Verifiably Safe Exploration for End-to-End Reinforcement Learning

Hunt, N.; Fulton, N.; Magliacane, S.; Hoang, T.N.; Das, S.; Solar-Lezama, A.

DOI
10.1145/3447928.3456653

Publication date
2021

Document Version
Final published version

Published in
HSCC2021

Verifiably Safe Exploration for End-to-End Reinforcement Learning

Nathan Hunt
nhunt@mit.edu
Massachusetts Institute of Technology
Cambridge, Massachusetts, USA

Nathan Fulton
nathan@ibm.com
MIT-IBM Watson AI Lab, IBM Research
Cambridge, Massachusetts, USA

Sara Magliacane
sara.magliacane@gmail.com
MIT-IBM Watson AI Lab
University of Amsterdam

Trong Nghia Hoang
tnhoang@amazon.com
MIT-IBM Watson AI Lab, IBM Research
Cambridge, Massachusetts, USA

Subhro Das
subhro.das@ibm.com
MIT-IBM Watson AI Lab, IBM Research
Cambridge, Massachusetts, USA

Armando Solar-Lezama
asolar@csail.mit.edu
Massachusetts Institute of Technology
Cambridge, Massachusetts, USA

Abstract

Deploying deep reinforcement learning in safety-critical settings requires developing algorithms that obey hard constraints during exploration. This paper contributes a first approach toward enforcing formal safety constraints on end-to-end policies with visual inputs. Our approach draws on recent advances in object detection and automated reasoning for hybrid dynamical systems. The approach is evaluated on a novel benchmark that emphasizes the challenge of safely exploring in the presence of hard constraints. Our benchmark draws from several proposed problem sets for safe learning and includes problems that emphasize challenges such as reward signals that are not aligned with safety constraints. On each of these benchmark problems, our algorithm completely avoids unsafe behavior while remaining competitive at optimizing for as much reward as is safe. We characterize safety constraints in terms of a refinement relation on Markov decision processes – rather than directly constraining the reinforcement learning algorithm so that it only takes safe actions, we instead refine the environment so that only safe actions are defined in the environment's transition structure. This has pragmatic system design benefits and, more importantly, provides a clean conceptual setting in which we are able to prove important safety and efficiency properties. These allow us to transform the constrained optimization problem of acting safely in the original environment into an unconstrained optimization in a refined environment.

CCS Concepts: • **Theory of computation** → **Logic and verification**; • **Computing methodologies** → **Reinforcement learning**; **Markov decision processes**; **Neural networks**.

Keywords: formal verification, reinforcement learning, neural networks, hybrid systems, safe artificial intelligence, differential dynamic logic

ACM Reference Format:
Nathan Hunt, Nathan Fulton, Sara Magliacane, Trong Nghia Hoang, Subhro Das, and Armando Solar-Lezama. 2021. Verifiably Safe Exploration for End-to-End Reinforcement Learning. In *24th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '21), May 19–21, 2021, Nashville, TN, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447928.3456653

HSCC '21, May 19–21, 2021, Nashville, TN, USA

ACM ISBN 978-1-4503-8339-4/21/05.
https://doi.org/10.1145/3447928.3456653

1 Introduction

Deep reinforcement learning algorithms [46] are effective at learning, often from raw sensor inputs, control policies that optimize for a quantitative reward signal. Learning these policies can require experiencing millions of unsafe actions. Even if a safe policy is finally learned – which will happen only if the reward signal reflects all relevant safety priorities – providing a purely statistical guarantee that the optimal policy is safe requires an unrealistic amount of training data [24]. The difficulty of establishing the safety of these algorithms makes it difficult to justify the use of reinforcement learning in safety-critical domains where industry standards demand strong evidence of safety prior to deployment [23].

Formal verification provides a rigorous way of establishing safety for traditional control systems [5]. The problem of providing formal guarantees in RL is called *formally constrained reinforcement learning (FCRL)*. Existing FCRL methods such as [2, 7, 10, 15–17, 19, 20, 35] combine the best of both worlds: they optimize for a reward function while safely exploring the environment.

Existing FCRL methods suffer from two significant disadvantages that detract from their real-world applicability: a) they enforce constraints over a completely symbolic state space that is assumed to be noiseless (e.g. the position of the safety-relevant objects are extracted from a simulator's internal state); b) they assume that the entire reward structure depends upon the same symbolic state-space used to enforce formal constraints. The first assumption limits the applicability of FCRL in real-world settings where the system's state must be inferred by an imperfect and perhaps even untrusted perception system. The second assumption implies a richer symbolic state that includes a symbolic representation of the reward, which we argue is unnecessary and may require more labelled data. Furthermore, this means these approaches may not generalize across different environments that have similar safety concerns, but completely different reward structures.

The goal of this paper is to *safely learn a safe policy* without assuming a perfect oracle that identifies the positions of all safety-relevant objects. I.e., unlike all existing FCRL methods, we do not rely on the internal state of the simulator. Prior to reinforcement learning, we train an object detection system to extract from RGB images the 2D positions of safety-relevant objects up to a certain precision. The pre-trained object detection system is used during reinforcement learning to extract the positions of safety-relevant objects, and that information is then used to enforce formal safety constraints. Absolute safety in the presence of untrusted perception is epistemologically challenging, but our formal safety constraints do at least account for a type of noise commonly found in object detection systems. Finally, although our system (called Verifiably

Safe Reinforcement Learning, or VSRL) uses a few labeled data to pre-train the object detection, we still learn an end-to-end policy that may leverage the entire visual observation for reward optimization.

Prior work from the formal methods community has demonstrated that safe RL is possible with full symbolic characterization of the environment and precise observations of the entire state. However, this is not realistic for actual robotic systems which have to interact with the physical world and can only perceive it through an imperfect visual system. This paper demonstrates that techniques inspired by formal methods can provide value even in this situation. First, we show that by using existing vision techniques to bridge between the visual input and the symbolic representation, one can leverage formal techniques to achieve highly robust behavior. Under certain assumptions on the vision system, we prove that our approach will learn safely. Second, we propose a new method of enforcing constraints which refines the environment instead of the learning algorithm. We prove that all policies are safe in this refined environment and that learning in this refined environment preserves expected rewards.

We also show that our method is capable of optimizing for reward even when significant aspects of the reward structure are not extracted as high-level features used for safety checking. Our experiments demonstrate that VSRL is capable of optimizing for reward structure related to objects whose positions we do *not* extract via supervised training. This is significant because it means that VSRL needs pre-trained object detectors only for objects that are safety-relevant.

Finally, we provide a novel benchmark suite for Safe Exploration in Reinforcement Learning that includes both environments where the reward signal is aligned with the safety objectives and environments where the reward-optimal policy is unsafe. Our motivation for the latter is that assuming reward-optimal policies respect hard safety constraints neglects one of the fundamental challenges of Safe RL: preventing "reward-hacking". For example, it is fundamentally difficult to tune a reward signal so that it has the "correct" trade-off between a pedestrian's life and battery efficiency. We show that in the environments where the reward-optimal policy is safe ("reward-aligned"), VSRL learns a safe policy with convergence rates and final rewards which are competitive or even superior to the baseline method. More importantly, VSRL learns these policies with zero safety violations during training; i.e., it achieves perfectly safe exploration. In the environment where the reward-optimal policy is unsafe ("reward-misaligned"), VSRL successfully avoids "reward-hacking" by violating safety constraints, optimizing only for the subset of reward that can be achieved without violating safety constraints.

Summarily, this paper contributes: **(1)** VSRL, a new approach toward formally constrained reinforcement learning that makes more realistic assumptions about access to symbolic features. This approach requires minimal supervision before reinforcement learning begins and explores safely while remaining competitive at optimizing for reward. **(2)** a new method of enforcing safety constraints by refining the environment so all policies are safe while preserving expected rewards **(3)** a novel benchmark suite for safe exploration in reinforcement learning with hard constraints that includes both properly specified and mis-specified reward signals.

