Safe Reinforcement Learning under Regularized Probabilistic Counterexample Guidance

XIAOTONG JI and ANTONIO FILIERI, Imperial College London, UK

Safe exploration in reinforcement learning remains a critical challenge for safety-critical autonomous systems, where the typical trial-and-error learning process can lead to hazardous outcomes. While several existing approaches incorporate kinematic models or external knowledge to limit the exploration of unsafe behaviors, their effectiveness is significantly weakened in the presence of incomplete or sparse knowledge. This paper introduces a counterexample-guided reinforcement learning method that navigates safe exploration in autonomous systems without prior knowledge, even when safety and optimality conflict. Our method geometrically abstracts discrete and continuous state-space systems into compact, PAC-learnable models that capture safety-relevant information. We then generate probabilistic counterexamples of the safety requirement to regulate online exploration toward minimizing safety violations, relying on minimal offline counterexample-guided simulations. We further propose a novel belief-based regularization method to address the distributional shift between online and offline learning and to balance optimization and safety, ensuring conservative behavior with theoretical guarantees. Our evaluations demonstrate the effectiveness of the method in significantly reducing safety violations without compromising cumulative rewards when benchmarked against other Q-learning or actor-critic methods with unconstrained or safety-constrained exploration.

CCS Concepts: • **Theory of computation → Reinforcement learning**; **Verification by model checking**; **Abstraction**; **Markov decision processes**.

Additional Key Words and Phrases: safe reinforcement learning, probabilistic model checking, counterexample guidance

ACM Reference Format:
Xiaotong Ji and Antonio Filieri. 2026. Safe Reinforcement Learning under Regularized Probabilistic Counterexample Guidance. 1, 1 (October 2026), 34 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Reinforcement Learning (RL) has emerged as a powerful paradigm for solving complex decision-making problems across diverse domains [8, 62, 72]. However, its deployment in real-world autonomous systems is constrained by the inherent lack of safety guarantees during learning [16]. At its core, RL operates through a trial-and-error process to systematically traverse the decision space and iteratively refines strategies in response to received rewards. While this process enables effective experiential learning, it simultaneously opens the door to potentially unsafe behaviors, posing severe consequences in safety-critical systems.

In response to these safety concerns, a diverse range of safe learning methods have been developed, integrating the robustness of model-driven approaches with the adaptability of data-driven strategies to encourage safer learning [23, 37]. These methods often involve analyzing the kinematic model of the learning system and then refining decision strategies

Authors' address: Xiaotong Ji, xiaotong.ji16@imperial.ac.uk; Antonio Filieri, a.filieri@imperial.ac.uk, Imperial College London, 180 Queen's Gate, South Kensington, London, UK, SW7 2AZ.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
Manuscript submitted to ACM

for safe exploration [2, 22, 56] or optimization under hard or soft safety constraints [1, 66, 68]. However, many state-of-the-art methods are limited in their applicability to systems with partially known or unknown dynamics, where insufficient information is available to characterize unsafe states or events *a priori*.

In [36], we proposed the integration of probabilistic model checking and reinforcement learning by automatically constructing probabilistic counterexamples to a safety requirement and employing these counterexamples as offline counterexample simulations to guide online exploration toward minimizing risky behaviors in a safer manner. A safety-relevant abstraction method provides the probabilistic model checker with a succinct surrogate model, iteratively refined during exploration, enabling scalable counterexample generation to support safer learning in systems without prior knowledge. The foundational framework in [36] produces a safer Q-learning method under counterexample guidance: when safety violations occur with unacceptable frequency during online exploration, controlled and small-scale counterexample simulations are generated to refine the Q-value estimation offline, thus reducing the recurrence of similar safety violations during subsequent online exploration. However, when no safe optimal policy exists, i.e., for a given reward scheme the agent is incentivized to pursue unsafe behaviors to maximize its average reward, offline training simulations focused only on avoiding unsafe behaviors as described in [36] may introduce oscillation in the learning process, where alternating online and offline phases pursue conflicting goals.

To address this potential limitation, in this paper, we extend [36] to introduce a novel belief-based regularization technique that manages the distribution shift between online and offline learning when the safety and optimization goals cannot be simultaneously satisfied. In particular, we extend and complement the foundation in [36] with a regularization strategy that shapes the reward the agent receives during online learning. This is achieved by penalizing actions that were discouraged during counterexample-guided offline learning phases. The proposed regularization helps prevent oscillatory behavior and can be tuned to reflect the agent's tolerance for risk during exploration while providing a theoretical guarantee for achieving a lower bound on the original value function. The resulting algorithm ensures conservative and robust learning performance, promoting safer exploration, and can be integrated into various state-of-the-art RL algorithms. In our implementation, we adapt it for both Q-learning and actor-critic methods, demonstrating the broad applicability of counterexample-guided learning.

Building on the abstraction and counterexample generation strategies from [36], our main contributions include:

- A regularized counterexample-guided learning approach that mitigates safety violations during online exploration through the deployment of small-scale offline counterexample simulations and belief-based regularization, resulting in a conservative performance with theoretical convergence guarantees under mild assumptions.
- Empirical evaluations demonstrating an average improvement of 25% in safety performance over Q-learning and actor-critic baselines across both discrete and continuous environments. In Q-learning, counterexample guidance directly influences Q-value estimation to reduce the likelihood of visiting unsafe states. In actor-critic methods, the guidance is incorporated more smoothly through the critic, enabling safer policy synthesis.

The rest of this paper is organized as follows: Sec. 2 provides foundational background information and preliminary definitions in the domain of RL and probabilistic model checking; Sec. 3 recalls our proposed abstraction strategy from [36], the foundational framework of counterexample-guided learning, and introduces the enhanced regularized and robust counterexample-guided learning algorithm; Sec. 4 presents our evaluation results, showcasing its effectiveness in Q-learning [74], DQN [51], and PPO [59], and in comparison with related safe RL methods [31, 48]; related work in the realm of safe online exploration and safe offline learning is discussed in Sec. 5; finally, Sec. 6 concludes the paper, summarizing our contributions and sketching potential future directions.

2 BACKGROUND

2.1 Problem Framework

Definition 2.1 (**Markov Decision Process (MDP)**). is defined as a tuple $M = (S, A, s_0, P, R, L)$, where:

- S denotes the state space,
- A represents the finite set of actions,
- $s_0 \in S$ is the initial state,
- $P : S \times A \times S \to [0, 1]$, such that $\forall s \in S, a \in A. \sum_{s'} P(s, a, s') = 1$, is the transition probability function,
- $R : S \times A \to \mathbb{R}_{\geq 0}$ is the reward function,
- $L : S \to 2^{AP}$ is the labeling function, assigning atomic propositions (AP) to each state.

A state $s \in S$ is represented as an n_S-dimensional vector, where $n_S \geq 1$. The state space is categorized as *discrete* if S is countable and typically represented as $S \subseteq \mathcal{Z}^{n_S}$, *continuous* if $S \subseteq \mathbb{R}^{n_S}$, and *hybrid* if it encompasses both discrete and continuous elements. For brevity, we omit n_S when it is clear from the context.

Definition 2.2 (**Trajectory**). A finite trajectory through an MDP is defined as a finite sequence $\sigma = \{s_0, a_0, \ldots s_i, a_i, \ldots s_n\}$, where s_0 is the initial state and $P(s_i, a_i, s_{i+1}) > 0$ for all i.

Definition 2.3 (**Policy**). A policy $\pi : S \times A \to [0, 1]$ is a probability distribution over actions $a \in A$ in each state $s \in S$. For a deterministic policy, $\pi(a|s) = 1$ for a specific action a and $\pi(a'|s) = 0$ for all $a' \neq a$.

The performance of the policy π is evaluated using value functions that summarize the long-term consequences of decisions taken by an agent. The *state value function* $V^\pi(s)$ measures the expected cumulative reward starting from state and following policy π: $V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \middle| s_0 = s\right]$. The *state-action value function* $Q^\pi(s, a)$, the Q-function, is defined as the expected return starting from state s taking action a following policy π: $Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \middle| s_0 = s, a_0 = a\right]$.
RL Algorithms aim to learn optimal decision-making policies by estimating value functions to maximize accumulated rewards. These methods can be classified into online RL, which involves direct interaction with the environment to capture real-time changes and feedback, and offline RL [44], which, for the sake of this work, leverages prior experiences to reinforce avoiding risky behavior in subsequent interactions with the real world. This work integrates the strengths of both approaches.
Q-learning (QL) [74], is a model-free RL method that optimizes the decision-making process by estimating the Q-value function Q. The Q-value function is updated by iteratively applying the Bellman operator $BQ(s, a) = \mathbb{E}[R(s, a) + \gamma \max_{a' \in A} Q(s', a')]$, that guarantees convergence to the optimal Q-value function Q^* at the fixed point of BQ, assuming each state-action pair is visited infinitely often. In practice, the direct application of QL is limited in the continuous domain, where the states are not enumerable. To address this limitation, Deep Q-Learning (DQL) methods introduce a parameterized Q-function $Q(s, a; \theta)$, where θ denotes the weights of a neural network used to approximate the Q function. DQL adapts the QL algorithm to apply the Bellman update rule on samples of experience collected at each iteration i to approximate the Q-value by minimizing the loss function [51]: $L_i(\theta_i) = \mathbb{E}_{s,a,r,s' \sim D}\left[\left(BQ(s, a; \theta_{i-1}) - Q(s, a; \theta_i)\right)^2\right]$.
Actor-Critic Methods [38] employ two parameterized networks: the critic network, which approximates the Q-value functions $Q(s, a; \theta)$ or the value function $V(s; \theta)$; and the actor network, which models the policy $\pi(a|s; \phi)$ to maximize the expected return based on the critic's feedback (θ and ϕ being the parameters of the respective networks). The critic network approximates the Q-value function $Q(s, a; \theta)$ under the Bellman operator coupled with the policy B^π

by minimizing the loss function: $L_i(\theta_i) = \mathbb{E}_{s,a,r,s' \sim D}\left[\left(B^\pi Q(s,a;\theta_{i-1}) - Q(s,a;\theta_i)\right)^2\right]$, aiming to reduce the temporal difference error by aligning with the Bellman principle.

To improve the stability and efficiency of policy updates, many actor-critic methods, including Proximal Policy Optimization (PPO) [59], employ the advantage function $A(s,a) = Q(s,a) - V(s)$ to reduce variance. PPO optimizes a clipped surrogate objective defined as $L^{\text{CLIP}}(\phi) = \mathbb{E}_{s,a\sim D}\left[\min\left(r(\phi)A(s,a), \text{clip}\left(r(\phi), 1-\epsilon, 1+\epsilon\right)A(s,a)\right)\right]$, where $r(\phi) = \frac{\pi_\phi(a|s)}{\pi_{\text{old}}(a|s)}$ is the probability ratio between the updated and old policies, and ϵ is a small positive hyperparameter that controls the extent of clipping.

From the safety perspective, this dual-network structure facilitates the direct synthesis of safer policies through a potential safety critic [65], avoiding extensively evaluating the value function for unsafe states and actions.

Regularization and Offline Learning. Offline and hybrid (combination of online and offline) RL algorithms often suffer from distribution shifts due to the lack of interaction with the environment during offline training phases. Regularization is commonly employed to address this problem and stabilize the learning process [18]. The regularized loss function is defined as $L_i(\theta_i) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}}\left[\left(BQ(s,a;\theta_{i-1}) - Q(s,a;\theta_i)\right)^2 + Ł_{reg}(s,a,\theta_i)\right]$, where the regularization function L_{reg} is introduced to adjust the learned Q function approximation by penalizing or rewarding the loss function used to learn it. Regularization functions can take different shapes depending on their intended effect on the learning process; for example, they can be used to learn more conservative strategies by penalizing choices with more uncertain outcomes, or to achieve smoother and more robust learning by penalizing large changes in the parameters θ during each iteration. In this work, we will aim to construct a regularization function to reconcile conflicting goals between online and offline learning experiences.

2.2 Probabilistic Model Checking and Counterexamples

Probabilistic model checking is a formal verification method to reason about stochastic systems. Given an MDP M and a property expressed in a convenient probabilistic temporal logic, probabilistic model checking can verify whether the model satisfies the property [3]. We assume Probabilistic Computational Temporal Logic (PCTL) [27] to specify probabilistic requirements for safety in this work. The syntax of PCTL is recursively defined as follows:

$$\Phi := true \mid \alpha \mid \Phi \wedge \Phi \mid \neg\Phi \mid P_{\bowtie p}\varphi \tag{1}$$

$$\varphi := X\Phi \mid \Phi U \Phi \tag{2}$$

Φ represents a state formula, determining the satisfaction of the property in each state of the model. The semantics of a PCTL property is summarized as follows: $true$ is a tautology satisfied in every state; $\alpha \in AP$ stands for atomic propositions, satisfied in any state labeled with α; $\wedge$ and $\neg$ are the Boolean operators for conjunction and negation, respectively; the modal operator $P_{\bowtie p}\varphi$ holds in a state s if, under any possible policy, the cumulative probability of all paths initiating in s and satisfying the path formula φ fulfills the relation $\bowtie p$. The Next operator $X\Phi$ is satisfied if Φ holds in the next state along a path. The Until operator $\Phi_1 U \Phi_2$ is satisfied by a path originating in s if there exists a time step $k \geq 0$ such that the state s' reached after k steps satisfies Φ_2 and $\forall j < k$, the states visited at step j satisfy Φ_1. The formula $true\ U\ \Phi$ is commonly abbreviated as $F\Phi$ and satisfied by any path that eventually reaches a state satisfying Φ. M satisfies a PCTL property Φ if Φ holds in the initial state s_0 [3].

PCTL allows specifying diverse safety requirements. In this study, we focus on a general safety requirement specified as the upper bound on the probability of eventually reaching a state labeled as `unsafe`[1]:

Definition 2.4 (**Safety Requirement**). The safety requirement for a learning system is formalized by the PCTL property $P_{\leq\lambda}\ [F\ \texttt{unsafe}]$, indicating the maximum probability of reaching any state $\{s \in S \mid \texttt{unsafe} \in L(s)\}$ must be less than or equal to the given safety threshold $\lambda \in (0, 1]$.

In the context of probabilistic model checking, counterexamples serve as evidence of property violation when a model M fails to satisfy a PCTL property [26]. We adapt the *minimal critical label set* method in [75], which formalizes the counterexample generation process as a Mixed-Integer Linear Programming (MILP) problem, to provide crucial insights when discrepancies arise between autonomous system behaviors and the specified safety requirement.

Definition 2.5 (**Counterexamples of the Safety Requirement**). Let $\mathcal{L}$ be the set of transition labels in the (finite-state) MDP model M, $x_\ell \in \{0, 1\}$ be the binary decision variable indicating whether label ℓ is selected, each label ℓ is assigned a non-negative weight $\omega(\ell) \in \mathbb{R}_{\geq 0}$, and $\omega_{\min} = \min\{\omega(\ell) \mid \ell \in \mathcal{L} \wedge \omega(\ell) > 0\}$ denotes the smallest strictly positive label weight. A minimal counterexample is a sub-model $M_{\text{cex}} = (S_{\text{cex}}, A_{\text{cex}}, s_0, P, R, L)$ derived from M, encompassing subsets $S_{\text{cex}} \subset S$ and $A_{\text{cex}} \subset A$ of the original decision space, containing violating behaviors of the safety requirement $P_{\leq\lambda}\ [F\ \texttt{unsafe}]$ initiating from s_0 given in the solution set x_ℓ of the following MILP (adapted from [75]):

$$\text{minimize } -\frac{1}{2}\omega_{\min} p_{s_0} + \sum_{\ell \in \mathcal{L}} \omega(\ell) x_\ell, \text{ such that}$$

$$p_{s_0} > \lambda \tag{3}$$

$$\forall s \in S_U.\ p_s = 1 \tag{4}$$

$$\forall s \in S \setminus S_U.\ \sum_{a \in A(s)} \pi_{s,a} \leq 1 \tag{5}$$

$$\forall s \in S \setminus S_U.\ p_s \leq \sum_{a \in A(s)} \pi_{s,a} \tag{6}$$

$$\forall s \in S \setminus S_U,\ \forall a \in A(s),\ \forall s' \in S \text{ s.t. } P(s,a,s') > 0,\ \forall \ell \in \mathcal{L}(s,a).\ p_{s,a,s'} \leq x_\ell \tag{7}$$

$$\forall s \in S \setminus S_U,\ \forall a \in A(s),\ \forall s' \in S \text{ s.t. } P(s,a,s') > 0.\ p_{s,a,s'} \leq P(s,a,s') \cdot p_{s'} \tag{8}$$

$$\forall s \in S \setminus S_U,\ \forall a \in A(s).\ p_s \leq (1 - \pi_{s,a}) + \sum_{s' : P(s,a,s') > 0} p_{s,a,s'} \tag{9}$$

$$\forall (s,a) \in P_T^{Prob}.\ \pi_{s,a} = \sum_{\ell \in \mathcal{L}(s,a)} x_\ell \tag{10}$$

$$\forall (s,a) \in P_T^{Prob},\ \forall s' : P(s,a,s') > 0,\ \forall \ell \in \mathcal{L}(s,a).\ r_s < r_{s'} + (1 - x_\ell) \tag{11}$$

where S_U is the set of states labeled as `unsafe`, p_s represents the probability of reaching any state in S_U from state s, $\pi_{s,a} \in \{0, 1\}$ indicates that action a is selected in state s, $\ell \in \mathcal{L}(s,a)$ is the label identifying a transition from s to s' via action a (not to be confused with the function L labeling states of the model) such that $x_\ell = 1$ iff the transition is included in the counterexample, $x_\ell = 0$ otherwise, and P_T^{Prob} represents the set of problematic state-action pairs if the minimal probability of reaching the target state is zero and the maximal probability of reaching the target state is non-zero from these state-action pairs.

