Explainably Safe Reinforcement Learning

Sabine Rieder[*]
Masaryk University
Technical University of Munich
sabine.rieder@mail.muni.cz,

Stefan Pranger[*]
Graz University of Technology
stefan.pranger@tugraz.at

Debraj Chakraborty
Masaryk University
chakraborty@fi.muni.cz

Jan Křetínský
Masaryk University
Technical University of Munich
jan.kretinsky@fi.muni.cz,

Bettina Könighofer
Graz University of Technology
bettina.koenighofer@tugraz.at

Abstract

Trust in a decision-making system requires both safety guarantees and the ability to interpret and understand its behavior. This is particularly important for learned systems, whose decision-making processes are often highly opaque. Shielding is a prominent model-based technique for enforcing safety in reinforcement learning. However, because shields are automatically synthesized using rigorous formal methods, their decisions are often similarly difficult for humans to interpret. Recently, decision trees became customary to represent controllers and policies. However, since shields are inherently non-deterministic, their decision tree representations become too large to be explainable in practice. To address this challenge, we propose a novel approach for explainable safe RL that enhances trust by providing human-interpretable explanations of the shield's decisions. Our method represents the shielding policy as a *hierarchy of decision trees*, offering top-down, case-based explanations. At design time, we use a world model to analyze the safety risks of executing actions in given states. Based on this risk analysis, we construct both the shield and a high-level decision tree that classifies states into risk categories (safe, critical, dangerous, unsafe), providing an initial explanation of why a given situation may be safety-critical. At runtime, we generate localized decision trees that explain which actions are allowed and why others are deemed unsafe. Altogether, our method facilitates the explainability of the safety aspect in the safe-by-shielding reinforcement learning. Our framework requires no additional information beyond what is already used for shielding, incurs minimal overhead, and can be readily integrated into existing shielded RL pipelines. In our experiments, we compute explanations using decision trees that are several orders of magnitude smaller than the original shield.

1 Introduction

Deep reinforcement learning (RL) [42] is a powerful machine learning technique for intelligent sequential decision-making. Despite its successes, its application in safety-critical systems remains limited due to safety concerns. RL agents learn by exploring their environment through trial and error, a process that inherently carries the risk of taking unsafe actions.
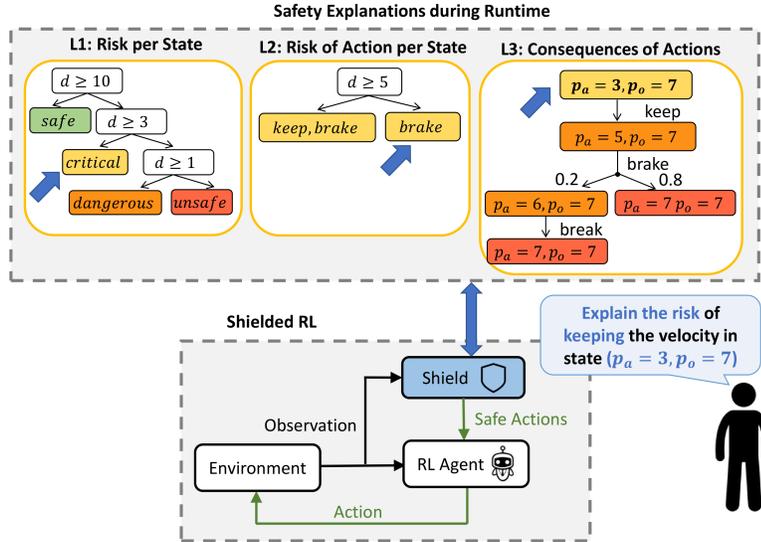
Figure 1: Overview of our method for explainable-safe RL.

Shielding [1] is a prominent approach towards safe RL. A shield blocks ("shields") unsafe actions from the learning agent at runtime, either during learning or evaluation, as depicted at the bottom of Fig. 1. At each step, the shield provides a list of all safe actions, from which the agent can select one. Shielded RL thus belongs to a class of methods that combine symbolic AI, which provides formal safety guarantees but suffers from limited scalability, with machine learning, which offers high scalability but lacks guarantees. However, even though shields add safety guarantees to RL, trusting a shielded system remains challenging due to the lack of *explainability*.

Policies learned via RL are notoriously difficult to interpret, as they are represented by highly opaque deep neural networks. Unfortunately, understanding the decisions made by a shield is almost equally challenging. Shields are typically implemented as large lookup tables that define the set of allowed actions for each state. These tables are inherently hard for humans to interpret, as they provide no insight into why a particular action is considered safe or unsafe in a given state. The combined opacity of both the shield and the RL policy makes it difficult to predict, understand, or trust the behavior of a shielded learning agent.

In this paper, we propose an approach to *explainably safe RL* by explaining the decisions of the shield to the user, making the safety aspect of the process more explainable.

Decision Trees (DTs) have recently gained popularity for representing controllers and policies due to their inherent simplicity and human-readability [17]. This is documented by numerous case studies and tool support, e.g. [5, 4, 7, 12]. However, directly computing DTs to represent a shield often results in excessively large trees, undermining their utility for explainability. Computing compact DTs is especially challenging in the context of shielding due to the shield's inherent non-determinism: a shield permits the agent to explore any action that is safe, resulting in more complex representations.

Our Approach for Explainable-Safe RL. Rather than presenting the user with a large tree that explains all safety-critical aspects at once, we propose to provide explanations in a *hierarchical* manner. Our method offers a compact, *top-down*, *case-based explanation*. An overview of our method is shown in Fig. 1. First, our approach computes the shield along with its explanations to help users understand how critical the current state is and the risks associated with executing specific actions. Second, the risk of each state is explained using a decision tree constructed at design time. Third, at runtime, additional explanations identify which actions are safe and clarify why others are considered unsafe. Our framework takes as input an abstract world model $\mathcal{M}$ in the form of a Markov Decision Process (MDP), along with a safety specification φ in temporal logic.

At Design-Time. Based on the model $\mathcal{M}$, we apply value iteration to compute, for every state-action pair (s, a), the risk of violating the safety specification φ when executing action a in state s. Based on this safety analysis, we categorize the states in $\mathcal{M}$ into four risk categories:

2

- *Safe*: From safe states, executing any action carries a low risk of violating safety in the future.
- *Critical*: In critical states, some, but not all, actions carry a too-high risk of violating safety.
- *Dangerous*: In dangerous states, all actions carry an unacceptably high risk of violating safety.
- *Unsafe*: States in which safety has already been violated.

Shield Computation. From this safety analysis, we first derive the *shield*. The task of the shield is to block actions from the RL agent that pose too high a risk. Thus, in safe states, the shield allows all actions. In critical states, the shield allows all actions whose risk is below the user-defined threshold. If no absolute-safety guarantees are enforced (i.e., the user sets the threshold to a value less than 1, allowing the agent to take some risks), the agent may end up in a dangerous state or even an unsafe one. In dangerous states, all actions induce a risk inevitably above the safety threshold, meaning safety is at risk but, for now, has not been violated. In this case, a fallback shielding strategy must be defined, such as selecting a predefined action (e.g., braking or landing). We follow a common approach and implement a shield that, in such situations, allows only the safest available action [23]. In unsafe states, safety is violated and the shield allows no actions (deadlocks). The resulting shield is a non-deterministic policy mapping states to allowed actions.

Level 1: Explaining the Risk of States. The highest-level explanation is represented in the form of a decision tree (DT) that categorizes states into the four risk categories (safe, critical, dangerous, and unsafe), helping users to determine whether the system is currently in a safety-critical situation and to assess the severity of the risk. For a given state, the predicates along the path in the DT provide an explanation for its categorization. *Example: In Fig. 1, a human operator asks for an explanation why the action of* keeping *the velocity is not allowed in the current state s, where the agent is at position 3 and an object is at position 7. The Level 1 DT explains, that the current state is* critical *because the distance d to the object is $3m \leq d \leq 10m$. Therefore, not all actions are safe to execute.*

At Runtime. The user may wish to understand why certain actions are considered safe or unsafe in the current situation. To explain the safety risk of actions, we compute a decision tree that explains the categorization of actions in the current state. If the user requests more detailed information about why a particular action is classified as unsafe, we compute an execution tree that summarizes the potential consequences of executing that action.

Level 2: Explaining the Risk of Actions. A Level 2 DT explains the shielding policy for the set of states represented by a leaf in the Level 1 DT. If the current state is categorized as critical in Level 1, the Level 2 DT explains which actions pose a low safety risk and are therefore permitted. If the current state is categorized as dangerous, the Level 2 DT explains what action is the safest one to execute. *Example: Continuing the scenario from Fig. 1, the Level 2 DT explains that, in the current state, only breaking is classified as safe action because the distance to the obstacle is less than $5m$. If the distance were greater than $5m$, maintaining the current velocity would also be considered safe.*

Level 3: Explaining the Consequences of Actions. If executing action a in a given state s is categorized as unsafe, we compute yet another tree that provides evidence for the violation of the property φ resulting from executing a in s. This final explanation is provided in the form of an *Execution Tree* (ET), which summarizes traces in the MDP $\mathcal{M}$ that start with executing a in s, followed by taking only the safest available action thereafter. This tree demonstrates that, even when the safest actions are taken after a, the property φ is violated with a probability that exceeds the safety threshold. *Example: Returning to the scenario described above, the Level 3 ET demonstrates that, if the speed is maintained in $(p_a = 3, p_o = 7)$, the next state will be $(p_a = 5, p_o = 7)$. From there, even braking would result in an unsafe state $(d = 0)$ within the next two steps.*

To compute the trees, the user can optionally provide predicates to guide the DT learning process. Note that the world model $\mathcal{M}$ and the safety specification φ are required in any shielded RL setting. As a result, our method can be easily integrated into any shielded RL application, as it does not require any additional information beyond what is already used to construct the shield.

Key Contributions of this Paper. (1) To the best of our knowledge, we introduce the first framework that combines explainability with formal safety guarantees for RL. (2) We provide a method for generating concise, case-based explanations that enable humans to understand the cause of safety violations as they occur. We overcome the non-scalability of a naive approach to explainability-via-DT by the *hierarchical decomposition* of explanations. (3) We evaluate our implementation on several challenging RL benchmarks, showing that the resulting explainable shields are compact and comprehensible, in contrast to traditional shields, which typically involve thousands of states.

2   Related Work

Shields were initially introduced in the context of reactive systems [10]. Later, [1] extended the concept to RL, ensuring absolute safety guarantees. In contrast, [23] adopted a quantitative perspective on safety, designing shields that provide probabilistic safety guarantees, thereby permitting the agent to take some calculated risks. These two fundamental concepts of shielded RL assume environments modeled as MDPs with discrete state and action spaces. Many extensions exist [34, 43, 35, 26]. For example, [15] studied shielded RL for *partially observable environments* where only a part of the state can be observed. Shields for *multi-agent systems* have been studied, in both centralized and decentralized settings [18, 29]. Shields for quantitative fairness properties have been studied in [6, 14]. In this work, we make a step toward bridging the gap between formal methods and explainable AI by computing explanations for rigorously computed shields, as introduced by [23].

A comprehensive literature review on explainable RL can be found in [31, 49]. Research directions include human-in-the-loop approaches [40], policy summarization [2, 44], training process visualization [32], and methods that identify performance-critical states [16, 20, 36]. These methods primarily focus on explaining why the agent's decisions are optimal for maximizing the expected reward. The computation of case-based explanations has been explored in [46, 13]. In contrast to explaining the full decision-making intent of the RL policy, our approach explains why certain actions are considered safety-critical.