2 Problem Definition

A reinforcement learning (RL) system can be represented as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$ which includes a (possibly infinite) set $\mathcal{S}$ of system states, an action space $\mathcal{A}$, a transition function $T(s, a, s')$ which specifies the probability of the next system state being s' after the agent executes action a at state s, a reward function $R(s, a)$ that gives the reward for taking action a in state s, and a discount factor $0 < \gamma < 1$ that indicates the system preference to earn reward as fast as possible. We denote the set of initial states as $\mathcal{S}_{init} \subseteq \mathcal{S}$.

In our setting, $\mathcal{S}$ are images and we are given a safety specification $\texttt{safe} : \mathcal{O} \to \{0, 1\}$ over a set of high-level observations $\mathcal{O}$, specifically, the positions (planar coordinates) of the safety-relevant objects in a 2D or 3D space. Since $\mathcal{S} \neq \mathcal{O}$, it is not trivial to learn a safe policy π such that $\texttt{safe}(\mathcal{O}) = 1$ along every trajectory. We decompose this challenge into two well-formed and tractable subproblems:

1. Pre-training a system $\psi : \mathcal{S} \to \mathcal{O}$ that converts the visual inputs into symbolic states using synthetic data (without acting in the environment);
2. Learning policies over the visual input space $\mathcal{S}$ while enforcing safety in the symbolic state space $\mathcal{O}$.

This problem is not solvable without making some assumptions, so here we focus on the following:

Assumption 1 (ϵ-Bounded Detection Errors)**.** *The symbolic mapping ψ is correct up to ϵ. More precisely, the true position of every object o_i can be extracted from the image s through the object detector $\psi(s)_i$ so that the Euclidean distance between the actual and extracted positions is at most ϵ, i.e. $\forall i \; ||\psi(s)_i - o_i||_2 \leq \epsilon$.*

Assumption 1 is strong for two reasons. First, there is currently no method for verifying that an object detector has ϵ-bounded errors. Second, there is no compelling empirical evidence that ϵ-bounded errors provide an exhaustive model of modern object detectors' failure modes. However, without any assumption on the fault model for the vision system, safety cannot be guaranteed. Our assumption that errors are ϵ-bounded, although not exhaustive, does capture one of the most common failure modes for all object detection algorithms. We leave exploration of more sophisticated fault models for *specific* object detectors as future work, and note that the parametric nature of our approach allows us to integrate more complex and specific fault models into VSRL.

Assumption 2 (Liveness of Safe Action)**.** *Initial states, described by a set of properties denoted as* `init`*, are safe, i.e. $\forall s \in \mathcal{S}_{init}$: $\texttt{safe}(\psi(s)) = 1$. Moreover, every state we reach after taking only safe actions has at least one available safe action.*

Assumption 3 (Correctness up to Simulation of Dynamical Model)**.** *We are given a dynamical model of the safety-relevant dynamics in the environment, given as either a discrete-time dynamical system or a system of ordinary differential equations, denoted as* `plant`*. We assume that model is correct up to simulation; i.e., if $T(s_i, a, s_j) \neq 0$ for some action a, then the dynamical system* `plant` *maps $\psi(s_i)$ to a set of states that includes $\psi(s_j)$.*

For example, the model may be a system of ODEs that describes how the acceleration and angle impact the future positions of a robot, as well as the potential dynamical behavior of some hazards in the

environment. Note that this model only operates on O (the symbolic state space), not $\mathcal{S}$ (low-level features such as images or LiDAR).

The *"up to simulation"* portion of our assumption is simply a technical artifact that aligns the reachability structure of the dynamical system described by a hybrid program to the probabilistic structure of the dynamical system described by a Markov decision process. Up to simulation means that if there is a non-zero probability of action a transitioning from s_i to s_j, then there is a flow of the ODEs from the state obtained by performing action a in state s_i, to the state s_j. That is, the assumption that this model is correct *up to simulation* simply means that we assume the discrete transition relation T is a subset of the relation obtained by following the flows of the ODEs in the dynamical model.

Assumption 4 (Correctness up to Simulation of Controller)**.** *We have an abstract model of the agent's behavior, denoted as* `ctrl`*, that is correct up to simulation: if $T(s_i, a, s_j) \neq 0$ for some action a, then $\psi(s_j)$ is one of the possible next states after $a(\psi(s_i))$ by ctrl.*

An abstract model of the agent's behavior describes at a high-level a safe controller behavior, disregarding the fine-grained details an actual controller needs to be efficient. An example is a model that brakes if it is too close to a hazard and can have any other type of behavior otherwise. Note that `ctrl` is very different from a safe policy π, since it only models the safety-related aspects of π without considering reward optimization.

Assuming a known and correct model for both the environment and the agent up to simulation is necessary to fully characterize safe actions. This assumption is reasonable whenever accurate dynamical models of the system under control are available, which is often the case in traditional control systems. The success of model-based approaches toward controller design provides evidence for the reasonableness of this assumption. Notice that the model only needs to accurately describe safety-relevant portions of the environment's dynamics; in particular, we make no assumption about the accuracy of the plant model for potions of the state space that aren't relevant to safety. This paper assumes accurate environmental models are given; learning or repairing a model of the environment during exploration is an interesting direction for future work.

3 Background

The goal of an RL agent in an environment specified by an MDP $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$ is to find a policy π that maximizes its expected total reward from an initial state $s_0 \in \mathcal{S}_{init}$:

$$V^{\pi}(s) \quad \triangleq \quad \mathbb{E}_{\pi}\left[\sum_{i=0}^{\infty} \gamma^i r_i\right] \tag{1}$$

where r_i is the reward at step i. In a deep RL setting, we can use the DNN parameters θ to parametrize $\pi(a|s;\theta)$. One particularly effective implementation and extension of this idea is proximal policy optimization (PPO), which improves sample efficiency and stability by sampling data in batches and then optimizing a surrogate objective function that prevents overly large policy updates [45]. This enables end-to-end learning through gradient descent which significantly reduces the dependency of the learning task on refined domain knowledge. Deep RL thus provides a key advantage over traditional approaches which were bottle-necked by a manual, time-consuming, and often incomplete feature engineering process.

To ensure formal guarantees we use differential Dynamic Logic (dL) [36–38, 40], a logic for specifying and proving reachability properties of hybrid dynamical systems, which combine both discrete-time (e.g. a robot that decides actions at discrete times) and continuous-time dynamics (e.g. an ODE describing the position of the robot at any time). Hybrid systems can be described with hybrid programs (HPs), for which we give an informal definition in Table 1. Notably, besides the familiar program syntax, HPs are able to represent a non-deterministic choice between two programs $\alpha \cup \beta$, and a continuous evolution of a system of ODEs for an arbitrary amount of time, given a domain constraint F on the state space $\{x_1' = \tau_1, ..., x_n' = \tau_n \,\&\, F\}$ where τ_i are terms.

Program Statement	Meaning
$\alpha;\beta$	Run first α and then β
$\alpha \cup \beta$	Execute either α or β
α^*	Repeat α 0 or more times
$x := \theta$	Assign evaluation of θ to x
$x := *$	Assign any real value to x
$\{x_1' = \theta_1, ..., x_n' = \theta_n \,\&F\}$	ODE evolution in domain F
$?F$	Abort if formula F not true

Table 1. Definition of Hybrid Programs in dL. α and β are HPs, x_i are variables, $x' = \frac{\partial x}{\partial t}$, and θ_i are terms.