[1]Notice that while we focus for simplicity on reachability properties, the broader ω-regular fragment of PCTL can be reduced to reachability properties [3, Chapter 10.3], allowing safety specifications beyond the simple reachability of `unsafe` states with the same procedures presented in this paper.

Intuitively, this optimization problem aims to find the smallest counterexample sub-model, composed of all the transitions (s, a, s') from the original model whose corresponding x_ℓ is 1, that will make the agent violate the safety specification. The decision variables are $x_\ell \in \{0, 1\}$, $\pi_{s,a} \in \{0, 1\}$, $p_s \in [0, 1]$, $p_{s,a,s'} \in [0, 1]$, and $r_s \in \mathbb{R}$; all other quantities (e.g. the transition probabilities $P(s, a, s')$ and label sets $\mathcal{L}$) are constants derived from the given MDP M. To this goal, Eq. 3 requires the probability of reaching an unsafe state from the initial state s_0 to be $> \lambda$ (violation of the safety specification); Eq. 4 fixes $p_s = 1$ for all the unsafe states; Eq. 5 imposes the agent to select at most one action a for each state s; if no action is chosen in a state s, Eq. 6 ensures that $p_s = 0$. Other constraints for ensuring the minimal size and preventing deadlock loops are also defined [75]. Eq. 7 ensures that the contribution of the transition (s, a, s') is 0 if such transition is not included in the counterexample, otherwise, Eq. 8, $p_{s,a,s'}$ is bounded by the probability of the corresponding transitions in the model – $P(s, a, s')$) times the probability of reaching the target from s' $p_{s'}$. Eq. 9 ensures that if action a is selected in state s, the probability of reaching S_U from s is bounded by the sum of the probabilities of reaching S_U from its successors given a. Finally, Eq. 10 and 11 prevent the scheduler from staying within the problematic region P_T^{Prob}: Eq. 10 ensures that whenever a problematic state-action label is selected, at least one successor transition is activated as a progress step; Eq. 11 assigns ranks r_s to problematic states and enforces that these ranks strictly increase along such progress steps, this eliminates cycles within the problematic region and thereby guarantees eventual reachability of an unproblematic state [75].

Generating Multiple Counterexamples. An MDP M that does not satisfy the safety requirement may have multiple counterexamples, either minimal or non-minimal in size (i.e., in number of states). Each of these can highlight different strategies leading to safety violations [71]. In this paper, we aim at generating multiple counterexamples, if they exist, in order to penalize any strategy that leads to requirements violations, while ensuring each counterexample remains compact to expedite the training process. Given a solution of the MILP (Def. 2.5), we can derive additional counterexamples by incorporating a blocking clause $x_\ell = 0$ into the problem constraints, compelling the optimizer to exclude one or more previously selected action pairs. This results in a systematic method to generate a variety of diverse counterexamples with an increasing number of blocking clauses, offering a more comprehensive representation of the different violating behaviors the agent may explore over time.

3 SAFE REINFORCEMENT LEARNING WITH REGULARIZED COUNTEREXAMPLE GUIDANCE

We propose a safe reinforcement learning method under regularized counterexample guidance to regulate exploration with minimal risky behaviors, requiring no prior knowledge of the environment. This method operates through an iterative online-offline learning process, enabling the agent to bootstrap acquired safety-relevant knowledge with offline counterexample simulations and mitigate risky behaviors within online concrete environments. We begin by recalling from [36] an incremental abstraction method that preserves the safety-relevant information about the explored environment (Sec. 3.1), as well as the original counterexample-guided offline learning procedure (Sec. 3.2), demonstrating how it may introduce oscillations when safety and reward maximization conflict.

Finally, in Sec. 3.3, we introduce a framework for online-offline learning that extends the applicability of the approach to both Q-learning and actor-critic methods. In Sec. 3.4, we discuss the limitations of [36] and other safe learning methods that assume the existence of a safe-optimal policy. We then present a novel regularization technique designed to prevent oscillatory behavior arising from conflicting objectives between safety and optimization with a theoretical guarantee of conservative learning performance, serving as a lower bound on the original value function in Sec. 3.5.

3.1 Safety-relevant State-space Abstraction

We restate the abstraction procedure to construct a finite abstract MDP that retains sufficient safety-relevant information from the currently explored concrete MDP M, as originally introduced in [36]. With the absence of prior knowledge about the environment, we formalize each online (*episode*) as a sequence of explored states and actions, and refer to the states in the online environment as *concrete* states $s \in S$. Each state s is assigned a set of labels, representing properties related to the learning objective. Specifically, the `unsafe` label identifies the states associated with unsafe behaviors that terminate this episode with a failure signal. Any concrete states of the online environment that are not yet labeled as `unsafe` are assumed `safe` to explore. This `unsafe` label is time-invariant, i.e., a state recognized as unsafe is assumed to remain dangerous to visit through the proceeding of the learning process. In particular, this entails that, before exploring any concrete state for the first time, the agent assumes it to be safe – unless the user provided initial knowledge – and as soon as a new concrete state is explored and identified as unsafe, the abstraction is updated accordingly. The purpose of the abstraction is to incrementally record and refine the explored concrete state space while conservatively maintaining safety-relevant information.
Consider the aim of safe exploration, the abstraction must maintain a **safety invariant**: *every explored unsafe concrete state should be mapped to an unsafe abstract state.* This requires that the finite abstract state space must distinctly separate safe and unsafe regions of the concrete space, and, conservatively, only safe states may be inaccurately classified as unsafe, not vice versa. Inspired by [10, 60] that cast geometric learning concepts as a set cover problem, our abstraction approach frames the geometric separation task of safe and unsafe states as a minimal red-blue set cover problem. This yields a family of geometric predicates whose boundaries induce a finite partition of S into disjoint abstract regions.

Definition 3.1 (**Safety-relevant Abstract MDP**). The abstract MDP is a tuple $M_a = (S_a, A_a, s_{a0}, R_a, P_a, L_a)$ with an abstraction map $\alpha : S \to S_a$, where S_a is a finite set of abstract states forming a partition of the concrete state space S, i.e., $\bigcup_{s_a \in S_a} s_a = S$ and $\forall s_a \neq s'_a,\ s_a \cap s'_a = \emptyset$, A_a is the set of applicable actions, $s_{a0} = \alpha(s_0)$ is the initial abstract state, $P_a : S_a \times A_a \times S_a \to [0, 1]$ is the abstract transition function, R_a is the abstract reward function, and $L_a : S_a \to \{\texttt{safe}, \texttt{unsafe}\}$ is an abstract labeling function.

The abstract model M_a is constructed from online episodes by mapping each concrete transition (s_t, a_t, s_{t+1}) to an abstract transition $(\alpha(s_t), a_t, \alpha(s_{t+1}))$ and computing P_a from the corresponding empirical frequencies: for each abstract state–action pair (s_a, a), we count observed abstract transitions (s_a, a, s'_a) induced by the mapping α and set $P_a(s_a, a, s'_a)$ to the normalized count over all successors s'_a, so that $\sum_{s'_a} P_a(s_a, a, s'_a) = 1$ whenever (s_a, a) is observed. The labeling function L_a is constructed to preserve the safety invariant. Initially, the entire state space S_a is assumed safe to explore. As exploration proceeds, α is incrementally refined to incorporate newly discovered unsafe states and preserve the safety invariant. To instantiate the abstract partition and the map α from explored data, we learn a set of geometric predicates $C = \{C_i\}$ that conservatively captures the explored `unsafe` region via a minimal red-blue set cover formulation. We then derive S_a and α from the predicate boundaries, as described below.

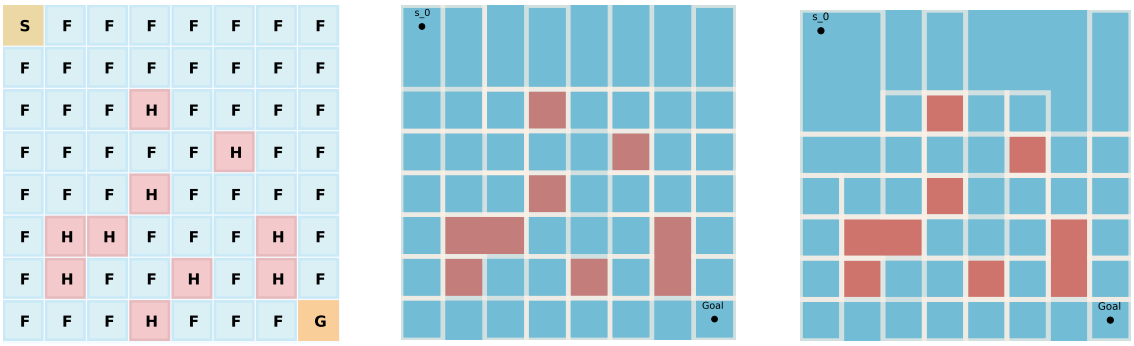
Definition 3.2 (**Minimal Red-Blue Set Cover Problem**). Let $\bar{S} \subseteq S = \{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_n\}$ be the set of all explored concrete states and $S_U \subseteq \bar{S} = \{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_m\}$ be the set of explored states assigned the `unsafe` label. Find a minimal predicates set $C = \{C_1, \ldots, C_K\}$ such that every element $\mathbf{s} \in S_U$ is covered by some predicate $C_i = \{\mathbf{s} \mid \boldsymbol{\omega}\mathbf{s} + \mathbf{b} \leq 0\}$, where $\boldsymbol{\omega}$ is the vector of slopes and $\mathbf{b}$ is the vector of biases corresponding to half-spaces enclosing the predicate, with an overall false positive rate $f' \leq f \in (0, 1]$ for safe concrete states ($\mathbf{s} \in \bar{S} \setminus S_U$) covered by $\bigcup_{i=1}^{K} C_i$.

Remark 3.1. The general affine form of the predicate C_i accommodates various common numerical abstract domains, including boxes (intervals), octagons, zonotopes, or polyhedra. We fix $\omega = 1$ to constrain the abstract domains to hyper-boxes in this work, each box with a minimum size $d > 0$, ensuring no dimension of the box is reduced to a length smaller than d. This specification coarsens the abstract domain and balances the trade-off between computational efficiency and the precision of the abstraction, and it showed promising results in our experiments (see Sec. 4 for additional discussion). The false positive rate threshold f cannot be zero in general, since the concrete state space (whether discrete or continuous) may not be perfectly partitioned by a finite set of polyhedra predicates. Lower values of f possibly result in more predicates in $|C|$. These parameter restrictions are adaptable for different applications, allowing recalibration of the trade-off, as discussed in [63].

This set-cover optimization problem can be solved by employing a branch and bound method [42] to systematically enumerate possible combinations of predicates and induce a solution set C with coverage of all unsafe concrete states by a Boolean combination of predicates with a maximum tolerable false positive rate (i.e., safe states misclassified as unsafe) f. At each branch point separating an unsafe region from a safe one, the points laying on the separating hyperplane are conservatively included within the unsafe regions ($\leq$). For hyperplanes separating two safe regions or two unsafe regions, the algorithm arbitrarily assigns the points along the hyperplane to either of the regions, inducing a partition of the concrete state space. The solution is incrementally updated as the exploration progresses to store the acquired knowledge and ensure that the safety invariant is consistently maintained throughout the exploration.
As safety-relevant information is acquired incrementally during exploration and given that the state space to be explored could be extremely large or infinite, there are potential computational challenges posed by the incremental branch-and-bound refinement, which may result in excessive fragmentation of the abstract state space. To mitigate this issue, we further adapt the concept of probabilistic approximate $\epsilon-$simulation [14] to merge adjacent states (states sharing at least one separating hyperplane) within the abstract state space S_a:

Definition 3.3 (**Adjacent ϵ-simulation**). Adjacent states $s \in S_a$ and $s' \in S_a$ are ϵ-similar if $|P_a(s, a, s_l) - P_a(s', a, s_l)| \leq \epsilon$ for all $s_l \in S_l$ and for all $a \in A_a$, where $\epsilon \in [0, 1]$ is a tolerance parameter and S_l is the partitions of S_a induced by the equivalence relation $s \sim s'$ iff $L_a(s) = L_a(s')$.

This simulation induces a hierarchical merging scheme. The initial level l_0 comprises the initial abstract states, partitioning the explored concrete state space into a finite set of boxes – one corresponding to each abstract state – and labeling as either safe or unsafe. At each subsequent level l_i, ϵ-similar adjacent states – i.e., states separated by a hyperplane – are merged into a single state iteratively until no further merge is possible. Beyond the size and cost reduction, this hierarchical scheme also benefits from a finer-grained approximation of concrete state space regions in proximity of explored unsafe states. After merging ϵ-similar adjacent states, we construct the new abstract transition matrix by aggregating the previously collected transition counts of the merged states (mapping old targets to their merged representatives) and renormalizing per merged state–action pair to obtain a valid stochastic matrix.
Example. Fig. 1 illustrates the construction on FrozenLake8×8. In the concrete MDP, each grid cell is a concrete state, and the cell labeled `S` is the initial state, `F` are frozen states, `G` is the goal state, and states labeled `H` are unsafe. During online exploration, the agent collects concrete transitions (s, a, s') and discovers unsafe cells when they are reached. The observed unsafe cells are treated as red points and the explored non-unsafe cells as blue points in the red-blue set cover problem. The learned box predicates cover all observed unsafe cells, while allowing at most a fraction f of explored safe cells to be conservatively included as false positives. The predicate boundaries induce the abstract states in S_a, each of which is a region of the concrete state space, possibly containing multiple concrete cells. The abstraction



(a) Concrete MDP (b) Safety-relevant abstract MDP (c) After adjacent ϵ-simulation

Fig. 1. Illustration of the abstract MDP construction on FrozenLake8×8: (a) the concrete MDP contains one concrete state per grid cell; the agent starts from S, aims to reach G, and treats hole cells H as unsafe; (b) the safety-relevant abstract MDP, which partitions the concrete state space into abstract regions induced by learned geometric predicates, where each colored region corresponds to one abstract state and may contain multiple concrete states; (c) the abstract MDP after adjacent ϵ-simulation merging.

map α sends each concrete state to the abstract region containing it. Each observed transition (s, a, s') is then mapped to $(\alpha(s), a, \alpha(s'))$, and the empirical frequencies of these mapped transitions define P_a. Finally, adjacent abstract states with the same safety label and ϵ-similar transition distributions may be merged. This design naturally preserves a finer abstraction near observed unsafe regions, where the learned predicate boundaries and the transition probabilities to unsafe states distinguish neighboring abstract states. The granularity of the abstract MDP is overall determined by the observed safety-relevant samples, the false-positive threshold f, the minimum box size d, and the ϵ-similarity merging criterion. The full incremental refinement process and the effect of different false-positive thresholds f and merging tolerances ϵ are reported in Appendix A.2.

