 \qquad (8)
\end{aligned}$$

We lift the abstraction function α for μ_{s_t} to an abstraction function $\hat{\alpha}$ for a TVLG state. Given a TVLG state (μ_{s_t}, Ω_t) at a timestep t, $\hat{\alpha}(\mu_{s_t}, \Omega_t) = (\alpha(\{\mu_{s_t}\}), \Omega_t)$. Similarly, we define the concretization function for an abstract TVLG state $S_t^{\mathcal{D}} = (\mu_{S_t}^{\mathcal{D}}, \Omega_t)$ simply as $\hat{\gamma}(S_t^{\mathcal{D}}) = (\gamma(\mu_{S_t}^{\mathcal{D}}), \Omega_t)$.

5.3 TVLG Policy Safety Loss

Similar to DSTS verification (Definition 4.8), to compute the worst-case safety loss among all sampled states subsumed by an abstract TVLG state $S_t^{\mathcal{D}} = (\mu_{S_t}^{\mathcal{D}}, \Omega_t)$ with respect to a safety property, we approximate $\hat{\gamma}(S_t^{\mathcal{D}})$ with a tight interval concretization $\hat{\gamma}_I(S_t^{\mathcal{D}})$. Because the value of each sampled state at timestep t is affected by the covariance Ω_t as well, we must take sampled states that deviate from the means into account.

Recall that a sampled TVLG state s can be considered drawn from a joint Gaussian distribution $\mathcal{N}(\mu_s, \Omega)$ with mean μ_s. The main diagonal ($diag$) of the covariance matrix Ω contains the variance of each state dimension. Ideally one could use a confidence hyper-ellipse (whose shape is determined by the eigenvectors and eigenvalues of Ω) to define the region that contains all states that can be drawn from the joint Gaussian distribution with high probability. For efficient verification, we instead use an interval (box) estimation to bound the confidence hyper-ellipse for approximating deviated states. A normal deviated state s lies within the interval range between $\left[\mu_s - n \cdot \sqrt{diag(\Omega)}, \mu_s + n \cdot \sqrt{diag(\Omega)}\right]$ with high probability. Here n is a parameter used to parameterize the probabilistic coverage of deviated samples from the mean μ_s. For example, taking $n = 5$ accounts for 99.99% of all the possible deviated states.

Given an abstract state $S^D = (\mu_S^{\mathcal{D}}, \Omega)$ of a TVLG system, define $\hat{\gamma}_I^n(S^D)$ as its approximated interval concretization:

$$\hat{\gamma}_I^n(S^{\mathcal{D}}) = \hat{\gamma}_I^n((\mu_S^{\mathcal{D}}, \Omega)) = [l - n \cdot \sqrt{diag(\Omega)}, u + n \cdot \sqrt{diag(\Omega)}]$$

where $[l, u] = \gamma_I(\mu_S^{\mathcal{D}})$. Intuitively, since $\gamma_I(\mu_S^{\mathcal{D}})$ is the interval lower and upper bounds of all concrete means over-approximated by the mean abstraction $\mu_S^{\mathcal{D}}$, further subtracting to the lower bound and adding to the upper bound a standard deviation yields an interval estimation of a probabilistic coverage of all sampled states that deviate from the concrete means constrained by Ω.

Given a safety property φ, the abstract safety loss function $\mathcal{L}_{\mathcal{D}}(S^{\mathcal{D}}, \varphi)$ for a TVLG abstract state $S^{\mathcal{D}}$ is defined as:

$$\mathcal{L}_{\mathcal{D}}(S^{\mathcal{D}}, \varphi) = \max_{s \in \hat{\gamma}_I^n(S^{\mathcal{D}})} (\mathcal{L}(s, \varphi))$$

The computation of this function is the same as Definition 4.6. We formally characterize its soundness property below:

Theorem 5.2 (Abstract Safety Loss Function Soundness). *Given an abstract TVLG state $(\mu_S^{\mathcal{D}}, \Omega)$ and a safety property φ, we have*

$$\mathcal{L}_{\mathcal{D}}((\mu_S^{\mathcal{D}}, \Omega), \varphi) = 0 \implies s \models \varphi \ \forall s \in \hat{\gamma}_I^n(\mu_S^{\mathcal{D}}, \Omega).$$

Our TVLG verification is essentially probabilistic and we use the parameter n to adjust the probabilistic coverage of deviated states from the Gaussian means. Under a chosen probabilistic coverage, given a time horizon T, a TVLG $\mathcal{G}[\pi]$ satisfies safety properties φ_{safe} and φ_{reach}, iff for its symbolic rollout $S_0^{\mathcal{D}}, S_1^{\mathcal{D}}, \ldots, S_{T-1}^{\mathcal{D}}, S_T^{\mathcal{D}}$ there exists $T' \leq T$ such that:

$$\underbrace{S_0^{\mathcal{D}}, S_1^{\mathcal{D}}, \ldots, S_{T'-1}^{\mathcal{D}}}_{\forall t \in 1 \ldots T'-1,\ \mathcal{L}_{\mathcal{D}}(S_t^{\mathcal{D}}, \varphi_{safe})=0} \qquad \underbrace{S_{T'}^{\mathcal{D}}}_{\mathcal{L}_{\mathcal{D}}(S_{T'}^{\mathcal{D}}, \varphi_{reach})=0}$$

If the symbolic rollout is unsafe, the abstract safety loss of $\mathcal{G}[\pi]$ is:

$$\mathcal{L}_{\mathcal{D}}(\mathcal{G}[\pi], \varphi_{safe}, \varphi_{reach}) = \mathcal{L}_{\mathcal{D}}(S_T^{\mathcal{D}}, \varphi_{reach}) + \sum_{t=1}^{T} \mathcal{L}_{\mathcal{D}}(S_t^{\mathcal{D}}, \varphi_{safe}) \tag{9}$$

In our implementation, the probabilistic coverage parameter n is learned to be sufficiently large (typically $n \leq 5$) so that the concretization of a symbolic rollout covers all sampled states from the true environment at each timestep. A theoretical analysis of this choice is given in Sec. 5.5.

5.4 TVLG Safe Policy Synthesis

A policy $\pi_\theta(\cdot|s_t) \sim \mathcal{N}(K \cdot s_t + k, \Gamma)$ is deemed as a function of its parameters θ. Policy parameters θ include K, k and Γ. To reduce verified safety loss, we could simply run gradient descent on the loss function $\mathcal{L}_{\mathcal{D}}(\mathcal{G}[\pi_\theta], \varphi_{safe}, \varphi_{reach})$, defined in Equation (9), to update θ by taking steps proportional to the negative of the gradient of the abstract safety loss. However, such a verification-based policy update only improves the safety of a policy with respect to an environment model. It does not incorporate MDP reward functions R that may additionally include performance goals than safety.