DTs are widely used in explainable AI due to their intuitive, human-readable structure [17]. In the context of explainable RL, DTs have been constructed to mimic the trained policy of RL agents [9, 30, 47, 19, 38]. In contrast, we compute DTs to explain the shield, resulting in compact explanations that specifically focus on the safety aspects of the problem Furthermore, our approach preserves both optimality and scalability by retaining the RL agent for decision-making. Recently, several works have explored the computation of DTs to represent optimal policies in MDPs [25, 3, 48]. However, these approaches focus on explaining the available actions, not the safety aspects, and are often not scalable to the large environments and non-deterministic policies of the RL context.

3   Background

Markov Decision Processes. A *Markov decision process (MDP)* [42] is a tuple $\mathcal{M} = \langle \mathcal{S}, s_0, \mathcal{A}, \mathcal{P} \rangle$ where $\mathcal{S}$ is a finite set of states, $s_0 \in \mathcal{S}$ is the initial state, $\mathcal{A}$ is a finite set of actions, and $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to Dist(\mathcal{S})$ is the probabilistic transition function. For all $s \in \mathcal{S}$, the available actions are $\mathcal{A}(s) = \{a \in \mathcal{A} \mid \exists s', \mathcal{P}(s, a)(s') \neq 0\}$ and we assume $|\mathcal{A}(s)| \geq 1$. Let $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ be a set of features. A state $s \in \mathcal{S}$ can be represented as a tuple $s = (\overline{v_1}, \ldots, \overline{v_n})$ where $\overline{v_i} = s(v_i) \in \mathbb{Z}$ is the assigned value to feature v_i.

Choices in an MDP are resolved via policies. A (memoryless non-randomizing) *non-deterministic policy* is a relation $\pi : \mathcal{S} \to 2^{\mathcal{A}}$. A (memoryless non-randomizing) *deterministic policy* is a function $\pi : \mathcal{S} \to \mathcal{A}$. Applying a deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$ to an MDP $\mathcal{M}$ induces a Markov chain (MC) $\mathcal{M}^\pi = \langle \mathcal{S}, s_0, \mathcal{P} \rangle$ with $\mathcal{P} : \mathcal{S} \to Dist(\mathcal{S})$ where all nondeterminism is resolved. A finite *trace* $\tau = \{s_0 s_1 s_2 ... s_n\}$ in a MC is a sequences of states such that $\mathcal{P}(s_i, s_{i+1}) > 0$. The probability mass of such a trace is defined as $\mathbb{P}(s_0 s_1 s_2 ... s_n) = \prod_{0 \leq i < n} \mathcal{P}(s_i, s_{i+1})$.

Probabilistic Model Checking. We consider safety properties expressed in Computation Tree Logic (CTL) [8]. Informally, a safety property specifies that 'something bad never happens'. Such properties can represent invariants like 'collisions are never allowed' as well as temporal properties like: 'The agent must not reach an unsafe position within 30 steps.' Probabilistic model checking [8] computes the probability of satisfying a safety property φ over a finite or infinite horizon, using adaptations of value iteration, policy iteration, or linear programming. We define the properties below with a bound n. For the unbounded horizon, $n = \infty$. For a given MDP $\mathcal{M}$, and a property φ in CTL, model checking computes the following probabilities:

- $\mathbb{P}_{\mathcal{M}^\pi, \varphi} : \mathcal{S} \times \mathbb{N} \to [0, 1]$ is the expected probability to satisfy φ in the MC $\mathcal{M}^\pi$ from a state $s \in \mathcal{S}$ within n steps for a deterministic policy π.
- $\mathbb{P}_{\mathcal{M}, \varphi}^{\min}(s, h) = \min_\pi \mathbb{P}_{\mathcal{M}^\pi, \varphi}(s, h)$ is the *minimal* expected probability in the MDP $\mathcal{M}$ *over all policies* from a state s within h steps.
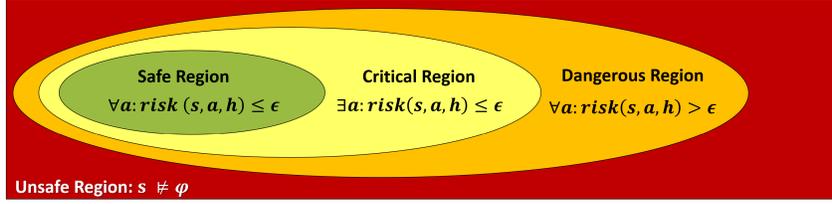
Figure 2: Risk-based Categorization of States.

Shields. A *shield* $\pi_{shield} : \mathcal{S} \to 2^{\mathcal{A}}$ for an MDP $\mathcal{M}$ can be represented as a *non-deterministic policy*. Shields are computed from a safety specification φ in CTL and an abstract model of the environment that captures the safety-relevant aspects of the full MDP [23]). Computing the shield using only a safety-relevant fragment of the full MDP enables the scalability of shielded RL. Different types of shields are distinguished by the safety guarantees they provide and the types of used world models. We follow the approach of Jansen et al.[23] and compute shields that offer probabilistic safety guarantees. We discuss the details in Section 4.

Reinforcement Learning. In reinforcement learning (RL) [42], an agent learns a task via interactions with an unknown environment modeled by an MDP $\mathcal{M}$ with an associated reward function $\mathcal{R} : \mathcal{S} \to \mathbb{R}$. In each state $s \in \mathcal{S}$, the agent chooses an action $a \in \mathcal{A}$. The environment then moves to a state s' with probability $\mathcal{P}(s, a, s')$. The return $\texttt{ret}_\rho$ of an execution ρ is the discounted cumulative reward defined by $\texttt{ret}_\rho = \Sigma_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t)$, using the discount factor $\gamma \in [0, 1]$. The objective of the agent is to learn an *optimal policy* $\pi^\star : \mathcal{S} \to \mathcal{A}$ that maximizes the expectation of the return.

Decision Trees. We use *decision trees* (DTs) [11] to represent nondeterministic policies in MDPs. A DT over the set of features $\mathcal{V}$ is a tuple $\mathcal{T} = \langle Tr, \Gamma, \rho, \Lambda, \lambda \rangle$ where Tr is a finite, rooted, binary, ordered tree consisting of a set of *inner nodes* N and a set of *leaves* L.

The set of *predicates* over $\mathcal{V}$ is denoted by Γ. We define a set of basic predicates $\Gamma[v_i \sim const]$, where $v_i \in \mathcal{V}$, $const \in \mathbb{Z}$, and $\sim \in \{\leq, <, \geq, >, =\}$. The function $\rho : N \to \Gamma$ assigns to every inner node $n \in N$ a predicate $p \in \Gamma$. The set Λ specifies a set of *labels*. The *labeling function* $\lambda : L \to \Lambda$ assigns to each leaf $l \in L$, a label $\gamma \in \Lambda$.

A DT $\mathcal{T}$ defines a function $f : \mathbb{Z}^d \to \mathbb{Z}$, with $d = |\mathcal{V}|$, as follows. Given an input vector (a state in the MDP) $s = (\overline{v_1}, \ldots, \overline{v_d}) \in \mathbb{Z}^d$, one follows a unique path p from the root to a leaf $l \in L$ s.t. for each inner node $n \in N$ on the path, the predicate $\rho(n)$ evaluates to true under the substitution $v_i = \overline{v_i}$ iff the first (typically left) child of n lies on p. We define $\mathcal{T}(s) = l$ to be the leaf the state s reaches. The value of the function f on s is defined as $f(s) = f(\overline{v_1}, \ldots, \overline{v_n}) = \lambda(l)$. We define $l_s : L \to 2^{\mathcal{S}}$ with $\mathcal{S} = \mathbb{Z}^d$ as the function that maps a leaf l to the set of states whose path ends in l.

Execution Trees. For a given MC $\mathcal{M} = \langle \mathcal{S}, s_0, \mathcal{P} \rangle$ and a set of finite traces $\Pi = \{\tau_1 \ldots \tau_n\}$ of $\mathcal{M}$, an *execution tree* (ET) represents $\tau_1 \ldots \tau_n$ of $\mathcal{M}$ in a tree structure. An execution tree $\mathcal{T} = \langle N, E \rangle$ is a rooted tree consisting of a set of nodes N and a set of edges $E : N \to N$. Nodes are labeled with states $s \in \mathcal{S}$. An edge $e \in E$ corresponds to a transition in $\mathcal{P}$. Each trace $\tau \in \Pi$ defines a path p in $\mathcal{T}$. Thus, each path $p = \{n_1, n_2, \ldots n_n\}$ corresponds to a trace $\tau_i = \{s_1, s_2, \ldots s_n\} \in \Pi$ such that the node n_i is labeled with the state s_i for $1 \leq i \leq n$.

4 Computing Hierarchical Safety Explanations

In this section, we present our algorithm for explainable safe RL in detail. We begin by describing the construction of the shield and the Level 1 DT in Sec. 4.1. In Sec. 4.2, we introduce the Level 2 and Level 3 trees, which provide case-based explanations at runtime. To compute the shield and the explanations, our algorithm takes the safety specification φ and an MDP $\mathcal{M} = \langle \mathcal{S}, s_0, \mathcal{A}, \mathcal{P} \rangle$. Optionally, it accepts user-defined predicates to guide the DT learning.

4.1 Shield Computation and High-Level Safety Explanations

Executing an action in a given state may carry some risk of violating safety at some point in the future. Based on this risk, states are categorized as *safe*, *critical*, *dangerous*, or *unsafe*.

Definition 1. *(Risk of Safety Violation) Given an MDP $\mathcal{M} = \langle \mathcal{S}, s_0, \mathcal{A}, \mathcal{P} \rangle$, a safety property φ, and a finite horizon h, we define the* risk *of violating φ from a state $s \in \mathcal{S}$ in the next h steps as a function $risk_{\mathcal{M},\varphi} : \mathcal{S} \times \mathbb{N} \to [0,1]$ as follows:*

$$\forall s \in \mathcal{S} : risk_{\mathcal{M},\varphi}(s, h) = \mathbb{P}^{\mathsf{min}}_{\mathcal{M}, \neg\varphi}(s, h).$$

The risk *of violating φ from a state $s \in \mathcal{S}$ after executing an action $a \in \mathcal{A}$ in the next h steps is defined via the function $risk_{\mathcal{M},\varphi} : \mathcal{S} \times \mathcal{A} \times \mathbb{N} \to [0,1]$ as follows:*

$$\forall s \in \mathcal{S}, \forall a \in \mathcal{A} : risk_{\mathcal{M},\varphi}(s, a, h) = \sum_{s' \in \mathcal{S}} (\mathcal{P}(s, a, s') \cdot risk_{\mathcal{M},\varphi}(s', h-1)).$$

The risk of a state s is defined as the *minimal expected probability* of reaching a state that violates φ within the next h steps, quantified over all policies; that is, the probability of a safety violation under the safest available policy. The risk of executing an action a in a given state s is the accumulated, weighted risk of the successor states reached by executing a in s. The risk across all states in the state space can be computed using standard probabilistic model checking algorithms, such as value iteration or dynamic programming [8], with tools like PRISM [27], STORM [22], or TEMPEST [34].