3.2 Counterexample-guided Offline Simulations

The abstract MDP M_a enables the generation of counterexamples to the safety requirement by solving the MILP (as detailed in Def. 2.5). Each counterexample M_{cex} is a (near-)minimal sub-model of the abstract MDP M_a, capturing potential safety violations in the concrete environment.

Within each simulation model M_{cex}, the agent starts from s_0, selects an action for each state $s_{\text{cex}} \in S_{\text{cex}}$ to simulate a transition based on the transition function P to a destination state s'_{cex}, and receives a penalty if s'_{cex} is assigned an *unsafe* label. Each abstract state s_{cex} is concretized to a concrete state s by sampling from historic concrete episodes, where the concrete states $s \sim s_{\text{cex}}$ and $s' \sim s'_{\text{cex}}$, to form a simulated trajectory following the same form as the concrete online trajectory. Each simulated trajectory terminates upon reaching an unsafe state or when an abstract transition fails to be concretized, which may occur when concrete safe states are misclassified as unsafe in the abstraction but no actual transition to unsafe states exists from them. In the latter scenario, the corresponding simulation path is discarded. Although every simulation within the counterexample is designed to eventually reach an unsafe state with probability 1 by construction [75], to prevent excessively lengthy simulations, it is practical to impose an (arbitrarily large) upper limit on the maximum number of steps per simulation run as an additional termination criterion.

It is worth acknowledging that the solution to the MILP problem in Def. 2.5 may not be unique, and different solutions can reveal different violating behaviors [71]. Hence, for more comprehensive training, multiple counterexamples

are generated from the current abstract model M_a to foster the diversity of violating behaviors to be simulated. Multiple counterexamples can be obtained by adding blocking clauses that force arbitrary labels selected in existing counterexamples to be deselected by constraining the corresponding $x_\ell = 0$, in random order as described in Sec. 2. Each solution is typically small-sized, resulting in short simulation traces and, in turn, cost-effective learning.

3.3 A Framework for Online-Offline Learning with Counterexample Guidance

In this section, we introduce a counterexample-guided RL framework that integrates online and offline learning, as outlined in Alg. 1. This framework is a generalization of the method we proposed in [36] to encompass both QL and actor-critic methods with the introduction of regularization, demonstrating the adaptability of counterexample guidance with different learning algorithms.

We will first recall the baseline implementation of the framework from [36], without regularization. We will then highlight in Sec. 3.4 its limitations in the presence of conflicts between reward maximization – pursued during online learning – and safety assurance – pursued during offline learning. Finally, Sec. 3.5 will introduce a new regularization strategy to make the learning converge to a stable policy that controllably trades off the conflicting goals.

Algorithm 1: Counterexample-Guided Reinforcement Learning

1 **Input:** Safety requirement threshold $\lambda > 0$, max number of CEX N_{cex}, number of simulations per CEX NS_{cex}, Bayes factor β
2 **Output:** Synthesized Policy π
3 **Initialize:** Abstract model M_a, experience dataset $D \leftarrow \emptyset$, Q-value function Q and policy π (randomly initialized)
4 **while** *not converged* **do**
5 `// Online Learning;`
6 $\pi, Q, D_{\text{online}} \leftarrow$ `onlineQLearning/onlineActorCritic`(π, Q) ;
7 $\delta_{\text{bht}} \leftarrow$ `BayesianHypothesisTester`$(D_{\text{online}}, \lambda, \beta)$;
8 $D \leftarrow D \cup D_{\text{online}}$;
9 **if** $\delta_{bht} > \beta$ **then**
10 `//Offline learning;`
11 $M_a \leftarrow$ `refine`(M_a, D) `// Update the abstract MDP;`
12 **for** *up to* N_{cex} **do**
13 $M_{\text{cex}} \leftarrow$ `cexGeneration`(M_a, λ);
14 simTraces $\leftarrow$ collectSimulationTraces$(M_{\text{cex}}, NS_{\text{cex}})$;
15 $Q \leftarrow$ updateQ/updateCritic(Q, simTraces) `// Q-Value / Critic update using cex-guided simulation traces;`
16 **end**
17 **end**
18 **end**

Initialization. Both the abstract model M_a and the explored dataset D are initially empty, or they can optionally be initialized with any prior knowledge if available (e.g., abstracting a set of traces collected during previous runs of the system). Similarly, the Q-value function and policy are randomly initialized, or they can optionally be initialized based on prior knowledge, if available.

Online learning. The `onlineQLearning/onlineActorCritic` procedure (line 6) represents the method that the agent uses to explore and learn from the concrete environment. In particular, it defines how rewards are used to update value estimation. The two alternative functions in Alg. 1 highlight the orthogonality of the algorithm to the specific

learning method adopted by the agent, where `onlineQLearning` represents the value update rule of Q-learning and `onlineActorCritic` the value update function of an actor-critic method, e.g., PPO. The safety of online exploration is estimated using a sequential Bayesian hypothesis testing [15] (line 7) to monitor the frequency of safety violations. This testing iteratively updates a Beta distribution estimating the probability of violation from the current experience of the agent. If the likelihood of "the violation probability exceeding λ" surpasses a specific Bayes factor β, the online learning phase is paused, the explored dataset D is expanded with the newly collected traces (line 8), and the subsequent offline learning process is triggered.

Remark 3.2 (**Bayesian hypothesis test**). Let $X_i \in \{0, 1\}$ denote whether the i-th online episode reaches an `unsafe` concrete state, and let $p = \Pr(X_i = 1)$ be the probability of reaching an unsafe state in the current online phase. A conjugate prior for this distribution is Beta $p \sim \text{Beta}(\alpha_0, \beta_0)$, $\alpha_0, \beta_0 > 0$ [15]. After $n \geq 1$ episodes in the online learning phase, we update the prior with $k = \sum_{i=1}^{n} X_i$ unsafe episodes, yielding the posterior $p \mid \mathcal{D} \sim \text{Beta}(\alpha_0 + k, \beta_0 + n - k)$. The hypothesis test compares $H_0 : p \leq \lambda$ (safety requirement satisfied) against $H_1 : p > \lambda$ (violation). We then trigger offline learning when the evidence for H_1 exceeds the Bayes factor β in Alg. 1. If a prior expectation is available, e.g., from historical data or similar applications, (α_0, β_0) can be set accordingly – for example, setting the ratio $\alpha_0/(\alpha_0 + \beta_0)$ equal to the a-priori expected failure probability, while the concentration $\alpha_0 + \beta_0$ determines the confidence in the initial prior, i.e., larger values of $\alpha_0 + \beta_0$ require more evidence from the collected data to move the posterior away from the prior expectation. In this paper, we do not assume any prior knowledge available and instead adopt the non-informative Jeffreys prior $\alpha_0 = \beta_0 = 0.5$, which is symmetric and converges quickly to the likelihoods observed from the sample data. After each triggered offline phase, we reset the test (i.e., restart from the prior) to assess safety under the updated behavior.

Offline learning. If the Bayesian test interrupts the online learning phase (line 9), an offline learning phase begins, aiming to prevent the recurrence of identified unsafe behaviors in subsequent online explorations. The abstract MDP M_a is refined with the current set of online explored paths D (line 11), enabling the generation of counterexamples. While the set of all possible counterexamples is finite [75], the computational cost for their exhaustive generation may be prohibitive for large M_a. Thus, we impose a maximum of N_{cex} counterexamples to be generated for each offline learning phase, using random blocking clauses to likely increase their diversity.

The offline simulation paths, derived from each M_{cex}, are submitted to the agent (line 15), penalizing the selection of potentially unsafe actions ahead of the next online learning phase. The Bayesian hypothesis test is then reset before entering the subsequent online learning phase (line 7), expecting a different agent behavior after each offline learning phase.

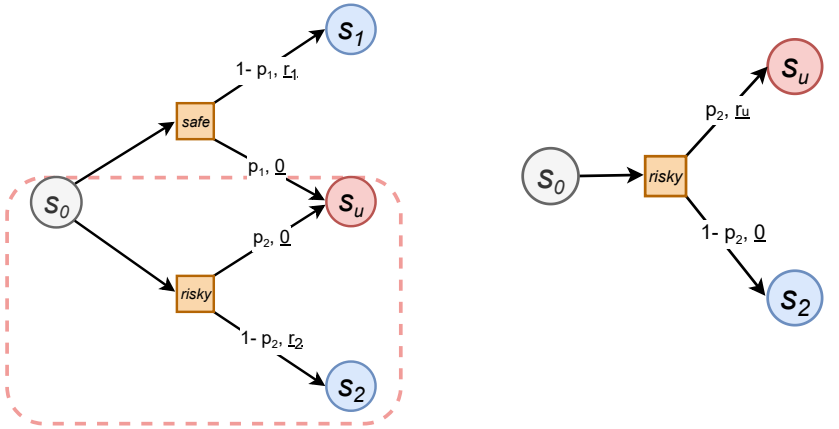
By exposing the agent to simulated execution traces that are focused on penalizing unsafe behaviors, Alg. 1 can be used with virtually any learning algorithm. Considering specifically QL and actor-critic methods, their approaches to incorporate the counterexample guidance are different: in QL, the counterexample guidance directly adjusts strategies through a penalized Q-value function to avoid visiting unsafe states; in actor-critic methods, the counterexample guidance adjusts the critic to implicitly influence the policy toward safer exploration. From a safety perspective, the explicit effect of counterexample guidance is to penalize the over-estimated Q-value and boost safer exploration in QL; on the other hand, its impact in actor-critic methods can be potentially smoother but more effective as it avoids extensive evaluations of the Q-value function in unsafe regions. We will further evaluate and discuss the characteristics of counterexample-guided learning in QL and actor-critic methods in Sec. 4.

3.4 Limitations when handling conflicting goals

The effectiveness of the basic counterexample-guided offline simulations as defined in [36] is influenced by the relationship between reward optimization and safety for the learning problem at hand. If the learning objective of maximizing rewards aligns inherently with the safety requirements, i.e., there exists a reward-optimal policy that is also safe, counterexample guidance can increase the safety of the agent during exploration and possibly speed up convergence to an optimal safe policy. This is captured by the following proposition adapted from [36]:

Proposition 3.1. If there exist maximal-reward policies that satisfy the safety requirement, then the counterexample-guided learning algorithm will eventually converge to one of them.

In other words, offline learning with counterexample guidance does not preclude the convergence to a maximal reward policy that satisfies the safety requirement, if such policy exists. This follows from the fact that the online exploration of the environment is never strictly limited by the use of counterexample guidance. Rather, offline phases aim at penalizing the relative value of actions that may eventually lead to safety violations, thereby reducing their likelihood of selection during subsequent online learning (a more detailed discussion can be found in Appendix A.1).
However, in the absence of a maximal-reward policy satisfying the safety requirement, alternating online learning with counterexample-guided offline simulations may cause oscillations in the agent's value estimation. Such oscillations arise from the fact that online learning steers toward a maximal-reward policy, while the offline policy, reinforced within counterexample-guided simulations, upholds the safety requirement, leading to a conflict. In such scenarios, counterexample guidance can still reduce the number of failures during exploration but it cannot entirely prevent the agent from converging again to an unsafe maximal-reward policy in the long run.



(a) A minimal MDP M_a, where $S_a = \{s_0, s_1, s_2, s_u\}, A_a = \{safe, risky\}$.

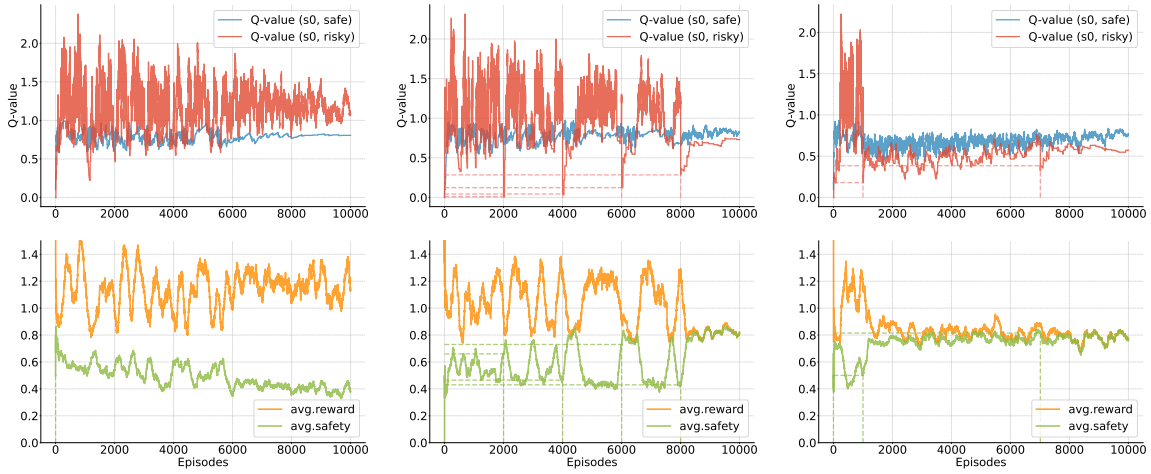
(b) Counterexample sub-model M_{cex} constructed from M_a.

Fig. 2. A minimal (abstract) MDP illustrating oscillating behavior induced by counterexample guidance in the presence of conflicts between the safety requirement and reward maximization. The safety requirement is $P_{\leq p_1}\,[F\,\texttt{unsafe}]$, with $p_1 < p_2$. s_u is the unsafe state.

Example of Oscillating Behavior. Consider the MDP M_a in Fig. 2(a), and assume the abstraction already converged to an exact representation of the environment. From the initial state s_0, the agent can choose between two actions:

- `safe` leads with probability p_1 to the unsafe state s_u and $1 - p_1$ to state s_1 where reward r_1 is granted
- `risky` leads with probability p_2 to the unsafe state s_u and $1 - p_2$ to state s_2 where reward r_2 is granted.

The expected value of the two actions is $v_1 = (1 - p_1) \cdot r_1$ for action `safe` and $v_2 = (1 - p_2) \cdot r_2$ for action `risky`, respectively. Let us assume that $v_2 > v_1$ and $p_1 < p_2$, and that the safety requirement is $P_{\leq p_1}\ [F\ \texttt{unsafe}]$.
During online learning, the agent's policy will converge toward selecting the `risky` action, as it would grant the highest expected value. However, such an exploration phase is bound to trigger an offline phase as soon as the Bayesian estimate of the failure rate exceeds p_1 while converging to p_2, signaling the violation of the safety requirement.
The counterexample sub-model that will guide the offline phase is shown in Fig. 2(b), where the offline episodes will result in a penalty $r_u < 0$ for reaching s_u via action `risky`. The net effect will be that after the offline phase, the agent will prefer action `safe`, only to eventually drift again toward `risky` to maximize its online reward.
Fig. 3(b) shows how the conflicting goals in this scenario lead to oscillations in the estimated values for the two actions (we use QL for this finite state example). The oscillating effect introduced by counterexample guidance can be observed by comparing Fig. 3(b) with the baseline QL process reported in Fig. 3(a). In the latter, without offline learning penalizing the selection of the `risky` action, the agent converges to selecting `risky` for its higher reward. In Fig. 3(b), the penalizing effect of offline learning phases on the estimated value of `risky` ends up only temporarily affecting the agent's preference for this action, ultimately resulting in the oscillating behavior visible in the plot (whose magnitude is smoothed over time by the decreasing learning rate of the agent).



(a) Classical QL (b) Online-offline Counterexample-guided QL (c) Regularized Counterexample-guided QL

Fig. 3. Evolution of Q-value estimations in $Q(s_0, \mathit{safe})$, $Q(s_0, \mathit{risky})$, average rewards and safety rates during the learning process for the model shown in Fig. 2(a) using classical QL, Online-offline counterexample-guided QL and Regularized Counterexample-guided QL (the model parameters used for this example are: $p_1 = 0.8, p_2 = 0.6, r_1 = 1, r_2 = 3, r_u = 0$ and $\rho = 2$ with a decaying learning rate a_t, where $a_0 = 0.01$ and the decay factor is $\gamma = 0.99998$

3.5 Safe Reinforcement Learning with Regularized Counterexample Guidance

To avoid oscillating behaviors in case no maximal-reward policy can assure the satisfaction of the safety requirement, in this section, we introduce a belief-based regularization term in the online learning goal that controls the distribution

shift between online and offline learning phases. We will first recall in more detail the effects of counterexample-based simulations under conflicting objectives of optimization and safety and then present the belief-based regularized Bellman operator we propose to control the distribution shift and the oscillations in value estimation.