We integrate MDP reward signals R to both seek a high cumulative reward and enforce verified safety. Reward improvement is subject to policy safety:

$$\begin{aligned} &\max_\theta J^R(\pi_\theta) \\ &\textit{subject to } \mathcal{L}_{\mathcal{D}}(\mathcal{G}[\pi_\theta], \varphi_{safe}, \varphi_{reach}) = 0 \end{aligned} \tag{10}$$

where an environment model $\mathcal{G}$ is fitted with respect to a current policy π_θ. A constrained optimization problem as such typically can be solved by the Lagrangian method [Platt and Barr 1988] that firstly reduces the constrained problem to an equivalent unconstrained optimization problem with an adaptive penalty coefficient λ:

$$\max_\theta \min_{\lambda \geq 0} J^R(\pi_\theta) - \lambda \mathcal{L}_{\mathcal{D}}(\mathcal{G}[\pi_\theta], \varphi_{safe}, \varphi_{reach}) \tag{11}$$

The Lagrangian method then solves the unconstrained optimization problem by performing gradient ascent on θ, to maximize the cumulative reward $J^R(\pi_\theta)$ and the negative abstract safety loss $\mathcal{L}(\mathcal{G}(\pi_\theta), \varphi_{safe}, \varphi_{reach})$, and gradient descent on λ based on the current value of the abstract safety loss. In our implementation, we alternatively use a simple strategy that fixes $\lambda = 1$ as a constant and find it sufficient for the benchmarks we consider in Sec. 6.

Specifically, to optimize the abstract safety loss $\mathcal{L}(\mathcal{G}(\pi_\theta), \varphi_{safe}, \varphi_{reach})$, we use the verification-based policy gradient $\nabla_\theta \mathcal{L}_{\mathcal{D}}$ defined in Sec. 4.3. To optimize the cumulative reward $J^R(\pi_\theta)$, in this work, we consider the standard model-free vanilla policy gradient $\nabla_\theta J^R(\pi_\theta)$ [Sutton et al. 1999]. This method directly optimizes policy parameters θ to maximize the cumulative reward using local search via the samples of the policy π_θ. We could simply update policy parameters θ via gradient ascent on the policy reward performance J^R and the negative abstract safety loss $\mathcal{L}_{\mathcal{D}}$ with a learning rate η:

$$\theta \leftarrow \theta + \eta \cdot \left(\nabla_\theta J^R(\pi_\theta) - \nabla_\theta \mathcal{L}_{\mathcal{D}}(\mathcal{G}(\pi_\theta), \varphi_{safe}, \varphi_{reach})\right)$$

However, a policy update as such can change the behavior of π_θ arbitrarily different from the old one at each training iteration. Because the fitted TVLG dynamics $\mathcal{G}[\cdot]$ is a local model and only valid in a local region of the state space around the sampled rollouts of the old policy, an unconstrained parameter update can cause the new policy to visit part of the state space where $\mathcal{G}[\cdot]$ is arbitrarily incorrect and unstable, leading to divergence. Therefore, our approach restricts the change of policies at each iteration to maintain the validity of a fitted time-varying linear model $\mathcal{G}[\cdot]$. Inspired by a few recently developed RL algorithms [Kakade 2001; Schulman et al. 2015], we address this issue by limiting the change by imposing a constraint on the KL-divergence between the old policy $\pi_{\theta_{old}}$ and the new policy π_θ by a threshold ξ:

$$\max_\theta J^R(\pi_\theta) - \mathcal{L}_\mathcal{D}(\mathcal{G}[\pi_\theta], \varphi_{safe}, \varphi_{reach})$$
$$\textit{subject to}\ \mathbb{E}_{s\sim d^{\pi_{\theta_{old}}}}[D_{KL}(\pi_{\theta_{old}}(\cdot|s)\ ||\ \pi_\theta(\cdot|s))] \le \xi \tag{12}$$

where $d^{\pi_{\theta_{old}}}$ is the state distribution under the policy $\pi_{\theta_{old}}$ estimated in the actual (true) environment by sampling[3] and the KL divergence D_{KL} is the expectation of the logarithmic difference between the probabilities of $\pi_{\theta_{old}}$ and π_θ:

$$D_{KL}(\pi_{\theta_{old}}\ ||\ \pi_\theta) = \mathbb{E}_{x\sim\pi_{\theta_{old}}}\left(\log\frac{\pi_{\theta_{old}}(x)}{\pi_\theta(x)}\right)$$

here the expectation is taken using the probabilities of $\pi_{\theta_{old}}$. Intuitively, D_{KL} measures how the probability distribution $\pi_{\theta_{old}}$ is different from a new policy π_θ.

The constrained optimization problem (12) can be analytically solved (similar to [Kakade 2001; Schulman et al. 2015]). We update θ as below. The derivation of this solution is given in Appendix A.4.

$$\theta = \theta_{old} + \sqrt{\frac{2\xi}{g^T H(\theta_{old})^{-1} g}}H(\theta_{old})^{-1}g \tag{13}$$
$$\textit{where}\ g = \left(\nabla_\theta J^R(\pi_\theta) - \nabla_\theta\mathcal{L}_\mathcal{D}(\mathcal{G}[\pi_\theta], \varphi_{safe}, \varphi_{reach})\right)|_{\theta=\theta_{old}}$$

$H(\theta_{old})$ is a Hessian matrix that measures the second-order derivative of the log probability of $\pi_{\theta_{old}}$ (after extending the KL divergence definition). Importantly, as opposed to improving policy parameters simply using the abstract safety loss gradient $\nabla_\theta\mathcal{L}_\mathcal{D}$ and the policy gradient $\nabla_\theta J^R$, the update rule in Equation (13) uses an additional KL divergence threshold ξ to restrain policy updates that can turn destructive to the validity of the fitted TVLG model $\mathcal{G}[\cdot]$. For sample efficiency, since $H(\theta_{old})$ is estimated based on sampled states from the old policy $\pi_{\theta_{old}}$, we can reuse the samples used to fit the TVLG model $\mathcal{G}[\cdot]$ for estimating $H(\theta_{old})$.