Definition 2. *(Safety of Actions) Given an MDP $\mathcal{M}$, a safety property φ, a finite horizon h, and a user-defined safety threshold $\epsilon \in [0,1]$. For a given state action pair $(s,a) \in \mathbf{S} \times \mathcal{A}$, action a is called* safe *in s if $risk_{\mathcal{M},\varphi}(s, a, h) \leq \epsilon$. Otherwise, a is* unsafe *in s.*

We partition the state space into $\mathcal{S} = \mathcal{S}_s \cup \mathcal{S}_c \cup \mathcal{S}_d \cup \mathcal{S}_u$ according to the risk per state as follows:

Definition 3. *(Safe States $\mathcal{S}_s$) $\forall s \in \mathcal{S}. \ s \in \mathcal{S}_s$ iff $\forall a \in \mathcal{A}: risk_{\mathcal{M},\varphi}(s, a, h) \leq \epsilon$.*

Definition 4. *(Critical States $\mathcal{S}_c$) $\forall s \in \mathcal{S}. \ s \in \mathcal{S}_c$ iff $\exists a \in \mathcal{A}: risk_{\mathcal{M},\varphi}(s, a, h) \leq \epsilon$ and $\exists a' \in \mathcal{A}: risk_{\mathcal{M},\varphi}(s, a', h) > \epsilon$.*

Definition 5. *(Dangerous States $\mathcal{S}_d$) $\forall s \in \mathcal{S}. \ s \in \mathcal{S}_d$ iff $s \models \varphi$ and $\forall a \in \mathcal{A}: risk_{\mathcal{M},\varphi}(s, a, h) > \epsilon$.*

**Definition 6.** *(Unsafe States $\mathcal{S}_u$) $\forall s \in \mathcal{S}. \ s \in \mathcal{S}_d$ iff $s \not\models \varphi$.*

The categorization of states is illustrated in Fig. 2. In safe states, all available actions are safe, i.e., they have a risk of at most ϵ. In critical states, only some actions are safe, while others may exceed the safety threshold. In dangerous states, all available actions are unsafe, i.e., each carries a risk of violating the safety property greater than ϵ, although no safety violation has occurred yet. A state is considered unsafe if it violates the safety property.

Computation of the shield $\pi_{\mathbf{shield}}$. Using the computed risk function $risk_{\mathcal{M},\varphi}$ and the classification of states, the shielding policy $\pi_{\mathrm{shield}} : \mathcal{S} \to 2^{\mathcal{A}}$ is computed as follows:

$$(s, a) \in \pi_{\mathrm{shield}} \iff \begin{cases} risk_{\mathcal{M},\varphi}(s, a, h) \leq \epsilon, & \text{or} \\ s \in \mathcal{S}_d \text{ and } risk_{\mathcal{M},\varphi}(s, a, h) = \min_{a' \in \mathcal{A}} risk_{\mathcal{M},\varphi}(s, a', h). \end{cases}$$

The shield permits all actions that are classified safe and blocks all unsafe actions. The only exception is in dangerous states, where the shield allows the agent to take the action with the lowest risk. Note that different approaches exist for defining shielding strategies in dangerous states, for example, selecting a predefined fallback action such as halting. Next, we compute the Level-1 DT $\mathcal{T}_{L1}$, which classifies states in the state space according to their risk category, providing an initial explanation of why a given state may be safety-critical.

Computation of DT $\mathcal{T}_{L1}$. We construct the decision tree $\mathcal{T}_{L1} = (Tr_{L1}, \Gamma_{L1}, \rho_{L1}, \Lambda_{L1}, \lambda_{L1})$ for a given MDP $\mathcal{M}$, finite horizon h, and safety threshold ϵ through the following steps. First, we define the set of predicates Γ_{L1} as follows. Γ_{L1} contains the basic predicates $[v_i \sim const]$, where $v_i \in \mathcal{V}, const \in \mathbb{Z}$, and $\sim \in \{\leq, <, \geq, >, =\}$. Additionally, this set can be extended by user-defined predicates to incorporate domain knowledge. For example, predicates measuring the distance of a state s to the unsafe region S_u often enable intuitive explanations of safety-critical dynamics. Such functions could capture, for example, the distance between two agents or the difference between the current temperature and an overheating threshold. These user-defined predicates are of the form $p_i = k_i(v_1, v_2, \ldots, v_n) \sim const_i$ with $k_i : \mathcal{S} \to \mathbb{Z}$. Second, the classification of the states serves as the labels for the leaves of $\mathcal{T}_{L1}$. Thus, $\Lambda_{L1} = \{s, c, d, u\}$. Lastly, we learn $\mathcal{T}_{L1}$ (i.e., its underlying tree Tr_{L1} and functions ρ_{L1} and λ_{L1}) using standard DT learning algorithms [33, 39]. We learn an exact DT s.t. for all leaves l of $\mathcal{T}_{L1}$, $l_s(l) \subseteq S_x$ for $x \in \{s, c, d, u\}$. Intuitively, every $s \in \mathcal{S}$ is classified exactly, and the tree achieves perfect accuracy.
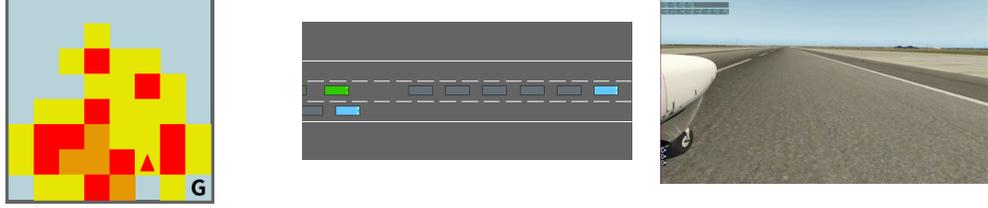
(a) A Frozen Lake environment.   (b) A Highway environment.   (c) The Taxiing environment.

Figure 3: The RL benchmarks used for evaluation of our approach.

4.2 Case-based Safety Explanations

At runtime, our framework provides case-based explanations via two additional trees. Given a current state s, a Level 2 decision tree explains which actions the RL agent is allowed to explore from s. For a given unsafe action a, the Level 3 execution tree explains why executing a is unsafe in s.

Computation of DT $\mathcal{T}_{L2}$. Let s be the current state. If $s \in \mathcal{S}_s$, all actions are classified as safe. If $s \in \mathcal{S}_u$, safety is already violated. Therefore, we compute action-dependent safety explanations only for critical and dangerous states. In *critical* states $s \in \mathcal{S}_c$, some actions are safe and some are unsafe (i.e., $\exists a\colon risk_{\mathcal{M},\varphi}(s,a,h) \leq \epsilon$ and $\exists a'\colon risk_{\mathcal{M},\varphi}(s,a',h) > \epsilon$.). A $\mathcal{T}_{L2}$ for a critical state s explains why actions are safe or unsafe in s. In *dangerous states*, all actions are unsafe, i.e., $\forall a\colon risk_{\mathcal{M},\varphi}(s,a,h) > \epsilon$. In such states, our shield allows the agent to explore the safest available action (which still exceeds the safety threshold ϵ). Then, the $\mathcal{T}_{L2}$ provides an explanation for why the selected action is considered the safest among the available options.

Let s be the current state and $l = \mathcal{T}_{L1}(s)$ be the leaf node reached by state s in $\mathcal{T}_{L1}$. If $s \in \mathcal{S}_c$ or $s \in \mathcal{S}_d$, we compute a Level-2 DT $\mathcal{T}_{L2}^l = (Tr_{L2}^l, \Gamma_{L2}^l, \rho_{L2}^l, \Lambda_{L2}^l, \lambda_{L2}^l)$ over the states $l_s(l)$ as follows. The set of predicates Γ_{L2}^l is equal to Γ_{L1}. The set of labels Λ_{L2}^l is defined as sets over actions, since the $\mathcal{T}_{L2}^l$ explains which actions are allowed by the shield, i.e., $\Lambda_{L2}^l = 2^{\mathcal{A}}$.

Computation of ET $\mathcal{T}_{L3}$. Given a current state s_{cur} and an unsafe action a_u, the ET $\mathcal{T}_{L3}$ explains that executing a_u in s_{cur} leads to unsafe states with a probability greater than ϵ. The tree provides this explanation by representing a set of traces that start in a state reached after executing a_u in s_{cur} and ending in an unsafe state, and that provide enough evidence to demonstrate that a_u is unsafe. For a given state pair (s_{cur}, a_u), we compute the $\mathcal{T}_{L3}^{s_{cur},a_u}$ as follows. First, the MDP $\mathcal{M}$ is transformed into an MC $\mathcal{M}^\pi = \langle \mathcal{S}, s_0, \mathcal{P} \rangle$ via the following policy π:

$$\forall s : \pi(s) = \begin{cases} a_u & \text{if } s = s_{cur}, \text{and} \\ \underset{a \in \mathcal{A}}{\arg\min}\ risk_{\mathcal{M},\varphi}(s,a,h) & \text{otherwise.} \end{cases}$$

Thus, the policy π picks in any state the safest available action. Only in the current state s_{cur}, the policy π selects the action a_u. Therefore, traces sampled from this policy demonstrate possible consequences of executing a_u in s_{cur}, if afterward only the safest actions were executed. We call a finite trace $\tau = \{s_0, \ldots, s_n\}$ *unsafe* if $s_n \in \mathcal{S}_u$. We compute a set of traces $\Pi = \{\tau_1, \tau_2, \ldots \tau_k\}$ that start in state s_{cur} in the MC $\mathcal{M}^\pi$ such that the probability mass of these traces exceeds the safety threshold ϵ, i.e. $\sum_{\tau \in \Pi} \mathbb{P}(\tau) \geq \epsilon$. We follow the algorithm from [21] to compute the most probable traces in the MC $\mathcal{M}^\pi$ that are unsafe and their probability mass exceeds the threshold ϵ. The algorithm uses the recursive enumeration algorithm for computing the k shortest paths on a weighted digraph [24], substituting probabilities in $\mathcal{M}^\pi$ with distances. This algorithm computes the shortest paths recursively until the probability mass of the traces is greater than ϵ. The resulting traces are prefix-merged to construct the execution tree $\mathcal{T}_{L3}^{(s_{cur},a_u)}$.

5   Experimental Evaluation

In this section, we present the experimental evaluation of our approach. We consider the size of a shield $|\pi_{shield}|$ as the size of the lookup table and the size of a tree $|\mathcal{T}|$ as the number of its nodes. The tree size serves as our metric for evaluating explainability. The model checking queries were computed using TEMPEST [34], and the DT representations of shields using DTCONTROL [5]. All experiments were conducted on a laptop with an Intel® Core™i7-11800H CPU at 2.3 GHz with 32

7

(a) $\mathcal{T}_{L1}$ (b) $\mathcal{T}_{L2}$, dangerous leaf   (c) $\mathcal{T}_{L3}^{(6,7),N}$
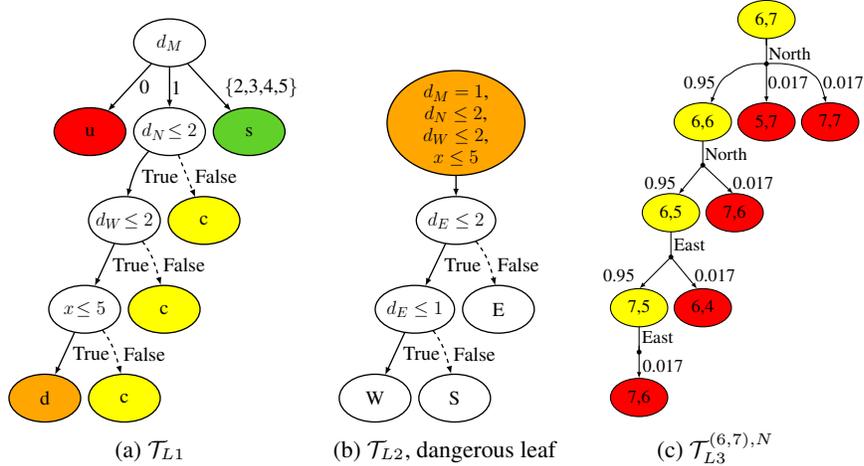
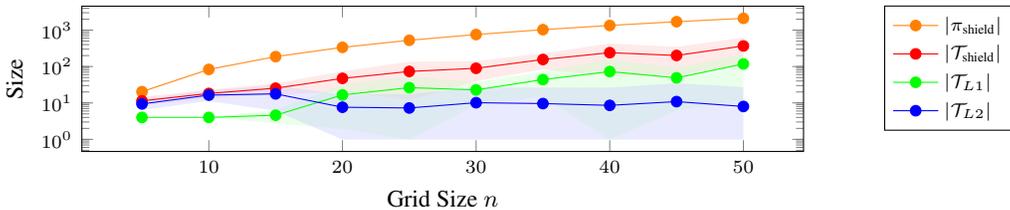Figure 4: Exemplary explanations for the Frozen Lake environment.