Consider again the model shown in Fig. 2(a). The agent can indeed reduce the frequency of visiting the unsafe state s_u through Alg. 1 by penalizing the Q-value $Q(s_0, \texttt{risky})$ within the offline counterexample-guided learning (counterexample in Fig. 2(b)), even when such actions may lead to higher rewards. For instance, with $p_1 = 0.8$ and $p_2 = 0.4$, rewards $r_1 = 1$ and $r_2 = 1.5$, the agent is encouraged to select the `safe` action with a smaller reward under the counterexample guidance. However, in real-world scenarios, risky behaviors in autonomous systems could potentially lead to significantly larger rewards, tempting the agent to opt for risky actions. For example, if the agent would instead receive $r_2 = 3$ for selecting the `risky` action toward s_2, it may eventually show a preference for this higher-risk action with higher Q-value during online learning. This preference leads to the oscillation of value estimations for counterexample-guided RL due to conflicting objectives of maintaining safety and maximizing reward shown in Fig. 3(b), resulting in recurring safety violations and frequent triggering of offline learning throughout the exploration process (the points in time when offline learning is triggered are shown with dashed vertical lines in the figure).

To manage a possible conflict between reward maximization and safety, we introduce a regularized Bellman operator [76] T that incorporates a belief-based regularization term for the Bellman update rule during online exploration. This operator is formulated to penalize the estimation of the value of an action according to the difference between the (latest) online and offline value estimations. In other words, if an action is exercised during offline learning – because part of a path sampled from counterexamples – its value will be lower after the offline phase is completed; then, the subsequent online phase will add a regularization term correlated to the drop in value of the action induced by the previous offline phase. The strength of the penalization is tunable between the extremes of ignoring the value drop after an offline phase (i.e., always prefer maximal reward) and penalizing future online estimates to quickly match the value drop at each online-offline iteration (i.e., always prefer maximal safety).

Regularization in Q-value. Our regularized Bellman operator for Q-value estimation is presented in Eq. (12):

$$TQ(s,a) = \mathbb{E}_{(s,a,r,s')\sim D}\left[R(s,a) + \gamma \max_{a' \in A} Q(s',a') - L_{\texttt{reg}}(s,a)\right] \tag{12}$$

where the regularization term $L_{\text{reg}}(s,a) = (1-b(s,a)) \cdot [\hat{Q}(s,a) - Q_{\text{cex}}(s,a)]$ accounts for the loss due to "overestimation" in the Q-value $\hat{Q}$ from the latest online learning compared to the corresponding counterexample-guided estimation Q_{cex}; the belief $b(s,a) = e^{-\rho[\hat{Q}(s,a) - Q_{\text{cex}}(s,a)]_+}$ modulates the magnitude of this loss by considering the divergence between the latest online estimated Q-value $\hat{Q}$ and the corresponding value Q_{cex} penalized via counterexample-guided simulations[2]; notice that we only regularize for actions where the difference $\hat{Q}(s,a) - Q_{\text{cex}}(s,a)$ is greater than zero ($[\cdot]_+$ returns its argument if the argument is positive and zero otherwise). The *meta-parameter* $\rho \geq 0$ serves as a risk preference indicator, allowing the user to strengthen or weaken the bias of the belief at each online-offline iteration. A smaller value of ρ encourages divergence from the counterexample-guided estimation, i.e., taking more risks during the following online phase and therefore preferring reward maximization. On the other hand, a larger value of ρ encourages conservative value updates that more strongly penalize unsafe actions (explored thus far). At its extremes, when $\rho \to 0$, the regularization term also tends to 0, reducing Eq. (12) to the standard Q-learning update rule for online learning;

[2]while in this work we formulated the regularization term using a belief structure, alternative divergence metrics, such as the Kullback-Leibler (KL) divergence [40] and Maximum Mean Discrepancy (MMD) [24], widely-used in several offline RL works [5, 41], can also be adapted to define regularization terms based on the distribution drift $[\hat{Q}(s,a) - Q_{\text{cex}}(s,a)]_+$.

instead, when $\rho \to \infty$, the Q-value estimation will be offset by the difference between the online and offline estimates, thus retaining the full penalization induced by the previous offline phase.

To intuitively demonstrate the effect of our regularization strategy, dubbed regularized counterexample-guided learning, we refer again to the evolution of Q-value estimations reported in Fig. 3 for the example model of Fig. 2.
As already discussed, pure online Q-learning (Fig. 3(a)), i.e., without any counterexample-guided offline action, converges to a policy maximizing the expected reward – action *risky* – irrespective of whether such policy satisfies the safety requirement or not. With online-offline counterexample-guided QL [36], offline phases penalize the `risky` action because its selection leads to violating the safety requirement. While this penalization temporarily reduces the frequency of visits to unsafe states, it does not prevent subsequent online phases from eventually converging again to the maximal reward policy – `risky` – thus repeatedly triggering offline phases leading to oscillations in the Q-value estimations (Fig. 3(b)). This is consistent with prop. 3.1, which ensures the convergence of online-offline counterexample guidance only when a safe maximal-reward policy exists [36], which is not the case for the model in Fig. 2. Finally, Fig. 3(c) shows the evolution of Q-value estimates when our new regularization strategy is applied ($\rho = 2$ in this example). The regularization term controllably biases the value estimation for the `risky` action, transferring a *safety bias* from counterexample-guided offline learning to the subsequent online phases. As a result, the agent will increasingly prioritize visiting safe actions also during online learning, eventually converging to a safe policy and reducing the need for offline corrections – the frequency of dashed vertical lines in the plots reduces over time.

Based on the regularized target induced by the operator T in Eq. (12) the learning procedures at line 6 of Alg. 1 are replaced by corresponding regularized versions, which shape the rewards received from the environment as appropriate for the learning method – in the case of Q-learning, the regularization term is directly added to the Q-value estimates, while in the case of actor-critic methods like PPO, the regularization term is added to the value estimate of the critic.

Theoretical discussion of TQ. At each online-offline iteration, the regularized Bellman operator T produces Q-values Q_T that lower-bound the latest Q-values estimated online $\hat{Q}$ for any actions that may contribute to the violation of safety requirements, i.e., that are exercised as part of a counterexample-guided offline simulation. When more offline interventions are triggered by persistent violations of the safety requirement throughout the learning process, the penalization effects of the offline phases compound in the regularized Q-value Q_T until the frequency of unsafe behaviors lowers enough not to trigger additional offline phases. In the remaining of this section, we will assume that:

(1) $\rho > 0$: when $\rho = 0$ the regularization term is also zero, thus Eq. (12) trivially reduces to the standard Q-value during online learning, neglecting any violations of the safety requirement
(2) a policy that satisfies the safety requirement exists. Otherwise, any policy exercised by the agent would eventually lead to the violation of the safety requirement, in turn triggering an unbounded number of offline phases
(3) the learning rate decays slowly enough for the agent to incorporate feedback from all the offline phases triggered during learning – the number of necessary offline phases is always finite because of the previous assumption[3].

These assumptions identify the target application scenarios for our method. We refer to Appendix A.1.3 for additional discussion, as well as for sufficient convergence conditions when assumptions (2) and (3) do not hold.

[3]when the learning rate approaches zero, the agent loses the capability of acquiring new information, including feedback from offline phases. This situation is exemplified in Fig. 3, where the amplitude of the oscillations ostensibly decreases over time. However, while the learning rate decay draws the process to a fixed point, there is no guarantee the agent incorporated sufficient information to achieve its goal, including possible safety requirements. This observation applies to any learning process and it is not specific to our method.

Proposition 3.2 (*Conservatism of the Regularized Q-Value*). For any state-action pair (s, a) along an unsafe trajectory $\tau = \{s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_n\}$ terminating at $s_n \in S_U$, sampled from a counterexample, the regularized Q-value $Q_T(s, a)$ lower-bounds the value $Q_B(s, a)$ that would be estimated by standard Q-learning (without regularization), i.e., $Q_T(s, a) \leq Q_B(s, a)$.

Proof sketch. Let $\hat{Q}(s, a)$ be the latest online Q-value estimation for an (s, a) pair. When an offline phase is triggered, paths that reach any unsafe state in S_U are sampled from a counterexample to train the agent offline. Upon reaching a state in s_U, the agent receives a strictly negative reward – the penalty $r_u < 0$ – which backpropagates along all decisions taken along the path. Hence, if an action is sampled as part of a sampled unsafe path, its Q-value will decrease during an offline phase otherwise, the offline phase has no effect on the action, leading to $Q_{\text{cex}}(s, a) \leq \hat{Q}(s, a)$. The coefficient $(1 - b(s, a))$ in the regularization term of Eq. (12) is always nonnegative (equal to zero for $\hat{Q}(s, a) = Q_{\text{cex}}(s, a)$). Therefore, the regularization term is strictly negative for any state-action pair part of a path sampled during offline counterexample-guided learning and the overall regularized Q-value TQ lower bounds the standard $Q - value$ the agent would converge to for that state-action pair. □

If after an offline phase, the agent converges again toward an unsafe policy, a new offline phase will be triggered, this time starting from a lower value $\hat{Q}'(s, a)$ estimated from the latest online phase for any state-action pair that was part of a path sampled during previous offline phases. The rationale in prop. 3.2 can then be reiterated: the penalization delta accrued during an offline phase does not depend on the value of a state-action pair when the offline phase starts, but only on the frequency of the state-action pair is sampled from a counterexample (and the value of $r_u < 0$, which is constant). Therefore, actions that repeatedly contribute to the agent violating the safety requirements are repeatedly penalized by the triggered offline phases until their lowered values shift the agent's decisions toward alternative actions. To complete our argument, we show that in between offline phases (and especially after the last offline phase), the regularized Bellman operator would eventually converge to a fixed point by proving its contraction property [34].

Theorem 3.1 (*Contraction of regularized counterexample-guided online learning*). During online learning phases, the regularization term $L_{\text{reg}}(s, a)$ is constant until the occurrence of a new offline phase. Therefore, the regularized Bellman operator $TQ(s, a) = r(s, a) - L_{\text{reg}}(s, a) + \gamma \max_{a'} Q(s', a')$ reduces to an affine operator and for any two Q-functions Q and Q',

$$\|TQ - TQ'\|_\infty \leq \gamma \|Q - Q'\|_\infty,$$

and T is a contraction under the max-norm if $\gamma < 1$.

Proof. With a fixed Q_{cex} after an offline phase, the difference $\Delta(s, a) = \hat{Q}(s, a) - Q_{\text{cex}}(s, a)$ between $\hat{Q}$ and Q_{cex} is a state–action constant. Therefore, the belief variable $b(s, a) = e^{-\rho[\Delta(s,a)]_+}$ is also a constant and $L_{\text{reg}}(s, a) = (1 - b)\,\Delta(s, a)$ no longer depends on Q. Until an offline phase occurs, the regularization term is identically zero; after the last offline phase occurs, the term will remain constant until the end of the learning process.

Let us define the regularized reward $R'(s, a) = r(s, a) - L_{\text{reg}}(s, a)$; then T reduces to the affine Bellman operator:

$$TQ(s, a) = R'(s, a) + \gamma \max_{a'} Q(s', a').$$

where for any Q, Q', $|TQ(s, a) - TQ'(s, a)| = \gamma \left|\max_{a'} Q(s', a') - \max_{a'} Q'(s', a')\right| \leq \gamma \|Q - Q'\|_\infty$. Taking the supremum over (s, a) gives the desired contraction when $\gamma < 1$. □

Conservative Advantages for Policy Gradient Updates. Policy gradient methods provide conservative value estimates and use them to perform policy updates. In actor–critic methods such as PPO [59], the policy is updated by maximizing a surrogate objective:

$$L(\theta) = \mathbb{E}_{s,a\sim\pi_{\text{old}}}\left[\min\left(r(\theta)\, A_B(s,a),\ \text{clip}\left(r(\theta), 1-\epsilon, 1+\epsilon\right) A_B(s,a)\right)\right].$$

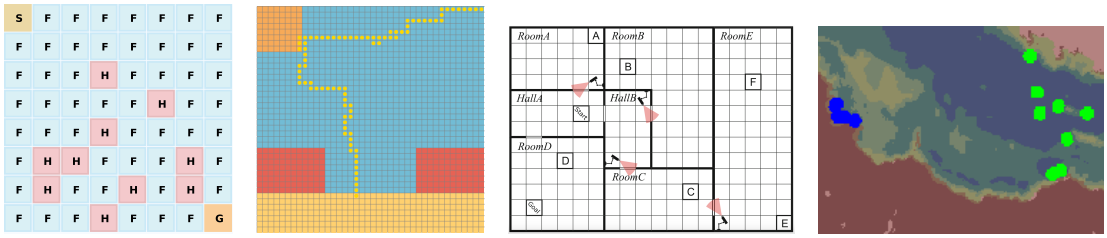
where $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)}$ is the probability ratio between the new and old policies, and the advantage function is $A_B(s,a) = Q_B(s,a) - V(s)$, where $Q_B(s,a)$ is computed under the standard Bellman operator B.

By Prop. 3.2, for state–action pairs occurring along unsafe trajectories, we have $Q_T(s,a) \le Q_B(s,a)$. The corresponding conservative advantage function that incorporates our regularization term is $A_T(s,a) = Q_T(s,a) - V(s) \le Q_B(s,a) - V(s) = A_B(s,a)$, so actions penalized during the offline phase are assigned lower advantage.

The advantage function also affects the trust region through a KL-divergence constraint, $D_{\text{KL}}(\pi_{\text{old}} \,\|\, \pi_\theta) \le \delta$, which ensures that each policy update remains close to the previous policy [58, 59]. Together, the conservative advantage $A_T(s,a)$ and the KL constraint imply that the policy update inherently penalizes unsafe actions: since unsafe actions are associated with lower Q_T (and hence lower A_T), the surrogate objective provides less incentive to increase their selection probability.

4 EVALUATION

In this section, we evaluate our method from two key aspects: the improvement in safety rates achieved through counterexample-guided exploration, and the impact on cumulative rewards as a result of the regularized counterexample-guided learning strategy. We also briefly discuss the computational overhead associated with counterexample generation and provide insights into the policies synthesized under counterexample guidance.



(a) Frozenlake8x8 (b) DiscreteGrid / HybridGrid (c) FlagMaze (d) MarsRover

Fig. 4. Layouts of Evaluation Environments

Environments. Our evaluation employs five environments with both discrete state space (`FrozenLake8x8`, `DiscreteGrid`, `FlagMaze`) and continuous state space (`HybridGrid`, `MarsRover`), as depicted in Fig. 4. The agent makes decisions between movements: `up`, `down`, `left`, and `right`, to achieve sequential goals through navigation in these environments while maintaining a probability above threshold λ to avoid unsafe behaviors during exploration. Additionally, `MarsRover` encounters an extra challenge of navigating a complex environment that cannot be completely approximated by a finite abstract state space, and `FlagMaze` encounters an extra challenge of conflicting objectives of optimization and safety during exploration:

- the slippery `FrozenLake8x8` (OpenAI Gym [7]), where the agent aims to navigate a safe path across an 8x8 grid, avoiding 'H' labeled states, with slippery actions. Under the slippery setting, the probability of moving

in the intended direction is one-third; otherwise, the agent slides into perpendicular directions with an equal probability

- `DiscreteGrid` (based on the implementation of [28, 29]), where the agent aims to sequentially reach the states labeled `goal1` first and then `goal2`, avoiding `unsafe` labeled states, with a 15% chance of random movement at each step
- `HybridGrid` – our adaptation of `DiscreteGrid` with a continuous state space, with movement distances sampling through a Gaussian distribution $\mathcal{N}(2, 0.5)$ at each step
- `MarsRover` (based on the implementation of [29, 30], simulating the Melas Chasma exploration [49], which cannot be perfectly abstracted by boxes or other affine abstract domains), where the agent seeks target states without entering unsafe regions. Movement distances per movement are derived from $\mathcal{U}(0, 10)$ with additional noise from $\mathcal{U}(-0.1, 0.5)$
- `FlagMaze` (based on the layout of [48] and the implementation of [53]), where the agent collects flags (6 in total) while avoiding detection by security cameras at all the doorways with varying probabilities. The more flags collected, the larger the rewards received, but also the higher the probability of being caught by the security cameras. Due to the conflicts of reward optimization and safety, this environment features several optimal policies that form a Pareto front for the accumulated reward and safety rate, as detailed in [48].