Formulas of dL are generated by the following grammar where α ranges over HPs:

$$\varphi, \psi ::= f \sim g \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi \mid \forall x.\varphi \mid \exists x.\varphi \mid [\alpha]\varphi$$

where f, g are polynomials over the state variables, ϕ and ψ are formulas of the state variables, $\sim$ is one of $\{\leq, <, =, >, \geq\}$. The formula $[\alpha]\varphi$ means that a formula φ is true in every state that can be reached by executing the hybrid program α.

Given a set of initial conditions `init` for the initial states, a discrete-time controller `ctrl` representing the abstract behaviour of the agent, a continuous-time system of ODEs `plant` representing the environment and a safety property `safe` we define the *safety preservation problem* as verifying that the following holds:

$$\texttt{init} \rightarrow [\{\texttt{ctrl};\texttt{plant}\}^*]\texttt{safe} \tag{2}$$

Intuitively, this formula means that if the system starts in an intial state that satisfies `init`, takes one of the (possibly infinite) set of control choices described by `ctrl`, and then follows the system of ordinary differential equations described by `plant`, then the system will always remain in states where `safe` is true.

Example 1 (Hello, World)**.** *Consider a 1D point-mass x that must avoid colliding with a static obstacle (o) and has perception error bounded by $\frac{\epsilon}{2}$. The following dL model characterizes an infinite set controllers that are all safe, in the sense that $x \neq o$* for **all** forward time and at every point throughout the entire flow of the ODE*:*

$$\texttt{init} \rightarrow [\{\texttt{ctrl}; t := 0; \texttt{plant}\}^*]x - o > \epsilon$$

where

$$\begin{aligned}
SB(a) &\equiv 2B(x - o - \epsilon) > v^2 + (a + B) * (aT^2 + 2Tv)) \\
\texttt{init} &\equiv SB(-B) \wedge B > 0 \wedge T > 0 \wedge A > 0 \wedge v \geq 0 \wedge \epsilon > 0 \\
\texttt{ctrl} &\equiv a := *; ? - B \leq a \leq A \wedge SB(a) \\
\texttt{plant} &\equiv \{x' = v, v' = a, t' = 1 \& t \leq T \wedge v \geq 0\}
\end{aligned}$$

and where x is the 1D coordinate of the point-mass under control, v is the velocity of x, a is the acceleration action taken by x, B is the maximum deceleration (i.e., maximum braking force), A is the maximum acceleration, T is the maximum time between control

decisions, t is the time elapsed during the current control step, and o is the 1D coordinate of the obstacle.

Starting from any state that satisfies the formula `init`*, the (abstract/non-deterministic) controller chooses* ***any*** *acceleration satisfying the* SB *constraint. After choosing any a that satisfies* SB*, the system then follows the flow of the system of ODEs in* `plant` *for any positive amount of time t less than T. The constraint* $v \geq 0$ *simply means that braking (i.e., choosing a negative acceleration) can bring the pointmass to a stop, but cannot cause it to move backwards.*

The full formula says that no matter how many times we execute the controller and then follow the flow of the ODEs, it will always be the case – again, for an infinite set of permissible controllers – that $x - o > \epsilon$*.*

Theorems of dL can be automatically proven in the KeYmaera X theorem prover [8, 9]. [33] explains how to synthesize action space guards from non-deterministic specifications of controllers (`ctrl`), and Fulton and Platzer [10, 11] explains how these action space guards are incorporated into reinforcement learning to ensure safe exploration.

3.1 Using Safe Controller Specifications to Constrain Reinforcement Learning

Hybrid programs have a denotational semantics that defines, for each program, the set of states that are reachable by executing the program from an initial state. A state is an assignment of variables to values. For example, the denotation of $x := t$ in a state s is:

$$[\![x := t]\!](s)(v) = s(v) \text{ for } v \neq x$$
$$[\![x := t]\!](s)(x) = t$$

Composite programs are given meaning by their constituent parts. For example, the meaning of $\alpha \cup \beta$ is:

$$[\![\alpha \cup \beta]\!](s) = [\![\alpha]\!](s) \cup [\![\beta]\!](s)$$

A full definition of the denotational semantics corresponding to the informal meanings given above is provided by [39].

Given a hybrid program and proven dL safety specification, Fulton and Platzer [10] explains how to construct safety monitors (which we also call *safe actions filters* in this paper) for reinforcement learning algorithms over a symbolic state space. In this section, we summarize their algorithm.

As opposed to our approach, Fulton and Platzer [10] employs both a controller monitor (that ensures the safety of the controller) and a model monitor (that ensures the adherence of the model to the actual system and checks for model mismatch).

The meaning of the controller monitor and model monitor are stated with respect to a specification with the syntactic form $P \rightarrow [\{\text{ctrl}; \text{plant}\}^*]Q$ where P is a dL formula specifying initial conditions, plant is a dynamical system expressed as a hybrid program that accurately encodes the dynamics of the environment, and Q is a post-condition. [10] assumes that ctrl has the form $?P_1; a_1 \cup \cdots \cup P_n; a_n$, where a_i are discrete assignment programs that correspond to the action space of the RL agent. For example, an agent that can either accelerate or brake has action space $A = \{A, -B\}$. The corresponding control program will be $?P_1; a := A \cup ?P_2; a := -B$ where P_1 is a formula characterizing when it is safe to accelerate and P_2 is a formula characterizing when it is safe to brake.

Given such a formula, [10] defines the controller and model monitors using the following conditions:

Corollary 1 (Meaning of Controller Monitor)**.** *Suppose CM is a controller monitor for* $P \rightarrow [\{ctrl; plant\}^*]Q$ *and* $s \in S$*. Let* $u : S \rightarrow S$ *be a deterministic controller. Then* $CM(u, s)$ *implies* $(s, u(s)) \in [\![ctrl]\!]$*.*

Corollary 2 (Meaning of Model Monitor)**.** *Suppose MM is a model monitor for* $init \rightarrow [\{ctrl; plant\}^*]Q$*, that U is a sequence of actions, and that s is a sequence of states. If* $MM(s_{i-1}, U_{i-1}, s_i)$ *for all i then* $s_i \models Q$*, and also* $(s_i, U_i(s_i)) \in [\![ctrl]\!]$ *implies* $(U_i(s_i), s_{i+1}) \in [\![plant]\!]$*.*

4 VSRL: Verifiably Safe RL on Visual Inputs

We present `VSRL`, a framework that can augment any deep RL algorithm to perform *safe exploration* on visual inputs. As discussed in Section 2, we decompose the general problem in two tasks:

1. learning a mapping of visual inputs s into a symbolic state o for safety-relevant properties using only a few examples (described in Section 4.1 and shown in Figure 1a);
2. learning policies over visual inputs, while enforcing safety in the symbolic state space (described in Section 4.2 and shown in Figure 1c).

This latter task requires a controller monitor, which is a function $\varphi : O \times A \rightarrow \{True, False\}$ that classifies each action a in each symbolic state o as "safe" or not. In this paper this monitor is synthesized and verified offline following [10, 11]. In particular, as discussed in the previous sections, the KeYmaera X theorem prover solves the safety preservation problem presented in Eq. (2) for a set of high-level reward-agnostic safety properties `safe`, a system of differential equations characterizing the relevant subset of environmental dynamics `plant`, an abstract description of a safe controller `ctrl` and a set of initial conditions `init` (shown in Figure 1b).

4.1 Object Detection

In order to remove the need to construct labelled datasets for each environment, we only assume that we are given a small set of images of each safety-critical object and a set of background images (in practice, we use 1 image per object and 1 background). We generate synthetic images by pasting the objects onto a background with different locations, rotations, and other augmentations. We then train a CenterNet-style object detector [50] which performs multi-way classification for whether each pixel is the center of an object. For speed and due to the visual simplicity of the environments, the feature extraction CNN is a truncated ResNet18 [21] which only keeps the first residual block. The loss function is the modified focal loss [29] from [26]. See Appendix A of [22] for full details on the object detector. Detection adds some run-time overhead for all environments, but many object detectors run quickly enough for real-time control. Section 4.4 contains a detailed discussion of scalability issues with object detection.