Figure 5: Shield and tree sizes for different configurations of the Frozen Lake environment.

GB of RAM. All details of the experimental setup can be found in the Appendix. We provide the implementation as supplementary material.

5.1 Frozen Lake

We performed a first set of experiments using the Farama Frozen Lake environment [45]. A Frozen Lake environment is an $n \times n$ grid, consisting of blue slippery tiles and holes. The agent can move in cardinal directions, where every movement carries a probability of 0.05 of slipping into a different direction. The RL agent has to reach the goal **G** while not falling into a hole. The shield preventing the agent from falling into a hole is computed with a horizon $h = \infty$ and a risk threshold $\epsilon = 0.075$. The state classification of an example environment is shown in Fig. 3a. Red tiles indicate unsafe states (holes), orange tiles indicate dangerous states, yellow tiles indicate critical states, and blue tiles indicate safe states. The set of predicates $\Gamma = \{x, y, d_M, d_N, d_E, d_S, d_W\}$ consists of the agent's coordinates (x, y), the Manhattan distance d_M to the nearest hole, and the distance d_a to the nearest hole in the cardinal direction $a \in \{N, S, E, W\}$.

Explaining Safety. Fig. 4a shows $\mathcal{T}_{L1}$ for the environment depicted in Fig. 3a. $\mathcal{T}_{L1}$ can concisely capture the states in which the shield needs to interfere, by using the user-defined distance predicates. Fig. 4b shows $\mathcal{T}_{L2}$ for the dangerous states in $\mathcal{T}_{L1}$. The tree summarizes that, to exit a dangerous state, the agent should move east if the distance to a hole in that direction permits it; otherwise, it needs to move to the nearest safe state. Finally, Fig. 4c shows why moving to the *North* at position $(6, 7)$ is not allowed. The represented traces in $\mathcal{T}_{L3}^{(6,7),N}$ have a probability mass of 0.08 of reaching an unsafe state, which exceeds the allowed risk of 0.075.

Results. We evaluated the scalability of our approach using randomly generated instances of Frozen Lake environments of increasing size $n \times n$ with $n \in \{5, 10, \ldots, 50\}$. Per size, we generate 10 random instances and compare the sizes of the computed shields and tree representations. Fig. 5 shows the average sizes of the shielding policy $|\pi_{shield}|$, the shield represented as one single tree $|\mathcal{T}_{\text{shield}}|$, and the trees $|\mathcal{T}_{L1}|$, and $|\mathcal{T}_{L2}|$ over grid size $n \times n$. The results show that, using our approach, we obtain DTs that are several orders of magnitude smaller than the shielding policy. The average
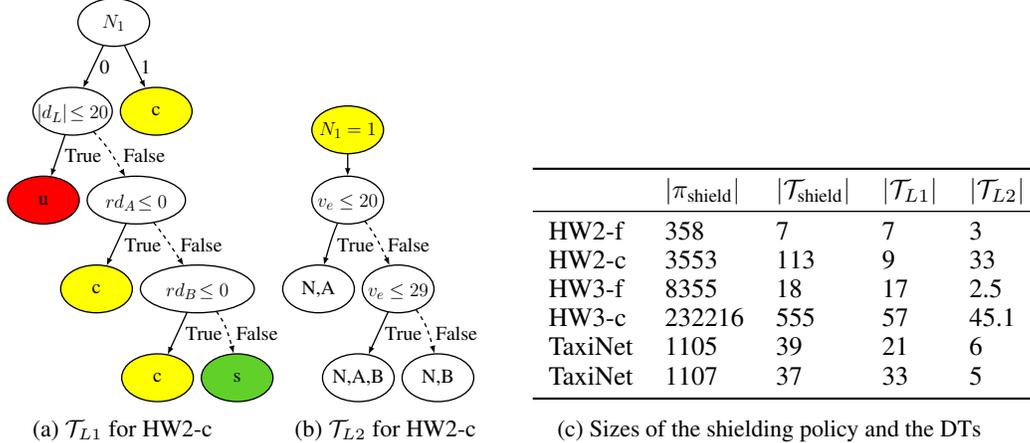
8

	$|\pi_{\text{shield}}|$	$|\mathcal{T}_{\text{shield}}|$	$|\mathcal{T}_{L1}|$	$|\mathcal{T}_{L2}|$
HW2-f	358	7	7	3
HW2-c	3553	113	9	33
HW3-f	8355	18	17	2.5
HW3-c	232216	555	57	45.1
TaxiNet	1105	39	21	6
TaxiNet	1107	37	33	5

(a) $\mathcal{T}_{L1}$ for HW2-c (b) $\mathcal{T}_{L2}$ for HW2-c (c) Sizes of the shielding policy and the DTs

Figure 6: Exemplary DTs for HW2-c and the comparison of the sizes of the shield and decision trees.

time for computing the shield ranges from 0.15s for $n = 5$ to $15.4s$ for $n = 50$. The average time to learn the DTs ($\mathcal{T}_{L1}$ or $\mathcal{T}_{L2}$) is approximately 2s for all grid sizes.

5.2 Highway Cruise Control

We conducted our second set of experiments in the Farama Highway environment [28], illustrated in Fig. 3b. In this environment, the agent controls a self-driving car operating on a highway populated with other vehicles. The agent's actions are: switching lanes (S_L, S_R), accelerating (A), braking (B), or performing no operation (N). The agent's goal is to reach the end of the road without collisions. Additionally, it receives a positive reward for driving in the leftmost lane. We compute a shield that ensures collision avoidance by enforcing a safe distance of 20m. In this example, the shield prohibits the agent from taking any risks (i.e., $\epsilon = 0$) with $h = \infty$. We consider two parameters: two or three lanes (HW2/HW3) and whether the agent can change its velocity (-f(ixed)/-c(hangeable)). This results in the scenarios HW2-f, HW2-c, HW3-f, and HW3-c. We discuss the results of HW2-c. Appendix A.2 provides further results and precise predicate definitions. The set of predicates $\Gamma = \{d_1, l_e, d_L, N_1, rd_A, rd_B\}$ includes: the distance d_1 to the nearest vehicle in lane 1; the current lane of the ego vehicle l_e; the distance d_L to the closest vehicle in the ego vehicle's lane; N_1 indicating whether the ego vehicle is next to a vehicle in lane 1; and final predicates encoding the remaining distance to other cars when the agent accelerates (rd_A) or brakes (rd_B).

Explaining Safety. Fig. 6a shows the $\mathcal{T}_{L1}$, and 6b shows the $\mathcal{T}_{L2}$ for the topmost critical node. The trees explain that if the ego car is next to a car in the adjacent lane, switching lanes is not allowed. Moreover, the agent may only accelerate or brake within its velocity bounds of $20 - 29$m/s.

Results. Table 6c shows the size of the shielding policy and the average sizes of DTs of different levels and scenarios. The results show, as expected, that representing shields via trees instead of lookup tables yields a more compact representation. Furthermore, using the hierarchical explanations $\mathcal{T}_{L1}$ and $\mathcal{T}_{L2}$ is, in most cases, more compact than using a single tree. The time for computing the shield ranges from 0.1s for HW2-f to $50s$ for HW3-c. The time to learn the DTs ranges from 1.3s for $\mathcal{T}_{L1}$ of HW2-f to 17.4s of $\mathcal{T}_{L1}$ for HW3-f.

5.3 Boeing Taxinet

For our final set of experiments, we applied our approach to an autonomous taxiing environment [41], as shown in Fig. 3c. In this environment, the agent must steer an aircraft to ensure it remains aligned with the centerline. Depending on the current heading, the aircraft's position relative to the centerline changes. The safety objective is to avoid exceeding a heading of $20°$ and a maximum deviation of 0.8m from the centerline. The shield is computed with a horizon $h = \infty$ and a risk threshold $\epsilon = 0.0$. The set of predicates $\Gamma = \{he, cte, |he|, |cte|, d\}$ consists of the heading error he, centerline error cte, their absolute values, and the distance d to the maximum allowed centerline error, depending on the current heading. The set of actions $\mathcal{A} = L, N, R$ allows the agent to steer left (L) or right (R) in steps of $5°$, or to take no action (N). We have applied our approach

9

to two different instances and show the resulting sizes for the shields and the DTs in Table 6c. The results again show that our approach yields compact representations of the shielding policy. We provide the corresponding DTs and further discussion in the Appendix.

6 Conclusion & Future Work

We presented a method for "explainably safe" RL via proposed explainable shields. Our approach provides case-based explanations of the shielding policy and a hierarchy of decision trees that explains the risks associated with states and actions, as well as the consequences of executing unsafe actions. Our experiments show the capability of our method in providing small trees even for complex scenarios. In future work, we will explore automated methods for predicate generation to reduce reliance on user input. Besides, we plan to investigate the potential of generative AI to render case-based explanations understandable to non-specialists. Furthermore, a user study on the understandability of our explanations is also of interest. Lastly, we plan to extend our approach by computing compact explanations for states that the trained RL policy visits frequently. As a trained RL policy rarely operates over the entire state space, focusing explanations on the most frequently visited states will provide clearer insights into the safety-relevant aspects of the agent's behavior.

References

[1] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 2669–2678. AAAI Press, 2018.

[2] Ofra Amir, Finale Doshi-Velez, and David Sarne. Summarizing agent strategies. *Autonomous Agents and Multi-Agent Systems*, 33(5):628–644, 2019.

[3] Roman Andriushchenko, Milan Ceska, Sebastian Junges, and Filip Macák. Small decision trees for mdps with deductive synthesis. In Ruzica Piskac and Zvonimir Rakamaric, editors, *Proceedings of the 37st International Conference on Computer Aided Verification, CAV 2025*, volume 15932 of *Lecture Notes in Computer Science*, pages 169–192. Springer, 2025.

[4] Pranav Ashok, Jan Kretínský, Kim Guldstrand Larsen, Adrien Le Coënt, Jakob Haahr Taankvist, and Maximilian Weininger. SOS: safe, optimal and small strategies for hybrid markov decision processes. In David Parker and Verena Wolf, editors, *Proceedings of the 16th International Conference on Quantitative Evaluation of Systems, QEST 2019*, volume 11785 of *Lecture Notes in Computer Science*, pages 147–164. Springer, 2019.