Model-free Modal-based Safe Policy Synthesis. We address the safe policy synthesis problem in a reinforcement learning loop. Our algorithm essentially alternates verification- and model-based policy update in Equations (12, 13) with model-free reinforcement learning, combining the strong safety guarantee of our model-based approach with the performance advantages of model-free RL. Figure 3 depicts Marvel's model-free model-based safe policy synthesis algorithm. Given an MDP M, the goal is to synthesize a policy π_θ and also learn a TVLG model $\mathcal{G}[\cdot]$ on which the safety guarantee of π_θ is conditioned. Policy search aims to have $\mathcal{L}_\mathcal{D}(\mathcal{G}(\pi_\theta), \varphi_{safe}, \varphi_{reach})$ reduced to 0, thereby ensuring π_θ safe with respect to the safety properties and the TVLG model $\mathcal{G}[\cdot]$ fitted at line 5. At Line 6, we estimate the model-free policy gradient $\nabla_\theta J^R(\pi_\theta)$ with the policy's sample rollouts for reward improvement. The policy's abstract safety loss, denoted as $\ell_\mathcal{D}$, is computed at Line 7. If $\ell_\mathcal{D} = 0$, Marvel dumps the safe policy as well as the learned TVLG model. The gradients of

[3] $d^\pi(s) = P(s_0 = s) + \beta P(s_1 = s) + \beta^2 P(s_2 = s) + \cdots$ where $s_0 \in S_0$, and $a_t \sim \pi(\cdot|s_t)$, $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$ $\forall t \ge 0$. If $\beta = 1$, $d_{\pi_{\theta_{old}}}$ is just the state visit frequency under $\pi_{\theta_{old}}$. P is the true transition probability distribution.

Require: MDP M as $(X, S, A, F, S_0, \varphi_{safe}, \varphi_{reach}, R, \beta)$ where the environment dynamics state transition function F is unknown, initial policy π_θ, KL divergence threshold ξ.
Ensure: Optimized policy π_θ and a learned TVLG model $\mathcal{G}[\cdot]$ such that $\mathcal{G}[\pi_\theta] \models \varphi_{safe}, \varphi_{reach}$.

```
 1: procedure Marvel
 2:     $\theta \leftarrow$ all weights in $\pi_\theta$ to optimize
 3:     while true do
 4:         Collect a set of rollouts $\mathcal{S}$ on policy $\pi_\theta$
 5:         Fit the TVLG dynamics $\mathcal{G}[\cdot]$ to the samples $\mathcal{S}$
 6:         Estimate vanilla policy gradient $\nabla_\theta J^R(\pi_\theta)$ with $\mathcal{S}$ and reward $R$ ([Sutton et al. 1999])
                                                            ▷ Abstract Interpretation
 7:         $\ell_\mathcal{D} \leftarrow \mathcal{L}_\mathcal{D}(\mathcal{G}[\pi_\theta], \varphi_{safe}, \varphi_{reach})$
 8:         if $\ell_\mathcal{D} = 0$ then
 9:             Dump $\pi_\theta$ and $\mathcal{G}[\cdot]$ to a safe policy list
10:         end if
                                                            ▷ Abstraction for Training
11:         $\theta_{old} \leftarrow \theta$
12:         Estimate $H(\theta_{old})$ based on the samples $\mathcal{S}$
13:         $g \leftarrow \big(\nabla_\theta J^R(\pi_\theta) - \nabla_\theta \mathcal{L}_\mathcal{D}(\mathcal{G}[\pi_\theta], \varphi_{safe}, \varphi_{reach})\big)|_{\theta=\theta_{old}}$
14:         $\theta \leftarrow \theta_{old} + \sqrt{\frac{2\xi}{g^T H(\theta_{old})^{-1} g}} H(\theta_{old})^{-1} g$
15:     end while
16: end procedure
```

Fig. 3. Marvel: Verification-guided safe policy synthesis.

the reward function and the abstract safety loss are used to guide the search of policy parameters at Line 13 to Line 14 according to Equation (13).

5.5 Theoretical Analysis

It is important to guarantee that a policy verified safe by Marvel is conditioned on an accurate environment model. We provide a theoretical analysis on the accuracy of Marvel's TVLG modeling.
Theoretical Guarantees on TVLG Modeling. Policy synthesis and verification in Marvel are based on abstraction. In this context, we define that a TVLG environment model $\mathcal{G}$ is accurate with respect to a policy π if all episode rollouts of π in the true environment lie within the interval conretization of the (abstract) symbolic rollout of the combined system $\mathcal{G}[\pi]$.

Specifically, as defined in Definition .5.1, a sample TVLG state of $\mathcal{G}$ at a timestep t is drawn from a normal distribution $\mathcal{N}(\mu_{s_t}, \Omega_t)$. An abstract TVLG state $S_t^D = (\mu_{S_t}^\mathcal{D}, \Omega_t)$ consists in the abstraction of the concrete means $\mu_{S_t}^\mathcal{D}$ and the covariance Ω_t at timestep t (Sec. 5.3). The interval concretization of the abstract state $S_t^\mathcal{D}$ is:

$$\hat{\gamma}_I^{n_t}(S_t^\mathcal{D}) = \hat{\gamma}_I^{n_t}((\mu_{S_t}^\mathcal{D}, \Omega_t)) = [l - n_t \cdot \sqrt{diag(\Omega_t)}, u + n_t \cdot \sqrt{diag(\Omega_t)}] \; where \; [l, u] = \gamma_I(\mu_{S_t}^\mathcal{D})$$

Recall that $\gamma_I(\mu_{S_t}^\mathcal{D})$ is the interval lower and upper bounds of all concrete means over-approximated by the mean abstraction $\mu_{S_t}^\mathcal{D}$. Further subtracting to the lower bound and adding to the upper bound a standard deviation $n_t \cdot \sqrt{diag(\Omega_t)}$ yields an interval estimation of a probabilistic coverage of all environment states at timestep t.

Marvel chooses the value of n_t at each individual timestep t to ensure that the interval concretization of the abstract state at t admits all the environment states sampled at that timestep for learning the TVLG model. In contrast to previous work on TVLG-based learning (e.g. [Levine and Abbeel 2014]), we have used results from statistical learning theory to obtain probabilistic guarantees on the correctness of TVLG modeling and provide the following theoretical guarantee:

Theorem 5.3 (TVLG Modeling Soundness). *Let $\epsilon, \delta \in \mathbb{R}_{>0}$, state dimensionality m, and a policy π be given. With the fixed policy π, the true environment MDP becomes a Markov chain and we can defined the state distribution at timestep t as $d_t^\pi(s)$. Consider N i.i.d. environment states $s_1, \dots, s_N \sim d_t^\pi$ sampled at timestep t where*

$$N \geq \frac{m(\ln \frac{2N}{m} + 1) + \ln \frac{4}{\delta}}{\epsilon^2} \tag{14}$$

and let $I = \hat{\gamma}_I^{n_t}(S_t^{\mathcal{D}})$ be an interval concretization of the abstract state $S_t^{\mathcal{D}}$ at timestep t with a suitable value of n_t that satisfies $s_i \in I$ for all $i \in \{1, \dots, N\}$. Then, with probability at least $1 - \delta$, we have

$$Pr_{s \sim d_t^\pi}(s \in I) \geq 1 - \epsilon$$

Intuitively, Theorem 5.3 says that at least a $1 - \epsilon$ fraction of true environment states at timestep t, sampled by the policy π, fall inside the interval concretization of the abstract state at t, and this guarantee holds with probability at least $1 - \delta$. We generalize the theoretical guarantee to the full (abstract) symbolic rollout of $\mathcal{G}[\pi]$: $S_0^{\mathcal{D}}, S_1^{\mathcal{D}}, \dots, S_T^{\mathcal{D}}$. If the number of rollouts N sampled for learning the TVLG model satisfies Equation (14), for any episode rollout of π drawn in the true environment $s_0, s_1, \dots, s_T$, with probability at least $1 - T\delta$, we have

$$\Pr_{(s_0, s_1, \dots, s_T) \sim \pi}(\forall t \in \{0, \dots, T\}, s_t \in \hat{\gamma}_I^{n_t}(S_t^{\mathcal{D}})) \geq 1 - T\epsilon$$