4.2 Enforcing Constraints

While `VSRL` can augment any existing deep RL algorithm, this paper extends PPO [44]. The algorithm performs RL as normal except that, whenever an action is attempted, the object detector and safety monitor are first used to check if the action is safe. If not, a safe action is sampled uniformly at random from the safe actions in the current state. This happens outside of the agent and can be seen as refining the environment using a safety check. Pseudocode for

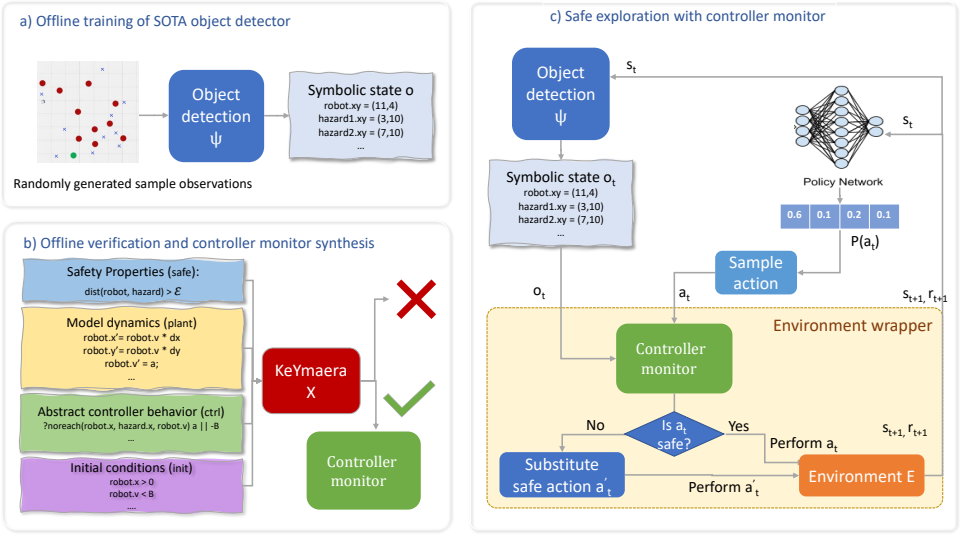


Figure 1. VSRL The left panels a) and b) represent offline pre-processing (described in Section 4.1) and verification. The right panel c) shows how these components are used to safely explore, as described in Section 4.2.

performing this refining is in Algorithm 1. The controller monitor is extracted from a verified dℒ model (see Page 3 of [10] for details).

Algorithm 1 The VSRL safety guard.

Input: s_t: input image; a_t: input action; ψ: object detector; φ: controller monitor; $E = (\mathcal{S}, \mathcal{A}, R, T)$: MDP of the original environment
if $\neg\varphi(\psi(s_t), a_t)$ **then**
  Sample substitute safe action a_t uniformly from $\{a \in \mathcal{A} \mid \varphi(\psi(s_t), a)\}$
Return $s_{t+1} \sim T(s_t, a_t, \cdot), r_{t+1} \sim R(s_t, a_t)$

4.3 Safety and Learning Results

We establish two theoretical properties about VSRL. First, we show that VSRL safely explores. Second, we show that using VSRL does not interfere with optimizing for the original reward objective.

If the object detector produces an accurate mapping, then Algorithm 1 will preserve the safety constraint associated with the φ monitor. We state this property formally in Theorem 1.

Theorem 1. *VSRL will remain safe if the following conditions hold along a trajectory $s_0, a_0, \ldots, s_n$ with $s_0 \in \mathcal{S}_{init}$:*

A1 *Initial states are safe: $s \in \mathcal{S}_{init}$ implies $\psi(s) \models init$.*
A2 *The model and symbolic mapping are correct up to simulation: If $T(s_i, a, s_j) \neq 0$ for some action a then $(\psi(s_i), a(\psi(s_i))) \in [\![ctrl]\!]$ and $(\psi(s_i), \psi(s_j)) \in [\![plant]\!]$.*

Proof. We begin the proof by pointing out that our assumption about how

$$init \rightarrow [\{ctrl; plant\}^*]safe$$

was proven provides us with the following information about some formula J:

$$\vdash init \rightarrow J \qquad \textbf{(LI1)}$$
$$\vdash J \rightarrow safe \qquad \textbf{(LI2)}$$
$$\vdash J \rightarrow [\{ctrl; plant\}^*]J \qquad \textbf{(LI3)}$$

Now, assume $s_0, a_0, s_1, a_1, \ldots, s_n$ with $s_0 \in \mathcal{S}_{init}$ is a trajectory generated by running an RL agent with actions selected by Algorithm 1 and proceed by induction on the length of the sequence with the inductive hypothesis that $\psi(s_i) \models J$.

If $i = 0$ then $s_0 \in \mathcal{S}_{init}$ by assumption. Therefore, $\psi(s_0) \models init$ by **A1**. We know by **LI1** that $\vdash init \rightarrow J$. Therefore, $\psi(s_0) \models J$ by Modus Ponens and the soundness of the dℒ proof calculus.

Now, suppose $i > 0$. We know $\psi(s_i) \models J$ by induction. Furthermore, we know $T(s_i, a_i, s_{i+1}) \neq 0$ because otherwise this trajectory could not exist. By **A2** and the denotation of the ; operator, we know $(\psi(s_i), \psi(s_{i+1})) \in [\![ctrl; plant]\!]$. By **LI3**, we know $\vdash J \rightarrow [ctrl; plant]J$ Therefore, $\psi(s_i) \models J$ and $(\psi(s_i), \psi(s_{i+1})) \in [\![ctrl; plant]\!]$ implies $\psi(s_i + 1) \models J$ by the denotation of the box modality and the soundness of dℒ.

We have now established that $\psi(s_i) \models J$ for all $i \geq 0$. By **LI2**, Modus Ponens, and soundness of the dℒ proof calculus, we finally conclude that $\psi(s_i) \models safe$. □

Note that if all actions a_i along the trajectory are generated using Algorithm 1, and if the model is accurate, then **A2** will hold.

4.3.1 Learning with Safety.

In order to enforce safety, we refine the original environment to create an environment without unsafe actions. By not modifying the agent or training algorithm, any theoretical results (e.g. convergence) which the algorithm already has will still apply in our safety-refined environment, provided these do not rely on specific properties of the original MDP. However, it is still necessary to show the relation between the (optimal) policies that may be found in the safe environment and the policies in the original environment. We show that 1) all safe policies in the original environment have the same transition probabilities and

expected rewards in the refined environment and 2) all policies in the refined environment correspond to a policy in the original environment which has the same transition probabilities and expected rewards. This shows that making progress (finding a policy with higher expected reward) in the safe environment leads to an equivalent amount of progress in the original environment. In particular, the optimal policies in the safe environment are optimal among safe policies in the original environment.

Let the original environment be the MDP $E = (\mathcal{S}, \mathcal{A}, T, R)$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ the set of actions, T the transition function, and R the reward function. Recall that we have a controller monitor $\varphi : \mathcal{S} \times \mathcal{A} \to \{True, False\}$ such that $\varphi(s, a)$ is $True$ if taking action a is safe in state s in E and $False$ otherwise (for simplicity, we won't worry about extracting symbolic states from visual observations here; that can be seen as happening inside of φ). When we refer to an action as safe or unsafe, we always mean in the original environment E. A policy π in E is safe iff

$$\forall s \in \mathcal{S}\ \forall a \in \mathcal{A}\ \pi(a|s) > 0 \implies \varphi(s, a).$$

The safety-refined environment will be $E' = (\mathcal{S}, \mathcal{A}, T', R')$ where the transition and reward functions will be modified to ensure 1) there are no unsafe actions and 2) expected rewards in E' correspond with those from acting safely in E.