[5] Pranav Ashok, Mathias Jackermeier, Jan Kretínský, Christoph Weinhuber, Maximilian Weininger, and Mayank Yadav. dtcontrol 2.0: Explainable strategy representation via decision

tree learning steered by experts. In *Proceedings of the 27th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2007*, volume 12652 of *Lecture Notes in Computer Science*, pages 326–345. Springer, 2021.

[6] Guy Avni, Roderick Bloem, Krishnendu Chatterjee, Thomas A. Henzinger, Bettina Könighofer, and Stefan Pranger. Run-time optimization for learned controllers through quantitative games. In Isil Dillig and Serdar Tasiran, editors, *Proceedings of the 31st International Conference on Computer Aided Verification, CAV 2019*.

[7] Muqsit Azeem, Debraj Chakraborty, Sudeep Kanav, Jan Kretínský, MohammadSadegh Mohagheghi, Stefanie Mohr, and Maximilian Weininger. 1-2-3-go! policy synthesis for parameterized markov decision processes via decision-tree learning and generalization. In Shankaranarayanan Krishna, Sriram Sankaranarayanan, and Ashutosh Trivedi, editors, *Proceedings of the 26th International Conference on Verification, Model Checking, and Abstract Interpretation, VMCAI 2025*, volume 15530 of *Lecture Notes in Computer Science*, pages 97–120. Springer, 2025.

[8] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT press, 2008.

[9] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 2499–2509, 2018.

[10] Roderick Bloem, Bettina Könighofer, Robert Könighofer, and Chao Wang. Shield synthesis: - runtime enforcement for reactive systems. In Christel Baier and Cesare Tinelli, editors, *Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2015*, volume 9035 of *Lecture Notes in Computer Science*, pages 533–548. Springer, 2015.

[11] Tomás Brázdil, Krishnendu Chatterjee, Martin Chmelik, Andreas Fellner, and Jan Kretínský. Counterexample explanation by learning small strategies in markov decision processes. In Daniel Kroening and Corina S. Pasareanu, editors, *Proceedings of the 27th International Conference on Computer Aided Verification, CAV 2015*, volume 9206 of *Lecture Notes in Computer Science*, pages 158–177. Springer, 2015.

[12] Carlos E. Budde, Pedro R. D'Argenio, and Arnd Hartmanns. Digging for decision trees: A case study in strategy sampling and learning. In Bernhard Steffen, editor, *Proceedings of the 2nd International Conference on Bridging the Gap between AI and Reality, AISoLA 2023*, volume 15217 of *Lecture Notes in Computer Science*, pages 354–378. Springer, 2024.

[13] Carrie J. Cai, Jonas Jongejan, and Jess Holbrook. The effects of example-based explanations in a machine learning interface. In Wai-Tat Fu, Shimei Pan, Oliver Brdiczka, Polo Chau, and Gaelle Calvary, editors, *Proceedings of the 24th International Conference on Intelligent User Interfaces, IUI 2019*, pages 258–262. ACM, 2019.

[14] Filip Cano, Thomas A. Henzinger, Bettina Könighofer, Konstantin Kueffner, and Kaushik Mallik. Fairness shields: Safeguarding against biased decision makers. In Toby Walsh, Julie Shah, and Zico Kolter, editors, *Proceedings of the 39nd AAAI Conference on Artificial Intelligence, AAAI 2025*, pages 15659–15668. AAAI Press, 2025.

[15] Steven Carr, Nils Jansen, Sebastian Junges, and Ufuk Topcu. Safe reinforcement learning via shielding under partial observability. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Proceedings of the 37th AAAI Conference on Artificial Intelligence, AAAI 2023*, pages 14748–14756. AAAI Press, 2023.

[16] Zelei Cheng, Xian Wu, Jiahao Yu, Wenhai Sun, Wenbo Guo, and Xinyu Xing. Statemask: Explaining deep reinforcement learning through state mask. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems, NeurIPS 2023*, 2023.

[17] Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Omer F. Rana, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, and Rajiv Ranjan. Explainable AI (XAI): core ideas, techniques, and solutions. *ACM Computing Survey*, 55(9):194:1–194:33, 2023.

[18] Ingy Elsayed-Aly, Suda Bharadwaj, Christopher Amato, Rüdiger Ehlers, Ufuk Topcu, and Lu Feng. Safe multi-agent reinforcement learning via shielding. In Frank Dignum, Alessio Lomuscio, Ulle Endriss, and Ann Nowé, editors, *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2021*, pages 483–491. ACM, 2021.

[19] Claire Glanois, Paul Weng, Matthieu Zimmer, Dong Li, Tianpei Yang, Jianye Hao, and Wulong Liu. A survey on interpretable reinforcement learning. *Machine Learning*, 113(8):5847–5890, 2024.

[20] Wenbo Guo, Xian Wu, Usmann Khan, and Xinyu Xing. EDGE: explaining deep reinforcement learning policies. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, NeurIPS 2021*, pages 12222–12236, 2021.

[21] Tingting Han and Joost-Pieter Katoen. Counterexamples in probabilistic model checking. In Orna Grumberg and Michael Huth, editors, *Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2007*, volume 4424 of *Lecture Notes in Computer Science*, pages 72–86. Springer, 2007.

[22] Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. The probabilistic model checker storm. *International Journal on Software Tools for Technology Transfer*, 24(4):589–610, 2022.

[23] Nils Jansen, Bettina Könighofer, Sebastian Junges, Alex Serban, and Roderick Bloem. Safe reinforcement learning using probabilistic shields. In Igor Konnov and Laura Kovács, editors, *Proceedings of the 31st International Conference on Concurrency Theory, CONCUR 2020*, volume 171 of *LIPIcs*, pages 3:1–3:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[24] Víctor M. Jiménez and Andrés Marzal. Computing the K shortest paths: A new algorithm and an experimental comparison. In Jeffrey Scott Vitter and Christos D. Zaroliagis, editors, *3rd International Workshop on Algorithm Engineering, WAE '99*, volume 1668 of *Lecture Notes in Computer Science*, pages 15–29. Springer, 1999.

[25] Florian Jüngermann, Jan Kretínský, and Maximilian Weininger. Algebraically explainable controllers: decision trees and support vector machines join forces. *International Journal on Software Tools for Technology Transfer*, 25(3):249–266, 2023.

[26] Bettina Könighofer, Roderick Bloem, Rüdiger Ehlers, and Christian Pek. Correct-by-construction runtime enforcement in AI - A survey. In *Principles of Systems Design - Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday*, volume 13660 of *Lecture Notes in Computer Science*, pages 650–663. Springer, 2022.

[27] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Proceedings of the 23rd International Conference on Computer Aided Verification, CAV 2011*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011.

[28] Edouard Leurent. An environment for autonomous driving decision-making. https://github.com/eleurent/highway-env, 2018.

[29] Daniel Melcer, Christopher Amato, and Stavros Tripakis. Shield decentralization for safe multi-agent reinforcement learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems, NeurIPS 2022*. Curran Associates, Inc., 2022.

[30] Stephanie Milani, Zhicheng Zhang, Nicholay Topin, Zheyuan Ryan Shi, Charles A. Kamhoua, Evangelos E. Papalexakis, and Fei Fang. MAVIPER: learning decision tree policies for interpretable multi-agent reinforcement learning. In Massih-Reza Amini, Stéphane Canu, Asja Fischer, Tias Guns, Petra Kralj Novak, and Grigorios Tsoumakas, editors, *Proceedings of the 19th European Conference on Machine Learning and Knowledge Discovery in Databases, ECML PKDD 2022*, volume 13716 of *Lecture Notes in Computer Science*, pages 251–266. Springer, 2022.

[31] Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. Explainable reinforcement learning: A survey and comparative review. *ACM Computing Surveys*, 56(7):168:1–168:36, 2024.

[32] Aditi Mishra, Utkarsh Soni, Jinbin Huang, and Chris Bryan. Why? why not? when? visual explanations of agent behaviour in reinforcement learning. In *15th IEEE Pacific Visualization Symposium, PacificVis 2022*, pages 111–120. IEEE, 2022.

[33] Tom M. Mitchell. *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997.

[34] Stefan Pranger, Bettina Könighofer, Lukas Posch, and Roderick Bloem. TEMPEST - Synthesis Tool for Reactive Systems and Shields in Probabilistic Environments. In *Proceedings of the 19th International Symposium on Automated Technology for Verification and Analysis, ATVA 2021*, volume 12971, pages 222–228. Springer, 2021.

[35] Stefan Pranger, Bettina Könighofer, Martin Tappler, Martin Deixelberger, Nils Jansen, and Roderick Bloem. Adaptive shielding under uncertainty. In *Proceedings of the 2021 American Control Conference, ACC 2021*, pages 3467–3474. IEEE, 2021.

[36] Stefan Pranger, Hana Chockler, Martin Tappler, and Bettina Könighofer. Test where decisions matter: Importance-driven testing for deep reinforcement learning. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024*, 2024.

[37] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[38] Aaron M Roth, Dinesh Manocha, Ram D Sriram, and Elham Tabassi. *Explainable and Interpretable Reinforcement Learning for Robotics*. Springer, 2024.

[39] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, USA, 2014.

[40] Mohan Sridharan and Ben Meadows. Towards a theory of explanations for human-robot collaboration. *Künstliche Intell.*, 33(4):331–342, 2019.

[41] Tyler C Staudinger, Zachary D Jorgensen, and Dragos D Margineantu. Xplane-ml-an environment for learning and decision systems for airplane operations. 2018.

[42] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning - An Introduction, 2nd Edition*. MIT Press, 2018.

[43] Martin Tappler, Stefan Pranger, Bettina Könighofer, Edi Muskardin, Roderick Bloem, and Kim G. Larsen. Automata Learning Meets Shielding. In *Proceedings of the 11th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, ISoLA 2022*, pages 335–359. Springer, 2022.

[44] Martin Tappler, Ignacio D. Lopez-Miguel, Sebastian Tschiatschek, and Ezio Bartocci. Rule-guided reinforcement learning policy evaluation and improvement. In James Kwok, editor, *Proceedings of the 34th International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 6254–6262. International Joint Conferences on Artificial Intelligence Organization, 8 2025. Main Track.

[45] Mark Towers, Ariel Kwiatkowski, Jordan K. Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. *CoRR*, abs/2407.17032, 2024.

[46] Jasper van der Waa, Elisabeth Nieuwburg, Anita H. M. Cremers, and Mark A. Neerincx. Evaluating XAI: A comparison of rule-based and example-based explanations. *Artificial Intelligence*, 291:103404, 2021.

[47] Marko Vasic, Andrija Petrovic, Kaiyuan Wang, Mladen Nikolic, Rishabh Singh, and Sarfraz Khurshid. Moët: Interpretable and verifiable reinforcement learning via mixture of expert trees. *CoRR*, abs/1906.06717, 2019.

[48] Daniël Vos and Sicco Verwer. Optimal decision tree policies for markov decision processes. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence, IJCAI 2023*, pages 5457–5465. ijcai.org, 2023.