Baselines. The effectiveness of our method is benchmarked against established algorithms: classical Q-Learning (QL) [74], DQN [50] and PPO [59] for discrete and continuous MDPs, respectively. Considering similar motivation on incorporating logical guidance and abstract MDPs, we further compare our method against QL-LCRL [28] in `FrozenLake` and `DiscreteGrid`, with the risk preference parameter $\rho = 0$ (equivalent to Alg. 1 in [36]) and $\rho = 5$ respectively. QL-LCRL also guides exploration in an initially unknown MDP by reshaping rewards via an automaton derived from an LTL specification, encouraging behaviors that satisfy the specification while avoiding `unsafe` states. We additionally report results for ARL [48] on `FlagMaze` in Appendix A.1.3. We do not treat ARL as a general baseline because it is a model-based assured approach that relies on an abstract MDP and quantitative verification under known dynamics to enforce a PCTL property, whereas our setting focuses on model-free safe exploration from interaction data. We therefore use ARL as a reference point: by tuning ρ, our model-free method can realize reward–safety trade-offs that approach the verified model-based behavior in practice.

Table 1. Learning results with corresponding hyperparameters in QL, QL-LCRL, and QL-CEX ($\rho = 0$ and $\rho = 5$)

Environment	$\lvert S\rvert$	$\lvert S_A\rvert$	λ	α	γ	ϵ	f	QL		QL-LCRL		QL-CEX	
								safe. prob	reward	safe. prob	reward	safe. prob	reward
DiscreteGrid	3200	30	0.2	0.9	0.9	0.01	0.05	0.31	0.924	0.387	**0.99**	0.57/**0.62**	**0.99/0.99**
FrozenLake8x8	64	50	0.35	0.1	0.9	0.01	0.05	0.339	0.55	0.401	0.54	**0.44/0.45**	**0.6/0.62**
FlagMaze	14976	470	0.15	0.1	0.99	0.01	0.05	0.75	**4.05**	-	-	0.75/**0.92**	**4.1**/2.81

Table 2. Learning results with corresponding hyperparameters in DQN, DQN-CEX ($\rho = 0$ and $\rho = 5$), PPO and PPO-CEX

Environment	$\lvert S\rvert$	$\lvert S_A\rvert$	λ	α	ϵ	f	DQN		DQN-CEX		PPO		PPO-CEX	
							safe. prob	reward	safe. prob	reward	safe. prob	reward	safe. prob	reward
HybridGrid	∞	41	0.2	5e-4	0.01	0.05	0.55	0.76	0.7/0.74	0.8/0.83	0.7	0.94	**0.78**	**0.94**
MarsRover	∞	149	0.2	5e-4	0.05	0.15	0.68	0.94	0.82/0.86	0.92/0.96	0.89	0.97	**0.98**	**0.99**

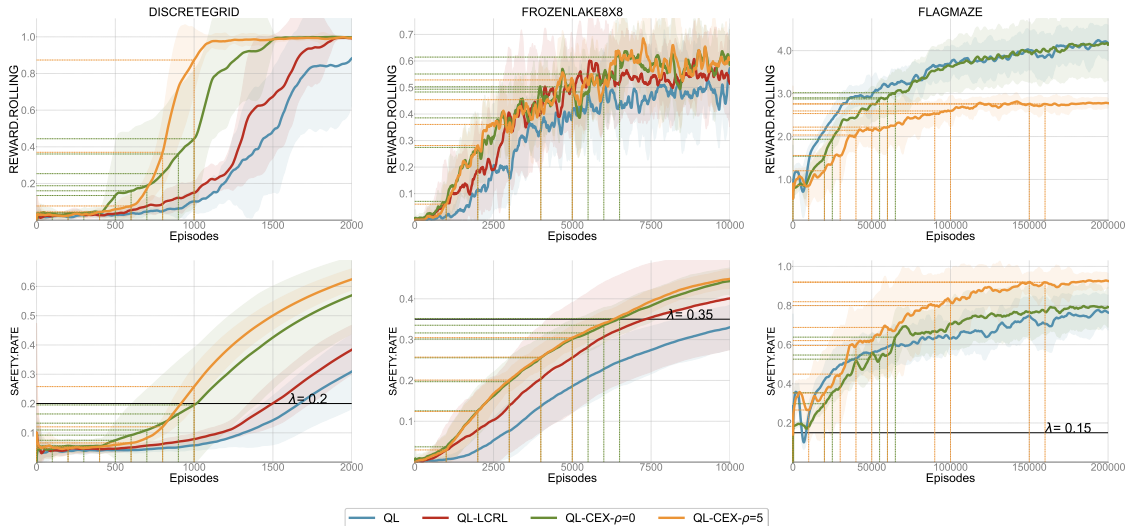


Fig. 5. Average over 10 runs of the accumulated safety rates and rolling rewards in QL, QL-LCRL [31] and QL-CEX (our method). The shaded region shows ± standard deviation

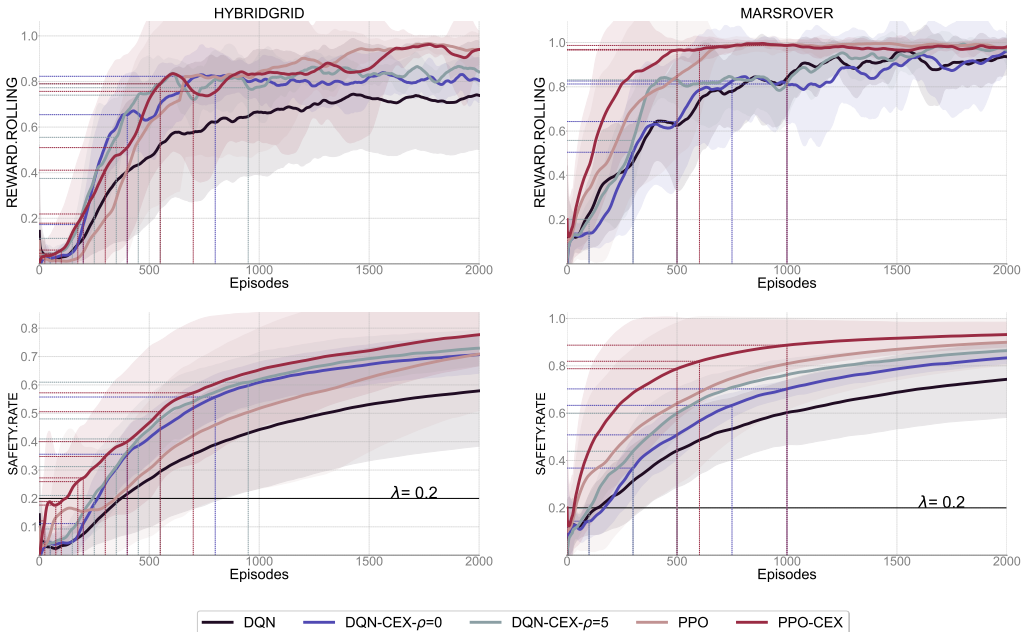


Fig. 6. Average over 10 runs of the accumulated safety rates and rolling rewards in DQN, DQN-CEX (our method), PPO and PPO-CEX (our method).

Experimental Results. We implemented QL and employed DQN and PPO furnished by the OpenAI Gym Baselines [55] in Python. Key hyper-parameters include the learning rate α, the discount factor γ, the ϵ for adjacent bisimulation and

a safety specification parameter λ modulated in line with the inherent uncertainties of each environment. The agents traverse Cartesian planes with a state size of $|S|$, while the state space minimization via abstraction reduces this to $|S_a|$. This abstraction process also considers false positive rates f for safe state inclusion in unsafe abstract states, and the minimal dimensions for each abstract box are set at 1 and 0.01 for discrete and continuous spaces, respectively. Hyper-parameters employed for each environment are listed in the left columns of Tab. 1 and 2. In our experiments, the Bayesian hypothesis test is executed on a sliding window of 50 samples, to reduce false positive triggers as learning starts and not enough data is available for a reliable test.

Fig. 5 and 6 show the learning performance, combining the accumulated safety rates (bottom) and the rolling average of accumulated rewards during exploration. Without prior knowledge, the agent starts exploration, with the average of 10 runs depicted by the line, and the standard deviation by the shaded area. Dashed vertical lines indicate the average episode number where an offline learning phase in counterexample-guided methods (our method) is triggered, while the solid horizontal line indicates the target safety rate λ in corresponding safety specifications. The cumulative rewards of different methods converge to similar values, suggesting that our method achieves cumulative rewards comparable to the baselines under counterexample guidance. Due to this conservative strategic guidance that prevents potentially unsafe actions during exploration, our method achieves a generally higher safety rate during all exploration processes. The intensity of this guidance is modifiable with the risk factor ρ – higher ρ imposing stricter penalties on unsafe behaviors. Compared to the learning result with $\rho = 0$ (no regularization), the learning result with a higher ρ value (e.g., $\rho = 5$) tends to be more conservative, particularly in conflict scenarios like `FlagMaze`, where risk minimization and reward maximization are in tension.

In `DiscreteGrid`, the safety rate for online learning via our method exceeds the preset threshold much faster than baseline methods. This can be attributed to the effective abstraction and minimization enabled by the geometric structure of grids, which eliminates the need for further offline phases after the initial 1000 episodes.

In slippery `FrozenLake8x8`, the inherent uncertainty in movements results in more extensive online exploration to facilitate effective counterexample guidance. This large uncertainty also leads to additional time to stabilize its performance, triggering offline learning phases more frequently and requiring a larger number of episodes to reliably meet the safety requirement.

In `FlagMaze`, the conflicting objectives of maximizing rewards and avoiding security cameras lead to a set of Pareto-optimal policies that show a trade-off between safety and rewards, as outlined in [48]. We also include the model-based ARL reference on this benchmark, and further analyze the relationship between the risk factor ρ and synthesized policies in our method for approximating the Pareto-front optimal policies in Appendix A.1.3. While both QL and our method achieve the safety requirement, our method guides the agent toward approximated Pareto-optimal policies with a higher safety rate than QL.

In `HybridGrid` and `MarsRover`, PPO-based approaches generally outperform DQN-based ones in learning performance with higher variance due to the different characteristics of these two learning methods. Despite these inherent methodological differences, our methods (PPO-CEX and DQN-CEX) achieve higher safety rates. While the initial accumulated rewards of our method may be lower due to cautious exploration, our approach eventually converges toward high-reward policies. Moreover, the frequency of triggering offline learning decreases as learning proceeds. QL-LCRL is not applicable to these two environments with continuous state space. Although LCRL [31] also employs an NFQ-based approach for continuous MDPs, its NFQ-LCRL variant solely relies on offline learning with randomly sampled data, making it incomparable to our method on the safe exploration aspect.

Overhead. Counterexample generation necessitates solving a MILP problem within an abstract state space. The evaluation in this section utilizes a Macbook Air equipped with an M1 CPU and 8GB of RAM, the average time ± standard deviation to resolve these optimization challenges were 0.71±0.98s for `FrozenLake8x8`, 0.08±0.06s for `DiscreteGrid`, 5.16±3.01s for `FlagMaze`, 2.31±3.94s for `HybridGrid`, and 3.87±4.07s for `MarsRover`, respectively, with the Gurobi Optimizer version 9.1.0 [25] employed for these calculations. On average, the numbers of counterexamples generated per offline learning stage were 11, 6, 32, 19, and 21 for each environment respectively. It is worth noting that the processes for generating counterexamples and conducting simulations are amenable to parallelization. While the MILP-solving stage may incur greater costs, offline learning is only resorted to when recent online explorations result in an unsatisfactory safety rate. As depicted in Fig. 5 and 6, with the aid of counterexample guidance, our method realizes the necessary safety exploration rate more swiftly, thereby offsetting the initial costs of solving MILP. Furthermore, the optimization challenge could be simplified at the expense of the solution's optimality and comprehensiveness of counterexamples – for computation time reduction.

Artifact availability. The implementation of the regularized counterexample-guided learning algorithms and the scripts used to reproduce the experiments are available at https://github.com/xtji/CEX-guided-RL.

5 RELATED WORK

Safe Exploration. Surveys and taxonomies on safe reinforcement learning [9, 23, 37] present a broad spectrum of strategies to reduce hazards during exploration. Most existing methods assume some form of prior knowledge or model dynamics [2, 13, 20, 52, 69], predefined safety constraints or runtime monitors [28, 35, 48, 56, 78], or expert demonstrations and additional data sources [22, 33, 57, 81]. A major line of work formalizes safety as explicit constraints in constrained MDPs and performs constrained optimization during training (e.g., trust-region or Lagrangian-style updates) [1, 66, 68]. While these approaches provide principled mechanisms to trade off reward and safety, they typically require a well-defined cost signal and careful tuning, and may exhibit instability or oscillation in constraint satisfaction. Another line of work focuses on uncertainty-aware safe exploration; for example, methods based on Gaussian-process surrogates [46, 67, 73] characterize uncertain dynamics under assumed model structures and restrict exploration accordingly. Only a few methods directly address safe exploration in unknown environments: [30] partitions unknown continuous MDPs into smaller tasks using LTL specifications (often prioritizing specification compliance rather than minimizing safety violations); [5] introduces a safety critic to filter potentially unsafe actions using an offline dataset; [32] employs a shielding discriminator trained from simulation/offline data; and [45, 79] learn safety constraints from demonstrations or raw observations. Complementary to these, recent verification-in-the-loop approaches use counterexamples or formal feedback to refine learned policies [12, 17, 21].

Our approach differs from most prior work by not assuming an a priori dynamics model, predefined safe shields, or a pre-collected offline dataset. Instead, we start from no prior exploration data, dynamically update safety-relevant information in a data-driven manner via abstraction, and enforce a probabilistic safety requirement by generating counterexamples and using targeted counterexample-guided simulations to regulate online learning.

Offline RL. As conceptualized in [44], offline RL learns near-optimal policies from a fixed dataset without online interaction, which is appealing for safety-critical systems because it avoids trial-and-error in the real environment. However, offline RL is challenged by limited dataset coverage and distribution shift between the learned policy and the behavior distribution. Many methods address this issue by discouraging out-of-distribution actions and preventing value overestimation via conservative regularization or optimization constraints [11, 41, 47, 61, 76], or by learning risk-averse policies using behavior cloning and safe policy priors [70, 77, 80]. While these approaches align with our

motivation for risk-averse learning, purely offline learning remains limited by partial dataset coverage and the inability to incorporate newly observed safety-critical evidence.

Recent work therefore emphasizes hybrid pipelines that combine offline pre-training with online fine-tuning [39, 54] and improved data-mixing strategies such as balanced replay buffers or sampling [4, 43, 64] to alleviate coverage limitations and stabilize online learning. Our method also follows a hybrid online–offline pattern, but differs in how offline supervision is obtained and preserved: we update abstract safety-relevant knowledge online and generate counterexample-guided simulations as targeted offline experience, mitigating catastrophic forgetting of safety-critical information and providing a probabilistic guarantee grounded in the most current counterexample-based evidence.

6 CONCLUSION

In this work, we have developed and analyzed a novel approach to enhance safety in model-free reinforcement learning through counterexample guidance, augmented by a combination of online and offline reinforcement learning with belief-based regularization. We demonstrated the effectiveness of our method with theoretical analysis and experimental evaluations, comparing it with state-of-the-art methods in RL.