For actions that are safe in T, we keep T' identical. For actions that are unsafe in T, we modify their effects in T' to be the same as taking a safe action in T. Without prior knowledge about which safe actions are better, we have T' emulate the effects of uniformly sampling a safe action and following T. Thus we set

$$T'(s, a, s') = \begin{cases} T(s, a, s') & \text{if } \varphi(s, a) \\ \frac{1}{|\mathcal{A}_{\varphi(s)}|} \sum_{a' \in \mathcal{A}_{\varphi(s)}} T(s, a', s') & \text{otherwise} \end{cases}$$

where $\mathcal{A}_{\varphi(s)} = \{a \in \mathcal{A} \mid \varphi(s, a)\}$ is the set of safe actions in state s.

R' is defined similarly so that it simulates the reward achieved by replacing unsafe actions with safe ones uniformly at random:

$$R'(s, a) = \begin{cases} R(s, a) & \text{if } \varphi(s, a) \\ \frac{1}{|\mathcal{A}_{\varphi(s)}|} \sum_{a' \in \mathcal{A}_{\varphi(s)}} R(s, a') & \text{otherwise} \end{cases}.$$

Lemma 1. *For every safe policy π in E, following that policy in E' leads to the same transitions with the same probabilities and gives the same expected rewards.*

Proof. By definition of safety, π has zero probability for any (s, a) where $\varphi(s, a)$ isn't true. Thus actions sampled from π lead to transitions and rewards from the branch of T' and R' where they are identical to T and R. □

Lemma 2. *For every policy π' in E' there exists a safe policy π in E such that π' has the same transition probabilities and expected rewards in E' as π does in E.*

Proof. For any π' in E', let $g(\pi') = \pi$ be defined such that

$$\pi(a|s) = \begin{cases} \pi'(a|s) + \frac{1}{|\mathcal{A}_{\varphi(s)}|} \sum_{a' \in \overline{\mathcal{A}}_{\varphi(s)}} \pi'(a'|s) & \text{if } \varphi(s, a) \\ 0 & \text{otherwise} \end{cases}$$

where $\overline{\mathcal{A}}_{\varphi(s)} = \{a \in \mathcal{A} \mid \neg\varphi(s, a)\}$ is the set of unsafe actions in state s. This evenly redistributes the probability that π' assigns to actions which would be unsafe in E among the safe actions.

We show first that the transition probabilities of π in E and π' in E' are the same.

$$\begin{aligned}
&P_{\pi,E}(s'|s) \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) T(s, a, s') \\
&= \sum_{a \in \mathcal{A}_{\varphi(s)}} \pi(a|s) T(s, a, s') + \sum_{a \in \overline{\mathcal{A}}_{\varphi(s)}} \underbrace{\pi(a|s)}_{=0} T(s, a, s') \\
&= \sum_{a \in \mathcal{A}_{\varphi(s)}} \left(\pi'(a|s) + \frac{1}{|\mathcal{A}_{\varphi(s)}|} \sum_{a' \in \overline{\mathcal{A}}_{\varphi(s)}} \pi'(s, a') \right) T(s, a, s') \\
&= \sum_{a \in \mathcal{A}_{\varphi(s)}} \pi'(a|s) T(s, a, s') \\
&\quad + \sum_{a' \in \overline{\mathcal{A}}_{\varphi(s)}} \pi'(s, a') \frac{1}{|\mathcal{A}_{\varphi(s)}|} \left(\sum_{a \in \mathcal{A}_{\varphi(s)}} T(s, a, s') \right) \\
&= \sum_{a \in \mathcal{A}_{\varphi(s)}} \pi'(a|s) T'(s, a, s') \\
&\quad + \sum_{a \in \overline{\mathcal{A}}_{\varphi(s)}} \pi'(a|s) \frac{1}{|\mathcal{A}_{\varphi(s)}|} \left(\sum_{a' \in \mathcal{A}_{\varphi(s)}} T(s, a', s') \right) \\
&= \sum_{a \in \mathcal{A}_{\varphi(s)}} \pi'(a|s) T'(s, a, s') + \sum_{a \in \overline{\mathcal{A}}_{\varphi(s)}} \pi'(a|s) T'(s, a, s') \\
&= \sum_{a \in \mathcal{A}} \pi'(a|s) T'(s, a, s') \\
&= P_{\pi',E'}(s'|s)
\end{aligned}$$

Let $\mathbb{E}_{\pi,E}[R_s]$ be the expected reward of following the policy π in environment E at state s. The equality of the expected reward for π in every state of E and π' in every state of E' can be shown similarly:

$$
\begin{aligned}
&\mathbb{E}_{\pi,E}\left[R_s\right] \\
&= \sum_{a\in\mathcal{A}} \pi(a|s)R(s,a) \\
&= \sum_{a\in\mathcal{A}} R(s,a) \begin{cases} \pi'(a|s) + \frac{1}{|\mathcal{A}_{\varphi(s)}|}\sum_{a'\in\overline{\mathcal{A}}_{\varphi(s)}} \pi'(a'|s) & \text{if } \varphi(s,a) \\ 0 & \text{otherwise} \end{cases} \\
&= \sum_{a\in\overline{\mathcal{A}}_{\varphi(s)}} R(s,a)\left(\pi'(a|s) + \frac{1}{|\mathcal{A}_{\varphi(s)}|}\sum_{a'\in\overline{\mathcal{A}}_{\varphi(s)}} \pi'(a'|s)\right) \\
&= \left(\sum_{a\in\mathcal{A}_{\varphi(s)}} \pi'(a|s)R(s,a)\right) \\
&\quad + \sum_{a\in\mathcal{A}_{\varphi(s)}} R(s,a)\frac{1}{|\mathcal{A}_{\varphi(s)}|}\sum_{a'\in\overline{\mathcal{A}}_{\varphi(s)}} \pi'(a'|s) \\
&= \left(\sum_{a\in\mathcal{A}_{\varphi(s)}} \pi'(a|s)R(s,a)\right) \\
&\quad + \sum_{a\in\overline{\mathcal{A}}_{\varphi(s)}} \pi'(a|s)\frac{1}{|\mathcal{A}_{\varphi(s)}|}\sum_{a'\in\mathcal{A}_{\varphi(s)}} R(s,a') \\
&= \sum_{a\in\mathcal{A}} \pi'(a|s) \begin{cases} R(s,a) & \text{if } \varphi(s,a) \\ \frac{1}{|\mathcal{A}_{\varphi(s)}|}\sum_{a'\in\mathcal{A}_{\varphi(s)}} R(s,a') & \text{otherwise} \end{cases} \\
&= \sum_{a\in\mathcal{A}} \pi'(a|s)R'(s,a) \\
&= \mathbb{E}_{\pi',E'}\left[R'_s\right]
\end{aligned}
$$

□

Lemma 3. *Running Algorithm 1 on E produces E', where E' is as defined at the beginning of this section.*

Proof. Trivial. □

Theorem 2. *Let $\mathbb{E}_{\pi,E}[R]$ be the expected reward of following policy π in environment E. Given two policies π'_1, π'_2 in E' with $\mathbb{E}_{\pi'_1,E'}[R] - \mathbb{E}_{\pi'_2,E'}[R] = c$, the corresponding policies $\pi_1 = g(\pi'_1), \pi_2 = g(\pi'_2)$ in E have $\mathbb{E}_{\pi_1,E}[R] - \mathbb{E}_{\pi_2,E}[R] = c$.*