[49] Lindsay Wells and Tomasz Bednarz. Explainable AI and reinforcement learning - A systematic review of current approaches and trends. *Frontiers in Artificial Intelligence*, 4:550030, 2021.
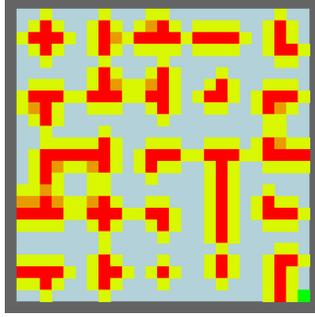
Figure 7: A randomly generated Frozen Lake environment with approximately 15% of holes.
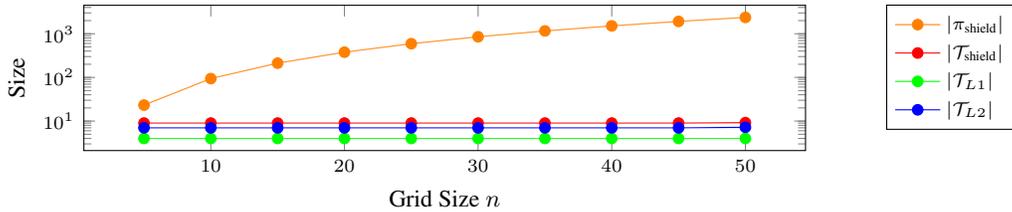

Figure 8: Size comparison between the different approaches using random Frozen Lake instances with approximately 4% of holes.
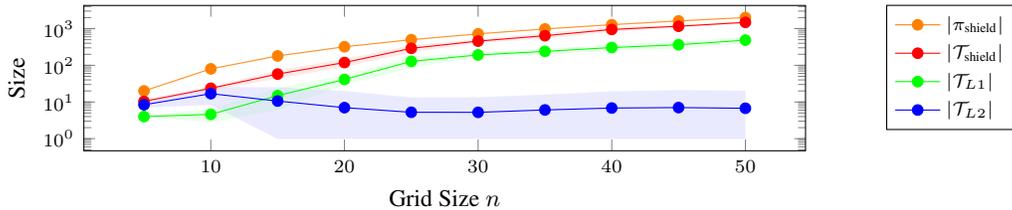
A   Appendix

Figure 9: Size comparison between the different approaches using random Frozen Lake instances with approximately 20% of holes.

A.1   Additional Results —Frozen Lake

Scalability. For the experiments on the scalability of our approach, we randomly generated instances of the Frozen Lake environment of various sizes. To ensure that all environments have a meaningful structure, we divide the overall size $N \times N$ of the instance into tiles of size 5×5. In the center of each tile, we place a randomly shaped connected hole of size 0 to 5. Fig 7 shows such a randomly generated environment. Red areas represent holes, orange areas the dangerous states, and yellow the critical states. For the experiment presented in the main part of the paper, Figure 5, we create the holes in such a way that the expected value of holes is 15% of all fields. The figure shows that the size of a DT is already smaller than the size of the original shield. Our method reduces the size further, even for large problem instances.

In Figure 8, we perform the same experiment with randomly generated instances, but only allow holes of sizes 0 to 2, which leads to an expected value of 4% of all fields being holes. In this case, all DTs are small and remain small for very large instances.

On the other hand, when creating instances with 20% of the entire grid being holes, the size of the DTs grows. This is caused by the fact that holes are close to each other because of their size. When a field is surrounded by several holes, it becomes dangerous instead of critical. Therefore, having many holes interacting with each other causes the overall problem to become more complicated and makes it harder to distinguish critical from dangerous states.
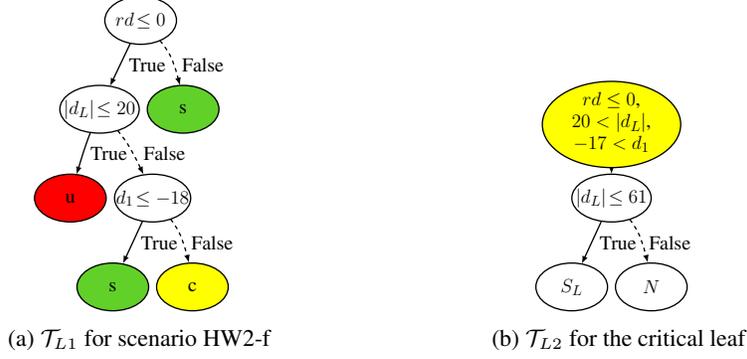
(a) $\mathcal{T}_{L1}$ for scenario HW2-f   (b) $\mathcal{T}_{L2}$ for the critical leaf

Figure 10: Decision Trees for scenario HW2-f

A.2   Additional Results —Highway Cruise Control

In this section, we discuss the remaining scenarios for the Highway environment and elaborate on the suggested predicates.

HW2-f. In the base scenario, the highway has two lanes, and all vehicles are driving at a fixed speed. The agent is slightly faster with a value of v_d than the vehicle on the right lane. The available actions are *switch left (S_L), switch right (S_R),* and *noop (N)*. All actions succeed with probability 1. The feature set of the environment is $\mathcal{V} = \{d_1, l_e\}$, the distance to the vehicle on lane 1, which is the rightmost lane, and the current lane of the ego vehicle. We extend the set of basic predicates $\Gamma = \{d_1, l_e, d_L, N_1, rd\}$ with the following predicates:

- $d_L := (d_1$ if $l_e = 1$ else $100)$ is the distance to the vehicle on the current lane of the ego vehicle. If there is no vehicle on the lane, the distance is set to a high value.
- $rd := d_1 - v_d - 20$ is the remaining distance until action has to be taken immediately. It can be seen as the distance to the safety buffer around the other vehicle. To avoid comparison with the constant v_d, we subtract it.

The DT $\mathcal{T}_{L1}$ describing this instance is depicted in Fig. 10a. It consists of 7 nodes. The DT investigates whether the remaining distance to the car on lane 1 remains large assuming no action is taken. If this is the case, every action is allowed. Otherwise, the DT compares to the safety threshold to filter out unsafe states. Lastly, when the agent is in front of the vehicle on lane 1, it may perform any action. Otherwise, as the root tests that the agent is close to the vehicle on lane 1, the state is critical. Figure 10b shows $\mathcal{T}_{L2}$ for the critical state. If the agent is on the left lane, shown by a high lane distance, it may not switch to the right lane. Otherwise, it has to switch to the left lane to avoid a collision. Computation of $\mathcal{T}_{L1}$ was completed within less than 1.4 seconds, $\mathcal{T}_{L2}$ computation required on average 1.6 seconds and shield creation 0.1 seconds.

HW2-c. In this scenario, the agent is allowed to change its speed and accelerate or decelerate by a fixed value of Δ_v. Therefore, the set of features is extended with the velocity of the ego vehicle and the vehicle on lane 1 to $\mathcal{V} = \{d_1, l_e, v_e, v_1\}$. The agent has the additional actions *accelerate (A)* and *brake (B)*. We provide the predicates $\Gamma = \{d_1, l_e, v_e, v_1, d_L, N_1, rd_A, rd_B\}$ with the addition of:

- $rd_A := (d_1 - (v_e + \Delta_v - v_1) - 20$ if $l_e = 1$ else $100)$ is the remaining distance until the safety requirement is violated, assuming the agent accelerates and the other vehicle is on the same lane.
- $rd_B := (d_1 - (v_e - \Delta_v - v_1) + 20$ if $l_e = 1$ else $100)$ is the remaining distance until the safety requirement is violated, assuming the agent decelerates and the agent is on lane 1.
- $N_1 := (d_1 \in [-20 + v_d, 20 + vd])$ which captures whether in the next time step, after the ego car has approached the other vehicle by v_d space units, it will be within the safety zone of the other vehicle. In this case, the ego vehicle is considered to be next to the other vehicle.

The new predicates are a refinement of rd to allow for a better understanding of the behavior of different actions.
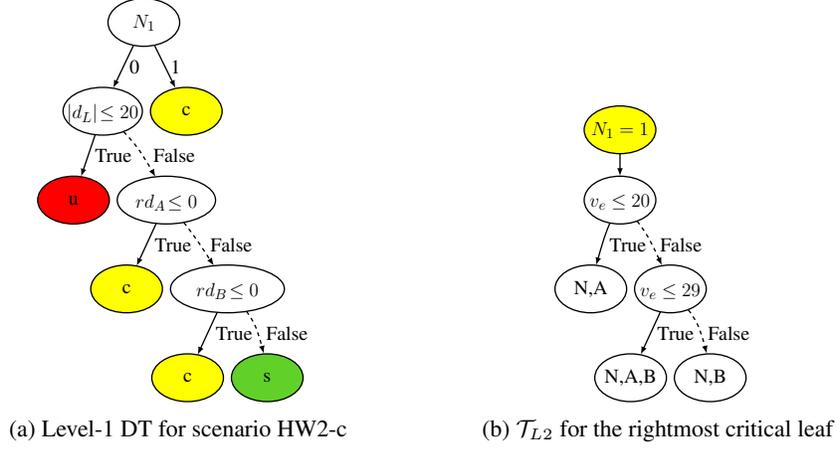
16

(a) Level-1 DT for scenario HW2-c

(b) $\mathcal{T}_{L2}$ for the rightmost critical leaf

Figure 11: DTs for scenario HW2-c

(a) $\mathcal{T}_{L1}$ for scenario HW3-f

(b) $\mathcal{T}_{L2}$ for the critical leaf with $N_1 = 0$ and $N_2 = 1$

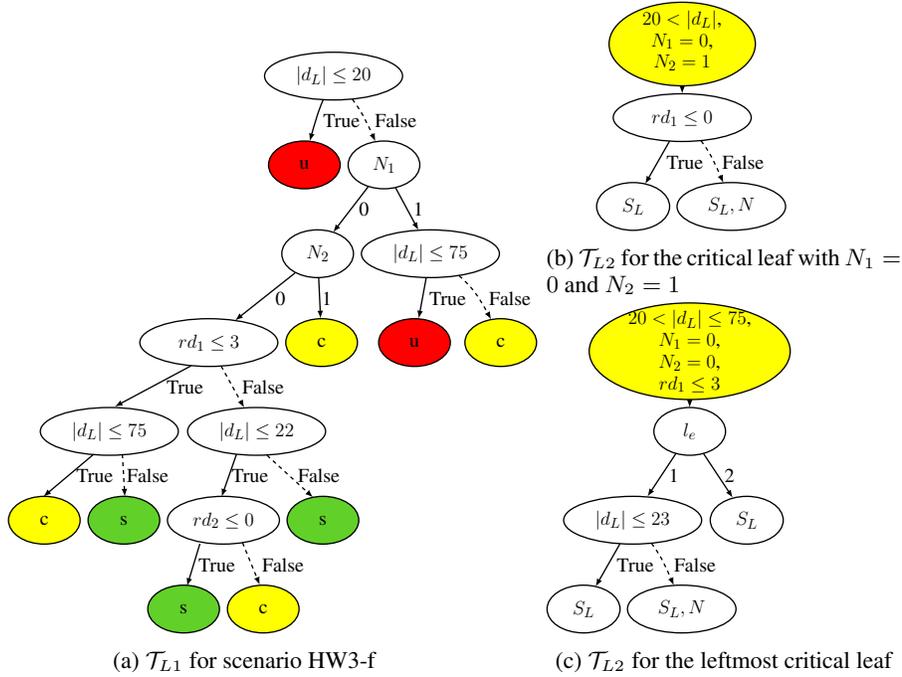(c) $\mathcal{T}_{L2}$ for the leftmost critical leaf

Figure 12: DTs for scenario HW3-f

$\mathcal{T}_{L1}$ is depicted in Figure 11a. It explains that when the ego vehicle is next to another vehicle, some actions are prohibited. Furthermore, when the current distance falls below the safety distance, the state is unsafe. In all other cases, the criticality depends on whether acceleration and braking are safely possible. In this example, no dangerous states exist as the environment is deterministic. Level-2 DTs consist, on average, of 33 nodes. This is caused by one tree having 81 nodes while all others remain small, similar to the one in Figure 6b, which is also depicted in Figure 11b. Computing $\mathcal{T}_{L1}$ required 1.3 seconds, the average time for $\mathcal{T}_{L2}$ was again 1.3 seconds, and creation of the shield took 0.7 seconds.