Statistical Verification of TVLG Model Accuracy. To meet the theoretical guarantee in Theorem 5.3, Marvel needs a very large number of sample rollouts to guarantee the accuracy of a learned TVLG model with high confidence (i.e. very small values of ϵ and δ). Thus it is computationally expensive to directly apply Theorem 5.3 in the learning loop of the Marvel algorithm in Fig. 3. We observed that, thanks to fitted GMM priors (Sec. 5.1), our modelling approach is very accurate even with a small number of sample rollouts (50 in our experiment). Based on this observation, for sample efficiency, we instead use post-learning verification to guarantee the probabilistic soundness of TVLG modeling in the implementation. The algorithm in Fig. 3 outputs verified policies and their environment models. Upon learning convergence, we statistically verify the accuracy of a TVLG model $\mathcal{G}$ with respect to a policy π whose verified safety is conditioned on $\mathcal{G}$.

Specifically, given the (abstract) symbolic rollout $S_0^{\mathcal{D}}, S_1^{\mathcal{D}}, \dots, S_T^{\mathcal{D}}$ of a combined TVLG system $\mathcal{G}[\pi]$, we apply a statistical model checking procedure (SMC_Verify) to verify that the inaccuracy of the TVLG model $\mathcal{G}$ with respect to the policy π is below ϵ:

$$\text{SMC\_Verify}(\Pr_{(s_0, \dots, s_T) \sim \pi}(\forall t \in \{0, \dots, T\}, s_t \in \hat{\gamma}_I^{n_t}(S_t^{\mathcal{D}})) \geq 1 - \epsilon)$$

Based on Clopper-Pearson confidence levels [Wang et al. 2019], we develop SMC_Verify as a sequential verification algorithm. A desired significance level $\delta > 0$ is used to capture the upper bound of the probability that SMC_Verify returns a wrong answer. At each iteration, we sample a batch of rollouts in the true environment using π to compute the significance level of confidence that ϵ bounds the fraction of true environment rollouts falling outside the symbolic rollout of $\mathcal{G}[\pi]$. We repeatedly do so until the significant level becomes less than the threshold δ. SMC_Verify terminates with probability 1. SMC_Verify can also be integrated with counterexample-guided refinement that increases the value of n_t to allow an abstraction to admit all safe rollouts sampled during statistical verification. The detailed algorithm for SMC_Verify is provided in Appendix. A.2.

In our experience, since learned TVLG models are already accurate despite a small amount of samples used in learning ($N = 50$), SMC_Verify only needs a few hundreds of sample rollouts to verify that a learned TVLG model captures at least 99% ($\epsilon = 0.01$) of true environment rollouts and guarantees to correctly derive this statistical result with probability at least 99% ($\delta = 0.01$). As our policy verification algorithm is sound with respect to a learned TVLG model, we have the same statistical guarantee on the verified safety of the policy conditioned on the model.

Marvel vs. Statistical Policy Verification. Marvel learns an environment model to assist policy verification with high probability safety guarantee. One may wonder why learning such a model is important and why not trivially learn a policy using RL and then directly verify the policy using statistical model checking? Simply doing so loses the benefits of exploiting verification-based policy updates to improve a policy's safety in the proof space, a key reason that Marvel can learn verified safe policies. In contrast, it is unclear of how statistical verification results might directly help optimize the policy if the confidence level is not desirable. In our experience, Marvel can learn policies that are empirically safe in cases RL algorithms cannot. As a result, RL algorithms alone cannot provide the same statistical guarantee that Marvel can offer (Sec. 6).

Worst-case Performance Update. Marvel blends reward improvement via RL and safety update via verification in a reinforcement learning loop. We show in Appendix. A.3 that Marvel has a worst-case performance degradation guarantee, i.e., a lower bound on reward improvement.

6 EXPERIMENTAL RESULTS

We have implemented our verification-based safe policy synthesis algorithm (Fig. 3) in a tool called Marvel. For abstract interpretation, Marvel uses the Zonotope abstract domain to reason about nonlinear state transition dynamics [Althoff 2015] and the DeepPoly abstract domain [Singh et al. 2019] to reason about control policies (including neural network policies). Marvel uses natural policy gradient (NPG) [Kakade 2001; Schulman et al. 2015] for the RL policy update in line 6 of the Marvel algorithm in Fig. 3. We compared Marvel with state-of-the-art baselines (details are presented below) and demonstrate that the verification guarantee provided by Marvel can lead to safe policies in scenarios that fail the baselines.

6.1 Deterministic State Transition Systems

In our first experiment, we evaluated Marvel on a diverse set of cyber-physical systems with known state-transition dynamics. We show that directly optimizing policies with verification-produced policy gradients in the proof space can yield provably safe policies that otherwise cannot be learned.

Baseline. In Sec. 6.1, we only consider one RL baseline, the natural policy gradient approach (NPG) used in Marvel for reward improvement. Without verification-based policy updates, Marvel (Fig. 3) is equivalent to the NPG-RL algorithm. We shaped the reward functions for Marvel and the RL baseline so that they have a penalty for states that do not satisfy φ_{safe} or φ_{reach}, e.g. Equation (2). We exclude the other model-free baseline considered in Sec. 6.2 and model-based control theory methods such as LQR [Levine and Abbeel 2014] because Marvel significantly outperforms them on learning verifiably safe policies for the benchmarks in this section.

Neural Network Policies. Our benchmarks are taken from the literature [Abate et al. 2017]. For all the benchmarks, φ_{safe} is a conjunction of interval constraints over system variables, meaning that the goal is to learn a policy that can maintain the system within a safety boundary. Similar to the CartPole problem in Sec.2, to prove that φ_{safe} holds in infinite timesteps, we set φ_{reach} as an inductive invariant, logically equivalent to the set of initial states, requiring that any rollout starting from a state in φ_{reach} eventually turns back to φ_{reach} and does not encounter a state that fails φ_{safe}. The policies are all two-hidden-layer neural networks (4×4) with tanh activation functions. Marvel was run without TVLG inference because the environment models are known.