Proof. Trivial by Lemma 2 which gives us that $\mathbb{E}_{\pi',E'}[R] = \mathbb{E}_{g(\pi),E}[R]$ for any π'. □

Theorem 3. *Any optimal policy in E' is optimal among the safe policies in E.*

Proof. Let π'^* be an optimal policy in E' and π^* be optimal among the safe policies in E. By Lemma 1, we know that the expected reward of π^* in E' is the same as in E ($\mathbb{E}_{\pi^*,E'}[R] = \mathbb{E}_{\pi^*,E}[R]$). Because π'^* is optimal in E', we have $\mathbb{E}_{\pi'^*,E'}[R] \geq \mathbb{E}_{\pi^*,E'}[R]$. Because π^* is optimal in E, we have $\mathbb{E}_{\pi^*,E}[R] \geq \mathbb{E}_{g(\pi'^*),E}[R]$. By Lemma 2, $\mathbb{E}_{g(\pi'^*),E}[R] = \mathbb{E}_{\pi'^*,E'}[R]$. Combining gives $\mathbb{E}_{g(\pi'^*),E}[R] = \mathbb{E}_{\pi^*,E}[R]$. □

Theorem 3 shows that we can find the optimal policy in E by learning the optimal policy in E'. Theorem 2 means that any progress we make in finding a better policy in E' translates to an equivalent amount of progress at optimizing for the objective in E. These results show that using E' to safely learn a policy that optimizes reward in E is reasonable.

A few notes regarding this approach:

- The intuitive approach to making an agent safe, if we know the set of safe actions in each state, might be to sample from the safe subset of the agent's policy distribution (after renormalization). Because this is not actually sampling from the distribution the agent learned, this may interfere with training the agent.
- While we keep $\mathcal{S}$ the same in E and E', there may be states which become unreachable in E' because only unsafe transitions in E lead to them. Thus the effective size of E''s state space may be smaller which could speed up learning effective safe policies.
- Our approach can be viewed as transforming a constrained optimization problem (being safe in E) into an unconstrained one (being safe in E').

4.4 Scalability of `VSRL`

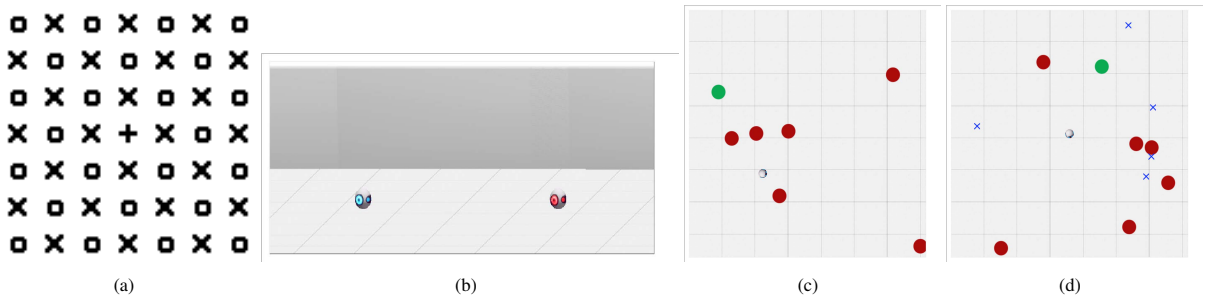
There are three sources of scalability concerns in `VSRL`: object detection, offline verification, and online controller monitoring.

We use neural networks for object detection. Fast and real-time inference for neural networks operating on rich visual inputs is an active line of research. Note that many object detection papers do achieve excellent results on very complex visual environments including in 3D environments. For example, there are many object detection methods that are designed to run at over 100 frames per second. In this work we use CenterNet, which runs in between 52 and 142 frames per second (depending on the amount of preprocessing, the number of passes, the resolution of the input images, etc.). Centernet works on real world images.

A second source of scalability challenges is in offline verification of safety-relevant dynamics (Box b in Figure 1). The models that this paper are based on were verified by writing Bellerophon tactics in KeYmaera X [8]. The work in this paper assumes an a priori verified hybrid systems model. Naturally, applying this work to different environment requires first building and verifying a model of the environment. Verifying hybrid systems is undecidable in theory and difficult in practice.

Given a verified model, constraint *checking* is not a source of scalability issues. A larger state-space or 3-dimensional environment would have minimal impact on constraint checking at runtime due to the nature of the runtime monitors extracted from KeYmaera X. These monitors are quantifier-free formulas of real arithmetic; therefore, checking that the monitor evaluates to true in a specific state and for a given action involves simply evaluating a single boolean expression. The size of this safety check grows linearly in the number of types of objects, these checks are trivially parallelizable, and the constant time is quite small as it only requires plugging into and evaluating one quantifier-free expression.

Although *checking* constraints is not a significant source of scalability challenges, *sampling* points from an arbitrary set such that those points match the safety constraint can be challenging. In all of the environments considered in this paper, the action space constraints are simple enough that we can explicitly characterize the set of safe actions in each state and sample uniformly from that set. However, in environments with more complex constraints, efficiently



(a) (b) (c) (d)

Figure 2. Visualizations of evaluation environments. **(a)** XO environment **(b)** ACC environment **(c)** Goal-finding environment **(d)** Pointmess environment.

sampling from arbitrary semi-algebraic sets becomes an important algorithmic consideration.

5 Experimental Validation of VSRL

We evaluate VSRL on four environments: a discrete **XO** environment [13], an adaptive cruise control environment (ACC), a 2D goal-finding environment (GF) similar to the Open AI Safety Gym Goal environment [43] but without a MuJoCo dependency and with simpler dynamics, and a pointmesses environment that emphasizes the problem of preventing reward hacking in safe exploration systems (PM). VSRL explores each environment without encountering any unsafe states.

The **XO** Environment is a simple setting introduced by [13] for demonstrating symbolic reinforcement learning algorithms (the implementation by Garnelo et al. [13] was unavailable, so we reimplemented this environment). The **XO** environment, visualized in Figure 2(a), contains three types of objects: **X** objects that must be collected (+1 reward), **O** objects that must be avoided (-1 reward), and the agent (marked by a **+**). There is also a small penalty (-0.01) at each step to encourage rapid collection of all **X**s and completion of the episode. This environment provides a simple baseline for evaluating VSRL. It is also simple to modify and extend, which we use to evaluate the ability of VSRL to generalize safe policies to environments that deviate slightly from implicit modeling assumptions. The symbolic state space includes the position of the **+** and the **O**, but not the position of the **X**s because they are not safety-relevant. The purpose of this benchmark is to provide a benchmark for safe exploration in a simple discrete setting.

The remainder of our experimental environments consider control of a point mass in 1D or 2D space. We extract positions from RGB images, and also assume independent sensors for both velocity and heading of the agent, so that these values do not have to be inferred from visual inputs. Inferring this information from visual inputs is possible, but assuming separate sensors for velocity and heading is reasonable.

The adaptive cruise control (ACC) environment has two objects: a follower and a leader. The follower must maintain a fixed distance from the leader without either running into the leader or following too far behind. We use the verified model from [42] to constrain the agent's dynamics.

The 2D goal-finding environment consists of an agent, a set of obstacles, and a goal state. The obstacles are the red circles and the goal state is the green circle. The agent must navigate from its (random) starting position to the goal state without encountering any of the obstacles. Unlike the OpenAI Safety Gym, the obstacles are *hard* safety constraints; i.e., the episode ends if the agent hits a hazard. We use the verified model from [32] to constrain the agent's dynamics.