HW3-f. This scenario is an extension of HW2-f with an additional lane. All vehicles drive at a fixed speed, with the ego vehicle being slightly faster by a value of v_d. The set of features is extended with the distance to the vehicle on lane 2 $\mathcal{V} = \{d_1, d_2, l_e\}$. We extend the set of predicates $\Gamma = \{d_1, d_2, l_e, d_L, N_1, N_2, rd, co\}$ We define N_1 and N_2 accordingly for the vehicles on lanes 1 and 2, and define rd_1 and rd_2 as rd for the vehicles on lanes 1 and 2, respectively.

17

The Level-1 DT is depicted in Figure 12a. It shows that the classification of the states depends on the distance to the vehicle in front and whether the agent is next to another vehicle. If the vehicle is next to one on lane 1 and has a vehicle close in front, it cannot avoid a collision.

On average, the Level-2 DTs have a size of 2.5 nodes, with the largest one having 5 nodes. Examples are shown in Figures 12b and 12c. Figure 12b corresponds to the critical leaf where $N_1 = 0$ and $N_2 = 1$. $N_2 = 1$ already encapsulates that the agent is currently on the middle lane. The available actions then depend on whether it is too close to the vehicle on lane 1, which disallows action N. In any case, switching to the leftmost lane is allowed. For the leftmost critical leaf, which is shown in Figure 12c, we know that the agent is not next to any vehicle. However, the current remaining distance to the vehicle on lane 1 is low. Therefore, if the agent is on lane 2 it should switch left to avoid a scenario where overtaking the vehicle on lane 2 is no longer possible. If it is on lane 1, the available actions depend to the distance of the vehicle on lane 1. Computation of $\mathcal{T}_{L1}$ was completed within 1.8 seconds, $\mathcal{T}_{L2}$ required, on average, less than 1.3 seconds, and shield creation less than 2.7 seconds.

HW3-c. In the last scenario, we consider a highway with three lanes and allow the ego vehicle to change its velocity. The state space $\mathcal{V}$ is extended with *pa*, which shows what action was taken in the previous step. It is required to precisely model the Highway RL environment, where the speed after no operation or switching lanes also depends on the action of the previous time step.

We allow all predicates of the previous cases and add a new predicate *sbd*, safe braking distance, which compares the braking distance (*bd*) needed for the ego vehicle to brake until it is at most as fast as the vehicle in front of it, to the remaining distance: $sbd := (d_L \geq bd)$. It is an extension of rd_B as rd_B considers switching lanes as a possible action, whereas *rbd* only allows braking. We obtain a Level-1 DT of slightly below 70 nodes, which is depicted in Figure 13. The size of the tree results from the special cases in which the ego vehicle can switch to the right-most lane and break before hitting the vehicle in front of it, or switch back to the middle lane. Introducing a predicate for these special cases reduces the tree size drastically but requires more detailed domain knowledge. This shows the need for the automatic finding of intelligent predicates. Similar behavior can be observed for the Level-2 DTs, which have an average size of 36 nodes. Computation of the $\mathcal{T}_{L1}$ required 17.4 seconds. Level-2 DTs are computed on average in less than 3.6 seconds. Shield computation was performed within 50 seconds.
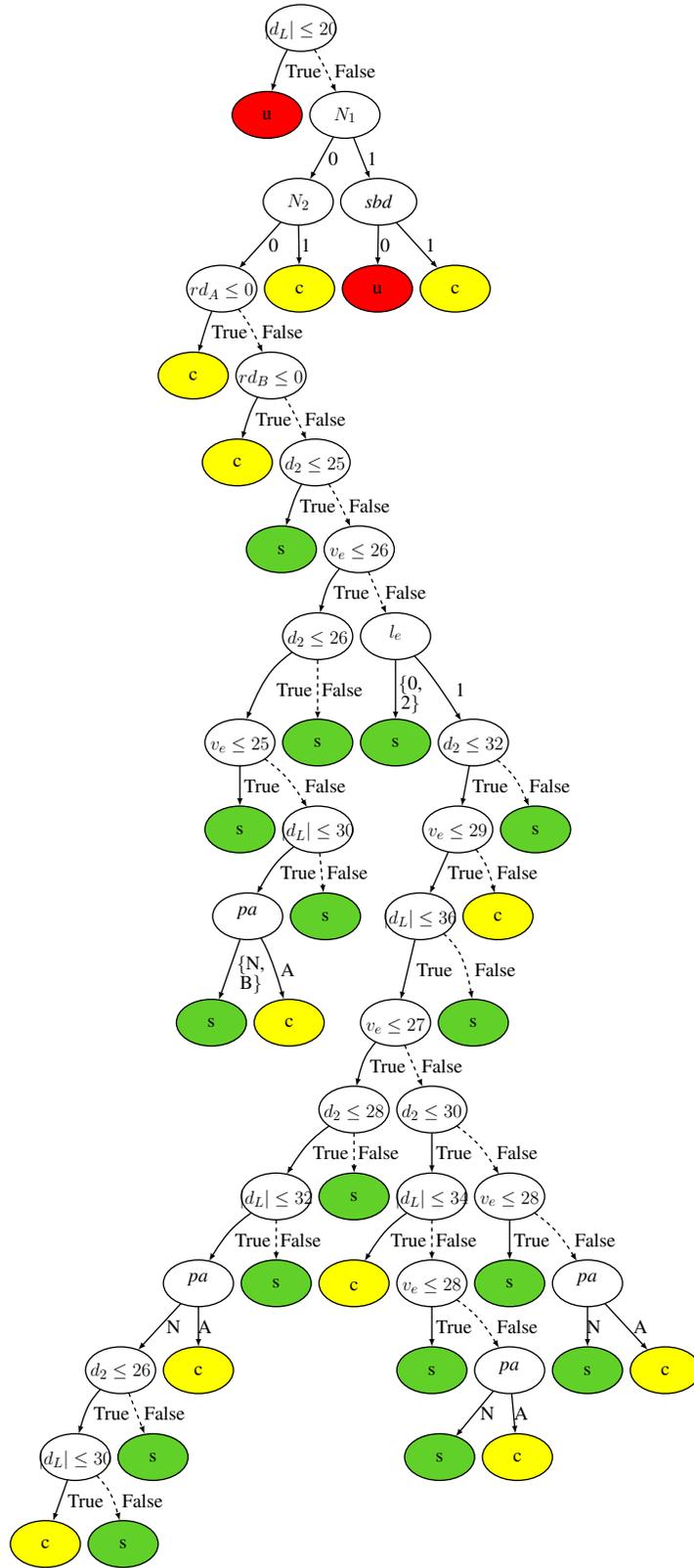
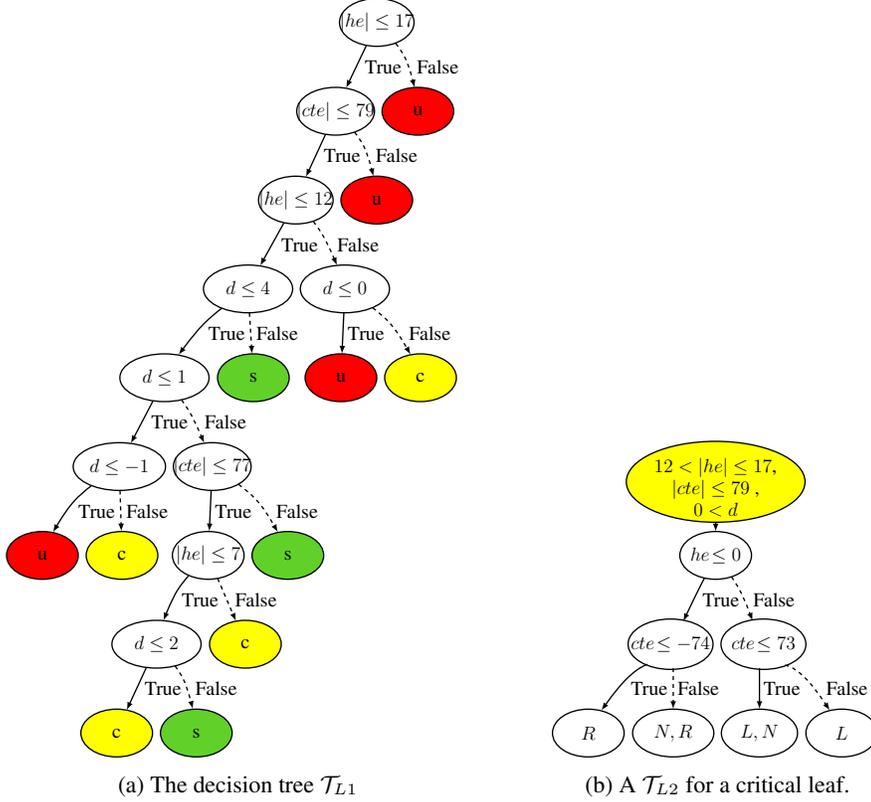Figure 13: Level-1 DT for scenario HW3-f

(a) The decision tree $\mathcal{T}_{L1}$ (b) A $\mathcal{T}_{L2}$ for a critical leaf.

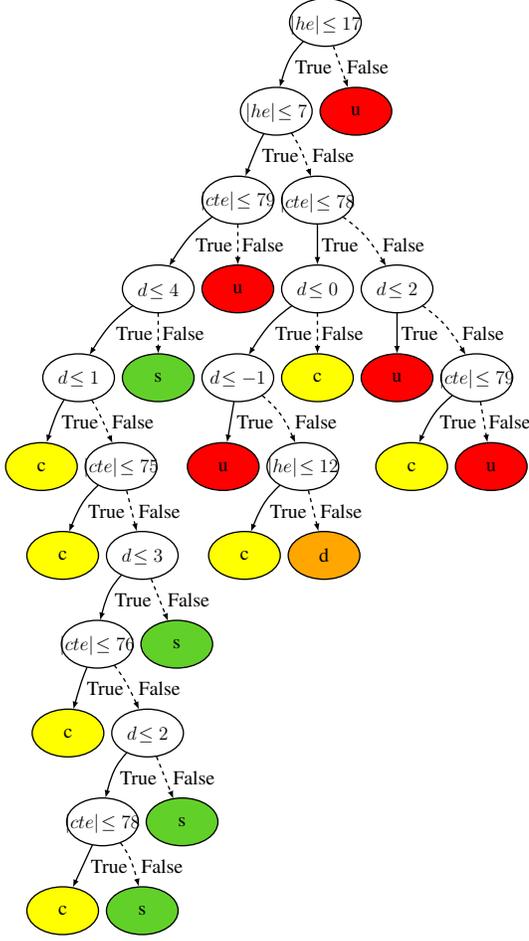Figure 14: Decision trees for the taxiing environment.

A.3 Additional Results —Boeing Taxinet

In this section, we discuss the results for two different autonomous taxiing environments: a model where steering always follows a deterministic update and a slippery model, where steering might cause the airplane to slip, causing its heading to change by $10°$ instead of $5°$. Initially, the airplane is positioned at the centerline with a heading of $0°$. Depending on its current heading, the airplane moves $\frac{he}{5}$ decimeter per time step.