The counterexample generation process is based on a conservative geometric abstraction method, distilling the knowledge obtained from historical exploration into a compact, finite MDP model. This model accurately captures safety-relevant information through limited online interactions with the environment and is updated incrementally to maintain the safety invariant, addressing potential catastrophic forgetting problems during the learning process. We generate counterexamples of the safety requirement in the abstract model, and create targeted, small-scale offline simulations, where the agent augments the penalized reward of making decisions that lead to previously encountered unsafe states under the belief-based regularization. In the subsequent online exploration, the agent receives a regularized state-action value that penalizes decisions with a large divergence from counterexample-guided policies that are likely to result in unsafe states. The regularization term addresses the distributional shift and oscillation of online and offline environments, enhancing the stability of learning performance with conservative learning behaviors, and thereby guiding the agent away from risky actions in future online interactions with tunable penalization strength.

The iterative process of online exploration followed by refinement through regularized counterexample-guided learning enables steady improvement in safety rates without substantially reducing cumulative rewards. The efficacy of this balanced approach is evident from our evaluation on standard benchmarks, including OpenAI Gym environments, and comparisons with baseline methods. While the initial evaluation focused on enhancing Q-Learning, DQN, and PPO algorithms in both discrete and continuous state spaces, the core principle of our framework is algorithm-agnostic. One limitation of our current formulation is that counterexample generation assumes a finite action set in the abstract MDP, which is used to enumerate state-action pairs and transition labels. Extending the approach to continuous action spaces would require an additional action abstraction or discretization step. Our future study will investigate this extension to continuous action spaces, a broader spectrum of reinforcement learning algorithms, and multi-agent systems, aiming to improve safety and sample efficiency for online exploration in unknown dynamic learning systems.

REFERENCES

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained Policy Optimization. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (Sydney, NSW, Australia) *(ICML'17)*. JMLR.org, 22–31.

[2] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe Reinforcement Learning via Shielding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence* (New Orleans, Louisiana, USA) *(AAAI'18/IAAI'18/EAAI'18)*.

AAAI Press, Article 326, 10 pages.
[3] Christel Baier and Joost-Pieter Katoen. 2008. Principles of Model Checking.
[4] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. 2023. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*. PMLR, 1577–1594.
[5] Homanga Bharadhwaj, Aviral Kumar, Nicholas Rhinehart, Sergey Levine, Florian Shkurti, and Animesh Garg. 2020. Conservative safety critics for exploration. *arXiv preprint arXiv:2010.14497* (2020).
[6] Anselm Blumer, A. Ehrenfeucht, David Haussler, and Manfred K. Warmuth. 1989. Learnability and the Vapnik-Chervonenkis Dimension. *J. ACM* 36, 4 (oct 1989), 929–965. https://doi.org/10.1145/76359.76371
[7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).
[8] Noam Brown and Tuomas Sandholm. 2018. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* 359, 6374 (2018), 418–424.
[9] Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. 2022. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems* 5 (2022), 411–444.
[10] Nader H. Bshouty, Sally A. Goldman, H. David Mathias, Subhash Suri, and Hisao Tamaki. 1998. Noise-Tolerant Distribution-Free Learning of General Geometric Concepts. *J. ACM* 45, 5 (sep 1998), 863–890. https://doi.org/10.1145/290179.290184
[11] Jacob Buckman, Carles Gelada, and Marc G Bellemare. 2020. The importance of pessimism in fixed-dataset policy optimization. *arXiv preprint arXiv:2009.06799* (2020).
[12] Steven Carr, Nils Jansen, Ralf Wimmer, Alexandru C Serban, Bernd Becker, and Ufuk Topcu. 2019. Counterexample-guided strategy improvement for pomdps using recurrent neural networks. *arXiv preprint arXiv:1903.08428* (2019).
[13] 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).
[14] Josée Desharnais, François Laviolette, and Mathieu Tracol. 2008. Approximate Analysis of Probabilistic Processes: Logic, Simulation and Games. In *Proceedings of the 2008 Fifth International Conference on Quantitative Evaluation of Systems (QEST '08)*. IEEE Computer Society, USA, 264–273. https://doi.org/10.1109/QEST.2008.42
[15] Allen B Downey. 2021. *Think Bayes*. " O'Reilly Media, Inc.".
[16] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning* 110, 9 (2021), 2419–2468.
[17] Pierre El Mqirmi, Francesco Belardinelli, and Borja G. León. 2021. An Abstraction-Based Method to Check Multi-Agent Deep Reinforcement-Learning Behaviors. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems* (Virtual Event, United Kingdom) *(AAMAS '21)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 474–482.
[18] Rafael Figueiredo Prudencio, Marcos R. O. A. Maximo, and Esther Luna Colombini. 2024. A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems. *IEEE Transactions on Neural Networks and Learning Systems* 35, 8 (2024), 10237–10257. https://doi.org/10.1109/TNNLS.2023.3250269
[19] Antonio Filieri, Corina S. Păsăreanu, Willem Visser, and Jaco Geldenhuys. 2014. Statistical Symbolic Execution with Informed Sampling. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Hong Kong, China) *(FSE 2014)*. Association for Computing Machinery, New York, NY, USA, 437–448. https://doi.org/10.1145/2635868.2635899
[20] Nathan Fulton and André Platzer. 2018. Safe Reinforcement Learning via Formal Methods: Toward Safe Control through Proof and Learning *(AAAI'18/IAAI'18/EAAI'18)*. AAAI Press, Article 794, 8 pages.
[21] Briti Gangopadhyay and Pallab Dasgupta. 2021. Counterexample guided RL policy refinement using bayesian optimization. *Advances in Neural Information Processing Systems* 34 (2021), 22783–22794.
[22] Javier García and Fernando Fernández. 2012. Safe Exploration of State and Action Spaces in Reinforcement Learning. *J. Artif. Int. Res.* 45, 1 (sep 2012), 515–564.
[23] Javier Garcıa and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16, 1 (2015), 1437–1480.
[24] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *The Journal of Machine Learning Research* 13, 1 (2012), 723–773.
[25] Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual. https://www.gurobi.com
[26] Tingting Han, Joost-Pieter Katoen, and Damman Berteun. 2009. Counterexample generation in probabilistic model checking. *IEEE transactions on software engineering* 35, 2 (2009), 241–257.
[27] Hans Hansson and Bengt Jonsson. 1994. A Logic for Reasoning about Time and Reliability. *Form. Asp. Comput.* 6, 5 (sep 1994), 512–535. https://doi.org/10.1007/BF01211866
[28] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. 2018. Logically-constrained reinforcement learning. *arXiv preprint arXiv:1801.08099* (2018).
[29] Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. [n. d.]. LCRL: Certified Policy Synthesis via Logically-Constrained Reinforcement Learning - implementation. https://github.com/grockious/lcrl

[30] Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. 2020. Deep reinforcement learning with temporal logics. In *Formal Modeling and Analysis of Timed Systems: 18th International Conference, FORMATS 2020, Vienna, Austria, September 1–3, 2020, Proceedings 18*. Springer, 1–22.

[31] Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. 2022. LCRL: Certified Policy Synthesis Via Logically-Constrained Reinforcement Learning. In *Quantitative Evaluation of Systems: 19th International Conference, QEST 2022, Warsaw, Poland, September 12–16, 2022, Proceedings* (Warsaw, Poland). Springer-Verlag, Berlin, Heidelberg, 217–231. https://doi.org/10.1007/978-3-031-16336-4_11

[32] Kai-Chieh Hsu, Allen Z Ren, Duy P Nguyen, Anirudha Majumdar, and Jaime F Fisac. 2023. Sim-to-lab-to-real: Safe reinforcement learning with shielding and generalization guarantees. *Artificial Intelligence* 314 (2023), 103811.

[33] Jessie Huang, Fa Wu, Doina Precup, and Yang Cai. 2018. Learning Safe Policies with Expert Guidance. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 9123–9132.

[34] Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. 1994. On the Convergence of Stochastic Iterative Dynamic Programming Algorithms. *Neural Computation* 6, 6 (1994), 1185–1201. https://doi.org/10.1162/neco.1994.6.6.1185

[35] Nils Jansen, Bettina Könighofer, Sebastian Junges, and Roderick Bloem. 2018. Shielded decision-making in MDPs. *arXiv preprint arXiv:1807.06096* (2018).

[36] Xiaotong Ji and Antonio Filieri. 2023. Probabilistic Counterexample Guidance for Safer Reinforcement Learning. In *Quantitative Evaluation of Systems*, Nils Jansen and Mirco Tribastone (Eds.). Springer Nature Switzerland, Cham, 311–328. https://doi.org/10.1007/978-3-031-43835-6_22

[37] Youngmin Kim, Richard Allmendinger, and Manuel López-Ibáñez. 2020. Safe Learning and Optimization Techniques: Towards a Survey of the State of the Art. In *Trustworthy AI - Integrating Learning, Optimization and Reasoning: First International Workshop, TAILOR 2020, Virtual Event, September 4–5, 2020, Revised Selected Papers* (Santiago de Compestela, Spain). Springer-Verlag, Berlin, Heidelberg, 123–139. https://doi.org/10.1007/978-3-030-73959-1_12

[38] Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. *Advances in neural information processing systems* 12 (1999).

[39] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2021. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169* (2021).

[40] S Kullback and RA Leibler. 1951. On Information and Sufficiency. *Ann. Math. Statist.* 22, 4 (1951), 79–86.

[41] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative Q-Learning for Offline Reinforcement Learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS'20)*. Curran Associates Inc., Red Hook, NY, USA, Article 100, 13 pages.

[42] E. L. Lawler and D. E. Wood. 1966. Branch-and-Bound Methods: A Survey. *Oper. Res.* 14, 4 (aug 1966), 699–719. https://doi.org/10.1287/opre.14.4.699

[43] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. 2022. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*. PMLR, 1702–1712.

[44] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* (2020).

[45] David Lindner, Xin Chen, Sebastian Tschiatschek, Katja Hofmann, and Andreas Krause. 2024. Learning safety constraints from demonstrations with unknown rewards. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2386–2394.

[46] Anqi Liu, Guanya Shi, Soon-Jo Chung, Anima Anandkumar, and Yisong Yue. 2020. Robust regression for safe exploration in control. In *Learning for Dynamics and Control*. PMLR, 608–619.

[47] Yixiu Mao, Qi Wang, Chen Chen, Yun Qu, and Xiangyang Ji. 2024. Offline reinforcement learning with ood state correction and ood action suppression. *arXiv preprint arXiv:2410.19400* (2024).

[48] George Rupert Mason, Radu Constantin Calinescu, Daniel Kudenko, and Alec Banks. 2017. Assured reinforcement learning with formally verified abstract policies. In *9th International Conference on Agents and Artificial Intelligence (ICAART)*. York.

[49] Alfred S McEwen, Colin M Dundas, Sarah S Mattson, Anthony D Toigo, Lujendra Ojha, James J Wray, Matthew Chojnacki, Shane Byrne, Scott L Murchie, and Nicolas Thomas. 2014. Recurring slope lineae in equatorial regions of Mars. *Nature geoscience* 7, 1 (2014), 53–58. https://doi.org/10.1038/ngeo2014

[50] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[51] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[52] Teodor Mihai Moldovan and Pieter Abbeel. 2012. Safe Exploration in Markov Decision Processes. In *Proceedings of the 29th International Coference on International Conference on Machine Learning* (Edinburgh, Scotland) *(ICML'12)*. Omnipress, Madison, WI, USA, 1451–1458.

[53] Pierre El Mqirmi, Francesco Belardinelli, and Borja G León. 2021. An abstraction-based method to check multi-agent deep reinforcement-learning behaviors. *arXiv preprint arXiv:2102.01434* (2021).

[54] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. 2020. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359* (2020).

[55] OpenAI. [n. d.]. Stable baselines version 3 - DQN. https://stable-baselines3.readthedocs.io/en/master/modules/dqn.html

[56] Tu-Hoa Pham, Giovanni De Magistris, and Ryuki Tachibana. 2018. OptLayer - Practical Constrained Optimization for Deep Reinforcement Learning in the Real World. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (Brisbane, Australia). IEEE Press, 6236–6243. https://doi.org/10.1109/ICRA.2018.8460547

[57] Bharat Prakash, Mohit Khatwani, Nicholas Waytowich, and Tinoosh Mohsenin. 2019. Improving safety in reinforcement learning using model-based architectures and human intervention. *arXiv preprint arXiv:1903.09328* (2019).

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

[59] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[60] Rahul Sharma, Saurabh Gupta, Bharath Hariharan, Alex Aiken, and Aditya V Nori. 2013. Verification as learning geometric concepts. In *Static Analysis: 20th International Symposium, SAS 2013, Seattle, WA, USA, June 20-22, 2013. Proceedings 20*. Springer, 388–411.

[61] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. 2020. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396* (2020).

[62] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.

[63] Gagandeep Singh, Markus Püschel, and Martin Vechev. 2017. A Practical Construction for Decomposing Numerical Abstract Domains. *Proc. ACM Program. Lang.* 2, POPL, Article 55 (dec 2017), 28 pages. https://doi.org/10.1145/3158143

[64] Yuda Song, Yifei Zhou, Ayush Sekhari, J Andrew Bagnell, Akshay Krishnamurthy, and Wen Sun. 2022. Hybrid rl: Using both offline and online data can make rl efficient. *arXiv preprint arXiv:2210.06718* (2022).

[65] Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. 2020. Learning to be safe: Deep rl with a safety critic. *arXiv preprint arXiv:2010.14603* (2020).

[66] Adam Stooke, Joshua Achiam, and Pieter Abbeel. 2020. Responsive Safety in Reinforcement Learning by PID Lagrangian Methods. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. JMLR.org, Article 847, 11 pages.

[67] Yanan Sui, Alkis Gotovos, Joel W. Burdick, and Andreas Krause. 2015. Safe Exploration for Optimization with Gaussian Processes. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37* (Lille, France) *(ICML'15)*. JMLR.org, 997–1005.

[68] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. 2018. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074* (2018).

[69] Garrett Thomas, Yuping Luo, and Tengyu Ma. 2021. Safe reinforcement learning by imagining the near future. *Advances in Neural Information Processing Systems* 34 (2021), 13859–13869.

[70] Núria Armengol Urpí, Sebastian Curi, and Andreas Krause. 2021. Risk-averse offline reinforcement learning. *arXiv preprint arXiv:2102.05371* (2021).

[71] Milan Češka, Christian Hensel, Sebastian Junges, and Joost-Pieter Katoen. 2019. Counterexample-Driven Synthesis for Probabilistic Program Sketches. In *Formal Methods – The Next 30 Years: Third World Congress, FM 2019, Porto, Portugal, October 7–11, 2019, Proceedings* (Porto, Portugal). Springer-Verlag, Berlin, Heidelberg, 101–120. https://doi.org/10.1007/978-3-030-30942-8_8

[72] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.

[73] Akifumi Wachi, Yanan Sui, Yisong Yue, and Masahiro Ono. 2018. Safe Exploration and Optimization of Constrained MDPs Using Gaussian Processes. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence* (New Orleans, Louisiana, USA) *(AAAI'18/IAAI'18/EAAI'18)*. AAAI Press, Article 802, 8 pages.

[74] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8 (1992), 279–292.

[75] Ralf Wimmer, Nils Jansen, Ã Erika, Joost-Pieter Katoen, et al. 2015. High-level counterexamples for probabilistic automata. *Logical Methods in Computer Science* 11 (2015).

[76] Yifan Wu, George Tucker, and Ofir Nachum. 2019. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361* (2019).

[77] Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. 2022. Constraints penalized q-learning for safe offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 8753–8760.

[78] Wen-Chi Yang, Giuseppe Marra, Gavin Rens, and Luc De Raedt. 2023. Safe reinforcement learning via probabilistic logic shields. *arXiv preprint arXiv:2303.03226* (2023).

[79] Sinong Zhan, Yixuan Wang, Qingyuan Wu, Ruochen Jiao, Chao Huang, and Qi Zhu. 2024. State-wise safe reinforcement learning with pixel observations. In *6th Annual Learning for Dynamics & Control Conference*. PMLR, 1187–1201.

[80] Yinan Zheng, Jianxiong Li, Dongjie Yu, Yujie Yang, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. 2024. Safe offline reinforcement learning with feasibility-guided diffusion model. *arXiv preprint arXiv:2401.10700* (2024).