The 2D pointmesses environment consists of an agent, a set of obstacles, a goal state, and a set of pointmesses (blue Xs). The agent receives reward for picking up the pointmesses, and the episode ends when the agent picks up all messes and reaches the goal state. Unlike the 2D goal-finding environment, hitting an obstacle does not end the episode. Instead, the obstacle is removed from the environment and a random number of new pointmesses spawn in its place. Notice that this means that the agent may reward hack by taking an unsafe action (hitting an obstacle) and then cleaning up the resulting pointmesses. We consider this the incorrect behavior. We use the verified model from [32] to constrain the agent's dynamics.

We compare VSRL to PPO using two metrics: the number of safety violations during training and the cumulative reward. These results are summarized in Table 2. VSRL is able to perfectly preserve safety in all environments from the beginning of training even with the ϵ-bounded errors in extracting the symbolic features from the images. In contrast, vanilla PPO takes many unsafe actions while training and does not always converge to a policy that entirely avoids unsafe objects by the end of training.

In some environments, preserving safety specifications also substantially improves sample efficiency and policy performance early in the training process. In the ACC environment, in particular, it is very easy to learn a safe policy which is reward-optimal. In the GF and PM environments, both the baseline agent and the VSRL agent struggle to learn to perform the task well (note that these tasks are quite difficult because encountering an obstacle ends the episode). However, VSRL remains safe without losing much reward relative to the amount of uncertainty in both policies.

6 Related Work

Recently, there has been a growing interest in safe RL, especially in the context of *safe exploration*, where the agent has to be safe also during training. A naive approach to RL safety is reward shaping, in which one defines a penalty cost for unsafe actions. This approach has several drawbacks, e.g. the choice of the penalty is brittle, so a naive choice may not outweigh a shorter path to the reward, as shown by Dalal et al. [6]. Therefore, recent work on safe RL addresses the challenge of providing reward-agnostic safety guarantees for

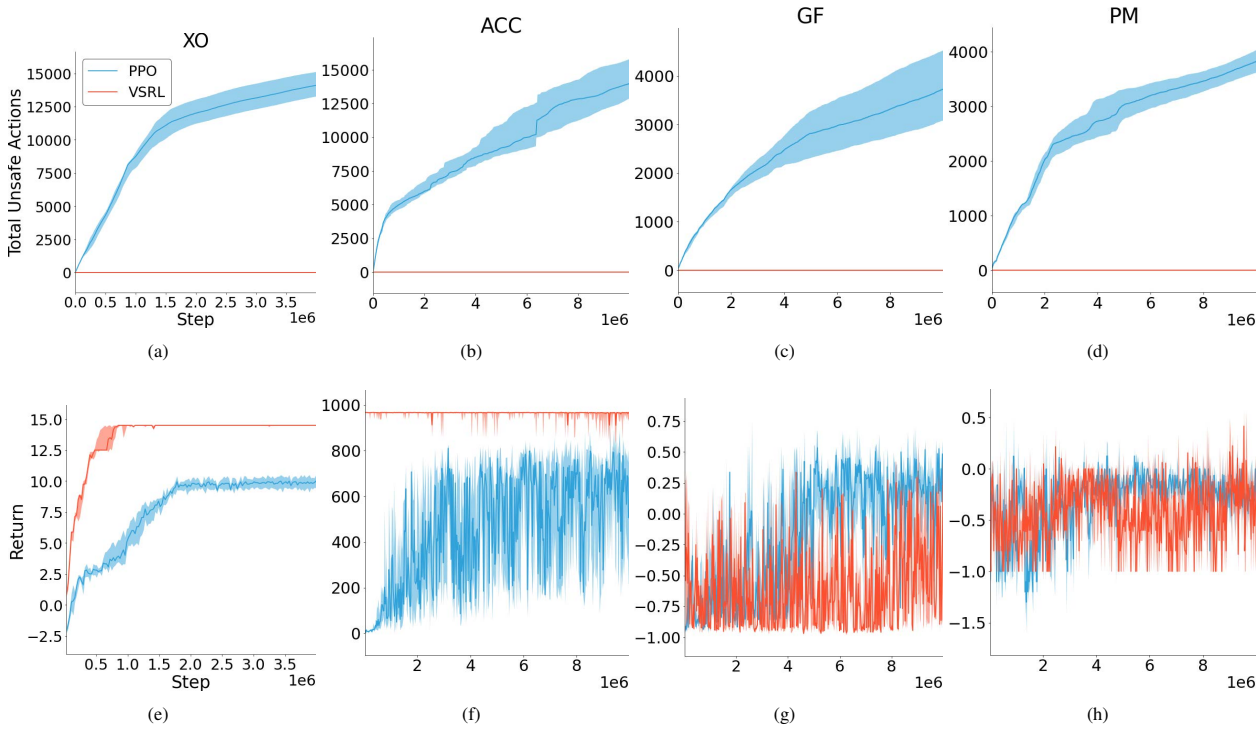


Figure 3. Empirical evaluation of VSRL on all environments. All plots show the median and interquartile range of 4+ repeats. The return is the sum of the reward over an entire episode.

	XO		ACC		GF		PM	
Method	R	U	R	U	R	U	R	U
PPO	10	14108	529	13983	0.233	3733	-0.25	3819
VSRL	14.5	0	967	0	0.228	0	-0.225	0

Table 2. Final reward (R; higher is better) and total number of unsafe actions (U; lower is better) on all environments. All results are the median over at least 4 replicates.

deep RL [12, 47]. Many recent safe exploration methods focus on safety guarantees that hold in expectation (e.g., [1, 44]) or with high probability (e.g., [3, 4, 6, 25]. Some of these approaches achieve impressive results by drawing upon techniques from control theory, such as Lyapunov functions [3] and control barrier certificates.

On the other hand, ensuring safety in expectation or with high probability is generally not sufficient in safety-critical settings where guarantees must hold *always*, even for rare and measure-zero events. Numerical testing alone cannot provide such guarantees in practice [24] or even in theory [41]. The problem of providing formal guarantees in RL is called *formally constrained reinforcement learning (FCRL)*. Existing FCRL methods such as [2, 7, 10, 15, 17–20, 35] combine the best of both worlds: they optimize for a reward function while still providing formal safety guarantees. While most FCRL method can only ensure the safety in discrete-time environments known a priori, Fulton and Platzer [10, 11] introduce *Justified Speculative Control*, which exploits Differential Dynamic Logic[39] to prove the safety of *hybrid systems*, systems that combine an agent's discrete-time decisions with a continuous time dynamics of the system.

A major drawback of current FCRL methods is that they only learn control policies over handcrafted symbolic state spaces. Relative to Fulton and Platzer [10, 11], our primary contribution is that we learn *end-to-end* policies instead of assuming that an oracle extracts a state-space from visual inputs. This has two advantages – unlike the approach in Fulton and Platzer [10, 11], VSRL considers the challenge of imperfect perception and is able optimize for latent features that are *not* in the symbolic safety model's state space. Similarly, while many methods extract a symbolic mapping for RL from visual data, e.g. [13, 14, 27, 28, 30, 31, 48, 49], they all require that all of the reward-relevant features are explicitly represented in the symbolic space. As shown by the many successes of Deep RL, e.g. [34], handcrafted features often miss important signals hidden in the raw data.

Our approach aims at combining the best of FCRL and end-to-end RL to ensure that exploration is always safe with formal guarantees,

while allowing a deep RL algorithm to fully exploit the visual inputs for reward optimization.

7 Conclusion and Discussions

Safe exploration in the presence of hard safety constraints is a challenging problem in reinforcement learning. We contribute VSRL, an approach toward safe learning on visual inputs. Through theoretical analysis and experimental evaluation, this paper establishes that VSRL maintains perfect safety during exploration while obtaining comparable reward. Because VSRL separates safety-critical object detection from RL, next steps should include applying tools from adversarial robustness to the object detectors used by VSRL.

References

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained Policy Optimization. In *International Conference on Machine Learning (ICML 2017) (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 22–31.