Taxiing. In this base scenario the agent will only be restricted by the shield if either it would reach the maximum heading error of $20°$ or in case it would diverge from the centerline by 80 decimeter. The decision tree $\mathcal{T}_{L1}$ and a $\mathcal{T}_{L2}$ for a critical leaf are shown in Fig. 14a and 14b, respectively. The level-1 DT $\mathcal{T}_{L1}$ nicely shows how user-defined predicates can help explain the shield in a concise way. After classifying the states in which the property has already been violated, the $\mathcal{T}_{L1}$ differentiates between states in which the heading error is smaller or larger than $12°$. It can then immediately make use of the defined distance function. In case the heading error is still small and the distance is sufficiently large, all states are safe and no interference is needed. If the distance is ≤ 40 decimeter $\mathcal{T}_{L1}$ classifies the different scenarios depending on concrete distance, heading and centerline errors. If the heading error exceeds $12°$ and safety has not yet been violated, the shield has to interfere. The restrictions the shield imposes are shown in 14b. This $\mathcal{T}_{L2}$ nicely shows how a compact representation allows us to legibly explain how the shield interferes.

The synthesis time for the shielding policy is 0.19 seconds and the time needed to compute either a $\mathcal{T}_{L1}$ or a $\mathcal{T}_{L2}$ tree is 1.5 seconds.

Taxiing on Slippery Ground. In this second experiment, the aircraft might slip an additional $5°$ with a probability of 10% in case it is steered to the left or to the right. The resulting decision trees are shown in 15. The Level-1 DT $\mathcal{T}_{L1}$, shown in 15a, captures this by more finely distinguishing between the current centerline error and distance to the maximum centerline error. Similarly to the Level-1 DT from the previous experiment, it classifies all states in which the heading error is

(a) The decision tree $\mathcal{T}_{L1}$
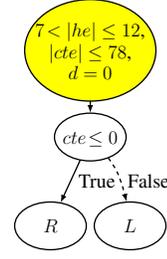
(b) A $\mathcal{T}_{L2}$ for a critical leaf.
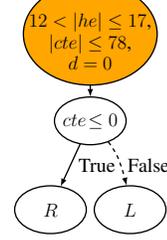
(c) A $\mathcal{T}_{L2}$ for a dangerous leaf.

Figure 15: Decision trees for the slippery taxiing environment.

sufficiently small, $\leq 7°$, and the distance to the maximum centerline error is large enough, as safe. In case the heading error is small, but the centerline error is close to the maximum allowed value, $\mathcal{T}_{L1}$ distinguishes between safe and critical states by capturing the different values for the distance and centerline error. In case the heading error is $> 7°$, the decision tree $\mathcal{T}_{L1}$ has to classify states that are either critical, dangerous, or are already violating the safety specification. Fig. 15b and Fig. 15c show two examples for $\mathcal{T}_{L2}$. We want to highlight these two examples for $\mathcal{T}_{L2}$, as they show how our approach is able to explain both how the shield has to interfere in order to enforce the safety specification, as well as when the safety specification cannot be adhered to anymore and only safest actions can be allowed by the shield anymore.

As above, the synthesis time for the shielding policy is 0.19 seconds and the time needed to compute either a $\mathcal{T}_{L1}$ or a $\mathcal{T}_{L2}$ tree is 1.5 seconds.
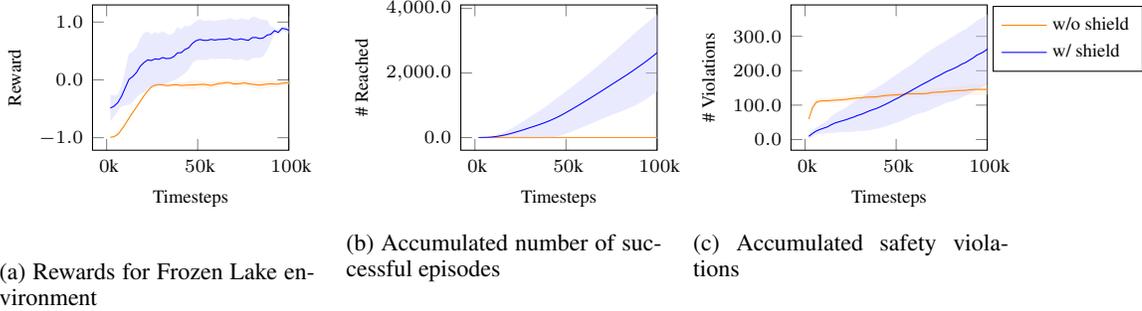
(a) Rewards for Frozen Lake environment

(b) Accumulated number of successful episodes

(c) Accumulated safety violations

B Supplementary Material

We have trained agents on the Frozen Lake environment from Fig. 3a and the four different settings in the Highway environment using the implementations from Stable-Baselines3 [37]. The library provides a *MaskablePPO* implementation, which is perfectly suited for safe RL via shielding. We have compared shielded training with training without a shield using *PPO*. All training runs have been conducted using the default parameters.

Due to the simulator for the Boeing TaxiNet environment being closed-source, we were not able to train agents for these problem instances.

B.1 Frozen Lake Environment

The task of the agent in this environment is to reach the goal state without falling into a hole. The agent can move in any cardinal direction and will succeed with a probability of 0.95. With a probability of 0.05 it will slip in any of the other cardinal directions where it is not obstructed by a wall. Upon reaching the goal the agent receives a reward of 1, otherwise, when the agent falls into a hole it receives a negative reward of -1.

Fig. 16a, 16b, and 16c show the training results averaged over 5 runs. These results show that the shield enables the agent to finish its task, while training without the shield does not succeed. When training without a shield, the agent is not able to explore the critical areas around the holes and therefore stays at the safe area near the initial position only. We want to remark that due to the stochasticity exhibited in this environment, complete safety cannot be guaranteed.

B.2 Highway Environment

The task of the agent in this environment is to reach the end of the highway as fast as possible without causing a crash. The agent can switch lanes, do nothing, or change its velocity, if it is not fixed. The agent receives a reward of 1.0 for driving on the rightmost lane, a reward of 0.5 on the leftmost lane, and a reward of 0.75 for driving on the middle lane in the environments HW3-f and HW3-c. If the agent crashes into another car, it receives a negative reward of -1.0.

We show the training results, averaged over 5 runs in Figures 17a to 17h. We want to use the example of environment HW3-c to highlight the need for accurate world models: Due to the world model not being 100% accurate, we have exhibited a small number of crashes in the shielded training run.
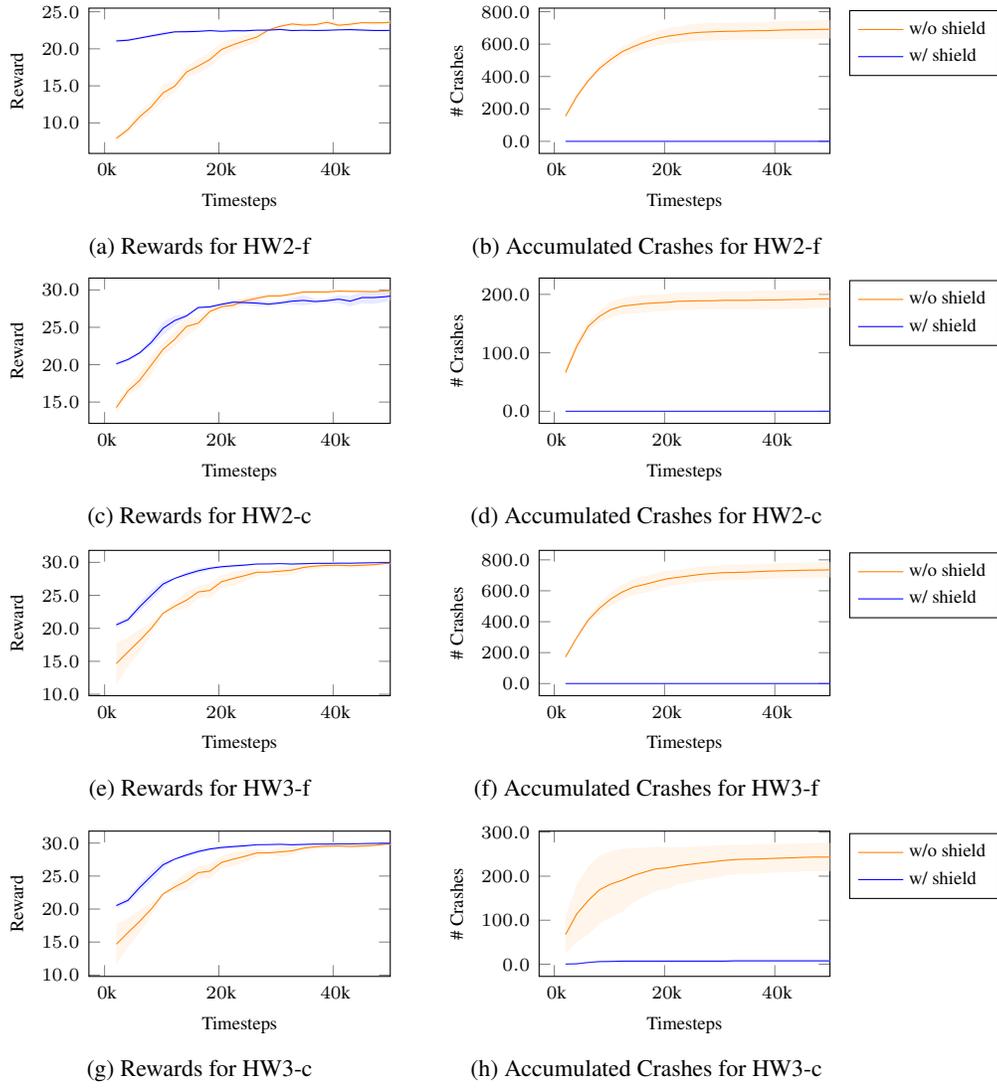
(a) Rewards for HW2-f

(b) Accumulated Crashes for HW2-f

(c) Rewards for HW2-c

(d) Accumulated Crashes for HW2-c

(e) Rewards for HW3-f

(f) Accumulated Crashes for HW3-f

(g) Rewards for HW3-c

(h) Accumulated Crashes for HW3-c

Figure 17

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We explain how we generate the described explanations and show in the scalability study in Figure 5 the size of the shield and the decision trees.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We clearly highlight in the conclusion and experiments how our method depends on well-defined predicates for larger problem instances.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: There are no theoretical results that require a proof.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We explain the set of predicates we use for computing the DTs. Furthermore, we name the used tools. Furthermore, we provide the code to reproduce the results in the supplementary material.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide the code in the supplementary material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We explain the parameters we used for instantiating the RL environments and provide the world models in the supplementary material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For the randomized scalability experiments, the plots show the mean and highlight the area plus/minus standard deviation around it. Each test case was run 10 times with different random seeds. In all other experiments, the results are deterministic.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We state in the beginning of our section on evaluation which machine we used.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: No human participants were involved in the research. Furthermore, all experiments are based on models, which are not recorded from human interactions. We cannot forsee a negative usecase of our research.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: In the introduction we explain the need for safe RL learning. Our method can have an impact in this direction. We forsee no negative societal impacts.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We do not create any data or model that has a high risk of misuse.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite the sources of the used RL environments.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.

28

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: We do not introduce new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This paper does not involve crowdsourcing or human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: Our proposed method does not involve LLMs.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.