Table 1. Safe policy synthesis with neural networks. Dim is the number of variables. AbsLoss is the best abstract safety loss achieved in 2000 training iterations. K is the number of timesteps a verified policy can steer back to an inductive invariant. Iters is the number of iterations to synthesize a verifiably safe policy.

		RL	**Marvel**	
Benchmark	**Dim**	**AbsLoss**	**K**	**Iters**
MagneticSup	2	33.78	1	352
Pendulum	2	2.03	3	400
Satellite	2	6.17	3	224
Helicopter	3	0.43	3	574
MagneticPtr	3	0.47	9	859
InverPendulum	4	0.31	3	1738
4CarPlatoon	8	1.33	10	121

The result is listed in Table 1. Marvel was able to produce verifiably safe neural policies for all the considered benchmarks. Although on most benchmarks the RL baseline can achieve similar rewards to Marvel, the learned policies cannot be verified safe. This is because the RL baseline policies were not trained with verification feedback and can converge to arbitrary policy parameters; abstract interpretation produced very conservative abstractions for their tanh functions that fail verification. Marvel can use verification feedback to learn well-performing policies that additionally allow tight abstractions for the neural network policies.

Nonlinear Systems. We also evaluated Marvel on two *nonlinear* continuous cyber-physical systems taken from the ARCH-COMP'20 competition [Johnson et al. 2020]. The first benchmark, Adaptive Cruise Control, involves an ego vehicle and a lead vehicle with six variables representing the position, velocity and acceleration of the two vehicles. Our training objective is to learn a linear policy that, when the lead vehicle suddenly reduces its speed, the ego car can decelerate to maintain a safe distance. φ_{safe} specifies the minimum relative distance that a policy should maintain at each timestep as well as an upper bound to prevent the ego car from stopping. A rollout lasts 50 timesteps and each timestep is 0.1s. Fig. 4 depicts the training performance of Marvel where we show the abstract safety loss at each training iteration. The RL baseline (blue) failed to further significantly reduce the worst-case safety loss. Marvel (olive) learned safe policies because it uses verification feedback to directly optimize the worst-case safety loss in the proof space. Previous work [Tran et al. 2020] used verification to detect safety issues for a well-trained policy for this system. Our result shows that verification can also be used for policy safe-by-construction. The result for the second nonlinear benchmark Single Pendulum is left in Appendix A.6.

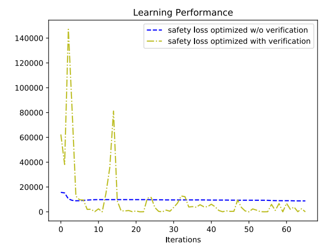
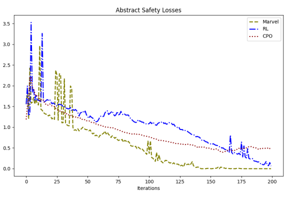


Fig. 4. Abstract Safety Losses on Adaptive Cruise Control Training.

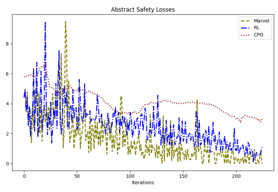
6.2 MDPs with Unknown or Complex Dynamics

We evaluated Marvel under the condition that either an MDP's state transition function is unknown or its complex dynamics does not allow scalable verification. We collected a set of state-of-the-art verfied RL benchmarks including classic control problems and robotics applications. We learn a TVLG environment model for such an environment to allow verification-based safe policy synthesis.

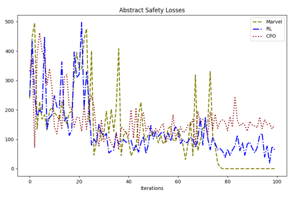
Baselines. Other than the NPG-RL baseline, we include a state-of-the-art safe-RL baseline: constrained policy optimization (CPO) [Achiam et al. 2017]. Safe-RL algorithms rely on specifying safety constraints as an additional cost function in the optimization objective and improve reward



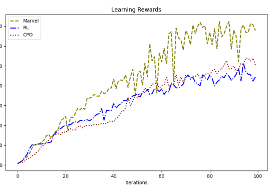
(a) ACC (3 dimensions)



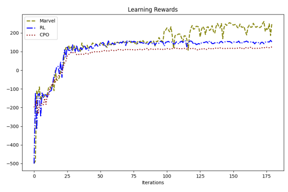
(b) Obstacle (4 dimensions)



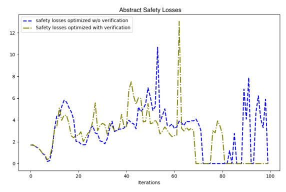
(c) DoublePendulum (6 dimensions)



(d) Hopper (11 dimensions)



(e) LunarLander (8 dimensions)



(f) Thrower (19 dimensions)

Fig. 5. Comparison between Marvel (olive), RL baseline (blue), and CPO baseline (red) in terms of abstract safety losses (closer to 0 is better) or rewards (a higher value is better).

performance subject to the cost function. CPO evaluates safety based on sample rollouts and does not use verification to provide any formal guarantees.

The result is depicted in Fig. 5 where we compare the policies learned using Marvel against that learned using the baselines in terms of their abstract safety loss and reward performance during training. After policy reward performance converges, Marvel learned verifiably safe policies for all the benchmarks while the baseline policies cannot even be shown empirically safe except ACC.

The first benchmark ACC taken from [Anderson et al. 2020] models a variant of the adaptive cruise control system. The lead car can apply an acceleration at any time. The goal is to maintain a very close distance to the lead car by the end of a rollout. As depicted in Fig. 5a, without model-based learning and verification, the baseline policies cannot be guaranteed provably safe. We found that they either accelerate quickly to catch up the lead car to achieve a high reward but fail to slow down soon enough to avoid a crash, or fail to maintain a close distance to the lead car because of being overly conservative. This result demonstrates that integrating safety verification can efficiently correct policy search directions toward safe policies in a policy training loop.

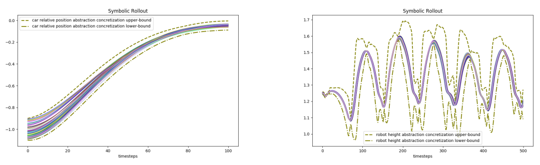
We observed similar results for Obstacle taken from [Anderson et al. 2020], DoublePendulum and Hopper from Mujoco [Todorov et al. 2012]. For Obstacle, Marvel learned safe policies that can arrive a goal state without hitting obstacles, a behavior that the baselines failed to reinforce as depicted in Fig. 5b. DoublePendulum is a more challenging version of the CartPole problem, where the pole has another pole on top of it. Verification feedback again is the key for safe policy synthesis as shown in Fig. 5c. The training objective of Hopper is to make a one-legged robot hop as fast as possible. Its safety property is that the robot height is always greater than 0.7 and the robot angle is smaller than 0.2. A rollout lasts 500 timesteps. NPG-RL and CPO failed to learn a safe policy and encountered very large safety penalty during training. Marvel was instead much more stable in training. We therefore show the average rewards (including safety rewards) collected by the three algorithms instead and leave the abstract safety loss comparison in Appendix. A.7. Fig. 5d indicates that Marvel outperforms the baselines significantly.