[81] Weichao Zhou and Wenchao Li. 2018. Safety-aware apprenticeship learning. In *Computer Aided Verification: 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I 30*. Springer, 662–680.

A APPENDIX

A.1 Proof Sketches and Additional Theoretical Discussion

In this section, we detail the theoretical discussion and proofs of the proposed abstraction method and the counterexample-guided reinforcement learning algorithm.

A.1.1 PAC Learnability of Safety-Relevant Abstraction. The core of the proposed safety-relevant abstraction is solving a minimal red-blue set coverage by polyhedra (restricted to hyperboxes in this work) predicates, as introduced in sec. 3.1. We aim to discuss how an abstraction up to an arbitrary accuracy will almost surely eventually be constructed. We remind that the set cover problem allows only over-approximation of unsafe regions, meaning that unsafe concrete points are always mapped to unsafe abstract states and safe concrete points may possibly be misclassified as unsafe with a maximum prescribed false positive rate. For simplicity, we focus on the MDPs and Q update rules in a finite setting. Because we allow hyperboxes of minimum size $d > 0$, a continuous space is implicitly discretized, with every sample from a continuous space included within a box of size d or larger.

Proposition A.1. The maximum misclassification error of a concrete state $s \in S$ incorrectly mapped into an alternate abstract state $s_a \in S_a$ with the set-cover solution set $C = \{\cup_{i=1} C_i\}$ can eventually be reduced below an arbitrary bound $0 < \bar{u} \leq 1$ with a probability of at least $1 - \delta$ $(0 < \delta < 1)$ throughout the exploration.

Proposition A.1 relies on a PAC learnability argument. During the online exploration, *every reachable concrete state will eventually be reached with probability 1 during online exploration, in particular, every reachable unsafe state will eventually be reached.* This follows from the fact that the online exploration of the environment is never strictly limited by the use of counterexample guidance. Rather, offline phases aim at reducing the relative Q-value of actions that may eventually lead to the violation of the safety requirement, thus reducing the likelihood of their selection during online learning. Because the agent is always allowed, with a controllable probability, to explore any action from the current state, every reachable state maintains a strictly positive probability of being reached. Therefore, according to our incremental abstraction strategy with the preservation of the safety invariant, which ensures unsafe concrete states are always mapped to unsafe abstract states: *Every reachable unsafe state will eventually be mapped to an unsafe abstract state.*

Proof. We use $\underline{\epsilon_{QL}} > 0$ to denote the lower bound of the values that ϵ_{QL} can take during exploration. Then, given $\underline{\epsilon_{QL}}$ and the concrete MDP's transition relation, the probability of visiting any (reachable) concrete state is bounded from below by a value $\underline{p_{QL}}$.

Recalling Theorem 2.1 in [6], if the solution set C of the minimal red-blue set cover problem has a finite Vapnik–Chervonenkis (VC) dimension, and if C is consistent with a uniform sample size $\max(\frac{4}{\bar{u}} \log \frac{2}{\delta}, \frac{8VC}{\bar{u}} \log \frac{13}{\bar{u}})$, then the prediction error of C can be limited to $u \in (0, 1]$, with a probability of at least $1 - \delta$. The expressiveness and also VC dimension of the solution set C is decided by the abstraction domain. For the set coverage learning concept we defined in sec. 3.1, the general VC dimension of a single abstract predicate of the form $C_i = \{\mathbf{x} | \omega \mathbf{x} + \mathbf{b} \leq 0\}$ is finite. The actual VC dimension of a specific abstract domain and scenario can be easily verified, e.g. the VC dimension of an axis-parallel box is $v = 4$.

According to Lemma 3.2.3 in [6], the VC dimension of a union set C of convex polygons C_i is less than $2vs \log(3s)$, for all $s \geq 1$, where s is the number of sets in C. Hence, the required sample size to limit the abstraction error $u \leq \bar{u}$ for the set cover solution C is $\max(\frac{4}{\bar{u}} \log \frac{2}{\delta}, \frac{16vs \log(3s)}{\bar{u}} \log \frac{13}{\bar{u}})$. By underapproximating the number of samples considering

conservatively that each concrete state has probability $\underline{p_{QL}}$ of being sampled, we can conclude that an abstraction with misclassification error less than a prescribed $\bar{u}$ can eventually be learned with arbitrary probability $1 - \delta$. □

Caveats and practical limitations. Considering our abstract domain uses boxes (with sides parallel to the axes of the state space domain), it is likely to misclassify safe regions where their boundaries cannot be covered exactly with a union of boxes. In general, a similar argument can be formulated for any finite-accuracy abstract domain. The corner case of using this abstraction is that small safe regions located between two unsafe ones placed at a distance, along any dimension, smaller than the minimum size of a box d may remain misclassified even if the agent happens to sample a concrete point that could discriminate them. In turn, if the optimal policy requires passing through one such small misclassified region, counterexample guidance could reduce the likelihood of the agent exploring it – but never entirely prevent its exploration. In practice, the problem can be mitigated by choosing a smaller value of d, or using an abstract domain with a more appropriate performance/accuracy trade-off for the problem at hand, e.g., octagons or polyhedra.

A.1.2 Convergence of Counterexample-Guided Learning.

Proposition 3.1. If there exist maximal-reward policies that satisfy the safety requirement, then the counterexample-guided learning algorithm will eventually converge to one of them.

The core idea of Alg. 1 is to discourage the re-exploration of state-action pairs that lead to violation of the safety requirement by penalizing corresponding Q-values. The penalization of explored unsafe behaviors has the effect of encouraging the exploration of alternative actions in a state because their Q-values "grow" relatively to the Q-values of the penalized actions and are thus more likely to be selected next. An underlying assumption for the stability of the method is that *there exists an optimal policy that satisfies the safety requirement.* In the following, we will discuss how introducing offline learning does not prevent convergence to such a policy (in fact, as demonstrated experimentally, it accelerates the convergence to it), including sufficient conditions for such convergence to occur. Finally, we will discuss what may happen if all optimal policies violate the safety requirement.

The online learning phases are bound to eventually converge to an optimal policy as long as each state-action pair can be visited infinitely often. The offline phases can decrease the relative likelihood of actions involved with policies leading to a violation of the safety requirement but never prevent their exploration altogether. As a result, when the online phases converge to an optimal policy that satisfies the safety requirement, and assuming the abstract model converges as well, no further offline phases will be triggered (except for possible occasional false positive triggers from the Bayesian hypothesis test). This happened in all our experiments, where occasional offline phases triggered after the agent converged to mainly explore policies that satisfy the safety requirement could introduce transient fluctuations in the cumulative reward but do not affect convergence to maximum reward in the long run.

A.1.3 Regularized Learning under Conflicting Objectives.

If there exists no maximal-reward policy satisfying the safety requirement, the introduction of offline learning may result in oscillations in the Q-values of actions involved in the maximal-reward policy discovered by the agent. Such oscillations arise from the fact that the maximal-reward policy the online phase converged to is itself, by hypothesis, a counterexample to the safety requirement. In this situation, Alg. 1 may still reduce the number of failures during exploration by reducing the frequency of visiting unsafe states, but it will not prevent the agent from converging to such unsafe but maximal-reward policy in the long run. To address this limitation, we proposed in sec. 3.5 a regularized Bellman operator to the Q-update rule that privileges safety over reward, such that the regularized Q-value serves as a lower bound of the original Q-values, with the following theorems.

Absence of a safe policy. The theoretical discussion of TQ in section 3.5 is based on three assumptions, including that there exists at least one policy satisfying the safety requirement. However, if assumptions 2 and/or 3 do not hold – i.e. there exists no safe policy and/or the learning rate decays too quickly – offline phases may be triggered indefinitely. In the worst case, the agent can oscillate between successive unsafe policies, each time incurring an additional penalty. In this case, we further discuss Theorem A.1, which still provides a sufficient condition for stability guarantee according to the value of ρ. Intuitively, a small ρ limits the per-iteration penalty so that updates remain a contraction even while Q_{cex} keeps changing. If ρ is larger, stability is still possible in practice provided the learning-rate schedule decays slowly enough to absorb the incremental penalties, i.e., the bound in the following theorem is conservative.

Theorem A.1 (*Contraction property of the regularized Bellman operator*). The regularized Bellman operator T satisfies the contraction property under the max-norm. That is, for any pair of Q-functions Q and Q',

$$\|TQ - TQ'\|_\infty \leq C\|Q - Q'\|_\infty,$$

where the constant $C = b\gamma + 3\rho\, r_{\max}$ with a discount factor $\gamma \in (0,1)$, an upper bound on the reward $r_{\max}$, and the risk factor ρ). In particular, T is a contraction if $\rho < \frac{1-\gamma}{3r_{\max}}$ such that $C < 1$.

Proof. To demonstrate that T is a contraction mapping, we first expand the terms in the modified Bellman updates:

$$\begin{aligned} TQ(s,a) &= r(s,a) + \gamma \max_{a'} Q(s',a') - L_{\text{reg}}(s,a) \\ &= r(s,a) + \gamma \max_{a'} Q(s',a') - (1 - b(s,a,Q))[\hat{Q} - Q_{\text{cex}}(s,a)] \end{aligned}$$

Consider the absolute difference in the modified Bellman operator for any arbitrary Q-functions Q and Q' with max-norm:

$$\begin{aligned} \|TQ - TQ'\|_\infty &= \max_{s,a} |r(s,a) + \gamma \max_{a'} Q(s',a') - (1 - b(s,a,Q))(\hat{Q} - Q_{\text{cex}}(s,a)) \\ &\quad - r(s,a) - \gamma \max_{a'} Q'(s',a') + (1 - b(s,a,Q'))(\hat{Q}' - Q_{\text{cex}}(s,a))| \\ &= \max_{s,a} |b(s,Q)(\hat{Q} - Q_{\text{cex}}(s,a)) - b(s,a,Q')(\hat{Q}' - Q_{\text{cex}}(s,a))| \\ &= \max_{s,a} |b(s,Q)(r(s,a) + \gamma \max_{a'} Q(s',a') - Q_{\text{cex}}(s,a)) - b(s,a,Q')(r(s,a) + \gamma \max_{a'} Q'(s',a') - Q_{\text{cex}}(s,a))| \\ &= \max_{s,a} |(b(s,a,Q) - b(s,a,Q'))Q_{\text{cex}}(s,a) + b(s,a,Q)(r(s,a) + \gamma \max_{a'} Q(s',a')) - b(s,a,Q')(r(s,a) + \gamma \max_{a'} Q'(s',a'))| \end{aligned}$$

The belief variable is defined as $b(s,a,Q) = \exp\left(-\rho \left[Q(s,a) - Q_{\text{cex}}(s,a)\right]_+\right)$, where the function $[x]_+$ is 1-Lipschitz and the exponential function $\exp(-\rho x)$ is smooth, with its derivative bounded by ρ on any bounded domain $\left|\exp(-\rho x) - \exp(-\rho y)\right| \leq \rho\,|x - y|$ for x, y in the range of $[Q(s,a) - Q_{\text{cex}}(s,a)]_+$. Therefore, by composing these two Lipschitz continuous functions, we deduce that

$$\|b(s,a,Q) - b(s,a,Q')\|_\infty \leq L_b\, \|Q - Q'\|_\infty,$$

with the Lipschitz constant L_b essentially upper bounded by ρ.

The Lipschitz constant L_b quantifies how sensitive the belief variable $b(s,a,Q)$ is to changes in the Q-function; that is, it bounds the maximum rate at which b can change as Q changes.

Then, assume that $\|b - b'\|_\infty = \varepsilon$ and that $\varepsilon \ll b$ when $Q \sim Q'$. We rearrange terms and apply the triangle inequality (noting that the Q-value is always less than the maximum reward when $\gamma < 1$ in episodic tasks with terminal rewards).

$$\begin{aligned}\|TQ - TQ'\|_\infty &= \max_{s,a} |\varepsilon(Q_{\text{cex}}(s,a) + r(s,a) + \gamma \max_{a'} Q'(s',a')) + b(s,a,Q)\gamma(\max_{a'} Q(s',a') - \max_{a'} Q'(s',a'))| \\ &\leq \max_{s,a} |3\varepsilon r_{\max} + b(s,a,Q)\gamma(\max_{a'} Q(s',a') - \max_{a'} Q'(s',a'))| \\ &\leq 3\varepsilon r_{\max} + b(s,a,Q)\gamma\|Q - Q'\|_\infty\end{aligned}$$

Dividing both sides by $\|Q - Q'\|_\infty$ yields

$$\frac{\|TQ - TQ'\|_\infty}{\|Q - Q'\|_\infty} \leq b\gamma + \frac{3\varepsilon r_{\max}}{\|Q - Q'\|_\infty},$$

so that T is a contraction if $C = b\gamma + \frac{3\|b-b'\|_\infty r_{\max}}{\|Q-Q'\|_\infty} < 1$. Therefore, the contraction condition can be satisfied if $\gamma + 3\rho\, r_{\max} < 1$, since $b \leq 1$ and $L_b \leq \rho$; alternatively, we have $\rho < \frac{1-\gamma}{3r_{\max}}$, meaning that this contraction would hold under certain trade-offs between the sensitivity ρ and the reward bound $r_{\max}$ in the environmental setting. □

Proposition A.2 (*Asymptotic Penalization of Unsafe Actions*). If the risk factor $\rho > 0$ and the learning rate a_t decays slowly enough, the regularized Bellman operator T progressively reduces the Q-values of unsafe actions (s,a) for which $Q(s,a) > Q_{\text{cex}}(s,a)$. When ρ is large, these Q-values are consistently penalized and kept close to or below $Q_{\text{cex}}(s,a)$, steering the policy away from unsafe behaviors.

This proposition characterizes how large risk factor ρ amplifies regularization effects during learning. For state-action pairs with $Q(s,a) > Q_{\text{cex}}(s,a)$, the belief variable $b(s,a,Q) = \exp(-\rho[Q(s,a) - Q_{\text{cex}}(s,a)]_+)$ tends toward zero as $\rho \to \infty$, which causes the regularization term $L_{\text{reg}}(s,a) = (1-b)[\hat{Q}(s,a) - Q_{\text{cex}}(s,a)]$ to approach the full gap $\hat{Q} - Q_{\text{cex}}$. ρ does not directly determine the fixed point of the Q-function, rather, it governs the intensity of penalization. The final Q-values depend on the balance between the penalty magnitude induced by ρ and the ability to update over time induced by the learning rate a_t: if the learning rate a_t decays too fast, the agent may stop adapting before unsafe actions are sufficiently penalized; if a_t remains large enough over time, repeated updates drive Q-values of unsafe actions downward.

This effect complements the contraction condition in Theorem A.1, which provides a sufficient (but not necessary) condition for convergence. We note however, that the sufficient condition in Theorem A.1 is conservative, while, in practice, convergence can be achieved also for pairs of (ρ, γ) that do not satisfy these conditions, as we will demonstrate with an example in the following discussion on the interaction between ρ and γ.

Effect of ρ on reward–safety trade-offs. We study the policies synthesized by our method in `FlagMaze` under the specification $P_{\leq\lambda}[F\,\texttt{unsafe}]$, where maximizing reward (collecting more flags) and maintaining safety (avoiding cameras) are inherently conflicting. In Fig. 7, we compare the final greedy policies obtained by classical QL and QL-CEX under a sweep of the risk parameter ρ (cross markers) against a set of Pareto-front reference policies for `FlagMaze` reported in prior model-based, verified work [48] (solid line). In particular, ARL [48] constructs an abstract MDP under known dynamics and uses quantitative verification to synthesize policies satisfying the PCTL constraint while optimizing reward, which yields the reported Pareto-front trade-off. We follow the hyperparameter template used in [17, 48] ($\alpha = 0.1$, $\epsilon_Q = 0.8$, $\gamma = 0.99$) and use a decaying learning-rate schedule.