[2] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe Reinforcement Learning via Shielding. In *AAAI Conference on Artificial Intelligence*.

[3] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. 2017. Safe model-based reinforcement learning with stability guarantees. In *Advances in neural information processing systems*. 908–918.

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

[5] Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem (Eds.). 2018. *Handbook of Model Checking*. Springer.

[6] Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. 2018. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757* (2018).

[7] Giuseppe De Giacomo, Luca Iocchi, Marco Favorito, and Fabio Patrizi. 2019. Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf Restraining Specifications. In *International Conference on Automated Planning and Scheduling (ICAPS 2019)*.

[8] Nathan Fulton, Stefan Mitsch, Brandon Bohrer, and André Platzer. 2017. Bellerophon: Tactical Theorem Proving for Hybrid Systems. In *International Conference on Interactive Theorem Proving*.

[9] Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. 2015. KeYmaera X: An Axiomatic Tactical Theorem Prover for Hybrid Systems. In *CADE*.

[10] Nathan Fulton and André Platzer. 2018. Safe Reinforcement Learning via Formal Methods: Toward Safe Control Through Proof and Learning. In *AAAI Conference on Artificial Intelligence*.

[11] Nathan Fulton and André Platzer. 2019. Verifiably Safe Off-Model Reinforcement Learning. In *TACAS 2019 (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

[12] Javier Garcıa and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* (2015).

[13] Marta Garnelo, Kai Arulkumaran, and Murray Shanahan. 2016. Towards deep symbolic reinforcement learning. *arXiv preprint arXiv:1609.05518* (2016).

[14] Vikash Goel, Jameson Weng, and Pascal Poupart. 2018. Unsupervised video object segmentation for deep reinforcement learning. In *Advances in Neural Information Processing Systems*.

[15] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. 2019. Omega-Regular Objectives in Model-Free Reinforcement Learning. In *TACAS 2019*.

[16] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. 2018. Logically-constrained reinforcement learning. *arXiv preprint arXiv:1801.08099* (2018).

[17] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. 2018. Logically-Correct Reinforcement Learning. *CoRR* abs/1801.08099 (2018). arXiv:1801.08099

[18] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. 2019. Certified Reinforcement Learning with Logic Guidance. *CoRR* abs/1902.00778 (2019). arXiv:1902.00778 http://arxiv.org/abs/1902.00778

[19] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. 2020. Cautious reinforcement learning with logical constraints. *arXiv preprint arXiv:2002.12156* (2020).

[20] Mohammadhosein Hasanbeig, Yiannis Kantaros, Alessand ro Abate, Daniel Kroening, George J. Pappas, and Insup Lee. 2019. Reinforcement Learning for Temporal Logic Control Synthesis with Probabilistic Satisfaction Guarantees. *arXiv e-prints*, Article arXiv:1909.05304 (Sept. 2019), arXiv:1909.05304 pages. arXiv:1909.05304 [cs.LO]

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[22] Nathan Hunt, Nathan Fulton, Sara Magliacane, Nghia Hoang, Subhro Das, and Armando Solar-Lezama. 2020. Verifiably Safe Exploration for End-to-End Reinforcement Learning. arXiv:2007.01223 [cs.AI]

[23] ISO-26262. 2011. International Organization for Standardization 26262 Road vehicles – Functional safety. (2011).

[24] Nidhi Kalra and Susan M. Paddock. 2016. *Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?* RAND Corporation.

[25] Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. 2018. Learning-based model predictive control for safe exploration. In *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 6059–6066.

[26] Hei Law and Jia Deng. 2018. Cornernet: Detecting objects as paired keypoints. In *European Conference on Computer Vision*.

[27] Yuezhang Li, Katia Sycara, and Rahul Iyer. 2018. Object-sensitive deep reinforcement learning. *arXiv preprint arXiv:1809.06064* (2018).

[28] Junchi Liang and Abdeslam Boularias. 2018. Task-Relevant Object Discovery and Categorization for Playing First-person Shooter Games. *arXiv preprint arXiv:1806.06392* (2018).

[29] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *IEEE international conference on computer vision*.

[30] Keting Lu, Shiqi Zhang, Peter Stone, and Xiaoping Chen. 2018. Robot representing and reasoning with knowledge from reinforcement learning. *arXiv preprint arXiv:1809.11074* (2018).

[31] Daoming Lyu, Fangkai Yang, Bo Liu, and Steven Gustafson. 2019. SDRL: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *AAAI'19*.

[32] Stefan Mitsch, Khalil Ghorbal, and André Platzer. 2013. On Provably Safe Obstacle Avoidance for Autonomous Robotic Ground Vehicles. In *Robotics: Science and Systems*, Paul Newman, Dieter Fox, and David Hsu (Eds.).

[33] Stefan Mitsch and André Platzer. 2016. ModelPlex: Verified Runtime Validation of Verified Cyber-Physical System Models. *Form. Methods Syst. Des.* 49, 1 (2016), 33–74. Special issue of selected papers from RV'14.

[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari With Deep Reinforcement Learning. In *NIPS Deep Learning Workshop*.

[35] Dung Phan, Nicola Paoletti, Radu Grosu, Nils Jansen, Scott A. Smolka, and Scott D. Stoller. 2019. Neural Simplex Architecture. (2019).

[36] André Platzer. 2008. Differential Dynamic Logic for Hybrid Systems. *J. Autom. Reas.* 41, 2 (2008), 143–189.

[37] André Platzer. 2010. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg.

[38] André Platzer. 2012. Logics of Dynamical Systems. In *LICS*. IEEE, 13–24.

[39] André Platzer. 2015. A Uniform Substitution Calculus for Differential Dynamic Logic. In *CADE*.

[40] André Platzer. 2017. A Complete Uniform Substitution Calculus for Differential Dynamic Logic. *J. Autom. Reas.* 59, 2 (2017), 219–266.

[41] André Platzer and Edmund M. Clarke. 2007. The Image Computation Problem in Hybrid Systems Model Checking. In *HSCC (LNCS, Vol. 4416)*, Alberto Bemporad, Antonio Bicchi, and Giorgio Buttazzo (Eds.). Springer, 473–486. https://doi.org/10.1007/978-3-540-71493-4_37

[42] Jan-David Quesel, Stefan Mitsch, Sarah M. Loos, Nikos Arechiga, and André Platzer. 2016. How to model and prove hybrid systems with KeYmaera: a tutorial on safety. *STTT* 18, 1 (2016), 67–91.

[43] Alex Ray, Joshua Achiam, and Dario Amodei. 2019. Benchmarking Safe Exploration in Deep Reinforcement Learning. (2019).

[44] 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) (JMLR Workshop and Conference Proceedings, Vol. 37)*, Francis R. Bach and David M. Blei (Eds.). 1889–1897.

[45] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. (2017). arXiv:1707.06347 http://arxiv.org/abs/1707.06347

[46] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

[47] Weiming Xiang, Patrick Musau, Ayana A Wild, Diego Manzanas Lopez, Nathaniel Hamilton, Xiaodong Yang, Joel Rosenfeld, and Taylor T Johnson. 2018. Verification for machine learning, autonomy, and neural networks survey. *arXiv* (2018).

[48] Fangkai Yang, Steven Gustafson, Alexander Elkholy, Daoming Lyu, and Bo Liu. 2019. Program Search for Machine Learning Pipelines Leveraging Symbolic Planning and Reinforcement Learning. In *Genetic Programming Theory and Practice XVI*.

[49] Fangkai Yang, Daoming Lyu, Bo Liu, and Steven Gustafson. 2018. Peorl: Integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. *arXiv preprint arXiv:1804.07779* (2018).

[50] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. 2019. Objects as Points. *arXiv preprint arXiv:1904.07850* (2019).