LunarLander, taken from OpenAI Gym [Brockman et al. 2016], is an environment in which a safe policy must move the lander from the top of the screen to a landing pad with a low speed, specified as φ_{reach}. The safety property φ_{safe} is that the horizontal position of the lander must be between $[-1, 1]$. The baseline policies can easily satisfy φ_{safe} but cannot be formally verified satisfying φ_{reach}. The lander under these policies tends to get stuck on a location off the landing pad slightly. Since it is difficult to visualize the small gap of abstract safety losses between landing on the pad and off the pad in this context, we show the average rewards collected by the three algorithms in Fig. 5e. Marvel learned fully verified safe policies and received much higher rewards for safe landing.

Thrower is a 19-dimensional benchmark taken from Mujoco [Todorov et al. 2012]. The goal is to manipulate a robotic arm to hit a ball to goal areas, specified as φ_{reach}. The learning curve of Marvel for this benchmark is more stable than the baseline as shown in Fig. 5f. Marvel also achieved higher rewards than the RL baseline. We do not show the result of CPO for this benchmark because the environment does not include a particular safety property. In general, we found that CPO, like many other safe-RL algorithms, is not good at handling reachability constraints as it is designed for optimizing aggregate safety costs along a rollout. This partially explains why CPO performs even worse than the RL basedline in terms of verified reachability on a few benchmarks.

The results of the other two Mujoco benchmarks, Reacher and Pusher, are shown in Appendix. A.7.

We depict the results of running abstract interpretation over a learned environment model and a learned policy as a combined closed-loop system for two benchmarks in Fig. 6. We draw the interval concretization of each abstract state as well as concrete rollouts from the real environments. For ACC, Fig. 6a shows that the relative vehicle positions in all sampled rollouts are safe below the threshold 0 and subsumed by the abstraction. We depict the abstraction of robot heights of Hopper in Fig. 6b. It shows that the movement of the robot (e.g. up and down) is accurately captured by the learned environment model.



(a) ACC (b) Hopper

Fig. 6. Abstract interpretation: interval concretization of abstract states (olive dashed lines) and sampled rollouts (colored trajectories) from the real environments are depicted.

Model Accuracy. An environment model is *accurate* if sampled rollouts lie within the interval conretization of a system's symbolic rollouts (Sec. 5.5). Marvel only relies on 50 rollouts to learn an environment model at each training iteration. This is to ensure that model-based verification is inexpensive. As indicated in Sec. 5.5, we instead used post-learning statistical verification (Appendix. A.2) to validate the accuracy of a learned environment model with respect to the synthesized policy conditioned on the model. We verify that a learned TVLG model captures at least 99% ($\epsilon = 0.01$) of sampled environment rollouts from the true environment and this statistical guarantee holds with probability at least 99% ($\delta = 0.01$). We only needed a few hundreds of sample rollouts to validate this statistical property for ACC, Obstacle, DoublePendulum, Hopper and LunarLander in Fig. 5. None of the states in the sample rollouts during statistical verification fall out of the interval concretization of the systems' abstraction. The learned environment model of the high-dimensional Thrower benchmark (19 dimensions) was also verified accurate. Only 0.2% of the sample rollouts (0.007% of the total sample states) lie out of the model's interval concretization. The model of the Pusher benchmark (20 dimensions) was also verified and is only inaccurate at 0.03% of the total states sampled during verification. Even if a sample rollout temporarily falls out of the interval concretization of a dimension of the two systems at a timestep, the leave is marginal, lower than 10^{-3} in our observation, and the rollout eventually comes back. Because these states and rollouts are still safe, we could continue to adjust the probability coverage parameters (n_t) of the models to obtain even higher verified accuracy.

As our policy verification algorithm is sound conditioned on a TVLG model, we have the same statistical guarantee on policy safety as environment-model accuracy. In our experiment, the baseline policies cannot be verified to have the same statistical guarantees that Marvel offers above.

7 RELATED WORK

Robust Machine Learning. Our work on using abstract interpretation [Cousot and Cousot 1977] for safe policy synthesis is inspired by the recent advances in verifying neural network robustness, e.g. [Anderson et al. 2019; Gehr et al. 2018; Singh et al. 2019; Weng et al. 2018]. These approaches apply abstract transformers to relax nonlinearity of activation functions in a neural network into convex representations, based on linear approximation [Singh et al. 2018, 2019; Weng et al. 2018; Wong and Kolter 2018; Zhang et al. 2020] or interval approximation [Gowal et al. 2018; Mirman et al. 2018]. Since the abstractions are differentiable, neural networks can be optimized toward tighter concertized bounds to improve verified robustness [Balunovic and Vechev 2020; Lin et al. 2019; Mirman et al. 2018; Wang et al. 2018a; Zhang et al. 2020]. These approaches do not consider the environment in which a neural network is deployed. Safety verification of a control policy over a time horizon with complex system dynamics remains challenging.

Recent work [Dutta et al. 2019; Fan et al. 2020; Ivanov et al. 2018; Sun et al. 2019; Tran et al. 2020] achieved initial results about verifying learning-enabled autonomous systems. Abstract interpretation was previously used to verify nonlinear dynamics systems [Althoff 2015; Koller et al. 2018; Oulamara and Venet 2015]. However, these approaches do not attempt to perform verification within a learning loop. Marvel demonstrates the substantial benefits of training policies with verification feedback.

Safe Reinforcement Learning. Safe reinforcement learning is a fundamental problem in machine learning [Moldovan and Abbeel 2012; Turchetta et al. 2016]. Verification is considered in [Akametalu et al. 2014] but the focus is on stability. Most safe-RL algorithms form a constraint optimization problem specifying safety constraints as a cost function in addition to an objective reward function [Achiam et al. 2017; Berkenkamp et al. 2017; Dalal et al. 2018; Le et al. 2019; Li and Belta 2019; Wen and Topcu 2018]. Their goal is to train a policy that maximizes the accumulated reward and bounds the amount of aggregate safety violation under a threshold. In contrast, Marvel ensures that a learned policy is formally verified safe with respect to an environment model and can better handle reachability constraints beyond safety (Sec. 6). Model-based safe learning was previously combined with formal verification in [Fulton and Platzer 2019]. However, a hand-crafted global model is required, which is updated as learning progresses to take into account the deviations between the model and the actual system behavior. Marvel infers environment models locally from data. Our contribution is significant because, without a global model, even small policy updates can cause safety guarantees based on a local model incorrect.