As ρ increases, QL-CEX exhibits a shift from reward-seeking behavior toward more conservative, safety-seeking policies: small values of ρ (e.g., $\rho = 0.2, 0.8$) yield reward levels comparable to classical QL but with lower safety, whereas larger values (e.g., $\rho = 3, 5$) concentrate near the safety-dominant end of the reference frontier. Mid-range ρ values (e.g.,

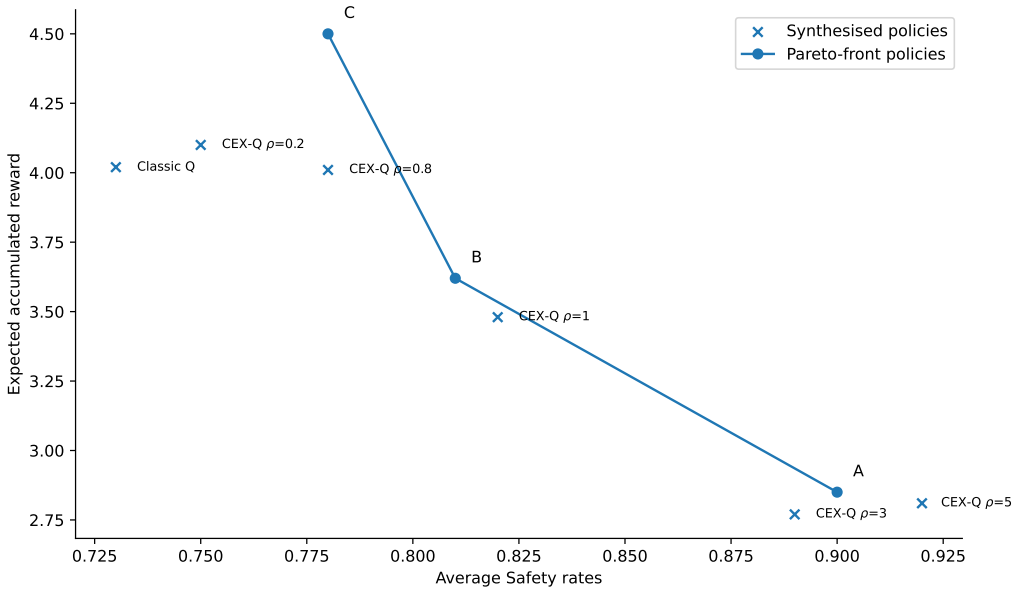


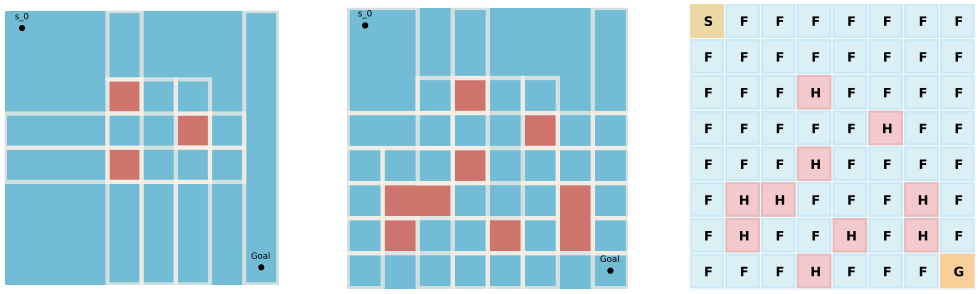
Fig. 7. Comparison of average safety rates and rewards of synthesized policies of QL, QL-CEX (our method), and Pareto-front policies.

$\rho = 1$) interpolate between these regimes, producing policies with more balanced reward–safety trade-offs. Overall, the observed monotone shift with increasing ρ traces the same reward–safety trade-off captured by the model-based verified Pareto frontier, indicating that our model-free method can be steered toward operating points that are consistent with ARL's verified reference behavior in practice.

Conservative Q-values and Policy Gradient Methods. We report here an additional discussion of how the conservative Q-value estimates derived from the counterexample-guided Bellman operator T affect policy updates in policy gradient methods. The policy in policy gradient methods can be updated using the advantage function $A(s,a) = Q(s,a) - V(s)$. The policy update step is governed by the maximization of the clipped surrogate objective:

$$L(\theta) = \mathbb{E}_{s,a\sim\pi_{\text{old}}}\Big[\min\big(r(\theta)A(s,a), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)A(s,a)\big)\Big], \tag{13}$$

where the probability ratio is $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)}$ and ϵ is the clipping threshold that defines the trust region.
We now substitute the conservative Q-value estimate $Q_T(s,a)$ under the counterexample-guided Bellman operator into the advantage function $A_T(s,a) = Q_T(s,a) - V(s)$. By Theorem 3.1, along any unsafe trajectory τ, the regularized estimate satisfies $Q_T(s,a) \leq Q_B(s,a)$, where Q_B is the Q-value under the standard Bellman operator. This directly implies $A_T(s,a) \leq A_B(s,a)$, meaning the advantage for actions leading to unsafe states is downweighted.
The conservative $A_T(s,a)$ further affects the surrogate loss, leading to a pessimistic update for unsafe actions even if they yield high immediate rewards. Due to the KL-divergence constraint $D_{\text{KL}}(\pi_{\text{old}} \parallel \pi_\theta) \leq \delta$, the policy update step also remains close to the previous policy. This overall process steers the policy π_θ by iteratively maximizing $L(\theta)$ using $A_T(s,a)$, ensuring convergence to a solution within the safe region defined by the conservative Q-values. This reasoning aligns with the convergence guarantees for policy gradient methods when using advantage estimates derived from Q-functions that satisfy contraction properties.



(a) Initial M_a (b) Refined M_a (c) Concrete MDP

Fig. 8. Incremental abstraction demonstration in Frozenlake8x8

A.2 Hyperparameter Effects on Abstraction and Learning Performance

In this section, we provide supplemental details about the abstraction evaluation, and the sensitivity of hyperparameters in the quality of abstraction and the learning outcomes. We first provide sets of additional hyperparameters in the QL, DQN, PPO, and our method to ease reproducibility. Then, we illustrate the incremental abstraction with concrete examples to demonstrate the incremental process and its influence and robustness on reinforcement learning systems with different hyperparameters. Next, we analyze the robustness of our learning performance with respect to hyperparameter tuning.

Additional Hyperparameters We implement our methods with the same values for all fundamental hyper-parameters against these benchmarks, all common hyper-parameters used with QL, DQN, PPO, and our method are the same as listed in Tab. 3 and 4. Additional hyper-parameters for QL-LCRL can refer to the values detailed in [31] and additional hyper-parameters of DQN and PPO can refer to the default values given in [51]. Then, we further assessed the robustness of our proposed method in terms of hyperparameter tuning related to our main algorithm, focusing on two main aspects: the quality of the resulting abstraction model and the learning performance, measured by accumulated safety rates and average rewards.

Table 3. Hyperparameters in QL for DiscreteGrid, FrozenLake8x8, and FlagMaze

Environment	β	N_{sim}	N_{iter}	N_{cex}	ϵ	ϵ_{decay}
DiscreteGrid	1	50	5000	20	0.3	0.9995
FrozenLake8x8	1	100	199	50	0.1	0.9995
FlagMaze	1	100	20000	50	0.9	0.99999

Table 4. Hyperparameters in DQN and PPO for HybridGrid and MarsRover

Environment	**Common Parameters**						**DQN Parameters**				**PPO Parameters**		
	β	N_{sim}	N_{iter}	max-grad-norm	batch-size	target-update	learning-starts	%-exp	gradient-steps	train-freq	n-steps	clip-range-vf	n-epochs
HybridGrid	1	50	5000	1	128	1000	10000	0.2	128	100	1000	0.1	5
MarsRover	1	50	3000	0.1	128	1000	5000	0.2	128	100	500	0.1	5



(a) Initial M_a (b) Refined M_a (c) Concrete MDP

Fig. 9. Incremental abstraction Demonstration in MarsRover

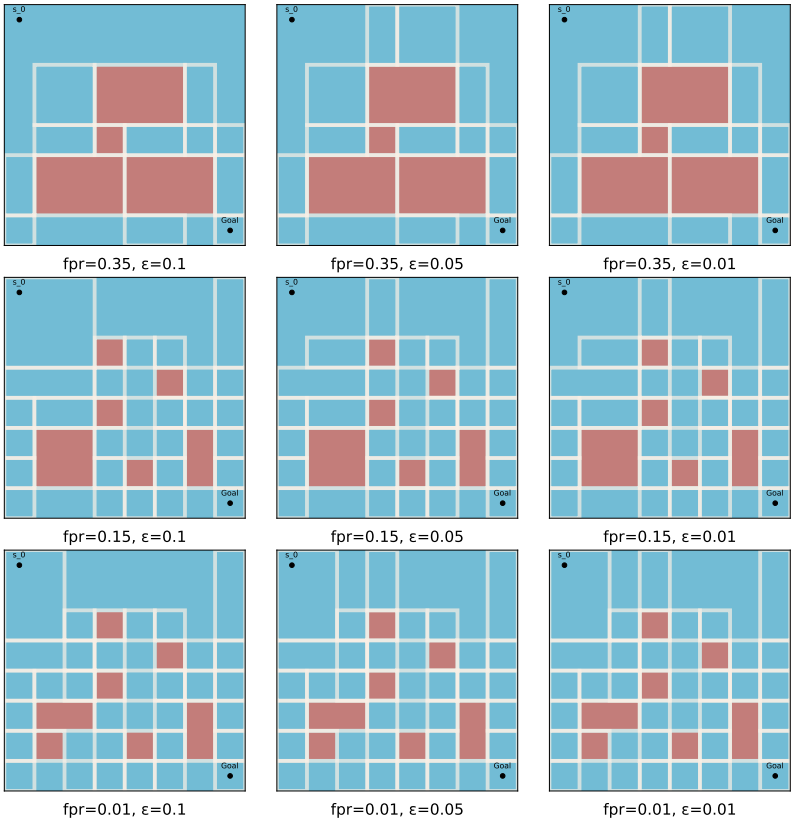


Fig. 10. Abstraction of FrozenLake8x8 with different fpr and ϵ

Hyperparameter Effects on Incremental Geometric Abstraction. When new unsafe states are discovered during online exploration, which was previously abstracted as safe, the abstract state space is updated incrementally using a branch-and-bound strategy to separate the new unsafe point. We demonstrate this process in the Frozenlake8x8 environment and the MarsRover environment in Fig. 8 and Fig. 9. The initial abstract MDP, used for counterexample generation, after the first 50 online episodes are shown in Fig. 8(a) and 9(a). With the increasing number of explored points, more

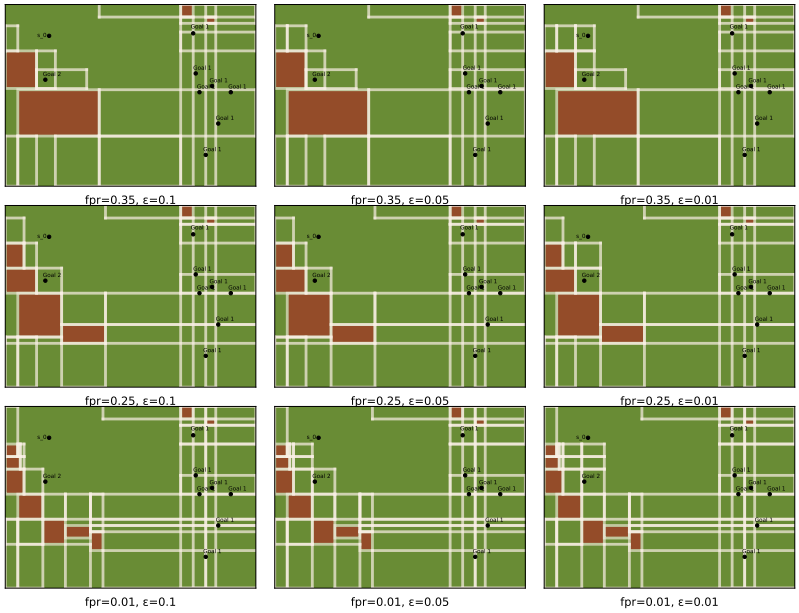


Fig. 11. Abstraction of MarsRover with different fpr and ϵ

safety-relevant information can be acquired, and we employ the branch and bound method incrementally to cover the newly discovered unsafe states and refine the safe states adjacent to the newly discovered unsafe regions, as shown in Fig. 8(b) and 9(b). The ground-truth layout is included in Fig. 8(c) and Fig. 8(c) for reference, where H and the dark-red regions indicate unsafe regions.

To evaluate simulation model quality, we present results using varying false positive rates f and ϵ for ϵ-bisimulation merging in both Frozenlake8x8 in Fig. 10 and MarsRover environments in Fig. 11. A lower FPR results in a more precise abstraction, which includes fewer safe concrete states from the unsafe abstract states (in environments like MarsRover, safe concrete states cannot be completely excluded with boxes).

Smaller ϵ values lead to less merging of similar states in the abstract state space. We recall the hierarchical ϵ-simulation concept, demonstrating that its inherent structure provides a beneficial trade-off between computational complexity and accuracy in proximity to unsafe states. Because the merge operation is constrained to states with identical labels and follows a hierarchical fashion (i.e., states from lower levels merge into higher ones, with transition probabilities at the higher level normalized by the sum from merged lower-level states), a lesser degree of merge due to ϵ-simulation is anticipated in the vicinity of unsafe states.

This hierarchical process indirectly optimizes the resolution in the state space where it is most crucial, specifically near unsafe states, while permitting a coarser merge in safer regions. Such non-uniform merging fosters a more efficient balance between the computational complexity of the counterexample generation optimization problem, which depends on the number of states in the abstraction, and the effectiveness of the counterexamples in accurately selecting concrete transitions from the agent's past experience near unsafe concrete regions.

Intuitively, given a specific ϵ, the final minimized abstraction is expected to be more precise near states labeled as unsafe due to the merge operation's restriction to adjacent abstract states. In these areas, a limited number of steps

and corresponding traversal of adjacent states suffices to differentiate the trajectory outcome under consideration. In contrast, a coarser abstraction is produced when more steps are required to determine the trajectory's outcome. This characteristic of the hierarchical ϵ simulation provides a tailored abstraction mechanism that varies with the safety of the region, enhancing the effectiveness of the abstract model.

Table 5. Robustness of Cex-Guided RL with different penalties and simulation episodes in slippery FrozenLake8x8

Penalty / $N_{\textbf{sim}}$	**20**		**50**		**100**	
	safe.prob	reward	safe.prob	reward	safe.prob	reward
-0.1	0.43	0.59	0.46	0.59	0.44	0.6
-0.5	0.43	0.61	0.43	0.62	0.45	0.6
-1	0.45	0.6	0.44	0.61	0.46	0.62

Table 6. Robustness of Cex-Guided RL with different Bayes factors and safety-check frequencies in slippery FrozenLake8x8

β / $N_{\textbf{safety-check}}$	**500**		**1000**		**2000**	
	safe.prob	reward	safe.prob	reward	safe.prob	reward
1	0.44	0.57	0.45	0.6	0.44	0.62
2	0.41	0.55	0.42	0.55	0.4	0.54
10	0.39	0.55	0.38	0.57	0.39	0.52

Regarding learning performance, we demonstrate the average accumulated safety rate and the average rolling reward are robust under different hyperparameters, by varying the penalty and number of offline episodes for each simulation model during the offline learning phase Tab. 5, and varying the Bayes factor and safety check interval during the online exploration in Tab. 6.

To estimate the probability of safety requirement violation and trigger offline learning when necessary, we utilize the Bayesian hypothesis testing estimator [19] with the Bayes factor as a hyperparameter. This estimator can be initialized with a minimum number of samples prior to accepting its decision and then determines whether the safety requirements are satisfied or violated based on the number of samples collected during the most recent online learning session. If the likelihood of violating the safety requirement significantly surpasses that of satisfying it, the offline learning phase is subsequently initiated.

Within the online learning session, we assess the Bayes factor of these two hypotheses, represented as $\frac{P(H_0|S)}{P(H_1|S)}$, using the real-time experience dataset. The magnitude of the given Bayes factor decides the frequency of triggering offline learning. This frequency setting engenders a trade-off between safety and computational cost: higher frequency leads to increased computational cost. A high-frequency setting may also induce over-conservative learning performance. This happens when excessive offline learning guidance inadvertently penalizes the learning process that entails a reasonable level of risk. In corner cases with very small Bayes factors, the agent may opt to explore solely safe regions, failing to achieve the learning objective.