Revel [Anderson et al. 2020] introduces a mechanism to learn neural-symbolic RL agents to ensure safe exploration during training. It can efficiently synthesize adaptive safety shields for complex neural network policies. Similar approaches [Bastani et al. 2018; Verma et al. 2018; Zhu et al. 2019] synthesize a symbolic program to approximate an RL policy based on imitation learning [Ross et al. 2011] and show that the symbolic program has better interpretability and safety. In contrast, Marvel directly synthesize safe policies by integrating verification into a policy learning loop.

8 CONCLUSIONS

We present Marvel that bridges program synthesis and verification for control policy safe-by-construction. Formal verification is integrated into a reinforcement learning loop to enable symbolic counterexample-guided inductive policy synthesis. Our experiments demonstrate that policy updates based on verification feedback can lead to provably safe control policies.

REFERENCES

Alessandro Abate, Iury Bessa, Dario Cattaruzza, Lucas Cordeiro, Cristina David, Pascal Kesseli, Daniel Kroening, and Elizabeth Polgreen. 2017. Automated Formal Synthesis of Digital Controllers for State-Space Physical Plants. In *Computer Aided Verification (CAV) (LNCS, Vol. 10426)*. Springer, 462–482.

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained Policy Optimization. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 22–31. http://proceedings.mlr.press/v70/achiam17a.html

Anayo K. Akametalu, Shahab Kaynama, Jaime F. Fisac, Melanie Nicole Zeilinger, Jeremy H. Gillula, and Claire J. Tomlin. 2014. Reachability-based safe learning with Gaussian processes. In *53rd IEEE Conference on Decision and Control, CDC 2014, Los Angeles, CA, USA, December 15-17, 2014*. IEEE, 1424–1431. https://doi.org/10.1109/CDC.2014.7039601

M. Althoff. 2015. An Introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*.

Greg Anderson, Shankara Pailoor, Isil Dillig, and Swarat Chaudhuri. 2019. Optimization and abstraction: a synergistic approach for analyzing neural network robustness. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22-26, 2019*. 731–744. https://doi.org/10.1145/3314221.3314614

Greg Anderson, Abhinav Verma, Isil Dillig, and Swarat Chaudhuri. 2020. Neurosymbolic Reinforcement Learning with Formally Verified Exploration. *CoRR* abs/2009.12612 (2020). arXiv:2009.12612 https://arxiv.org/abs/2009.12612

Mislav Balunovic and Martin T. Vechev. 2020. Adversarial Training and Provable Defenses: Bridging the Gap. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=SJxSDxrKDr

Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. 2018. Verifiable Reinforcement Learning via Policy Extraction. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 2499–2509. http://papers.nips.cc/paper/7516-verifiable-reinforcement-learning-via-policy-extraction

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 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 908–918. http://papers.nips.cc/paper/6692-safe-model-based-reinforcement-learning-with-stability-guarantees

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

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

Patrick Cousot and Radhia Cousot. 1977. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977*. 238–252. https://doi.org/10.1145/512950.512973

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerík, Todd Hester, Cosmin Paduraru, and Yuval Tassa. 2018. Safe Exploration in Continuous Action Spaces. *CoRR* abs/1801.08757 (2018). arXiv:1801.08757 http://arxiv.org/abs/1801.08757

Souradeep Dutta, Xin Chen, and Sriram Sankaranarayanan. 2019. Reachability analysis for neural feedback systems using regressive polynomial rule inference. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019*, Necmiye Ozay and Pavithra Prabhakar (Eds.). ACM, 157–168. https://doi.org/10.1145/3302504.3311807

Jiameng Fan, Chao Huang, Xin Chen, Wenchao Li, and Qi Zhu. 2020. ReachNN*: A Tool for Reachability Analysis of Neural-Network Controlled Systems. In *Automated Technology for Verification and Analysis - 18th International Symposium, ATVA 2020, Hanoi, Vietnam, October 19-23, 2020, Proceedings (Lecture Notes in Computer Science, Vol. 12302)*, Dang Van Hung and Oleg Sokolsky (Eds.). Springer, 537–542. https://doi.org/10.1007/978-3-030-59152-6_30

Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. 2018. An Introduction to Deep Reinforcement Learning. *Foundations and Trends® in Machine Learning* 11, 3-4 (2018), 219–354. https://doi.org/10.1561/2200000071

Nathan Fulton and André Platzer. 2019. Verifiably Safe Off-Model Reinforcement Learning. In *Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 11427)*, Tomás Vojnar and Lijun Zhang (Eds.). Springer, 413–430. https://doi.org/10.1007/978-3-030-17462-0_28

Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. 2018. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. 3–18. https://doi.org/10.1109/SP.2018.00058

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy A. Mann, and Pushmeet Kohli. 2018. On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models. *CoRR* abs/1810.12715 (2018). arXiv:1810.12715 http://arxiv.org/abs/1810.12715

Radoslav Ivanov, James Weimer, Rajeev Alur, George J. Pappas, and Insup Lee. 2018. Verisig: verifying safety properties of hybrid systems with neural network controllers. *CoRR* abs/1811.01828 (2018). arXiv:1811.01828 http://arxiv.org/abs/1811.01828

Taylor T Johnson, Diego Manzanas Lopez, Patrick Musau, Hoang-Dung Tran, Elena Botoeva, Francesco Leofante, Amir Maleki, Chelsea Sidrane, Jiameng Fan, and Chao Huang. 2020. ARCH-COMP20 Category Report: Artificial Intelligence and Neural Network Control Systems (AINNCS) for Continuous and Hybrid Systems Plants. In *ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20) (EPiC Series in Computing, Vol. 74)*, Goran Frehse and Matthias Althoff (Eds.). EasyChair, 107–139. https://doi.org/10.29007/9xgv

Sham Kakade. 2001. A Natural Policy Gradient. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic* (Vancouver, British Columbia, Canada) *(NIPS'01)*. MIT Press, Cambridge, MA, USA, 1531–1538.

Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. 2018. Learning-Based Model Predictive Control for Safe Exploration. In *57th IEEE Conference on Decision and Control, CDC 2018, Miami, FL, USA, December 17-19, 2018*. IEEE, 6059–6066. https://doi.org/10.1109/CDC.2018.8619572

Hoang Minh Le, Cameron Voloshin, and Yisong Yue. 2019. Batch Policy Learning under Constraints. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 3703–3712. http://proceedings.mlr.press/v97/le19a.html

Sergey Levine and Pieter Abbeel. 2014. Learning Neural Network Policies with Guided Policy Search under Unknown Dynamics. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger (Eds.). 1071–1079. http://papers.nips.cc/paper/5444-learning-neural-network-policies-with-guided-policy-search-under-unknown-dynamics

Xiao Li and Calin Belta. 2019. Temporal Logic Guided Safe Reinforcement Learning Using Control Barrier Functions. *CoRR* abs/1903.09885 (2019). arXiv:1903.09885 http://arxiv.org/abs/1903.09885

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*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1509.02971

Xuankang Lin, He Zhu, Roopsha Samanta, and Suresh Jagannathan. 2019. ART: Abstraction Refinement-Guided Training for Provably Correct Neural Networks. *CoRR* abs/1907.10662 (2019). arXiv:1907.10662 http://arxiv.org/abs/1907.10662

Horia Mania, Aurelia Guy, and Benjamin Recht. 2018. Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 1805–1814. http://papers.nips.cc/paper/7451-simple-random-search-of-static-linear-policies-is-competitive-for-reinforcement-learning

Matthew Mirman, Timon Gehr, and Martin T. Vechev. 2018. Differentiable Abstract Interpretation for Provably Robust Neural Networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 3575–3583. http://proceedings.mlr.press/v80/mirman18b.html

Teodor Mihai Moldovan and Pieter Abbeel. 2012. Safe Exploration in Markov Decision Processes. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress. http://icml.cc/2012/papers/838.pdf

Mendes Oulamara and Arnaud J. Venet. 2015. Abstract Interpretation with Higher-Dimensional Ellipsoids and Conic Extrapolation. In *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 9206)*, Daniel Kroening and Corina S. Pasareanu (Eds.). Springer, 415–430. https://doi.org/10.1007/978-3-319-21690-4_24

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An

Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*. 8024–8035. http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library

John C. Platt and Alan H. Barr. 1988. Constrained Differential Optimization. In *Neural Information Processing Systems (NIPS'88)*. 612–621.

Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham M. Kakade. 2017. Towards Generalization and Simplicity in Continuous Control. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 6550–6561. http://papers.nips.cc/paper/7233-towards-generalization-and-simplicity-in-continuous-control

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011 (JMLR Proceedings, Vol. 15)*, Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík (Eds.). JMLR.org, 627–635. http://proceedings.mlr.press/v15/ross11a/ross11a.pdf

John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. 2015. Trust Region Policy Optimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015 (JMLR Workshop and Conference Proceedings, Vol. 37)*, Francis R. Bach and David M. Blei (Eds.). JMLR.org, 1889–1897. http://proceedings.mlr.press/v37/schulman15.html

Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T. Vechev. 2018. Fast and Effective Robustness Certification. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 10825–10836. http://papers.nips.cc/paper/8278-fast-and-effective-robustness-certification

Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev. 2019. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.* 3, POPL (2019), 41:1–41:30. https://doi.org/10.1145/3290354

Armando Solar-Lezama. 2008. *Program synthesis by sketching*. Ph.D. Dissertation. University of California, Berkeley.

Xiaowu Sun, Haitham Khedr, and Yasser Shoukry. 2019. Formal verification of neural network controlled autonomous systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019*, Necmiye Ozay and Pavithra Prabhakar (Eds.). ACM, 147–156. https://doi.org/10.1145/3302504.3311802

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller (Eds.). The MIT Press, 1057–1063. http://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation

Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control.. In *IROS*. IEEE, 5026–5033. http://dblp.uni-trier.de/db/conf/iros/iros2012.html#TodorovET12

Hoang-Dung Tran, Xiaodong Yang, Diego Manzanas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak, and Taylor T. Johnson. 2020. NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems. In *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 12224)*, Shuvendu K. Lahiri and Chao Wang (Eds.). Springer, 3–17. https://doi.org/10.1007/978-3-030-53288-8_1

Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. 2016. Safe Exploration in Finite Markov Decision Processes with Gaussian Processes. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (Eds.). 4305–4313. http://papers.nips.cc/paper/6358-safe-exploration-in-finite-markov-decision-processes-with-gaussian-processes

Jonathan Uesato, Ananya Kumar, Csaba Szepesvári, Tom Erez, Avraham Ruderman, Keith Anderson, Krishnamurthy (Dj) Dvijotham, Nicolas Heess, and Pushmeet Kohli. 2019. Rigorous Agent Evaluation: An Adversarial Approach to Uncover Catastrophic Failures. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=B1xhQhRcK7

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. 2018. Programmatically Interpretable Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 5052–5061. http://proceedings.mlr.press/v80/verma18a.html

Shiqi Wang, Yizheng Chen, Ahmed Abdou, and Suman Jana. 2018a. MixTrain: Scalable Training of Formally Robust Neural Networks. *CoRR* abs/1811.02625 (2018). http://arxiv.org/abs/1811.02625

Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. 2018b. Formal Security Analysis of Neural Networks using Symbolic Intervals. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, William Enck and Adrienne Porter Felt (Eds.). USENIX Association, 1599–1614. https://www.usenix.org/conference/usenixsecurity18/presentation/wang-shiqi

Yu Wang, Mojtaba Zarei, Borzoo Bonakdarpour, and Miroslav Pajic. 2019. Statistical Verification of Hyperproperties for Cyber-Physical Systems. *ACM Trans. Embed. Comput. Syst.* 18, 5s (2019), 92:1–92:23. https://doi.org/10.1145/3358232

Min Wen and Ufuk Topcu. 2018. Constrained Cross-Entropy Method for Safe Reinforcement Learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 7461–7471. https://proceedings.neurips.cc/paper/2018/hash/34ffeb359a192eb8174b6854643cc046-Abstract.html

Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane S. Boning, and Inderjit S. Dhillon. 2018. Towards Fast Computation of Certified Robustness for ReLU Networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 5273–5282. http://proceedings.mlr.press/v80/weng18a.html

P. J. Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proc. IEEE* 78, 10 (1990), 1550–1560. https://doi.org/10.1109/5.58337

Eric Wong and J. Zico Kolter. 2018. Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 5283–5292. http://proceedings.mlr.press/v80/wong18a.html

Mojtaba Zarei, Yu Wang, and Miroslav Pajic. 2020. Statistical verification of learning-based cyber-physical systems. In *HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control, Sydney, New South Wales, Australia, April 21-24, 2020*, Aaron D. Ames, Sanjit A. Seshia, and Jyotirmoy Deshmukh (Eds.). ACM, 12:1–12:7. https://doi.org/10.1145/3365365.3382209

Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane S. Boning, and Cho-Jui Hsieh. 2020. Towards Stable and Efficient Training of Verifiably Robust Neural Networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=Skxuk1rFwB

He Zhu, Zikang Xiong, Stephen Magill, and Suresh Jagannathan. 2019. An inductive synthesis framework for verifiable reinforcement learning. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22-26, 2019*, Kathryn S. McKinley and Kathleen Fisher (Eds.). ACM, 686–701. https://doi.org/10.1145/3314221.3314638